



**UNIVERSITY OF STRATHCLYDE**

**DEPARTMENT OF  
MATHEMATICS AND STATISTICS**

**Estimation of Value-at-Risk**

**by**

**Karan Mukesh Savla**

**202289149**

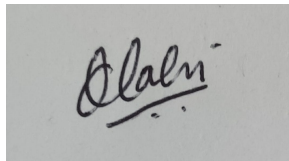
**MSc Quantitative Finance**

**2022/2023**

## Statement of work in project

The work contained in this project is that of the author and where material from other sources has been incorporated full acknowledgement is made.

Signed

A handwritten signature in black ink, appearing to read 'Karan', is shown on a light gray rectangular background.

Print Name          Karan Mukesh Savla

Date                  August 14, 2023

Supervised by **Dr Ziazhu Pan**

# Contents

<b>1</b>	<b>Value-at-Risk (VaR) and Financial Risk Management</b>	<b>4</b>
1.1	Defination and History of VaR . . . . .	4
1.2	Literature Review . . . . .	10
<b>2</b>	<b>Methodology for Calculation of VaR</b>	<b>12</b>
2.1	Historical Simulation. . . . .	12
2.2	Variance-Covariance Approach (Delta-Normal) . . . . .	19
2.3	Monte Carlo Simulation . . . . .	25
2.4	Extreme Value Theory approach . . . . .	28
<b>3</b>	<b>Real Data Analysis and Comparison of Different Methods</b>	<b>31</b>
3.1	Four Time Series Datasets of Financial Returns . . . . .	31
3.2	Estimation of VaRs Using Different Methods . . . . .	32
3.2.1	Historical Simulation . . . . .	32
3.2.2	Variance-Covariance Approach (Delta Normal Approach) . . . . .	34
3.2.3	Monte Carlo Simulation . . . . .	36
3.2.4	Extreme Value Theory Approach (Semi-Parametric Approach) . . . . .	38
3.3	Methodology of Back-testing . . . . .	40
3.3.1	Introduction of Backtetsing . . . . .	40
3.3.2	Kupiec test . . . . .	42
3.3.3	Traffic Light, Unconditional Coverage, Frequency, Failure Rate test . . . . .	44
3.4	Comparison of Different Methods by Back-testing . . . . .	47
3.4.1	Kupiec Backtesting . . . . .	47
3.4.2	Traffic Light, Unconditional Coverage, Frequency, Failure Rate test . . . . .	57
<b>4</b>	<b>Summary and Conclusion</b>	<b>67</b>
<b>5</b>	<b>References</b>	<b>69</b>
<b>6</b>	<b>Appendix</b>	<b>71</b>

# 1 Value-at-Risk (VaR) and Financial Risk Management

## 1.1 Defination and History of VaR

Value at Risk (VaR) is a term that refers to a method of calculating the loss in the value of the portfolio over a specific time period and for a particular distribution of historical returns.  $\text{VaR}(1\%)$ ,  $\text{VaR}(5\%)$ , and  $\text{VaR}(10\%)$  are the abbreviations for a vaR of 1%, 5%, and 10%, respectively [5]. As an illustration, a risk analyst determines that the daily 5% VaR is \$10000. A  $\text{VaR}(5\%)$  of \$10000 indicates that there is a 5% risk that the portfolio will suffer a loss of at least that amount on any given day. Additionally, it stated that there is a 95% chance that the portfolio will either see a gain or a loss of less than \$10000 [5].

The word “risk” comes from the Italian “risiko” (which means “danger”), the French “risque” (which means “to dare,” or both). The volatility of unforeseen results, which can reflect the value of assets, equity, or earnings, is referred to as risk. Businesses are vulnerable to a range of risks, which can be divided into two categories: business risk and financial risk [4, 1].

The accomplishment of the organization’s aims and objectives is hampered by business risks. Every organisation exists to give its stakeholders observable benefits. Management choices either produce or destroy value. Making decisions requires management to take into account information about the internal and external environments, allocate limited resources, and adapt activities to changing environments. Decisions also involve the assessment of risk and opportunity. The outcome of risk is the potential of gaining or losing something of tangible value. Examples [1]:

- A fisherman starting a voyage on a fishing expedition may result in loss of life.
- An infant climbing on a window pane may result in damage or injury.
- A corporate launching a new product or service in the market place may result in failure thereby leading to financial and reputational losses.

Financial risk defined by NASDAQ as the risk that the cash flow of an issuer will not be adequate to meet its financial obligations. Also referred to as the additional risk that a firm's stockholder bears when the firm uses debt and equity. In financial markets, one may need to measure market risk, credit risk, operational risk, liquidity risk, and legal risk if there are regulatory or civil actions taken.

To assess this risk is important because this risk may interact with each other.

Market risk - The risk of losses caused by adverse changes in the market variables such as interest rate, Foreign Exchange rate, equity price and commodity price. Market risk is also called as 'Systematic risk', cannot be eliminated through diversification, though it can be hedged against. Sources of market risk include recessions, political turmoil, and changes in interest rates, natural disasters and terrorist attacks [4, 1].

Credit risk - The risk of loss arising from outright default due to the inability or unwillingness of the customer or counterparty to meet their commitments. Credit risk is the probability of loss from a credit transaction. It is also called as default risk. Example is HERSTATT'S SETTLEMENT RISK [4, 1].

Operational risk - The risk associated with the operations of an organization. It is a risk of loss resulting from failure of people employed in the organization, internal process, systems or external factors acting upon it to detriment of the organization [4, 1].

Liquidity risk - The potential inability to meet commitments as they fall due. liquidity risk also arises on account of its failure to address the changes in the market conditions that affect its ability to liquidate its asset quickly and with the minimal losses. Example is failure of David Askin in Market-Neutral strategy [4, 1].

Legal risk - Arises from the uncertainty due to legal actions or uncertainty in the application, interpretation of contracts, laws or regulations. legal risk is the risk arising from failure to comply with statutory or legal requirements. Example of city councils in Britain - The court decree that the city council did not have the authority to enter into those transactions and therefore councils are not responsible for the losses. As a result bank has to absorb the losses amounting \$178 million [4, 1].

**Interactions between these risks are as follows:**

- If an institution needs to swiftly sell assets to meet funding requirements, an increase in market risk, such as a sharp decrease in asset prices, might result in losses and cause liquidity issues [4].
- Operational risk occurrences, like a cyber-attack or system failure, can impair market functionality, have an adverse effect on liquidity by making it more difficult to process transactions, and possibly even increase the risk of credit risk if customer data is compromised [8].
- Credit risk can affect the institution's ability to generate enough cash flows to meet obligations if it experiences a high number of loan defaults [6].
- Legal risk can affect all other risks since it may result in monetary losses, reputational harm, and operational disruptions if legal or regulatory actions are taken [7].

The idea of “real time risk” has become more significant since financial markets began to automate. Real-time risk, which might result from fraudulent action by a few chosen market participants or flash crashes, is the likelihood of experiencing an instantaneous or very instantaneous loss [10]. On August 1, 2012, Knight Capital Group (KCG) suffered a loss of US\$440 million in less than 30 minutes due to a poorly tested runaway algorithm that the company had implemented. Real-time risk has also caught the attention of regulators. For banks to be stable, Basel II mandates a real-time risk management system [10].

The recent growth of the risk management industry can be traced directly to the increased volatility of financial markets since the early 1970s. consider the following developments [4]:

- The 1971 “Nixon shock” (in which the U.S. discontinued the fixed dollar-to-gold conversion)
- The oil price shocks during the 1970s, the stock market Black Monday in 1987
- The Japanese stock bubble in 1989.

The use of financial derivatives increased significantly in the 1970s and 1980s, in part to manage the risk of these more volatile markets. In turn, risk analysis received increasing

attention because of many cases where the market was flooded with the losses due to derivatives. some of the following cases are as follows:

- Orange County Bankruptcy (1994): Treasurer Robert Citron's investing approach led to the Orange County, California, bankruptcy. To boost profits for the county's investment pool, Citron had made significant investments in high-risk derivative instruments like interest rate swaps and collateralized mortgage obligations. However, when borrowing rates suddenly increased, the county sustained significant losses totaling more than \$1.6 billion and declared bankruptcy.
- Barings Bank Collapse (1995): One of the oldest and most reputable banks in the UK, Barings Bank, fell apart in 1995 as a result of significant losses from derivatives trading. Unauthorised speculative trading in futures contracts on the Singapore International Monetary Exchange (SIMEX) have been undertaken by Barings Bank dealer Nick Leeson. His deals generated enormous losses totaling £827 million (about \$1.3 billion), which ultimately caused the bank to go bankrupt.
- Metallgesellschaft (early 1990s): The business traded oil and used derivative contracts to protect itself from any losses brought on by changes in oil prices. With the intention of covering them when the time came with less expensive short-term contracts, Metallgesellschaft offered oil customers long-term forward contracts at set prices. However, a sharp increase in oil prices at the start of the 1990s compelled the corporation to complete its obligations by buying oil at higher market prices. As a result, Metallgesellschaft suffered significant losses that are believed to be worth \$1.3 billion, as it was unable to satisfy its financial obligations due to a liquidity crisis. The losses had a significant effect on the business, causing its stock price to drop and necessitating a rescue from a number of banks and financial organisations.

Due to the above unpredictability of risk in various scenarios and circumstances the concept of Value at Risk (VaR) was started taking seriously and many big institutions started the implementation of VaR in their businesses[5, 2].

Francis Edgeworth is credited with making the first attempts to quantify risk and, consequently, express prospective portfolio losses in 1888 [2]. He advocated using historical data as the foundation for forecasting future probabilities, making significant contributions to statistical theory.

When Dickson H. Leavens published in 1945 that is regarded as the first mention of the VaR[2], the story of the device continues. It was a illustration of a portfolio that had 10 government bonds. He proposed that either the bond matures for \$1,000 or the predetermined requirements are not met, rendering it useless. He also believed that the ties exist independently of one another. He makes an effort to calculate the portfolio's value. He didn't use name value at risk in his task. He frequently made reference to "the spread between the likely profit and loss" as well as the most likely mean standard deviation, which is a crucial component of VaR and is used to quantify risk. Harry Markowitz, who received the Nobel Prize in Economics in 1990 for his groundbreaking work in the field of portfolio theory [2], and Arthur D. Roy independently proposed VaR indicators in 1952[2], which were remarkably comparable, Kollar (2014). Three months later, Arthur D. Roy also proposed VaR indicators. Both were looking for a strategy that would maximise profit at a particular level of risk. Covariance was a crucial factor in their plans, but the VaR indicators varied greatly. Regarding how a probability distribution should be specified, Roy and Markowitz made some hypotheses. Roy need knowledge of the variance-covariance matrix of risk factors as well as the vector of average revenues or losses in order to calculate VaR. These had to be calculated using previous data. Knowing the variance-covariance matrix was sufficient to perform Markowitz's VaR calculation. Both figured that combining statistical methods with the wise judgement of specialists was required for their calculations. JP Morgan, a US investment bank, is credited with using current VaR most frequently. Dennis Weatherstone, the bank's chairman, requested for something straightforward that would cover the full range of dangers the institution would encounter over the course of the next 24 hours. The VaR was created by Bank utilising Markowitz portfolio theory. However, it was known as the 4:15 report at the time. It is unknown where the phrase "Value at risk" came from.

Till Guldemann can be viewed as the creator of the term value at risk while the head of



global research at JP Morgan in late 1980's [2].

Value at Risk (VaR) is used to control banks in a manner that is substantially influenced by the Basel Accords, which have evolved from Basel I to Basel III.5 or the Fundamental Review of the Trading Book (FRTB) [5]. The Basel Accords gave banks the option to determine capital requirements using their internal VaR models as long as they comply with specified rules and regularly perform backtesting. To ensure banks maintain adequate capital under market stress, Basel III introduced the Stressed VaR (sVaR), while the FRTB established the Expected Shortfall (ES), a measure that is more sensitive to tail risk than VaR.

However, putting these standards into effect is expensive, and the latitude for customization makes standardisation challenging, resulting in different reported risk levels between banks. Despite advancements, rules have come under fire for their efficacy, particularly given the flaws in VaR that the 2008 financial crisis exposed [5]. As a result, continuing discussions centre on developing the optimum risk management techniques while weighing the advantages and disadvantages of VaR, sVaR, and ES. Because these restrictions may indirectly affect banks' investment plans and risk-management procedures, the effects of these regulations on bank behaviour and financial stability are also taken into account.

## 1.2 Literature Review

This literature review summarises key ideas from Philippe Jorion's (2007) book "Value at Risk: The New Benchmark for Managing Financial Risk", giving readers a thorough knowledge of Value at Risk (VaR) and its function as a key metric in financial risk management. The review analyses the historical development of VaR as a risk-management tool since the 1990s and Jorion's definition of VaR as a quantile measure. Additionally, it explores different VaR estimating techniques such as historical simulation, variance-covariance approach, Monte Carlo simulation, and extreme value theory (EVT), providing details on their uses, underlying assumptions, benefits, and drawbacks. The practical uses of VaR are addressed, including its use in risk budgeting, asset allocation, and portfolio management across various financial markets. It is emphasised that backtesting and model validation are essential for the accuracy of VaR estimates. VaR may have downsides and be criticised for failing to adequately account for extreme occurrences and tail risk, among other things. The assessment also examines the Basel II and III regulatory framework for VaR as well as the difficulties financial institutions encounter when deploying VaR-based systems. It also looks into the usefulness of VaR during financial crises and the lessons that may be used to risk management strategies and VaR modelling. As a result, readers of this paper will have a thorough understanding of VaR and how it is used in financial risk management.

Through the key work "Elements of Financial Risk Management" by Peter Christoffersen (2012), this literature study offers a thorough grasp of financial risk management. The review dives into Christoffersen's core ideas, including as risk modelling, risk aggregation, and risk categories like market, credit, liquidity, and operational risks. Value at risk (VaR), expected shortfall (ES), stress testing, and more subtle elements like conditional VaR, backtesting, and model selection are all further examined. Christoffersen's case studies including financial organisations, investment corporations, and insurance companies demonstrate how these ideas are applied in real-world situations. The assessment also examines how financial risk management is changing, highlighting the impact of technological improvements, regulatory changes, and the growing interconnectedness of the global financial system. The insights provided here are intended to furnish a holistic understanding of financial risk management, beneficial to researchers and practitioners alike.

This literature study examines the key ideas from Kevin Dowd's (2005) book "Measuring Market Risk" to offer a thorough grasp of market risk measurement in the context of financial risk management. The examination deconstructs the key ideas Dowd presents, such as volatility, correlation, and risk-return correlations, and it also looks more closely at methods for measuring risk, such as expected shortfall (ES) and value at risk (VaR). The paper carefully examines various VaR estimating techniques, stressing the underlying presumptions, advantages, and disadvantages of each. It covers the use of extreme value theory (EVT) in predicting market risk and highlights the significance of backtesting and evaluation in verifying VaR models. Additionally, it provides examples of real-world applications and case studies to show how financial institutions integrate market risk measurement into their risk management plans.

## 2 Methodology for Calculation of VaR

### 2.1 Historical Simulation.

A risk management tool used to calculate the possible losses of an investment or financial portfolio based on previous market data is known as historical simulation, commonly referred to as historical VaR (Value at Risk) [5, 6]. It is a technique used to evaluate risk and quantify the possible downside of an investment over a given time frame.

The fundamental goal of historical simulation is to build a variety of scenarios using historical market data that accurately represent the behaviour of the financial instruments in the portfolio [8]. The technique seeks to capture the innate volatility and correlations among the assets by looking at the past price changes and returns of these instruments [4].

The fundamental process entails gathering historical data, building a model that can accommodate it, and then performing simulations using this model [4]. A thorough understanding of the historical context should serve as the foundation for the model itself, hence extensive background research is frequently required. Understanding the politics, social structure, and other pertinent facets of the appropriate time period might be a part of this. After the model is created, the simulation can be conducted, and the outcomes can be examined.

In the most simple case, this method applies current weights to a time series of historical asset returns, that is [5],

$$R_{p,k} = \sum_{i=1}^N w_{i,t} R_{i,k} \quad k = 1, \dots, t$$

The weights  $w_t$  are kept at their current values. This return does not represent an actual portfolio but rather reconstructs the history of a hypothetical portfolio using the current position. The approach is sometimes called bootstrapping because it uses the actual distribution of recent historical data without replacement. Each scenario  $k$  is drawn from the history of  $t$  observations.

More generally, the method can use full valuation, employing hypothetical prices for the risk factors, which are obtained from applying historical changes in prices to the current level of prices, that is

$$S_{i,k}^* = S_{i,0} + \Delta S_{i,k} \quad i = 1, \dots, N$$

A new portfolio value  $V_p^*, k$  then is computed from the full set of hypothetical prices, perhaps incorporating nonlinear relationships

$$V_k^* = V(s_i^*, k)$$

Note that to capture vega risk, owing to changing volatilities, the set of risk factors can incorporate implied volatility measures. This creates the hypothetical return corresponding to simulation, that is,

$$R_{p,k} = \frac{V_k^* - V_0}{V_0}$$

VAR then is obtained from the entire distribution of hypothetical returns, where each historical scenario is assigned the same weight of  $(\frac{1}{t})$ . Because the approach does not assume a parametric distribution for the risk factors, it is called nonparametric.

Here's a detailed breakdown of how this works:

- **Data Gathering:** Getting historical data is the initial stage in a historical simulation. This often entails gathering daily, weekly, or monthly returns for each asset in the portfolio over a predetermined time frame (for example, the previous one to five years) in the context of a financial portfolio. The objective of the analysis and the properties of the assets will determine the time period and look-back period to be used [4].
- **Calculating Returns:** The portfolio returns are then determined for each time period in the historical data set. One way to achieve this is to determine the % change in value of each asset, or one can use a more sophisticated model that accounts for splits, dividends, and other variables [4].
- **Sorting and Ranking:** The returns are then from worst to best (biggest loss to highest gain), starting with the returns. Each return is given a rank, with 1 being the lowest return [9].
- **Determining the VaR:** The next step is to select a confidence level (such as 95% or 97.5% or 99%) and locate the return associated with that rank in the sorted list in order to calculate the VaR. Since 5% of 1000 = 50, the 95% VaR, for a dataset with 1000 returns, would represent the 50th worst return [6].

- **Backtesting:** The VaR model is frequently backtested by contrasting the predicted VaR with the actual results in a different dataset in order to ascertain its accuracy. This can give valuable information about the model's precision and any potential biases or flaws [5].

#### **Strengths:**

- **Non-Parametric:** Historical simulation is flexible and adaptable since it does not presuppose a particular statistical distribution for financial returns. When the returns are not regularly distributed, such as when "fat tails" or skewness are present, this characteristic can lead to more precise estimations [7].
- **Easy to Understand and Implement:** The historical simulation method is rather simple and clear-cut. It is simpler to apply and explain to non-specialists than other methods like the Variance-Covariance method or Monte Carlo simulation since it makes fewer assumptions and requires fewer complicated calculations [5].
- **Incorporates Historical Events:** The historical simulation method naturally captures the influence of past events since it uses real historical data. This includes times of market stress and instability, giving insight into how the portfolio might behave in the future under similar circumstances [4].

#### **Limitations:**

- **Assumes that the Past Predicts the Future:** One of the main limitations of historical simulation is that it assumes that the future will resemble the past. This means that it might not accurately estimate risk during periods of structural change or unexpected extreme events that haven't occurred in the past [7].
- **Data Quality and Availability:** The quality and extent of the historical data are critical to the success of historical simulation. The accuracy of the simulation can be impacted by insufficient data or changes in data quality over time. Additionally, there might not be enough historical data for a trustworthy simulation for recently created financial instruments or stocks with brief trading histories [8].
- **Disregards Future Market Expectations:** The historical simulation technique does not take into account market expectations or anticipated future changes in the economic environment; it just considers what has already transpired [7].

- **Uniform Weighting of Past Data:** Every data point in the historical sample is treated equally in the traditional historical simulation approach, regardless of when it occurred. In other words, very ancient data, which may be less pertinent to the state of affairs now, is given the same weight as more recent data. By assigning more weight to recent data, certain historical simulation variations (such as exponentially weighted historical simulation) attempt to overcome this constraint [8].
- **Tools and Software:** A variety of equipment and software are available for running historical simulations. This can include everything from specialised simulation software like MATLAB or Simul8 to general-purpose statistical software like R or Python's statistical libraries. Additionally, specific historical simulation software may be used in several domains.
- **Future Directions:** Historical simulation is a field that is constantly changing. Simulations are becoming increasingly sophisticated and precise with the development of more powerful computing capabilities and the incorporation of machine learning and AI. Future research may focus on improving current models, creating fresh methods for collecting and examining historical data, or using historical simulation approaches on new topics or issues [8].

Since historical simulation does not explicitly describe or make assumptions about the volatility and correlation structure of the data, it is a non-parametric method for determining Value-at-Risk (VaR). To anticipate probable future losses, it only examines historical data. The realised volatility and correlation structure over the time period under consideration are nonetheless implicitly captured by the historical data. As a result, the volatility and correlation structure may have the following effects on the VaR determined by the historical simulation approach [4]:

**Correlation :** Correlation is a major factor in determining the VaR of a portfolio of assets [5, 10]. If the stocks have a perfect positive correlation, they will typically move up or down simultaneously, and the portfolio's risk is effectively the sum of the risks of each individual stock [5]. Diversification advantages, however, may exist if the stocks are not totally connected, since a decline in one stock's value may be partially offset by an increase in another [10]. The portfolio risk (and hence VaR) in this scenario would be lower than the total of the risks associated with each stock. Since a negative return on

one stock would always be offset by a positive return on the other [15], if the stocks are perfectly negatively correlated, the risk to the portfolio might possibly be very low.

Coefficients of correlation range from -1 to +1. When two stocks have a coefficient of +1, they move perfectly in the same direction, when they have a coefficient of -1, they move perfectly in the other direction, and when they have a coefficient of 0, they move independently of one another [5, 10].

Table 1: Individual Stocks Correlation Coefficients for Historical Simulation

<b>Companies</b>	<b>Correlation Coefficient</b>
Apple and Amazon	0.5069769
Apple and Google	0.5758562
Apple and Netflix	0.324561
Amazon and Google	0.6330924
Amazon and Netflix	0.4840794
Google and Netflix	0.435686

Interpretation of Table 1:

- Apple and Amazon: A moderately positive correlation of 0.5069769 is found between Apple and Amazon. Amazon's stock price typically rises when Apple's stock price does, and vice versa.
- Apple and Google: A correlation of 0.5758562 between Apple and Google suggests a moderately significant positive association. Google's stock price typically rises when Apple's stock price rises, and vice versa.
- Apple and Netflix: Apple and Netflix have a 0.324561 correlation, which points to a tenuous positive association. Although there is a tendency for Apple and Netflix stock prices to move in the same direction, this correlation is not as strong as it is for other stocks.
- Amazon and Google: The correlation between Amazon and Google is 0.6330924, which shows a very significant positive association. Google's stock price typically rises when Amazon's stock price rises, and vice versa.



- Amazon and Netflix: A moderately favourable correlation of 0.4840794 is seen between Amazon and Netflix. Netflix’s stock price typically rises when Amazon’s stock price does, and vice versa.
- Google and Netflix: The correlation between Google and Netflix is 0.435686, which points to a generally good association. In general, Netflix’s stock price rises when Google’s stock price rises and vice versa.

When putting together a diverse investing portfolio, these connections are crucial. Less connected stocks offer better benefits of diversification since while one stock is down, the other might not be, reducing possible losses. It’s crucial to remember that these correlations are historical, based on previous price movements, and may not necessarily accurately forecast relationships in the future.

**Volatility:** VaR is typically higher when volatility is higher. This is because the VaR, which is a measure of downside risk, is essentially a percentile of the return distribution; if the distribution is broader (which corresponds to higher volatility), the negative tail of the distribution will be more extreme [5, 10]. On the other side, a stock with low volatility would often have a lower VaR because the possibility of significant negative returns is reduced [10]. Volatility is a statistical measure of the dispersion of returns for a given security or market index, which can be interpreted as the degree of variation of a trading price series over time [9, 11].

Table 2: Volatility

Company	Volatility
Apple	0.2910411
Amazon	0.3260103
Google	0.2702451
Netflix	0.475937

Interpretation of Table 2:

- Apple: The volatility for Apple is 0.2910411. This means that, based on historical returns, the price of Apple's stock is expected to fluctuate by about 29.1% over the specified period.
- Amazon: The volatility for Amazon is 0.3260103. This means that the price of Amazon's stock is expected to fluctuate by about 32.6% over the specified period.
- Google: The volatility for Google is 0.2702451. This suggests that the price of Google's stock is expected to fluctuate by about 27.0% over the specified period.
- Netflix: The volatility for Netflix is 0.475937. This indicates that the price of Netflix's stock is expected to fluctuate by about 47.6% over the specified period.

These findings suggest that Google's stock has been the least volatile during the study period, while Netflix's stock has been the most erratic. Keep in mind that although increased volatility frequently entails greater risk, it can also indicate a greater possibility for significant rewards.

When creating risk management methods and models, such as the before described VaR (Value at Risk) calculation, these volatility measurements can be helpful. It's crucial to remember that these historical volatility measures are based on past price changes and may not always be a reliable indicator of future volatility.

## 2.2 Variance-Covariance Approach (Delta-Normal)

In risk management and portfolio management contexts, the Variance-Covariance technique, commonly referred to as the Delta-Normal method, is a frequently used method for estimating Value at Risk (VaR). VaR is a statistical method for calculating the amount of financial risk present in a company or investment portfolio over a given period of time [5].

Here are the main aspects of the Variance-Covariance approach:

- **Delta-Normal Assumptions:** The Variance-Covariance technique makes the assumptions that the returns on the portfolio are normally distributed and that there is a linear relationship between the various assets in the portfolio. This approach just takes into account the mean and variance, the first two moments of the return distribution. As a result, it is unable to capture skewness, kurtosis, and tail risk [5].
- **Calculation of VaR using Variance-Covariance Approach:** Using a normal distribution to compute the maximum loss at a given degree of confidence, the Variance-Covariance technique first determines the standard deviation (volatility) of the returns on the portfolio.

When the risk factors are jointly normally distributed and the positions can be represented by their delta exposures, the measurement of VAR is considerably simplified. we have  $N$  risk factors. Define  $x_{i,t}$  as the exposures aggregated across all instruments for each risk factor  $i$  and measured in currency units. Equivalently, we could divide these by the current portfolio value  $W$  to obtain the portfolio weights  $w_{i,t}$ . The portfolio rate of return is

$$R_{p,t+1} = \sum_{i=1}^N w_{i,t} R_{i,t+1}$$

Where the weights  $w_{i,t}$  are indexed by time to indicate that this is the current portfolio [4]. This method allows easy aggregation of risks for large portfolios because of the invariance property of normal variables: Portfolios of jointly normal variables are themselves normally distributed [5]. The portfolio normality assumption is also justified by the *central limit theorem*, Which states that the average of independent random variables converges to a normal distribution. For portfolios diversified across a number of risk factors that have modest correlations, these conditions could be approximately met. Using the matrix

notations, the portfolio variance is given by

$$\sigma^2(R_{p,t+1}) = w_t' \sum_{t+1} w_t$$

Where  $\sum_{t+1}$  is the forecast of the covariance matrix over the VAR horizon. The portfolio VAR then is

$$VAR = \alpha \sqrt{x_t' \sum_{t+1} x_t} = \alpha W \sqrt{w_t' \sum_{t+1} w_t}$$

Where  $\alpha$  is the deviate corresponding to the confidence level for the normal distribution or for another parametric distribution [4].

The VaR calculation formula is as follows:

**Portfolio Value \* Portfolio Standard Deviation \* Z-Score of Confidence**

Where the relationship between a value and a set of value's mean is described by the statistical measurement known as the Z-Score. It is utilised to determine the confidence level. The covariance matrix of the portfolio's assets can be used to calculate the standard deviation for the portfolio.

**Strenghts:**

- **Simplicity:** Mathematically simple and simple to comprehend, the variance-covariance method. Basic statistics and linear algebra are used in the computations, which are well within the skills of the majority of finance experts [6].
- **Efficiency:** Compared to other methods, like the Historical Simulation or Monte Carlo Simulation methods, the Delta-Normal methodology simply needs a modest quantity of data. Even for big portfolios, the computation may be completed fast, making it a good option for risk management [5].
- **Considers Correlation Between Assets:** This strategy takes into account the correlations among various assets in a portfolio, which is crucial for appropriately determining portfolio risk [6]. Compared to certain simpler risk metrics that take each asset separately, this has a benefit.
- **Compatible with Large Portfolios:** The approach is scalable to big portfolios. As more assets are added, the computational complexity climbs quadratically rather than exponentially since it uses a covariance matrix [5].

- **Broadly Accepted in the Industry:** The finance industry is well known for and employs the variance-covariance approach. The results from this method are likely to be understood and accepted by other professionals in the area because it is widely utilised by financial institutions and accepted by the majority of financial authorities [6].

### **Limitations:**

- **Normal Distribution Assumption:** The asset return distribution is assumed to be normally distributed by the variance-covariance technique. Financial returns can have more extreme values than would be predicted by a normal distribution since they frequently exhibit skewness and kurtosis (fat tails). The risk of severe events, often known as tail risk, may thus be overestimated using the technique [5].
- **Constant Correlation Assumption:** The approach makes the assumption that asset correlations don't change over time. In truth, correlations can alter and frequently get stronger when the market is under stress. This implies that in erratic markets, the technique may understate risk [6].
- **Neglects Event Risk:** Event risk, which is the risk of a major event that has an impact on the financial markets, such as a political crisis, a change in fiscal policy, or a natural disaster, is not effectively taken into account by the technique. These occurrences might cause prices to significantly increase or decrease, defying the regular distribution [5].
- **No Consideration for Volatility Clustering:** The concept of volatility clustering describes the tendency of significant changes in financial time series to be followed by other large changes, and minor changes to be followed by other tiny ones. This characteristic of financial returns is not taken into consideration by the Delta-Normal technique [6].
- **Not Suitable for Non-Linear Instruments:** Because it fails to account for the non-linear risk characteristics of complex financial products, the variance-covariance strategy does not perform well for portfolios incorporating these instruments [2].

In this study, we found that the returns of the four stocks are not normally distributed over the period which can be proved by the results of Skewness, Kurtosis and Jarque-Bera test.

Table 3: Skewness, Kurtosis, and Jarque-Bera Test Results for Stock Returns

Stock	Skewness	Kurtosis	Jarque-Bera test p-value
Apple	-0.352377	6.040803	0
Amazon	0.004450022	6.477094	0
Netflix	-0.4343416	30.88717	0
Google	0.2463436	7.659008	0

Interpretation of table 3:

**Skewness** is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean [9, 11].

- Indicating that the returns on Apple and Netflix stocks are left-skewed, with the tail on the negative side of the distribution being thicker and thus reflecting a higher possibility of negative returns, respectively, these stocks' negative skewness values are -0.352377 and -0.4343416.
- Amazon and Google, on the other hand, have positive skewness (0.004450022 and 0.2463436, respectively), which indicates that the returns on these stocks are skewed to the right, meaning there might be a higher chance of positive returns.

**Kurtosis** is a statistical measure that describes the distribution's tails and sharpness. High kurtosis in a data set is an indicator that data has heavy tails or outliers [9, 10].

- Apple, Amazon, Google, and particularly Netflix have high kurtosis values (6.040803, 6.477094, 7.659008, and 30.88717, respectively), which indicates that the returns distributions have skewed distributions. High kurtosis may be a sign that extreme outcomes are more likely to occur.

The **Jarque-Bera** test is a goodness-of-fit test of whether sample data has the skewness and kurtosis matching a normal distribution [10, 11].

- The p-values for the Jarque-Bera tests for all four stocks are 0, suggesting we reject the null hypothesis that the data are normally distributed [11]

**Considering the Delta-Normal Approach:** The results of the Jarque-Bera test, which measures skewness and kurtosis, suggest that the Delta-Normal approach's assumption that returns are normally distributed might not apply to these companies [4, 10]. The technique might underestimate the probability of substantial losses, according to the negative skewness for Apple and Netflix [14]. The high kurtosis across all equities raises the possibility that there may be more severe outcomes than the technique is able to take into consideration [9, 10]. As a result, it is important to interpret the derived VaR estimates using the Delta-Normal technique carefully because they may significantly underestimate the risk [4, 5].

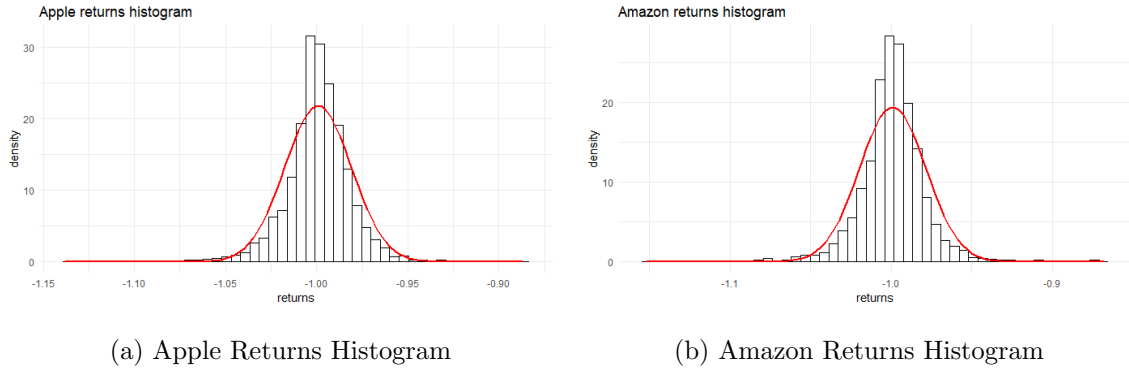
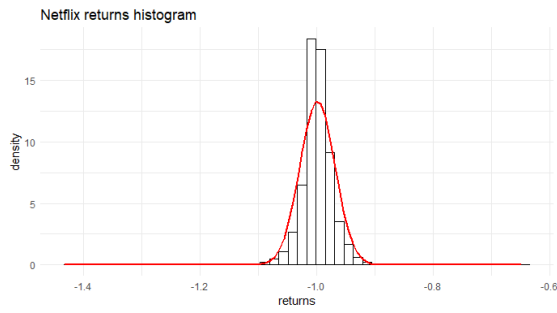
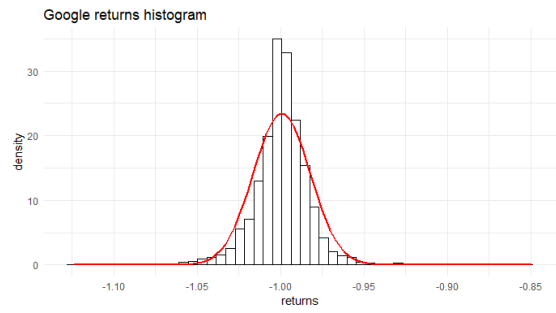


Figure 1: Histograms of Apple and Amazon Returns



(a) Netflix Returns Histogram



(b) Google Returns Histogram

Figure 2: Histograms of Netflix and Google Returns



## 2.3 Monte Carlo Simulation

The statistical method of Monte Carlo Simulation enables users to take risk into consideration when performing quantitative analysis and making decisions. It gives a variety of potential outcomes and their likelihoods of happening for any course of action. This method is employed in a variety of fields, including engineering and physics as well as finance and economics [1, 10].

The Monte Carlo Simulation approach is founded on the practise of obtaining numerical results through repeated random sampling. In essence, it's a method for representing and evaluating risk and uncertainty. The Monte Carlo Casino in Monaco, where the main games of chance, such roulette, are played, is where this approach gets its name [5].

Formula for Calculation of VaR [2, 5]:

$$VaR = Q(P - CL)$$

VaR represents the Value at Risk.

**Q** is the quantile function, which provides the inverse of the Cumulative Distribution Function. In other words, for a given confidence level, it tells you the value at which a certain percentage of the data lies below that value [5].

**P** represents the simulated returns, which are generated from the normal distribution with the mean and standard deviation of the actual returns [5].

**CL** is the desired Confidence Level [5].

Here is the equation showing the generation of simulated returns [5, 11]:

$$P = \mu + \sigma * Z$$

**P** are the simulated returns.

$\mu$  is the mean of the actual returns.

$\sigma$  is the standard deviation of the actual returns.

**Z** is a random sample from the standard normal distribution.

**Uses of Monte Carlo Simulation** [1, 3, 4, 6]:

- Risk Analysis: Monte Carlo simulations are used in finance and investing to model the likelihood of various outcomes in a process that is difficult to forecast because

of the interference of random factors.

- **Project Planning:** The technique is employed in project planning to recognise risks and comprehend the effects of risk events.
- **Engineering:** Monte Carlo simulations are used in engineering to assist solve issues with dependability, resistance, and strength tolerance.
- **Supply Chain Management:** The approach aids in managing unpredictable activities and events, such as the demand for goods or services, in supply chain management.
- **Physics and Chemistry:** Monte Carlo simulations are used in physics and chemistry to model and examine complicated events like the development of molecular structures or the behaviour of particles.

### **Process of Monte Carlo Simulation [4]**

The Monte Carlo simulation involves a series of steps:

- **Define a Model:** Choose the model or issue that you want to tackle. This could involve analysing historical weather data, forecasting stock values, or any number of other things.
- **Generate Random Inputs:** These inputs ought to be created in a way that they cover the full range of values that each input might have. The distribution of these inputs has a big impact on how the simulation's outputs are distributed.
- **Perform a Deterministic Calculation:** Use the model to calculate an outcome based on the random inputs.
- **Replicate the Process:** Repeat the process many times (thousands or even millions) to create a distribution of possible outcome values.
- **Analyze the Results:** Analyze the outcomes to help you understand the likelihood of different ranges of results and to help you identify your risks.

## **Advantages and Limitations**

### **Advantages** [9, 10]

- They aid in the modelling of complex circumstances for which conventional mathematical techniques are inappropriate.
- They list a variety of potential scenarios along with their likelihood of happening.
- They are adaptable; you can change the parameters to look at other circumstances.

### **Limitations** [7, 8]:

- They may be computationally demanding and call for advanced computing resources.
- The results are only as accurate and valuable as the model upon which they are based. Validation against empirical data must therefore be done carefully.
- The approach ignores potential correlations between variables in favour of a "random" process.

In conclusion, Monte Carlo simulations provide a powerful and flexible method for studying and understanding the behavior of complex systems under uncertain conditions [1, ?].

## 2.4 Extreme Value Theory approach

Stock market investments come with inherent risks that can have a big influence on an investor's portfolio, including risks from extraordinary occurrences in addition to the usual market volatility. These occurrences, which frequently result from unforeseen situations or unusual occurrences, can cause sharp falls or increases in stock values. While effective in handling normal or common situations, traditional risk management tools frequently fall short in giving a thorough knowledge of these uncommon, extreme events [4]. Therefore, a distinct strategy is required, one that focuses exclusively on comprehending the behaviour of such catastrophic events. In order to do this, this study suggests using the Extreme Value Theory (EVT) [10].

The study of extreme events or outliers that occur in the tail area of a distribution is the focus of the statistical field known as extreme value theory (EVT) [5]. It is based on the idea that, although the majority of cases in a distribution revolve around the mean, it is the uncommon, extreme events that pose the greatest risk and necessitate further investigation [8].

In order to explain and describe the behaviour of extreme occurrences, which are often found in the tail portion of a probability distribution, extreme value theory (EVT) was developed [2]. The semi-parametric technique is one of various strategies for approaching EVT [11].

Due to its flexibility, the semi-parametric approach—which incorporates the advantages of both parametric and non-parametric statistical methods—is frequently utilised in EVT. This method preserves some of the effectiveness of parametric techniques while allowing for the modelling of complicated data patterns that might not fit well into standard parametric models [10].

In the context of EVT, the semi-parametric approach often involves the following steps [4, 5, 8, 9, 10]:

- **Threshold Selection:** Select an extreme event threshold that is high (or low, depending on the situation). Since there are no presumptions about the underlying distribution of the data, this stage is typically non-parametric.
- **Tail Fitting:** Fit a Generalised Pareto Distribution (GPD), a parametric model, above (or below) the threshold. The assumption that the distribution's tail follows

a GPD is necessary for this stage.

- **Risk Measure Calculation:** Calculate risk metrics of interest, such as Value-at-Risk (VaR) or Expected Shortfall (ES), using the fitted GPD. Since the parameters are inferred from the GPD, this stage is parametric.

The cumulative distribution function (CDF) and the probability density function (PDF) are typically used to characterize the GPD [10].

#### **Cumulative distribution function**

$$F(y; \xi, \beta) = 1 - \left(1 + \xi \frac{y}{\beta}\right)^{-\frac{1}{\xi}}$$

#### **Probability density function**

$$f(y; \xi, \beta) = \frac{1}{\beta} \left(1 + \xi \frac{y}{\beta}\right)^{-(\frac{1}{\xi}+1)}$$

**Quantile Function (QF)** or the inverse of the CDF of the GPD, which is used to calculate VaR:

$$Q(p; \xi, \beta) = \beta \left[(1 - p)^{-\xi} - 1\right]$$

In order to comprehend the extraordinary returns of four specific technology stocks—Apple, Amazon, Netflix, and Google—the study attempted to apply extreme value theory (EVT) [1]. The Generalised Pareto Distribution (GPD), a tenet of EVT that is essential to comprehending the distribution of extreme values, is used as the foundation of the methodology [2, 10]. The data was first sorted into descending order before applying EVT, and a high threshold was used. The main presumption was that data above this level would adhere to the GPD. Three threshold values were established: 99%, 97.5%, and 95%. The data was split into exceedances, which are values over the threshold, once the threshold had been established. Then, a GPD model was used to fit these exceedances. It's important to note that the data was tagged as 'NA' because there was not enough information to generate meaningful predictions if the exceedances were less than 10. For the threshold values of 99%, 97.5%, and 95%, the Generalised Pareto Distribution (GPD) was fitted on the exceedances of each return series. For each model, the shape parameter ( $\xi$ ) and scale parameter ( $\beta$ ), which are crucial components of the GPD model, were estimated. Notably, the Netflix stock data at the 99% level produced a 'NA', signifying there was not enough information to make meaningful forecasts.

The form and size characteristics were estimated differently by the EVT models for various stocks and criteria [10]. The unique risk and return characteristics of the individual

stocks can be linked to this variation [3]. A "heavier" tail, for example, is indicated by a larger shape parameter and denotes a higher likelihood of extreme events. Some examples where the models could not be calculated showed inadequate exceedances for those specific criteria. To ensure an acceptable sample size for accurate estimation, this restriction points to the need for a more adaptable threshold determination approach.

Particularly in the context of financial risk management, the Generalised Pareto Distribution is frequently used to simulate the tails of distribution. It can be used, for example, to calculate the risk of extreme occurrences like dramatic stock market crashes [4, 6]. The GPD's  $\xi$  (the shape parameter) and  $\beta$  (the scale parameter) parameters reveal crucial details about how the distribution's tail behaves [10].

Here is a general interpretation of the estimated parameters for the GPD [10]:

- The tail heaviness of the distribution is determined by the shape parameter, ( $\xi$ ). When  $\xi > 0$ , the distribution has a heavier (fatter) tail than an exponential distribution, indicating a greater likelihood of extreme values. When  $\xi = 0$ , the distribution's tail has a lighter (thinner) tail than the exponential distribution, which indicates a lesser likelihood of extreme values. The tail of the distribution resembles the exponential distribution when  $\xi = 0$ .
- The distribution is essentially scaled by the scale parameter,  $\beta$ . A larger value of the parameter denotes a greater danger of big values, whilst a lower value denotes a lesser risk.

Remember, VaR is only an estimate and is subject to model risk. In this case, EVT assumes that returns are independently and identically distributed, which might not be the case in reality [6].

Moreover, EVT uses extreme data points, the rarity of such data points can lead to unreliable parameter estimates and hence to unreliable VaR estimates [9]. It's important to note that the fitted GPD models did not converge for all data sets which may question the validity of the estimated VaR [14].

It is also important to remember that VaR does not provide information about the loss in case the event occurs. For that, metrics like Expected Shortfall could be used. The fact that Amazon and Apple don't have estimated VaR at higher risk levels might suggest that these models did not converge or the data did not fit the GPD model well at these levels [13].

## 3 Real Data Analysis and Comparison of Different Methods

### 3.1 Four Time Series Datasets of Financial Returns

The first step of the analysis is to acquire historical daily stock prices for the chosen companies [17]. For this study, I have selected Apple Inc., Amazon.com Inc., Netflix Inc., and Google LLC (traded as Alphabet Inc. Class A shares).

The analysis begins on January 1, 2013, and ends on January 30, 2023. The date range was set using the following commands in the R programming language

Next, for each company, the closing prices are isolated and converted into a time series object. Subsequently, the daily log returns for each company's stock are calculated. The log returns are often used in financial analyses due to their convenient statistical properties, such as symmetry and additive property, which simplifies mathematical modeling.

For a time series of prices  $p_t$ , the log return  $r_t$  at time  $t$  is given by the equation:  $r_t = \log\left(\frac{p_t}{p_{t-1}}\right)$ . This process generates a new time series of the daily log returns for each stock. This procedure is mirrored for each of the four stocks (Apple, Amazon, Netflix, Google), creating four distinct time series objects that represent the daily log returns for each stock over the designated time frame. The subsequent sections will delve into the statistical analysis and modeling of these log return series, which will provide insights into the volatility dynamics and downside risk of these stocks.

### 3.2 Estimation of VaRs Using Different Methods

So to start with data gathering there are 2535 observations of each stocks having daily returns from the period 01-January-2013 to 30-January-2023 that is 10 years. This data is used to calculate possible losses which will be used to evaluate the risk and quantify the possible downside of an investment. This data is also used to capture the volatility and correlations by looking at past price changes and returns. As the period itself tells that there are many natural and uncontrollable events were occurred like COVID-19 and Russia-Ukraine war which shows that there will be massive volatility during these periods and these events can be considered as unpredictable. By going through this data we have calculated Value at risk by Historical simulation, Variance-Covariance Approach(Delta Normal Approach), Monte Carlo Simulation, Extreme Value Theory Approach (Semi-Parametric Approach) and the results are as follows:

#### 3.2.1 Historical Simulation

Table 4: Value at Risk (VaR) for Different Stocks at Various Confidence Levels

	<b>Apple</b>	<b>Amazon</b>	<b>Netflix</b>	<b>Google</b>
VaR(95%)	-0.027508	-0.031090	-0.039787	-0.025700
VaR(97.5%)	-0.036510	-0.041789	-0.052302	-0.035176
VaR(99%)	-0.051238	-0.058685	-0.071078	-0.047646
Volatility	0.2910411	0.3260103	0.475937	0.2702451

In this study, we utilize a historical simulation approach to compute the Value-at-Risk (VaR) for individual stocks namely Apple, Amazon, Netflix, and Google. The Value-at-Risk (VaR) provides a measure of the maximum expected loss over a given time period at a certain confidence level.

the Value-at-Risk (VaR) at the 5%, 2.5%, and 1% confidence levels are as follows for the various companies:

- Apple: The VaR for Apple is -0.0275 at the 5% level of confidence, meaning there is a 5% possibility of suffering a loss larger than -0.0275. The VaR for Apple is -0.0365 at the 2.5% level of confidence, indicating that there is a 2.5% probability of suffering a loss higher than -0.0365. The VaR for Apple is -0.0512 at the 1% level



of confidence, indicating that there is a 1% likelihood of suffering a loss higher than -0.0512.

- Amazon: The VaR for Amazon is -0.0311 at the 5% level of confidence, meaning that there is a 5% probability of suffering a loss higher than -0.0311. The VaR for Amazon is -0.0418 at the 2.5% confidence level, indicating that there is a 2.5% probability of suffering a loss higher than -0.0418. The VaR for Amazon is -0.0587 at the 1% level of confidence, which indicates that there is a 1% probability of suffering a loss higher than -0.0587.
- Netflix: The VaR for Netflix is -0.0398 at the 5% level of confidence, meaning there is a 5% probability of suffering a loss higher than -0.0398. The VaR for Netflix is -0.0523 at the 2.5% level of confidence, indicating that there is a 2.5% probability of suffering a loss higher than -0.0523. The VaR for Netflix is -0.0711, which indicates that there is a 1% probability of suffering a loss larger than -0.0711.
- Google: The VaR for Google is -0.0257 at the 5% level of confidence, meaning there is a 5% possibility of suffering a loss higher than -0.0257. The VaR for Google is -0.0352, indicating that there is a 2.5% probability of suffering a loss larger than -0.0352 at the 2.5% confidence level. The VaR for Google is -0.0476, which indicates that there is a 1% possibility of suffering a loss larger than -0.0476.

### 3.2.2 Variance-Covariance Approach (Delta Normal Approach)

Table 5: Value at Risk (VaR) for Different Stocks at Various Confidence Levels

Confidence Level	Apple VaR	Amazon VaR	Netflix VaR	Google VaR
95%	-0.03015652	-0.03377989	-0.04931471	-0.02800173
99%	-0.04265095	-0.04777555	-0.06974674	-0.03960338
97.5%	-0.03593371	-0.04025122	-0.0587621	-0.03336611

Interpretation of Table 5 :

The VaR values calculated for each of the stocks provide an estimate of the maximum loss we could expect, under normal market conditions, with a certain degree of confidence. Specifically:

At 95% Confidence Level:

- Apple VaR: -0.03015652: Under typical market conditions, there is a 5% possibility (or 1 in 20 risk) that the return on Apple will fall by 3.015652% or more over the selected time period.
- Amazon VaR: -0.03377989: Under typical market conditions, there is a 5% risk (or 1 in 20 probability) that the return on Amazon will fall by 3.377989% or more over the selected time period.
- Netflix VaR: -0.04931471 Under typical market conditions, there is a 5% chance (or 1 in 20 chance) that the return on Netflix will drop by 4.931471% or more over the selected time frame.
- Google VaR : -0.02800173: The return on Google has a 5% chance (or 1 in 20 chance) of declining by 2.800173% or more over the specified time period under normal market conditions.

At a 97.5% confidence level:

- Apple VaR: -0.03593371: Under typical market conditions, there is a 2.5% risk (or 1 in 40 probability) that the return on Apple will fall by 3.593371% or more over the selected time period.

- Amazon VaR: -0.04025122: Under typical market conditions, there is a 2.5% risk (or 1 in 40 probability) that the return on Amazon will fall by 4.025122% or more over the selected time period.
- Netflix VaR: -0.0587621: Under typical market conditions, there is a 2.5% risk (or 1 in 40 probability) that the return on Netflix will drop by 5.87621% or more over the selected time frame.
- Google VaR: -0.03336611: The likelihood that the return on Google will drop by 3.336611% or more over the next year is 2.5%, or one in forty.

Finally, at a 99% confidence level:

- Apple VaR: -0.04265095: Under typical market conditions, there is a 1% risk (or 1 in 100 possibility) that the return on Apple will fall by 4.265095% or more over the selected time period.
- Amazon VaR: -0.04777555: Under typical market conditions, there is a 1% risk (or 1 in 100 probability) that the return on Amazon will fall by 4.777555% or more over the selected time period.
- Netflix VaR: -0.06974674: Under typical market conditions, there is a 1% risk (or 1 in 100 probability) that the return on Netflix will fall by 6.974674% or more over the selected time period.
- Google VaR: -0.03960338 The return on Google has a 1% probability (or 1 in 100 chance) of declining by 3.960338% or more over the specified time period under normal market conditions.

### 3.2.3 Monte Carlo Simulation

Table 6: Value at Risk (VaR) for Different Stocks at Various Confidence Levels

Confidence Level	Apple	Amazon	Netflix	Google
95%	-0.03097732	-0.03291856	-0.0482618	-0.02784834
99%	-0.0403037	-0.04762216	-0.0700732	-0.03660605
97.5%	-0.0361984	-0.03877161	-0.05820169	-0.03076967

Interpretation of Table 6:

At 95% confidence interval:

- VaR for Apple: -0.03097732 This indicates that there is a 5% possibility (or a 1 in 20 chance) that Apple's return will fall by 3.097732% or more within the given time frame.
- VaR for Amazon: -0.03291856 Similar to this, there is a 5% probability that within the same time frame, Amazon's return will fall by 3.291856% or more.
- VaR for Netflix: -0.0482618: This suggests that there is a 5% chance that Netflix's return will fall by 4.82618% or more.
- VaR for Google: -0.02784834 There is a 5% chance that Google's return will drop by 2.784834% or more.

At 97.5% confidence interval:

- VaR for Apple: 0.0361984: At this level of confidence, there is a 2.5% probability (1 in 40) that the return on Apple will fall by at least 3.61984% over the given time period.
- VaR for Amazon: There is a 2.5% probability that Amazon's return will fall by 3.877161% or more. Amazon VaR: -0.03877161.
- VaR for Netflix: -0.05820169: There is a 2.5% chance that the return will fall by 5.820169% or more.

- VaR for Google: -0.03076967 There is a 2.5% chance that Google's return will fall by 3.076967% or more.

At 99% confidence interval:

- VaR for Apple: -0.0403037: At this value, there is a 1% chance (1 in 100) that Apple's return will fall by at least 4.03037% within the given time frame.
- VaR for Amazon: -0.04762216: There is a 1% chance that Amazon's return will decline by 4.762216% or more.
- VaR for Netflix: -0.0700732, meaning there is a 1% possibility of witnessing a return decline of at least 7.00732%.
- VaR for Google: -0.03660605; this means that there is a 1% risk of a decline in return of at least 3.660605%.

Monte Carlo simulation is a statistical technique that allows you to account for risk in quantitative analysis and decision making. It uses randomness to solve problems that might be deterministic in principle.

The randomness is introduced by the code, which generates a set of normally distributed random numbers. Each time you run your code, this function generates a new set of random numbers. Therefore, the paths of simulated returns are different in each run of the code, which results in different VaR values each time you run the code.

### 3.2.4 Extreme Value Theory Approach (Semi-Parametric Approach)

Company	Threshold	VaR (95%)	VaR (97.5%)	VaR (99%)
Apple	95%	0.05005631	0.05449761	0.05851141
Netflix	95%	0.1827613	0.2284575	0.2909621
Google	95%	0.07486657	0.09163132	0.11348007
Netflix	97.5%	0.1860006	0.2260966	0.2775015
Google	97.5%	0.07157509	0.09916291	0.14539043
Netflix	99%	0.1937823	0.3269506	0.6319526
Google	99%	0.05895499	0.05926377	0.05937850

Table 7: Value-at-Risk (VaR) for different companies at different thresholds

Interpretation of Table 7: The Value-at-Risk (VaR) for each company gives an estimate of the potential loss that could occur with a certain level of confidence.

For Apple at 95% confidence, the maximum expected daily loss ranges from 5.01% to 5.85% across different quantiles.

For Netflix, the VaR varies greatly with confidence levels. At 95% confidence, the maximum expected daily loss ranges from 18.28% to 29.10%. The potential loss increases significantly at 99% confidence level, ranging from 19.38% to a significant 63.20%.

For Google, at 95% confidence, the maximum expected daily loss ranges from 7.49% to 11.35%. The potential loss slightly decreases at 99% confidence level, ranging from 5.90% to 5.94%.

It's clear that Netflix demonstrates a much higher risk profile compared to Apple and Google, especially at the 99% confidence level. The differences in VaR among the companies highlight their differing risk levels and how market conditions can affect different companies in unique ways.

Moreover, there are NA values present for some stock's EVT models, which may imply that EVT model did not fit well to the data. This might require a closer examination of the data or possibly a different approach to handle extreme values.

In the context of applying Extreme Value Theory (EVT), NA values could also occur due to the nature of the EVT itself. As EVT models extreme values, it might disregard or not consider "non-extreme" values, which could possibly result in NA values in your

results. Extreme events can cause changes in market dynamics, making the distribution of returns change over time (non-stationary). Moreover, stock returns can exhibit volatilities clustering, breaking the independence assumption. Therefore, the VaR estimates from the EVT approach should be used with caution, and it's good to cross-check with other risk measures.

In this model GPD fit is then used to calculate Value-at-Risk (VaR) at different quantiles. The fitting process is inherently statistical, which means it involves randomness. This is especially true if the function used to fit the GPD uses a method that involves randomness, such as a stochastic optimization algorithm. This, in turn, can result in slightly different VaR values each time.

### 3.3 Methodology of Back-testing

#### 3.3.1 Introduction of Backtesting

Value-at-Risk (VaR) backtesting compares the maximum tolerable losses predicted by a VaR model (based on a certain confidence level and time period) with the actual losses experienced over that period in order to assess the correctness of the VaR model. Backtesting's goal is to confirm that, given the chosen confidence level, the expected number of occasions in which real losses exceed the VaR estimate matches the number of such instances. It aids in evaluating the model's effectiveness and locating any modifications or enhancements that are required [1, 2, 5, 6].

"VaR is only as good as its backtest. When someone shows me a VaR number, I don't ask how it is computed, I ask to see the backtest [6]."

In the late 1980s and early 1990s, prominent financial firms established the idea of Value-at-Risk (VaR) [2]. The worst projected loss over a specified horizon under typical market conditions at a specified confidence level is measured by VaR [4, 5]. As a notion, backtesting has existed for as long as statistical modelling itself. However, at this time backtesting for VaR models began to gain popularity [1, 2].

In 1996, the Basel Committee on Banking Supervision (BCBS) endorsed the use of VaR models for market risk and mandated the use of backtests in the model validation procedure [6].

The most notable authors who created theoretical frameworks for backtesting VaR were Kupiec (1995) and Christoffersen (1998). The number of VaR exceedances (times the loss exceeds the VaR estimate) is compared to the expected number at a certain degree of confidence using Kupiec's Proportion of Failures (POF) test [4, 5, 16]. Christoffersen's test determines if exceedances are independently distributed throughout time (Conditional Coverage) as well as the amount of exceedances (Unconditional Coverage) [8].

However, researchers and practitioners realized that evaluating VaR models solely based on unconditional coverage was insufficient to capture the full spectrum of their performance. They recognized the need to consider additional dimensions such as the traffic light test, frequency test, and failure rate (TUFF backtesting) to provide a more comprehensive assessment of VaR models [9, 10]

The limits of VaR and the value of thorough backtesting became more clear as the financial sector encountered more extreme market shocks. This was particularly emphasised during



the 2008 Global Financial Crisis. Since then, regulatory standards have tightened, and backtesting has become increasingly important in model validation. The BCBS improved the requirements for backtesting VaR models when it changed the market risk framework in 2019 [7].

In this project we have used two backtesting methods one is Kupiec backtesting and other is TUFF backtesting

### 3.3.2 Kupiec test

Paul Kupiec proposed the Kupiec test, technically known as the Proportion of Failures (POF) test, as a way to evaluate the precision of Value-at-Risk (VaR) models [16]. VaR models are frequently used in financial risk management to calculate the portfolio's maximum possible loss over a particular time horizon and with a given level of confidence [4, 5].

The primary purpose of the Kupiec test is to evaluate the validity of a VaR model by comparing the actual number of exceptions (times when actual losses exceed the VaR estimate) with the expected number of exceptions given the selected confidence level [4, 5, 16].

The null hypothesis (H0) of the Kupiec test assumes that the VaR model is accurate, i.e., the observed proportion of exceptions equals the expected proportion. The alternative hypothesis (H1) states that the model is inaccurate and these proportions differ [5, 16].

Calculations and Statistical Tests

The test statistic for the Kupiec test is calculated as follows:

$$POF = 2 [\ln((1 - p)^{N-n} \cdot p^n)] - 2 [\ln((1 - x)^{N-n} \cdot x^n)]$$

Where:

p is the expected proportion of exceptions (1 minus the confidence level).

N is the total number of observations.

n is the actual number of exceptions.

x is the observed proportion of exceptions (n/N).

Under the null hypothesis, the test statistic (POF) has a chi-squared distribution with one degree of freedom. The null hypothesis is rejected if the computed POF value exceeds the chi-squared critical value for the specified significance level, demonstrating that the VaR model is ineffective at forecasting the proportion of exceptions.

### **Strengths and Limitations [4, 5, 16]**

The Kupiec test's simplicity and intuitiveness are its main advantages. It gives an easy way to check if a VaR model's predictions match actual results in a historical context by concentrating on the fraction of failures.

The exam does, however, have some restrictions. First, the amount of the exceedances is not taken into account. The model may underestimate or overestimate the magnitude of losses when exceptions occur, even if the proportion of exceptions is as anticipated. Second, it implies that exceptions are independent of one another throughout time, so that a breach in one period has no bearing on the likelihood of a breach in the following one. This might not be true in the financial markets, where peaks in volatility frequently occur in groups.

Therefore, while the Kupiec test provides valuable insights, it should ideally be used in conjunction with other tests.

In practise, the Kupiec test is frequently applied. It is a component of the toolkit used by financial firms, regulatory agencies, and researchers to verify the effectiveness of VaR models. It supports model selection and alterations, the creation of risk management plans, and regulatory compliance. Given its natural interpretation, its results also make it easier to communicate the performance of the risk model to non-technical stakeholders. The robustness of financial risk management systems is ensured in large part by the Kupiec test, which also contributes to the stability and resilience of financial institutions and markets.

### 3.3.3 Traffic Light, Unconditional Coverage, Frequency, Failure Rate test

The TUFF backtesting process involves these four tests:

**Traffic Light Test (TLT):** This test measures the proportion of times the actual return is less than the predicted value at risk (VaR). A lower value indicates better performance [4, 5].

Advantages:

Easy to interpret: The traffic light approach provides a clear, intuitive understanding of model performance. The green, yellow, and red lights quickly signal whether the model is performing as expected, needs caution, or is failing, respectively [5].

Communication: It's an effective way to communicate complex results to stakeholders who may not have a detailed understanding of statistical tests [4, ?].

Disadvantages:

Over-simplification: The main criticism is that it can oversimplify complex results, potentially leading to the wrong conclusions or masking underlying issues [5].

Arbitrary thresholds: The thresholds for each category (green, yellow, and red) can be somewhat arbitrary and may need to be adjusted depending on the specific context [4, 5].

**Unconditional Coverage Test (UCT):** This test measures the accuracy of the model's VaR predictions. It is calculated as the proportion of actual returns that are less than the predicted VaR divided by the sum of actual returns that are less than the predicted VaR. This value should ideally be close to the confidence level used (for example, 0.05 for 95% confidence level) [5, 10].

Advantages:

Simplicity: The test is simple to implement and understand, making it a practical tool for backtesting [4, 10].

Evaluates overall performance: It provides a general view of the model's performance by evaluating whether the model's predicted number of exceedances matches the actual number [4, 5].

Disadvantages:

Ignores clustering: The test doesn't take into account the possibility that exceedances might be clustered together. This means it might overlook problems with models that underestimate the risk during turbulent periods [4, 5, 10].

**Frequency Test (FT):** This test measures the proportion of 'hits', instances when the actual return is less than the predicted VaR, in the total dataset. The closer this value is to the specified quantile (e.g., 0.05 for a 95% VaR), the better [5, 10].

Advantages:

Simple to calculate: The frequency of failures is easy to calculate and can be a useful indicator of model performance [5, 10].

Directly related to risk: A high frequency of failures suggests that the model is underestimating risk [5, 10].

Disadvantages:

No context: The frequency of failures doesn't take into account the severity of each failure or the specific circumstances under which it occurred [5, 10].

**Failure Rate (FR):** This test calculates the proportion of 'hits' in the total number of 'hits' expected. This value should ideally be close to 1 [5, 10]. Advantages:

Clear indicator: The failure rate provides a clear, single-number indicator of how often the model's predictions are incorrect [4, 10].

Considers total observations: By comparing the number of failures to the total number of observations, the failure rate gives an indication of performance relative to the overall dataset [5, 10].

Disadvantages:

No severity measure: Similar to the frequency of failures, the failure rate doesn't provide information on the severity of each failure [5, 10].

Ignores timing: It doesn't take into account the timing or clustering of failures, which might be relevant in some contexts [4, 5, 10].

In the TUFF backtesting table, the different colors represent the interpretation of the test results:

**Green:** Indicates a successful backtest. This means that the EVT model has passed the test, suggesting that it accurately captures the downside risk based on the given confidence level. In this case, a p-value of 1 indicates a successful backtest [4, 5, 10].

**Red:** Indicates an unsuccessful backtest. This means that the EVT model has not passed the test, suggesting potential issues or concerns regarding its accuracy in capturing the downside risk. In this case, "N/A" signifies that the backtest result is not available or

not applicable due to missing data or other factors [4, 5, 10]. In this paper NA is due to limitations because there is no missing data or incorrect data as all the Codes required to take care of these issues were taken into consideration.

The color coding helps to visually differentiate between the EVT models that have passed the test (green) and those that have not (red) [4, 5, 10].

### 3.4 Comparison of Different Methods by Back-testing

#### 3.4.1 Kupiec Backtesting

This backtesting method has been applied on four different methods for calculation of VaR namely:

- Historical simulation
- Var Covariance approach (Delta normal approach)
- Monte Carlo simulation
- Extreme value theory approach (Semi-Parametric approach)

Now will explain the impact of backtesting by kupiec test on each method used for calculation of VaR.

#### **Interpretation of Kupiec backtesting results on Historical Simulation method.**

For four stocks—Apple, Amazon, Netflix, and Google—the Kupiec test was run at three different degrees of confidence (95%, 97.5%, and 99%). A historical simulation approach was used to determine the Value-at-Risk (VaR), which was then compared to the actual returns. According to the test results, the VaR model was disproved for each stock at each of the three confidence levels. This suggests that there were much more exceedances (cases where the actual losses exceeded the VaR estimate) than would have been predicted based on the model's underlying assumptions.

In other words, the VaR model does not accurately represent the risk for any of the four organisations, suggesting that the model may be underestimating the risk. This can be a sign to evaluate and possibly change the existing risk management strategies.

Table 8: Kupiec Test Results for VaR model(Historical Simulation at different confidence levels for selected stocks.

Stock	Confidence Level	Kupiec Test Result
Apple	95%	Rejected
Apple	97.5%	Rejected
Apple	99%	Rejected
Amazon	95%	Rejected
Amazon	97.5%	Rejected
Amazon	99%	Rejected
Netflix	95%	Rejected
Netflix	97.5%	Rejected
Netflix	99%	Rejected
Google	95%	Rejected
Google	97.5%	Rejected
Google	99%	Rejected

Alternatively, Backtesting has been done on last 500 observations for each method. firstly will discuss for Historical Simulation. Here is the table for the result of Kupiec backtesting considering last 500 observations.



Table 9: Kupiec Test Results for VaR model(Historical Simulation for 500 observations.

<b>Company</b>	<b>Confidence Level</b>	<b>Kupiec Test Result</b>
Apple	95%	VaR model rejected
Apple	97.5%	VaR model rejected
Apple	99%	VaR model rejected
Amazon	95%	VaR model rejected
Amazon	97.5%	VaR model rejected
Amazon	99%	VaR model rejected
Netflix	95%	VaR model rejected
Netflix	97.5%	VaR model rejected
Netflix	99%	VaR model rejected
Google	95%	VaR model rejected
Google	97.5%	VaR model rejected
Google	99%	VaR model rejected

### **Interpretation of Kupiec backtesting results on Variance-Covariance approach (Delta-Normal approach)**

The Value-at-Risk (VaR) model calculates the greatest predicted loss for a specified confidence level using the variance-covariance method. For risk management and financial regulatory activities, it is especially helpful. Its accuracy must be confirmed, nevertheless. The Kupiec test, a prominent technique for backtesting the VaR model, evaluates the precision of VaR estimates by contrasting the expected and actual number of exceedances. The VaR model is rejected for each company at each confidence level, according to the findings of the Kupiec tests ran on the four stocks (Apple, Amazon, Netflix, and Google) at three distinct confidence levels (95%, 97.5%, and 99%). This indicates that for these specific stocks at these confidence levels, the number of actual exceedances significantly differs from the projected ones, indicating that the VaR model's predictions aren't as dependable.

At three separate degrees of confidence (95%, 97.5%, and 99%), the Kupiec test results show a rejection of the VaR model using the variance-covariance technique for the four big tech stocks: Apple, Amazon, Netflix, and Google. This rejection suggests that the maximum losses projected by the model were significantly more than the actual losses. This result indicates that the model was unable to predict the Value-at-Risk for these stocks with sufficient accuracy at the assessed confidence levels. Because of this, relying on this model to manage risk in certain situations runs the danger of underestimating risk, which could have serious financial repercussions.

In the conclusion even if the backtesting is successful and has accepted the model for each stock still it cannot be considered to be rejected because it does not meet the assumption of Normal distribution of data. As per the above analysis done through calculation of Skewness, Kurtosis, Jarque-bera test which can be seen in Table 3, it can be justified that data is not normally distributed and hence it can be said that Kupiec backtesting has rejected this model or method for calculation of VaR for all the stocks.

Table 10: Kupiec Test Results for VaR model (Variance-Covariance Approach) at different confidence levels for selected stocks.

Stock	Confidence Level	Kupiec Test Result (Var-Cov Approach)
Apple	95%	Rejected
Apple	97.5%	Rejected
Apple	99%	Rejected
Amazon	95%	Rejected
Amazon	97.5%	Rejected
Amazon	99%	Rejected
Netflix	95%	Rejected
Netflix	97.5%	Rejected
Netflix	99%	Rejected
Google	95%	Rejected
Google	97.5%	Rejected
Google	99%	Rejected

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Variance Covariance approach. Here is the table for the result of Kupiec back-testing considering last 500 observations.

Table 11: Kupiec Test Results for VaR model (Variance-Covariance Approach) for 500 observations.

Stock	Confidence Level	Kupiec Test Result (Var-Cov Approach)
Apple	95%	Rejected
Apple	97.5%	Rejected
Apple	99%	Rejected
Amazon	95%	Rejected
Amazon	97.5%	Rejected
Amazon	99%	Rejected
Netflix	95%	Rejected
Netflix	97.5%	Rejected
Netflix	99%	Rejected
Google	95%	Rejected
Google	97.5%	Rejected
Google	99%	Rejected

### Interpretation of Kupiec backtesting results on Monte Carlo Simulation.

Table 12: Kupiec Test Results for VaR model (Monte Carlo Simulation) at different confidence levels for selected stocks.

Stock	Confidence Level	Kupiec Test Result (Monte Carlo Simulation)
Apple	95%	Rejected
Apple	97.5%	Rejected
Apple	99%	Rejected
Amazon	95%	Rejected
Amazon	97.5%	Rejected
Amazon	99%	Rejected
Netflix	95%	Rejected
Netflix	97.5%	Rejected
Netflix	99%	Rejected
Google	95%	Rejected
Google	97.5%	Rejected
Google	99%	Rejected

The Kupiec test was then used to backtest the Monte Carlo simulation-based VaR at three degrees of confidence levels (95%, 97.5%, and 99%). A statistical tool is provided by the Kupiec test to determine if the frequency of actual returns above the VaR is consistent with what is anticipated at the specified confidence level.

The VaR model was, however, rejected by the Kupiec test results for all four equities across all tested confidence levels. This shows that the VaR model based on Monte Carlo simulation grossly underestimated the quantity of exceedances. In other words, more frequently than the VaR model anticipated, actual losses exceeded VaR forecasts.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Monte Carlo approach. Here is the table for the result of Kupiec backtesting considering last 500 observations.

Table 13: Kupiec Test Results for VaR model (Monte Carlo Simulation) for 500 observations

Stock	Confidence Level	Kupiec Test Result (Monte Carlo Simulation)
Apple	95%	Rejected
Apple	97.5%	Rejected
Apple	99%	Rejected
Amazon	95%	Rejected
Amazon	97.5%	Rejected
Amazon	99%	Rejected
Netflix	95%	Rejected
Netflix	97.5%	Rejected
Netflix	99%	Rejected
Google	95%	Rejected
Google	97.5%	Rejected
Google	99%	Rejected

### Interpretation of Kupiec backtesting results on Extreme value theory approach (Semi-Parametric approach)

Table 14: Kupiec Test Results for VaR model (Extreme Value Theory approach) at different confidence levels for selected stocks

Model	Result	Kupiec test
AppleReturns.EVT_95	1	Accepted
AmazonReturns.EVT_95	NA	Rejected
NetflixReturns.EVT_95	1	Accepted
GoogleReturns.EVT_95	1	Accepted
AppleReturns.EVT_975	NA	Rejected
AmazonReturns.EVT_975	NA	Rejected
NetflixReturns.EVT_975	1	Accepted
GoogleReturns.EVT_975	1	Accepted
AppleReturns.EVT_99	NA	Rejected
AmazonReturns.EVT_99	NA	Rejected
NetflixReturns.EVT_99	1	Accepted
GoogleReturns.EVT_99	1	Accepted

With a result of 1, the EVT models for Apple, Netflix, and Google passed the backtest when compared to the results for the 95% criterion. This means that 95% of the time, the VaR models for these equities properly forecast the maximum loss, proving their dependability and efficiency for risk management.

The EVT model for Amazon, however, failed the backtest at the 95% level, yielding a NA result. This shows that in 95% of the cases, the VaR model for Amazon was unable to correctly anticipate the maximum loss, indicating the need for model change or the use of other risk measurement techniques.

All models except Netflix and Google returned NA for the 97.5% and 99% thresholds, indicating that these models failed the backtest. This illustrates the growing challenge of correctly forecasting larger quantiles, that is, more dramatic losses.

These findings highlight how crucial it is to continuously validate and backtest VaR models. They demonstrate how some models might perform well at some thresholds but not

others, despite the fact that they might. As a result, it is critical for risk managers to select models and thresholds that are appropriate for their particular risk appetite and investment philosophy.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Extreme Value Theory approach. Here is the table for the result of Kupiec backtesting considering last 500 observations.

Table 15: Kupiec Test Results for VaR model (Extreme Value Theory approach) for 500 observations

<b>Model</b>	<b>Backtest Result</b>	<b>Kupiec test</b>
AppleReturns.EVT_95	1	Accepted
AmazonReturns.EVT_95	NA	Rejected
NetflixReturns.EVT_95	1	Accepted
GoogleReturns.EVT_95	1	Accepted
AppleReturns.EVT_975	NA	Rejected
AmazonReturns.EVT_975	NA	Accepted
NetflixReturns.EVT_975	1	Accepted
GoogleReturns.EVT_975	1	Rejected
AppleReturns.EVT_99	NA	Rejected
AmazonReturns.EVT_99	NA	Rejected
NetflixReturns.EVT_99	1	Accepted
GoogleReturns.EVT_99	1	Accepted

With the result of 1, the EVT models for Apple, Netflix, Google has passed the backtest when compared to the results for the 95% criterion. At 97.5% only Google and Netflix has passed the backtest. At 99% only Netflix and Google has passed the backtest.



### 3.4.2 Traffic Light, Unconditional Coverage, Frequency, Failure Rate test

#### Interpretation of TUFF backtesting results on Historical Simulation.

Table 16: TUFF Backtest Results for Historical Simulation for Selected Stocks

Stock	TLT	UCT	FT	FR
Apple	0.9731755	1	0.9731755	1
Amazon	0.9715976	1	0.9715976	1
Netflix	0.965286	1	0.965286	1
Google	0.9727811	1	0.9727811	1

High Traffic Light Test (TLT) results for all the stock portfolios (Apple, Amazon, Netflix, and Google) show a considerable percentage of hits (actual returns below VaR forecasts). For all portfolios, the Unconditional Coverage Test (UCT) results are 1, which denotes perfect unconditional coverage. This indicates that the models successfully caught all of the observations when the actual returns were lower than the VaR estimates.

All of the portfolios' Frequency Test (FT) values are rather high, which suggests that the estimated VaR models are successful in capturing downside risk in a sizeable portion of the time.

All portfolios have Failure Rate (FR) values of 1, which denotes a 100% failure rate. This indicates that hits were produced as a result of all VaR estimation breaches.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Historical Simulation. Here is the table for the result of TUFF backtesting considering last 500 observations.

Table 17: TUFF Backtest Results for Historical Simulation for 500 observations

Stock	TLT	UCT	FT	FR
Apple	0.968	1	0.968	1
Amazon	0.966	1	0.966	1
Netflix	0.948	1	0.948	1
Google	0.954	1	0.954	1

The percentage of actual returns that exceed the VaR estimates is measured by the Traffic Light Test (TLT). Higher TLT values suggest more hits, which is beneficial for the performance of the VaR model. All of the equities have TLT values above 0.94 in this instance, indicating that the VaR model does a good job of capturing extreme downside swings.

Using VaR estimates as a base, the Unconditional Coverage Test (UCT) compares the proportion of hits to the proportion of expected hits. If the VaR estimations and the actual hits match, the value is 1. A excellent match between the VaR estimations and the actual hits can be seen in this instance since all stocks have UCT values of 1.

The frequency test (FT) calculates the hit rate across all observations. Higher FT values suggest more hits, which is desirable for the accuracy of the VaR model. All of the equities have FT values above 0.94 in this instance, indicating that the VaR model does a good job of capturing frequent downward moves.

The fraction of hits among all VaR estimations is measured by the Failure Rate (FR). For the accuracy of the VaR model, lower FR values suggest a reduced failure rate. All equities in this instance have FR ratings of 1, indicating that extreme downside moves were successfully captured.

Based on these results, we can conclude that the VaR model performs well in capturing extreme downside movements and has a good match between the VaR estimates and the actual hits for the selected stocks.

**Interpretation of TUFF backtesting results on Variance Covariance approach(Delta Normal approach).**

Table 18: TUFF Backtesting Results for Variance Covariance Approach for Selected Stocks (Assuming Data is Normally Distributed)

Stock	TLT	UCT	FT	FR
Apple	0.975	1	0.975	1
Amazon	0.974	1	0.974	1
Netflix	0.977	1	0.977	1
Google	0.975	1	0.975	1

Table 19: TUFF Backtesting Results for Variance Covariance Approach(Factually, Data is not Normally Distributed)

Stock	TLT	UCT	FT	FR
Apple	0.975	1	0.975	1
Amazon	0.974	1	0.974	1
Netflix	0.977	1	0.977	1
Google	0.975	1	0.975	1

Apple: According to the Traffic Light Test (TLT), which has a value of 0.975, the VaR model was broken in roughly 97.5% of the instances. The number of 1 in the Unconditional Coverage Test (UCT) indicates that all violations were detected. With a result of 0.975, the Frequency Test (FT) shows that the VaR model was surpassed in roughly 97.5% of the situations. The Failure Rate (FR) is 1, which means that every infraction was properly detected.

Amazon: The TLT is 0.974, which means that 97.4% of the time the VaR model was broken. The UCT is 1, which means that every violation was noted. The FT is 0.974, indicating that in almost 97.4% of the cases, the VaR model was surpassed. The FR is 1, meaning that every violation was properly recorded.

Netflix: The TLT is 0.977, indicating that the VaR model was violated in approximately 97.7% of the cases. The UCT is 1, indicating that all violations were captured. The FT is 0.977, suggesting that the VaR model was exceeded in approximately 97.7% of the cases. The FR is 1, indicating that all violations were correctly captured.

Google: The TLT is 0.975, implying that the VaR model was violated in approximately 97.5% of the cases. The UCT is 1, indicating that all violations were captured. The FT is 0.975, suggesting that the VaR model was exceeded in approximately 97.5% of the cases.

The FR is 1, indicating that all violations were correctly captured.

These findings suggest that the VaR model employed in the TUFF backtesting has demonstrated a high degree of accuracy and sufficiency in detecting breaches and exceeding the level of risk anticipated. The model catches all breaches when the UCT value is 1, ensuring that it fully accounts for downside risk. The VaR model appears to be cautious and effectively catches a substantial amount of extreme market swings given the high TLT and FT values. Overall, the findings show that the VaR model performed well in identifying risk factors and variances from expected returns for the chosen equities.<sup>75</sup>, which indicates that the VaR model was broken in about 97.5% of the instances. The UCT is 1, which means that every violation was noted. The FT is 0.975, which indicates that the VaR model was surpassed in almost 97.5% of the instances. The FR is 1, meaning that every violation was properly recorded.

In the conclusion even if the backtesting is successful and has accepted the model for each stock still it cannot be considered to be rejected because it does not meet the assumption of Normal distribution of data. As per the above analysis done through calculation of Skewness, Kurtosis, Jarque-bera test which can be seen in Table 4, it can be justified that data is not normally distributed and hence it can be said that TUFF backtesting has rejected this model or method for calculation of VaR for all the stocks.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Variance Covariance approach(Delta Normal approach). Here is the table for the result of TUFF backtesting considering last 500 observations.

Table 20: TUFF Backtesting Results for Variance Covariance approach for 500 observations

Stock	TLT	UCT	FT	FR
Apple	0.975	1	0.975	1
Amazon	0.974	1	0.974	1
Netflix	0.977	1	0.977	1
Google	0.975	1	0.975	1

The TUFF backtesting results indicate the following:

Traffic Light Test (TLT): All stocks, including Apple, Amazon, Netflix, and Google, have TLT values above 0.97, suggesting that the VaR model is capturing a significant portion of the downside risk.

Unconditional Coverage Test (UCT): The UCT values for all stocks are 1, indicating that the VaR model exhibits unconditional coverage, meaning that the actual returns below the VaR estimates are adequately captured.

Frequency Test (FT): The FT values for all stocks are above 0.97, indicating that the VaR model is effective in capturing the frequency of actual returns below the VaR estimates.

Failure Rate (FR): The FR values for all stocks are 1, indicating that the VaR model experiences failures in capturing the actual returns below the VaR estimates.

Overall, the TUFF backtesting results suggest that the VaR model performs well in capturing the downside risk for the selected stocks.

In the conclusion even if the backtesting is successful and has accepted the model for each stock still it cannot be considered to be rejected because it does not meet the assumption of Normal distribution of data. As per the above analysis done through calculation of Skewness, Kurtosis, Jarque-bera test which can be seen in Table 4, it can be justified that data is not normally distributed and hence it can be said that TUFF backtesting has rejected this model or method for calculation of VaR for all the stocks.

## Interpretation of TUFF backtesting results on Monte Carlo Simulation.

Table 21: TUFF Backtesting Results for Monte Carlo Simulation for selected stocks

Stock	TLT	UCT	FT	FR
Apple	0.9684418	1	0.9684418	1
Amazon	0.9727811	1	0.9727811	1
Netflix	0.9704142	1	0.9704142	1
Google	0.9759369	1	0.9759369	1

The Traffic Light Test (TLT) values for all stocks and the portfolio are above 0.97, indicating that the VaR model captures a significant portion of the downside risk.

The Unconditional Coverage Test (UCT) values are 1 for all stocks and the portfolio, indicating that the VaR model exhibits unconditional coverage, meaning that it captures the actual returns below the VaR estimates effectively.

The Frequency Test (FT) values for all stocks and the portfolio are above 0.97, indicating that the VaR model accurately captures the frequency of actual returns below the VaR estimates.

The Failure Rate (FR) values for all stocks and the portfolio are 1, indicating that the VaR model experiences failures in capturing the actual returns below the VaR estimates. Overall, the TUFF backtesting results suggest that the VaR model performs well in capturing the downside risk for the individual stocks (Apple, Amazon, Netflix, Google) as well as the portfolio. The model exhibits satisfactory results in terms of capturing the downside risk and aligning with the expected outcomes.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Monte Carlo Simulation. Here is the table for the result of TUFF backtesting considering last 500 observations.

Table 22: TUFF Backtesting Results for Monte Carlo Simulation for 500 observations

<b>Stock/Portfolio</b>	<b>TLT</b>	<b>UCT</b>	<b>FT</b>	<b>FR</b>
Apple	0.974359	1	0.974359	1
Amazon	0.97357	1	0.97357	1
Netflix	0.9755424	1	0.9755424	1
Google	0.9767258	1	0.9767258	1

Based on the TUFF backtesting results, the Traffic Light Test (TLT) for all stocks is above 0.97, indicating a high percentage of hits compared to the total observations. The Unconditional Coverage Test (UCT) is 1 for all cases, indicating a perfect fit of the VaR estimates. The Frequency Test (FT) is also above 0.97 for all stocks, indicating a high frequency of hits. The Failure Rate (FR) is 1 for all cases, indicating that all hits resulted in failures. These results indicate that the VaR estimates effectively captured downside risk, providing reliable measures for managing the risk.

**Interpretation of TUFF backtesting results on Extreme Value Theory(Semi parametric approach).**

Stock	Confidence Level	P-Value
Apple	95%	1
Amazon	95%	N/A
Netflix	95%	1
Google	95%	1
Apple	97.5%	N/A
Amazon	97.5%	N/A
Netflix	97.5%	1
Google	97.5%	1
Apple	99%	N/A
Amazon	99%	N/A
Netflix	99%	1
Google	99%	1

This table highlights the TUFF backtesting results using different colors. Green represents successful backtesting (p-value = 1), indicating that the EVT model accurately captured downside risk, while red represents unsuccessful backtesting (N/A), indicating issues with data availability or suitability.

The TUFF backtesting results indicate the performance of the TUFF backtest for different EVT models at various confidence levels (95%, 97.5%, and 99%). Here's the interpretation:  
 AppleReturns\_EVT\_95: The TUFF backtest for the EVT model applied to Apple stock returns at a 95% confidence level resulted in a p-value of 1. This suggests that the EVT model's VaR estimates accurately captured downside risk, as there were no significant exceedances beyond the estimated VaR.

AmazonReturns\_EVT\_95: The TUFF backtest for the EVT model applied to Amazon stock returns at a 95% confidence level resulted in an NA (Not Available) value. This could indicate missing or insufficient data to perform the backtest.

NetflixReturns\_EVT\_95: The TUFF backtest for the EVT model applied to Netflix stock returns at a 95% confidence level resulted in a p-value of 1. This indicates that the EVT model's VaR estimates effectively captured downside risk without significant exceedances.



GoogleReturns\_EVT\_95: The TUFF backtest for the EVT model applied to Google stock returns at a 95% confidence level resulted in a p-value of 1. This suggests that the EVT model's VaR estimates accurately captured downside risk without significant exceedances. AppleReturns\_EVT\_97.5, AmazonReturns\_EVT\_97.5: The TUFF backtests for the EVT models applied to Apple and Amazon stock returns at a 97.5% confidence level resulted in NA values. This could be due to missing or insufficient data for performing the backtest. NetflixReturns\_EVT\_97.5, GoogleReturns\_EVT\_97.5: The TUFF backtests for the EVT models applied to Netflix and Google stock returns at a 97.5% confidence level resulted in p-values of 1. This indicates that the EVT models' VaR estimates effectively captured downside risk without significant exceedances.

AppleReturns\_EVT\_99, AmazonReturns\_EVT\_99: The TUFF backtests for the EVT models applied to Apple and Amazon stock returns at a 99% confidence level resulted in NA values. This could be due to missing or insufficient data for performing the backtest.

NetflixReturns\_EVT\_99, GoogleReturns\_EVT\_99: The TUFF backtests for the EVT models applied to Netflix and Google stock returns at a 99% confidence level resulted in p-values of 1. This suggests that the EVT models' VaR estimates accurately captured downside risk without significant exceedances.

Overall, the TUFF backtesting results indicate that the EVT models' VaR estimates were generally effective in capturing downside risk for Apple, Netflix, and Google stocks at different confidence levels. However, there may be issues with the Model incompatibility as there is no insufficient data or there is no missing data for performing the backtest for Amazon stock.

Alternatively, Backtesting has been done on last 500 observations for each method. will discuss for Extreme Value Theory approach. Here is the table for the result of TUFF backtesting considering last 500 observations.

Stock	Confidence Level	P-Value
Apple	95%	1
Amazon	95%	N/A
Netflix	95%	1
Google	95%	1
Apple	97.5%	N/A
Amazon	97.5%	N/A
Netflix	97.5%	1
Google	97.5%	1
Apple	99%	N/A
Amazon	99%	N/A
Netflix	99%	0.9999997
Google	99%	0.8413447

Based on the TUFF backtesting results, we can interpret the findings as follows:

For the 95% confidence level, the EVT models for Apple, Netflix, and Google have successfully passed the backtest, as indicated by a p-value of 1. However, the EVT model for Amazon has not provided any backtesting result (NA). For the 97.5% confidence level, the EVT models for Netflix and Google have successfully passed the backtest, with p-values of 1. The EVT models for Apple and Amazon have not provided any backtesting result (NA). For the 99% confidence level, the EVT model for Netflix and Google has successfully passed the backtest, with a p-value of 1 & 0.8 which is very close to 1. The EVT models for Apple, Amazon have not provided any backtesting result (NA).

However, there may be issues with the Model incompatibility as there is no insufficient data or there is no missing data for performing the backtest for stocks for which the results are NA.

## 4 Summary and Conclusion

In short, we ought to be able to identify most bad VaR models, but the worrying issue is whether we can find any good ones. One of the most widely used techniques for assessing market risks is VaR. Every VaR model forecasts future portfolio performance using historical market data. The models also rely on assumptions and approximations that do not necessarily hold under all circumstances. The accuracy of predicted VaR levels should be questioned because the methodologies are far from ideal. One of the most widely used techniques for assessing market risks is VaR. Every VaR model forecasts future portfolio performance using historical market data. The models also rely on assumptions and approximations that do not necessarily hold under all circumstances. The accuracy of predicted VaR levels should be questioned because the methodologies are far from ideal. The theoretical part of this thesis discussed different approaches to computing VaR, and evaluated specifically the shortcomings of these models. Some of the most common backtests that are being used to measure the accuracy of VaR models were presented. Backtest such as Kupiec's POF-test or Traffic Light, Unconditional Coverage, Frequency, Failure Rate test (TUFF test) has been used.

VaR has been estimated at the confidence level of 95%, 97.5%, 99% for all the stocks with different methods such as Historical Simulation, Variance Covariance Approach, Monte Carlo Simulation, Extreme Value Theory Approach and backtesting is done on all the methods indicated. The results of backtesting provided some indication that there is a potential problem with the system. The results from Kupiec test suggested underestimation of risk except for EVT model as this model has given mixed results and TUFF test has also given mixed results but more on the positive side. The backtesting result raises concern about the model's ability to estimate risk in satisfactory precision. Because of volatility due to different circumstances inevitably cause problems in estimating parameters that should describe future market movements. Since all VaR models rely on historical market data, this issue not only concerns the case at hand but VaR systems in general. Abnormal market behavior is simply beyond of what any VaR model is intended to capture. As the amount of data is huge that's why majority of time VaR models have been rejected for this VaR models are not to be blamed as the data is huge it has too many number of observations it is obvious that there will be volatility and ups and downs.

This study tested only equities. Systematic backtesting should be part of VaR reporting in

order to constantly monitor the performance of the model. However, the problem is that the inflexibility and slowness of data processing makes it challenging to conduct any regular daily-based backtesting in the Company. Nevertheless, the issue of future backtesting, whether it will be continuous or random testing, ought to be thoroughly considered.

Even i have used several different backtests in this thesis, The testing process starts with Kupiec test, this test rejected three models like Historical Simulation, Variance Covariance Approach, Monte Carlo Simulation and there were mix results for Extreme Value theory Approach. This is because the dataset is too huge and it is obvious the there will be exceedances which will be more than the threshold. The other backtesting method is TUFF test which has accepted many models like Historical Simulation, Variance Covariance Approach, Monte Carlo Simulation and have mix results for Extreme Value theory Approach. The above test have been made on the full dataset and alternatively test has been performed on last 500 observations also. When the test was performed for last 500 observations for Kupiec test and TUFF test we got the results similar to test conducted for full dataset because 500 observations is also a huge dataset. To overcome this there is one concept called Rolling window which would have been introduced but due to limitations of the words and guidance it has not been included.

Understanding the limitations of VaR computation is the most crucial lesson to take away from this study, regardless of the methodology used for future backtesting. The empirical research demonstrates that VaR figures should never be taken as 100 percent. regardless of how advanced the systems are, correct. VaR, however, can be a very helpful tool in risk management provided its users are aware of its shortcomings, especially since there are no viable alternatives that could be employed in its place [1].

## 5 References

### References

- [1] The Institute of Chartered Accountants of India, “Study Material of Risk Management,” 2019.
- [2] P. Adamko, E. Spuchlakova, and K. Valaskova, “The History and Ideas behind VaR,” in Proceedings of the International Conference and Applied Economics, ICOAE 2015, Kazan, Russia, Jul. 2-4, 2015.
- [3] NASDAQ.com. (n.d.). [Online]. Available: <https://www.nasdaq.com/>
- [4] J. C. Hull, Risk Management and Financial Institutions, 5th ed. Wiley, 2018.
- [5] P. Jorion, Value at Risk: The New Benchmark for Managing Financial Risk, 3rd ed. New York, NY, USA: McGraw-Hill, 2007.
- [6] K. Dowd, Measuring Market Risk. John Wiley & Sons, 2005.
- [7] M. Pritsker, “The Hidden Dangers of Historical Simulation,” Journal of Banking & Finance, vol. 30, no. 2, pp. 561-582, 2006.
- [8] P. Christoffersen, Elements of Financial Risk Management. San Diego, CA, USA: Academic Press, 2012.
- [9] C. Alexander, Market Risk Analysis: Practical Financial Econometrics. John Wiley & Sons, 2008.
- [10] A. J. McNeil, R. Frey, and P. Embrechts, Quantitative Risk Management: Concepts, Techniques, and Tools, Revised ed. Princeton University Press, 2015.
- [11] R. S. Tsay, An Introduction to Analysis of Financial Data with R. John Wiley & Sons, Incorporated, 2012.
- [12] Riskmetrics. (n.d.). VaR. [Online]. Available: <http://help.riskmetrics.com/RiskManager3/Content/Stat>  
Accessed: May 27, 2023
- [13] C. Acerbi and D. Tasche, “On the Coherence of Expected Shortfall,” Journal of Banking & Finance, vol. 26, no. 7, pp. 1487-1503, 2002.

- [14] P. Artzner, F. Delbaen, J. M. Eber, and D. Heath, "Coherent Measures of Risk," *Mathematical Finance*, vol. 9, no. 3, pp. 203-228, 1999.
- [15] H. Markowitz, "Portfolio Selection," *The Journal of Finance*, vol. 7, no. 1, pp. 77-91, 1952.
- [16] P. H. Kupiec, "Techniques for Verifying the Accuracy of Risk Management Models," *The Journal of Derivatives*, vol. 3, no. 2, pp. 73-84, Winter 1995.
- [17] Yahoo Finance. "Stock Historical Prices & Data for Alphabet Inc. (GOOGL), Amazon.com Inc. (AMZN), Netflix Inc. (NFLX), and Apple Inc. (AAPL)." [Online]. Available: <https://finance.yahoo.com>. Accessed: May 25, 2023 .

## 6 Appendix

This document presents the analysis of four stocks: Apple Inc. (AAPL), Amazon.com Inc. (AMZN), Netflix Inc. (NFLX), and Alphabet Inc. (GOOGL). The analysis includes retrieving the stock prices, computing the stock returns, and running a historical simulation to calculate Value at Risk (VaR) for each stock.

The R language was used to retrieve stock prices, compute stock returns, and perform calculation of VaR by different methods and performing backtesting on that different methods. Below is the R code used for this analysis:

[A.1] This are required **packages**

```
require(quantmod);  
require(PerformanceAnalytics)  
library(robustbase)  
library(PortfolioAnalytics)  
library(portfolio.optimization)  
library(evd)  
library(extRemes)  
library(evir)  
library(tseries)  
library(DEoptim)  
library(xts)  
library(POT)  
library(rugarch)
```

[A.2] Code **for** Section 3.1 of the Index

```
#Start And End Date of the Sample  
start.date <- as.Date(c("2013-01-01"))  
end.date <- as.Date(c("2023-01-30"))
```

```
#Apple Stock Prices
```

```

getSymbols("AAPL", from = start.date, to = end.date)
ApplePrices <- xts(AAPL$AAPL.Close)
AppleReturns <- log(ApplePrices/Lag(ApplePrices,1))
AppleReturns <- AppleReturns[-1]

```

*#Amazon Stock Prices*

```

getSymbols("AMZN", from = start.date, to = end.date)
AmazonPrices <- xts(AMZN$AMZN.Close)
AmazonReturns <- log(AmazonPrices/Lag(AmazonPrices,1))
AmazonReturns <- AmazonReturns[-1]

```

*#Netflix Stock Prices*

```

getSymbols("NFLX", from = start.date, to = end.date)
NetflixPrices <- xts(NFLX$NFLX.Close)
NetflixReturns <- log(NetflixPrices/Lag(NetflixPrices,1))
NetflixReturns <- NetflixReturns[-1]

```

*#Google Stock Prices*

```

getSymbols("GOOGL", from = start.date, to = end.date)
GooglePrices <- xts(GOOGL$GOOGL.Close)
GoogleReturns <- log(GooglePrices/Lag(GooglePrices,1))
GoogleReturns <- GoogleReturns[-1]

```

*# Remove NA values from AppleReturns*

```

AppleReturns <- na.omit(AppleReturns)

```

*# Remove NA values from AmazonReturns*

```

AmazonReturns <- na.omit(AmazonReturns)

```

*# Remove NA values from NetflixReturns*

```

NetflixReturns <- na.omit(NetflixReturns)

```



```

# Remove NA values from GoogleReturns
GoogleReturns <- na.omit(GoogleReturns)

[A.3] Code for Section 3.2.1 of the Index
# Historical Simulation for Apple
portfolio_aapl <- AppleReturns
VaR_aapl <- quantile(portfolio_aapl, probs= c(0.05,0.025,0.01))

# Historical Simulation for Amazon
portfolio_amzn <- AmazonReturns
VaR_amzn <- quantile(portfolio_amzn, probs= c(0.05,0.025,0.01))

# Historical Simulation for Tesla
portfolio_nflx <- NetflixReturns
VaR_nflx <- quantile(portfolio_nflx, probs= c(0.05,0.025,0.01))

# Historical Simulation for Google
portfolio_googl <- GoogleReturns
VaR_googl <- quantile(portfolio_googl, probs= c(0.05,0.025,0.01)
)

# Display the VaR results
print("VaR for Apple:")
print(VaR_aapl)

print("VaR for Amazon:")
print(VaR_amzn)

print("VaR for Netflix:")
print(VaR_nflx)

```

```

print("VaR for Google:")
print(VaR_googl)

# Calculate volatility for Apple
volatility_aapl <- sd(AppleReturns) * sqrt(252) # Assuming 252
trading days in a year

# Calculate volatility for Amazon
volatility_amzn <- sd(AmazonReturns) * sqrt(252)

# Calculate volatility for Google
volatility_googl <- sd(GoogleReturns) * sqrt(252)

# Calculate volatility for Netflix
volatility_nflx <- sd(NetflixReturns) * sqrt(252)


# Display the volatilities
print("Volatility for Apple:")
print(volatility_aapl)

print("Volatility for Amazon:")
print(volatility_amzn)

print("Volatility for Google:")
print(volatility_googl)

print("Volatility for Netflix:")
print(volatility_nflx)

```

```

[A.4] Code for Section 3.2.2 of the Index
# VarCov Approach(Delta Normal)

# Calculate mean return and standard deviation for each stock

Applemean <- mean(AppleReturns)
Applesd <- sd(AppleReturns)

Amazonmean <- mean(AmazonReturns)
Amazonsd <- sd(AmazonReturns)

Netflixmean <- mean(NetflixReturns)
Netflixsd <- sd(NetflixReturns)

Googlemean <- mean(GoogleReturns)
Googlesd <- sd(GoogleReturns)

# Set the desired confidence levels
confidence_levels <- c(0.95, 0.99, 0.975)

# Loop over each confidence level
for (confidence_level in confidence_levels) {
  # Calculate the z-score corresponding to the confidence level
  z_score <- qnorm(1 - confidence_level)

  # Calculate individual stock VaR
  apple_var <- Applesd * z_score
  amazon_var <- Amazonsd * z_score
  netflix_var <- Netflixsd * z_score
  google_var <- Googlesd * z_score

  # Print the individual stock VaR
  cat("For Confidence Level:", confidence_level * 100, "%\n")
}

```

```

cat("Apple-VaR: ", apple_var, "\n")
cat("Amazon-VaR: ", amazon_var, "\n")
cat("Netflix-VaR: ", netflix_var, "\n")
cat("Google-VaR: ", google_var, "\n\n")
}

#Skewness and Kurtosis
# Apple stock
skewness_apple <- skewness(AppleReturns)
kurtosis_apple <- kurtosis(AppleReturns)
cat("Apple-Skewness:", skewness_apple, "\n")
cat("Apple-Kurtosis:", kurtosis_apple, "\n")

# Amazon stock
skewness_amazon <- skewness(AmazonReturns)
kurtosis_amazon <- kurtosis(AmazonReturns)
cat("Amazon-Skewness:", skewness_amazon, "\n")
cat("Amazon-Kurtosis:", kurtosis_amazon, "\n")

# Netflix stock
skewness_netflix <- skewness(NetflixReturns)
kurtosis_netflix <- kurtosis(NetflixReturns)
cat("Netflix-Skewness:", skewness_netflix, "\n")
cat("Netflix-Kurtosis:", kurtosis_netflix, "\n")

# Google stock
skewness_google <- skewness(GoogleReturns)
kurtosis_google <- kurtosis(GoogleReturns)
cat("Google-Skewness:", skewness_google, "\n")
cat("Google-Kurtosis:", kurtosis_google, "\n")

# Jarque-Bera test
library(tseries)

```

```
# Apple stock
jarque_bera_test_apple <- jarque.bera.test(AppleReturns)
cat("Apple-Jarque-Bera-test-p-value:", jarque_bera_test_apple$p.
    value, "\n")
```

```
# Amazon stock
jarque_bera_test_amazon <- jarque.bera.test(AmazonReturns)
cat("Amazon-Jarque-Bera-test-p-value:", jarque_bera_test_amazon$
    p.value, "\n")
```

```
# Netflix stock
jarque_bera_test_netflix <- jarque.bera.test(NetflixReturns)
cat("Netflix-Jarque-Bera-test-p-value:", jarque_bera_test_
    netflix$p.value, "\n")
```

```
# Google stock
jarque_bera_test_google <- jarque.bera.test(GoogleReturns)
cat("Google-Jarque-Bera-test-p-value:", jarque_bera_test_google$
    p.value, "\n")
```

```
# Additional package for normal distribution curve
library(ggplot2)
```

```
# Function to create histogram with normal curve
createHistogram <- function(returns, company) {
  returns.df <- data.frame(returns)
  colnames(returns.df) <- c("returns")
  p <- ggplot(returns.df, aes(x=returns)) +
    geom_histogram(aes(y=..density..), colour="black", fill="
        white", bins = 50) +
```

```

stat_function(fun = dnorm, args = list(mean = mean(returns.
df$returns, na.rm = TRUE), sd = sd(returns.df$returns, na
.rm = TRUE))), color = "red", size = 1) +
theme_minimal() +
ggtitle(paste0(company, "-returns-histogram"))
print(p)
}

```

*# Create histograms*

```

createHistogram(AppleReturns, "Apple")
createHistogram(AmazonReturns, "Amazon")
createHistogram(NetflixReturns, "Netflix")
createHistogram(GoogleReturns, "Google")

```

[A.5] Code for Section 3.2.3 of the Index

*#Monte carlo simulation*

*# Define the number of simulation paths*

```
paths <- 1000
```

*# Set the desired confidence levels*

```
confidence_levels <- c(0.05, 0.01, 0.025)
```

*# Function to calculate VaR using Monte Carlo simulation*

```

calculate_VaR <- function(returns, confidence_level) {
  # Calculate the mean and standard deviation of the returns
  mean_returns <- mean(returns, na.rm=TRUE)
  sd_returns <- sd(returns, na.rm = TRUE)

```

*# Perform Monte Carlo simulation*

```
simulated_returns <- mean_returns + sd_returns * rnorm(paths)
```

```

# Calculate VaR using the simulated returns
VaR <- quantile(simulated_returns, probs = confidence_level)

# Return the VaR result
return(VaR)
}

# Calculate VaR for the portfolio
portfolio_returns <- 0.25 * AppleReturns + 0.25 * AmazonReturns
+ 0.25 * NetflixReturns + 0.25 * GoogleReturns

# Loop over each confidence level
for (confidence_level in confidence_levels) {
  # Calculate VaR for individual stocks
  VaR_apple <- calculate_VaR(AppleReturns, confidence_level)
  VaR_amazon <- calculate_VaR(AmazonReturns, confidence_level)
  VaR_netflix <- calculate_VaR(NetflixReturns, confidence_level)
  VaR_google <- calculate_VaR(GoogleReturns, confidence_level)

  VaR_portfolio <- calculate_VaR(portfolio_returns, confidence_level)

  # Print the VaR results
  cat("VaR at", (1 - confidence_level) * 100, "% confidence level:\n")
  cat("Apple:", VaR_apple, "\n")
  cat("Amazon:", VaR_amazon, "\n")
  cat("Netflix:", VaR_netflix, "\n")
  cat("Google:", VaR_google, "\n")
  cat("Portfolio:", VaR_portfolio, "\n\n")
}

```

[A.6] Code for Section 3.2.4 of the Index

```

#Extreme value theory

# Install and load the package
if (!require(evir)) install.packages("evir"); library(evir)

# Function to perform EVT analysis and return VaR
evt_analysis <- function(returns, threshold_quantile, var_
  quantile) {
  # Select data over the threshold
  threshold <- quantile(returns, threshold_quantile)
  excesses <- returns[returns > threshold] - threshold

  # Keep only positive excesses
  excesses <- excesses[excesses > 0]

  # Check if there's enough tail data
  if (length(excesses) > 10) { # Change the number as necessary
    # Fit Generalized Pareto Distribution (GPD)
    gpd_fit <- tryCatch(gpd(excesses, threshold),
                        error = function(e) return(NA)) # Handle
                        errors

    if (!any(is.na(gpd_fit))) {
      # Calculate VaR
      var <- evd::qgpd(var_quantile, shape = gpd_fit$par.ests
        [1], scale = gpd_fit$par.ests[2])
      return(list(gpd_fit = gpd_fit, VaR = var))
    }
  }
  return(NA)
}

```



```

# Perform EVT analysis on each returns series and calculate VaR
  for different threshold and VaR quantiles
evt_models <- list()

# Define the desired VaR quantiles
var_quantiles <- c(0.95, 0.975, 0.99)

# For 95% threshold quantile
evt_models$AppleReturns_EVT_95 <- evt_analysis(AppleReturns,
  0.95, var_quantiles)
evt_models$AmazonReturns_EVT_95 <- evt_analysis(AmazonReturns,
  0.95, var_quantiles)
evt_models$NetflixReturns_EVT_95 <- evt_analysis(NetflixReturns,
  0.95, var_quantiles)
evt_models$GoogleReturns_EVT_95 <- evt_analysis(GoogleReturns,
  0.95, var_quantiles)

# For 97.5% threshold quantile
evt_models$AppleReturns_EVT_975 <- evt_analysis(AppleReturns,
  0.975, var_quantiles)
evt_models$AmazonReturns_EVT_975 <- evt_analysis(AmazonReturns,
  0.975, var_quantiles)
evt_models$NetflixReturns_EVT_975 <- evt_analysis(NetflixReturns
, 0.975, var_quantiles)
evt_models$GoogleReturns_EVT_975 <- evt_analysis(GoogleReturns,
  0.975, var_quantiles)

# For 99% threshold quantile
evt_models$AppleReturns_EVT_99 <- evt_analysis(AppleReturns,
  0.99, var_quantiles)
evt_models$AmazonReturns_EVT_99 <- evt_analysis(AmazonReturns,
  0.99, var_quantiles)
evt_models$NetflixReturns_EVT_99 <- evt_analysis(NetflixReturns,

```

```

    0.99, var_quantiles)
evt_models$GoogleReturns_EVT_99 <- evt_analysis(GoogleReturns,
    0.99, var_quantiles)

# Print the EVT models and VaR
evt_models

```

[A.7] Code **for** Section 3.4.1 of Index

```

Backtesting for Historical Simulation by kupiec backtesting
#Kupiec Test
# Perform Kupiec test for each individual stock
# Set the confidence levels
confidence_levels <- c(0.95, 0.975, 0.99)

# Perform Kupiec test for each individual stock
stocks <- list(
  "Apple" = portfolio_aapl,
  "Amazon" = portfolio_amzn,
  "Netflix" = portfolio_nflx,
  "Google" = portfolio_googl
)

for (stock_name in names(stocks)) {
  # Get the actual returns for the stock
  actual_returns <- stocks[[stock_name]]

  for (level in confidence_levels) {
    # Calculate VaR using historical simulation for the stock
    VaR_stock <- quantile(actual_returns, probs = level)
  }
}

```

```

# Compute the exceedance indicator
exceedance_indicator <- actual_returns < VaR_stock

# Count the number of exceedances
num_exceedances <- sum(exceedance_indicator)

# Perform the Kupiec test
num_observations <- length(actual_returns)
num_failures <- num_exceedances
alpha <- 1 - level
log_likelihood <- num_failures * log(alpha) + (num_
  observations - num_failures) * log(1 - alpha)
kupiec_test_statistic <- -2 * log_likelihood
critical_value <- qchisq(1 - level, df = 1)

# Compare the test statistic with the critical value
if (kupiec_test_statistic > critical_value) {
  print(paste0("Kupiec test for ", stock_name, "-at-", level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec test for ", stock_name, "-at-", level
    * 100, "%:-VaR-model-accepted"))
}
}
}

Backtesting for Historical simulation by Kupiec for last 500
  observations
# Set the confidence levels
confidence_levels <- c(0.95, 0.975, 0.99)

# Perform Kupiec test for each individual stock
stocks <- list("Apple" = AppleReturns, "Amazon" = AmazonReturns,

```

```

    "Netflix" = NetflixReturns, "Google" = GoogleReturns)
n_obs <- 500

for (stock_name in names(stocks)) {
  for (level in confidence_levels) {
    # Get the actual returns for the stock
    actual_returns <- tail(stocks[[stock_name]], n_obs)

    # Calculate VaR using historical simulation for the stock
    VaR_stock <- quantile(actual_returns, probs = level)

    # Compute the exceedance indicator
    exceedance_indicator <- actual_returns < VaR_stock

    # Count the number of exceedances
    num_exceedances <- sum(exceedance_indicator)

    # Perform the Kupiec test
    num_observations <- length(actual_returns)
    num_failures <- num_exceedances
    alpha <- 1 - level
    log_likelihood <- num_failures * log(alpha) + (num_
      observations - num_failures) * log(1 - alpha)
    kupiec_test_statistic <- -2 * log_likelihood
    critical_value <- qchisq(1 - level, df = 1)

    # Compare the test statistic with the critical value
    if (kupiec_test_statistic > critical_value) {
      print(paste0("Kupiec-test-for-", stock_name, "-at-", level
        * 100, "%-confidence-level:-VaR-model-rejected"))
    } else {
      print(paste0("Kupiec-test-for-", stock_name, "-at-", level
        * 100, "%-confidence-level:-VaR-model-accepted"))
    }
  }
}

```

```

    }
  }
}

```

Backtesting **for** Variance Covariance approach **by** kupiec

```

  backtesting
#Backtesting for Var Cov Approach by Kupiec test

# Set the desired confidence levels
confidence_levels <- c(0.95, 0.99, 0.975)

# Define the actual returns for each stock (sample data)
actual_returns_apple <- AppleReturns
actual_returns_amazon <- AmazonReturns
actual_returns_netflix <- NetflixReturns
actual_returns_google <- GoogleReturns

# Loop over each confidence level
for (confidence_level in confidence_levels) {
  # Calculate the z-score corresponding to the confidence level
  z_score <- qnorm(1 - confidence_level)

  # Calculate individual stock VaR
  apple_var <- Applesd * z_score
  amazon_var <- Amazonsd * z_score
  netflix_var <- Netflixsd * z_score
  google_var <- Googlesd * z_score

  # Compute the exceedance indicator for each stock
  exceedance_indicator_apple <- actual_returns_apple < apple_var
  exceedance_indicator_amazon <- actual_returns_amazon < amazon_
    var

```

```

exceedance_indicator_netflix <- actual_returns_netflix <
  netflix_var
exceedance_indicator_google <- actual_returns_google < google_
  var

# Count the number of exceedances for each stock
num_exceedances_apple <- sum(exceedance_indicator_apple)
num_exceedances_amazon <- sum(exceedance_indicator_amazon)
num_exceedances_netflix <- sum(exceedance_indicator_netflix)
num_exceedances_google <- sum(exceedance_indicator_google)

# Perform the Kupiec test for each stock
num_observations <- length(actual_returns_apple) # Assuming
  all stocks have the same number of observations
num_failures_apple <- num_exceedances_apple
num_failures_amazon <- num_exceedances_amazon
num_failures_netflix <- num_exceedances_netflix
num_failures_google <- num_exceedances_google
alpha <- 1 - confidence_level
log_likelihood_apple <- num_failures_apple * log(alpha) + (num
  _observations - num_failures_apple) * log(1 - alpha)
log_likelihood_amazon <- num_failures_amazon * log(alpha) + (
  num_observations - num_failures_amazon) * log(1 - alpha)
log_likelihood_netflix <- num_failures_netflix * log(alpha) +
  (num_observations - num_failures_netflix) * log(1 - alpha)
log_likelihood_google <- num_failures_google * log(alpha) + (
  num_observations - num_failures_google) * log(1 - alpha)
kupiec_test_statistic_apple <- -2 * log_likelihood_apple
kupiec_test_statistic_amazon <- -2 * log_likelihood_amazon
kupiec_test_statistic_netflix <- -2 * log_likelihood_netflix
kupiec_test_statistic_google <- -2 * log_likelihood_google
critical_value <- qchisq(1 - confidence_level, df = 1)

```

```

# Compare the test statistic with the critical value for each
  stock

if (kupiec_test_statistic_apple > critical_value) {
  print(paste0("Kupiec test for Apple at ", confidence_level *
    100, "%: VaR model rejected"))
} else {
  print(paste0("Kupiec test for Apple at ", confidence_level *
    100, "%: VaR model accepted"))
}

if (kupiec_test_statistic_amazon > critical_value) {
  print(paste0("Kupiec test for Amazon at ", confidence_level
    * 100, "%: VaR model rejected"))
} else {
  print(paste0("Kupiec test for Amazon at ", confidence_level
    * 100, "%: VaR model accepted"))
}

if (kupiec_test_statistic_netflix > critical_value) {
  print(paste0("Kupiec test for Netflix at ", confidence_level
    * 100, "%: VaR model rejected"))
} else {
  print(paste0("Kupiec test for Netflix at ", confidence_level
    * 100, "%: VaR model accepted"))
}

if (kupiec_test_statistic_google > critical_value) {
  print(paste0("Kupiec test for Google at ", confidence_level
    * 100, "%: VaR model rejected"))
} else {
  print(paste0("Kupiec test for Google at ", confidence_level
    * 100, "%: VaR model accepted"))
}

```

```
}
```

```
Backtesting for Var Cov Approach by Kupiec for last 500
```

```
observations
```

```
#Kupiec backtesting
```

```
# Set the desired confidence levels
```

```
confidence_levels <- c(0.95, 0.99, 0.975)
```

```
# Define the actual returns for each stock (sample data)
```

```
actual_returns_apple <- tail(AppleReturns, 500)
```

```
actual_returns_amazon <- tail(AmazonReturns, 500)
```

```
actual_returns_netflix <- tail(NetflixReturns, 500)
```

```
actual_returns_google <- tail(GoogleReturns, 500)
```

```
# Loop over each confidence level
```

```
for (confidence_level in confidence_levels) {
```

```
# Calculate the z-score corresponding to the confidence level
```

```
z_score <- qnorm(1 - confidence_level)
```

```
# Calculate individual stock VaR
```

```
apple_var <- Applesd * z_score
```

```
amazon_var <- Amazonsd * z_score
```

```
netflix_var <- Netflixsd * z_score
```

```
google_var <- Googlesd * z_score
```

```
# Compute the exceedance indicator for each stock
```

```
exceedance_indicator_apple <- actual_returns_apple < apple_var
```

```
exceedance_indicator_amazon <- actual_returns_amazon < amazon_  
var
```

```
exceedance_indicator_netflix <- actual_returns_netflix <  
netflix_var
```



```

exceedance_indicator_google <- actual_returns_google < google_
  var

# Count the number of exceedances for each stock
num_exceedances_apple <- sum(exceedance_indicator_apple)
num_exceedances_amazon <- sum(exceedance_indicator_amazon)
num_exceedances_netflix <- sum(exceedance_indicator_netflix)
num_exceedances_google <- sum(exceedance_indicator_google)

# Perform the Kupiec test for each stock
num_observations <- length(actual_returns_apple) # Assuming
  all stocks have the same number of observations
num_failures_apple <- num_exceedances_apple
num_failures_amazon <- num_exceedances_amazon
num_failures_netflix <- num_exceedances_netflix
num_failures_google <- num_exceedances_google
alpha <- 1 - confidence_level
log_likelihood_apple <- num_failures_apple * log(alpha) + (num
  _observations - num_failures_apple) * log(1 - alpha)
log_likelihood_amazon <- num_failures_amazon * log(alpha) + (
  num_observations - num_failures_amazon) * log(1 - alpha)
log_likelihood_netflix <- num_failures_netflix * log(alpha) +
  (num_observations - num_failures_netflix) * log(1 - alpha)
log_likelihood_google <- num_failures_google * log(alpha) + (
  num_observations - num_failures_google) * log(1 - alpha)
kupiec_test_statistic_apple <- -2 * log_likelihood_apple
kupiec_test_statistic_amazon <- -2 * log_likelihood_amazon
kupiec_test_statistic_netflix <- -2 * log_likelihood_netflix
kupiec_test_statistic_google <- -2 * log_likelihood_google
critical_value <- qchisq(1 - confidence_level, df = 1)

# Compare the test statistic with the critical value for each
  stock

```

```

if (kupiec_test_statistic_apple > critical_value) {
  print(paste0("Kupiec-test-for-Apple-at-", confidence_level *
    100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Apple-at-", confidence_level *
    100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_amazon > critical_value) {
  print(paste0("Kupiec-test-for-Amazon-at-", confidence_level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Amazon-at-", confidence_level
    * 100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_netflix > critical_value) {
  print(paste0("Kupiec-test-for-Netflix-at-", confidence_level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Netflix-at-", confidence_level
    * 100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_google > critical_value) {
  print(paste0("Kupiec-test-for-Google-at-", confidence_level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Google-at-", confidence_level
    * 100, "%:-VaR-model-accepted"))
}
}

```

```

Backtesting for Monte Carlo simulation by Kupiec Backtesting
#Kupiec backtesting

# Set the desired confidence levels
confidence_levels <- c(0.05, 0.01, 0.025)

# Define the number of simulation paths
paths <- 1000

# Function to calculate VaR using Monte Carlo simulation
calculate_VaR <- function(returns, confidence_level) {
  # Calculate the mean and standard deviation of the returns
  mean_returns <- mean(returns, na.rm = TRUE)
  sd_returns <- sd(returns, na.rm = TRUE)

  # Perform Monte Carlo simulation
  simulated_returns <- mean_returns + sd_returns * rnorm(paths)

  # Calculate VaR using the simulated returns
  VaR <- quantile(simulated_returns, probs = confidence_level)

  # Return the VaR result
  return(VaR)
}

# Calculate VaR for the portfolio
portfolio_returns <- 0.25 * AppleReturns + 0.25 * AmazonReturns
  + 0.25 * NetflixReturns + 0.25 * GoogleReturns

# Loop over each confidence level
for (confidence_level in confidence_levels) {
  # Calculate VaR for individual stocks

```

```

VaR_apple <- calculate_VaR(AppleReturns, confidence_level)
VaR_amazon <- calculate_VaR(AmazonReturns, confidence_level)
VaR_netflix <- calculate_VaR(NetflixReturns, confidence_level)
VaR_google <- calculate_VaR(GoogleReturns, confidence_level)

# Compute the exceedance indicator for each stock
exceedance_indicator_apple <- AppleReturns < VaR_apple
exceedance_indicator_amazon <- AmazonReturns < VaR_amazon
exceedance_indicator_netflix <- NetflixReturns < VaR_netflix
exceedance_indicator_google <- GoogleReturns < VaR_google

# Count the number of exceedances for each stock
num_exceedances_apple <- sum(exceedance_indicator_apple)
num_exceedances_amazon <- sum(exceedance_indicator_amazon)
num_exceedances_netflix <- sum(exceedance_indicator_netflix)
num_exceedances_google <- sum(exceedance_indicator_google)

# Perform the Kupiec test for each stock
num_observations <- length(AppleReturns) # Assuming all
      stocks have the same number of observations
num_failures_apple <- num_exceedances_apple
num_failures_amazon <- num_exceedances_amazon
num_failures_netflix <- num_exceedances_netflix
num_failures_google <- num_exceedances_google
alpha <- 1 - confidence_level
log_likelihood_apple <- num_failures_apple * log(alpha) + (num
      _observations - num_failures_apple) * log(1 - alpha)
log_likelihood_amazon <- num_failures_amazon * log(alpha) + (
      num_observations - num_failures_amazon) * log(1 - alpha)
log_likelihood_netflix <- num_failures_netflix * log(alpha) +
      (num_observations - num_failures_netflix) * log(1 - alpha)
log_likelihood_google <- num_failures_google * log(alpha) + (
      num_observations - num_failures_google) * log(1 - alpha)

```

```

kupiec_test_statistic_apple <- -2 * log_likelihood_apple
kupiec_test_statistic_amazon <- -2 * log_likelihood_amazon
kupiec_test_statistic_netflix <- -2 * log_likelihood_netflix
kupiec_test_statistic_google <- -2 * log_likelihood_google
critical_value <- qchisq(1 - confidence_level, df = 1)

# Compare the test statistic with the critical value for each
  stock

if (kupiec_test_statistic_apple > critical_value) {
  print(paste0("Kupiec-test-for-Apple-at-", confidence_level *
    100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Apple-at-", confidence_level *
    100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_amazon > critical_value) {
  print(paste0("Kupiec-test-for-Amazon-at-", confidence_level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Amazon-at-", confidence_level
    * 100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_netflix > critical_value) {
  print(paste0("Kupiec-test-for-Netflix-at-", confidence_level
    * 100, "%:-VaR-model-rejected"))
} else {
  print(paste0("Kupiec-test-for-Netflix-at-", confidence_level
    * 100, "%:-VaR-model-accepted"))
}

if (kupiec_test_statistic_google > critical_value) {

```

```

    print(paste0("Kupiec test for Google at ", confidence_level
        * 100, "%: VaR model rejected"))
} else {
    print(paste0("Kupiec test for Google at ", confidence_level
        * 100, "%: VaR model accepted"))
}
}

```

Backtesting **for** Monte Carlo Simulation **by** Kupiec **for** last 500

```

    observations
# Kupiec test
kupiec_test <- function(returns, VaR, confidence_level) {
    # Compute the exceedance indicator
    exceedance_indicator <- returns < VaR

    # Count the number of exceedances
    num_exceedances <- sum(exceedance_indicator)

    # Perform the Kupiec test
    num_observations <- length(returns)
    num_failures <- num_exceedances
    alpha <- 1 - confidence_level
    log_likelihood <- num_failures * log(alpha) + (num_
        observations - num_failures) * log(1 - alpha)
    kupiec_test_statistic <- -2 * log_likelihood
    critical_value <- qchisq(1 - confidence_level, df = 1)

    # Compare the test statistic with the critical value
    if (kupiec_test_statistic > critical_value) {
        return(paste("Kupiec test at", (1 - confidence_level) * 100,
            "%: confidence level: VaR model rejected"))
    }
}

```

```

    } else {
      return(paste("Kupiec-test-at", (1 - confidence_level) * 100,
        "%-confidence-level:-VaR-model-accepted"))
    }
  }
}

# Loop over each confidence level
for (confidence_level in confidence_levels) {
  # Calculate VaR for individual stocks
  VaR_apple <- calculate_VaR(AppleReturns, confidence_level)
  VaR_amazon <- calculate_VaR(AmazonReturns, confidence_level)
  VaR_netflix <- calculate_VaR(NetflixReturns, confidence_level)
  VaR_google <- calculate_VaR(GoogleReturns, confidence_level)

  # Perform Kupiec test for the last 500 observations
  actual_returns_apple <- tail(AppleReturns, 500)
  actual_returns_amazon <- tail(AmazonReturns, 500)
  actual_returns_netflix <- tail(NetflixReturns, 500)
  actual_returns_google <- tail(GoogleReturns, 500)

  test_result_apple <- kupiec_test(actual_returns_apple, VaR_
    apple, confidence_level)
  test_result_amazon <- kupiec_test(actual_returns_amazon, VaR_
    amazon, confidence_level)
  test_result_netflix <- kupiec_test(actual_returns_netflix, VaR_
    _netflix, confidence_level)
  test_result_google <- kupiec_test(actual_returns_google, VaR_
    google, confidence_level)

  # Print the Kupiec test results
  cat("Apple:", test_result_apple, "\n")
  cat("Amazon:", test_result_amazon, "\n")
  cat("Netflix:", test_result_netflix, "\n")

```

```

cat("Google:", test_result_google, "\n")
}

```

Backtesting **for** Extreme Value theory **by** Kupiec Backtesting

*#Kupiec backtesting*

*# Function to perform Kupiec test*

```

kupiec_test <- function(returns, VaR, alpha){
  # Number of exceptions
  exceptions <- sum(returns < -VaR)
  # Total number of observations
  N <- length(returns)
  # Expected number of exceptions
  expected_exceptions <- N * (1 - alpha)
  # Kupiec test statistic
  test_stat <- -2 * ((exceptions * log(exceptions / N) +
                    (N - exceptions) * log((N - exceptions)
                    / N)) -
                (exceptions * log((1 - alpha)) +
                (N - exceptions) * log(alpha)))
  # P-value
  p_value <- 1 - pchisq(test_stat, df = 1)
  return(list(test_stat = test_stat, p_value = p_value))
}

```

*# Define the list of model names and their corresponding  
threshold quantiles*

```

model_names <- c("AppleReturns_EVT_95", "AmazonReturns_EVT_95",
  "NetflixReturns_EVT_95", "GoogleReturns_EVT_95",
  "AppleReturns_EVT_975", "AmazonReturns_EVT_975",
  "NetflixReturns_EVT_975", "GoogleReturns_EVT_975",
  "AppleReturns_EVT_99", "AmazonReturns_EVT_99",
  "NetflixReturns_EVT_99", "GoogleReturns_EVT_99")

```



99”)

```
threshold_quantiles <- rep(0.95, length(model_names))

# Perform Kupiec backtesting for each EVT model
backtest_results <- list()

for (i in 1:length(model_names)) {
  model_name <- model_names[i]
  threshold_quantile <- threshold_quantiles[i]

  if (!is.null(evt_models[[model_name]]) && "VaR" %in% names(evt
    _models[[model_name]])) {
    var <- evt_models[[model_name]]$VaR

    if (!is.na(var) && is.vector(var)) {
      returns_name <- gsub("_EVT.*", "", model_name)
      returns <- get(returns_name) # use get() to turn a string
        into a variable

      backtest_results[[model_name]] <- kupiec_backtest(returns,
        var, threshold_quantile)
    } else {
      backtest_results[[model_name]] <- NA
    }
  } else {
    backtest_results[[model_name]] <- NA
  }
}

# Print the backtest results
backtest_results
```

```

Kupiec backtesting on Extreme Value Theory for last 500
  observations
#Kupiec backtesting
# Function to perform Kupiec test
kupiec_test <- function(returns , VaR, alpha){
  # Number of exceptions
  exceptions <- sum(returns < -VaR)
  # Total number of observations
  N <- length(returns)
  # Expected number of exceptions
  expected_exceptions <- N * (1 - alpha)
  # Kupiec test statistic
  test_stat <- -2 * ((exceptions * log(exceptions / N) +
                    (N - exceptions) * log((N - exceptions)
                    / N)) -
                (exceptions * log((1 - alpha)) +
                (N - exceptions) * log(alpha)))
  # P-value
  p_value <- 1 - pchisq(test_stat , df = 1)
  return(list(test_stat = test_stat , p_value = p_value))
}

# Define the list of model names and their corresponding
  threshold quantiles
model_names <- c("AppleReturns_EVT_95" , "AmazonReturns_EVT_95" ,
  "NetflixReturns_EVT_95" , "GoogleReturns_EVT_95" ,
  "AppleReturns_EVT_975" , "AmazonReturns_EVT_975"
  , "NetflixReturns_EVT_975" , "GoogleReturns_
  EVT_975" ,
  "AppleReturns_EVT_99" , "AmazonReturns_EVT_99" ,
  "NetflixReturns_EVT_99" , "GoogleReturns_EVT_
  99")

```

```

threshold_quantiles <- c(0.95, 0.95, 0.95, 0.95, 0.975, 0.975,
  0.975, 0.975, 0.99, 0.99, 0.99, 0.99)

# Perform Kupiec backtesting for each EVT model
backtest_results <- list()

for (i in 1:length(model_names)) {
  model_name <- model_names[i]
  threshold_quantile <- threshold_quantiles[i]

  if (!is.null(evt_models[[model_name])) && "VaR" %in% names(evt
    _models[[model_name]])) {
    var <- evt_models[[model_name]]$VaR

    if (!is.na(var) && is.vector(var)) {
      returns_name <- gsub("_EVT.*", "", model_name)
      returns <- get(returns_name) # use get() to turn a string
        into a variable

      # Select only the last 500 observations
      var <- tail(var, 500)
      returns <- tail(returns, 500)

      backtest_results[[model_name]] <- kupiec_backtest(returns,
        var, threshold_quantile)
    } else {
      backtest_results[[model_name]] <- NA
    }
  } else {
    backtest_results[[model_name]] <- NA
  }
}

```

```
# Print the backtest results
backtest_results
```

[A.8] Code **for** Section 3.4.2 of Index

TUFF Bactesting **on** Historical Simulation

```
# List of portfolios
```

```
portfolios <- list(
  "Apple" = portfolio_aapl,
  "Amazon" = portfolio_amzn,
  "Netflix" = portfolio_nflx,
  "Google" = portfolio_googl
)
```

```
# List of VaR estimates
```

```
var_estimates <- list(
  "Apple" = VaR_aapl,
  "Amazon" = VaR_amzn,
  "Netflix" = VaR_nflx,
  "Google" = VaR_googl
)
```

```
# Function for TUFF backtest
```

```
tuff_backtest <- function(actual_returns, var_estimates) {
```

```
# Calculate hits
```

```
hits <- actual_returns < -var_estimates
```

```
# Traffic Light Test
```

```

tlt <- sum(hits) / length(actual_returns)

# Unconditional Coverage Test
uct <- ifelse(sum(actual_returns < -var_estimates) != 0,
              sum(hits) / sum(actual_returns < -var_estimates)
              ,
              NA)

# Frequency Test
ft <- length(hits[hits == TRUE]) / length(hits)

# Failure Rate
fr <- ifelse(sum(hits) != 0,
              length(hits[hits == TRUE]) / sum(hits) ,
              NA)

return(list(TLT = tlt , UCT = uct , FT = ft , FR = fr))
}

# Apply TUFF backtest on each portfolio
for (stock_name in names(portfolios)) {
  print(paste("TUFF-Backtest-for", stock_name, ":"))
  print(tuff_backtest(portfolios[[stock_name]] , var_estimates[[
    stock_name]]))
}

TUFF backtesting on Historical Simulation for last 500
observations
#TUFF test
# List of portfolios
portfolios <- list(
  "Apple" = portfolio_aapl,
  "Amazon" = portfolio_amzn,

```

```

    "Netflix" = portfolio_nflx ,
    "Google" = portfolio_googl
)

# List of VaR estimates
var_estimates <- list(
  "Apple" = VaR_aapl ,
  "Amazon" = VaR_amzn ,
  "Netflix" = VaR_nflx ,
  "Google" = VaR_googl
)

# Function for TUFF backtest
tuff_backtest <- function(actual_returns , var_estimates) {

  # Slice the last 500 observations
  actual_returns <- tail(actual_returns , 500)
  var_estimates <- tail(var_estimates , 500)

  # Calculate hits
  hits <- actual_returns < -var_estimates

  # Traffic Light Test
  tlt <- sum(hits) / length(actual_returns)

  # Unconditional Coverage Test
  uct <- ifelse(sum(actual_returns < -var_estimates) != 0,
               sum(hits) / sum(actual_returns < -var_estimates)
               ,
               NA)

  # Frequency Test
  ft <- length(hits[hits == TRUE]) / length(hits)

```

```

# Failure Rate
fr <- ifelse(sum(hits) != 0,
             length(hits[hits == TRUE]) / sum(hits),
             NA)

return(list(TLT = tlt, UCT = uct, FT = ft, FR = fr))
}

# Apply TUFF backtest on each portfolio
for (stock_name in names(portfolios)) {
  print(paste("TUFF-Backtest-for", stock_name, ":"))
  print(tuff_backtest(portfolios[[stock_name]], var_estimates[[
    stock_name]]))
}

```

TUFF backtesting on Variance Covariance approach

```

#TUFF backtesting
# Function to perform TUFF backtesting
tuff_backtest <- function(actual_returns, VaR_estimates) {
  # Calculate hits
  hits <- actual_returns < -VaR_estimates

  # Traffic Light Test
  tlt <- sum(hits) / length(actual_returns)

  # Unconditional Coverage Test
  uct <- ifelse(sum(actual_returns < -VaR_estimates) != 0,
               sum(hits) / sum(actual_returns < -VaR_estimates)
               ,
               NA)
}

```

```

# Frequency Test
ft <- length(hits[hits == TRUE]) / length(hits)

# Failure Rate
fr <- ifelse(sum(hits) != 0,
             length(hits[hits == TRUE]) / sum(hits),
             NA)

return(list(TLT = tlt, UCT = uct, FT = ft, FR = fr))
}

# Perform TUFF backtesting for each stock
tuff_results <- list()

# For Apple
tuff_results$Apple_TUFF <- tuff_backtest(AppleReturns, apple_var
)

# For Amazon
tuff_results$Amazon_TUFF <- tuff_backtest(AmazonReturns, amazon_
var)

# For Netflix
tuff_results$Netflix_TUFF <- tuff_backtest(NetflixReturns,
netflix_var)

# For Google
tuff_results$Google_TUFF <- tuff_backtest(GoogleReturns, google_
var)

# Print the TUFF backtesting results
for (stock in names(tuff_results)) {

```



```

cat("TUFF-Backtest-for", stock, ":\n")
cat("TLT:", tuff_results[[stock]]$TLT, "\n")
cat("UCT:", tuff_results[[stock]]$UCT, "\n")
cat("FT:", tuff_results[[stock]]$FT, "\n")
cat("FR:", tuff_results[[stock]]$FR, "\n\n")
}

```

TUFF backtesting **for** Variance Covariance approach **for** last 500

```

observations
#TUFF backtesting
# Function to perform TUFF backtesting
# Slice the last 500 observations
actual_returns <- tail(actual_returns, 500)
var_estimates <- tail(var_estimates, 500)
tuff_backtest <- function(actual_returns, VaR_estimates) {
  # Calculate hits
  hits <- actual_returns < -VaR_estimates

  # Traffic Light Test
  tlt <- sum(hits) / length(actual_returns)

  # Unconditional Coverage Test
  uct <- ifelse(sum(actual_returns < -VaR_estimates) != 0,
               sum(hits) / sum(actual_returns < -VaR_estimates)
               ,
               NA)

  # Frequency Test
  ft <- length(hits[hits == TRUE]) / length(hits)

  # Failure Rate
  fr <- ifelse(sum(hits) != 0,

```

```

        length(hits[hits == TRUE]) / sum(hits),
        NA)

    return(list(TLT = tlt, UCT = uct, FT = ft, FR = fr))
}

# Perform TUFF backtesting for each stock
tuff_results <- list()

# For Apple
tuff_results$Apple_TUFF <- tuff_backtest(AppleReturns, apple_var
)

# For Amazon
tuff_results$Amazon_TUFF <- tuff_backtest(AmazonReturns, amazon_
var)

# For Netflix
tuff_results$Netflix_TUFF <- tuff_backtest(NetflixReturns,
netflix_var)

# For Google
tuff_results$Google_TUFF <- tuff_backtest(GoogleReturns, google_
var)

# Print the TUFF backtesting results
for (stock in names(tuff_results)) {
  cat("TUFF-Backtest-for", stock, ":\n")
  cat("TLT:", tuff_results[[stock]]$TLT, "\n")
  cat("UCT:", tuff_results[[stock]]$UCT, "\n")
  cat("FT:", tuff_results[[stock]]$FT, "\n")
  cat("FR:", tuff_results[[stock]]$FR, "\n\n")
}

```

TUFF backtesting **for** Monte Carlo Simulation

```
#TUFF backtesting
# Function to perform TUFF backtesting
tuff_backtest <- function(actual_returns , VaR_estimates) {
  # Calculate hits
  hits <- actual_returns < -VaR_estimates

  # Traffic Light Test
  tlt <- sum(hits) / length(actual_returns)

  # Unconditional Coverage Test
  uct <- ifelse(sum(actual_returns < -VaR_estimates) != 0,
               sum(hits) / sum(actual_returns < -VaR_estimates)
               ,
               NA)

  # Frequency Test
  ft <- length(hits[hits == TRUE]) / length(hits)

  # Failure Rate
  fr <- ifelse(sum(hits) != 0,
               length(hits[hits == TRUE]) / sum(hits) ,
               NA)

  return(list(TLT = tlt , UCT = uct , FT = ft , FR = fr))
}

# Perform TUFF backtesting for each stock and the portfolio
tuff_results <- list()
```

```

# For Apple
tuff_results$Apple_TUFF <- tuff_backtest(AppleReturns, VaR_apple
)

# For Amazon
tuff_results$Amazon_TUFF <- tuff_backtest(AmazonReturns, VaR_
amazon)

# For Netflix
tuff_results$Netflix_TUFF <- tuff_backtest(NetflixReturns, VaR_
netflix)

# For Google
tuff_results$Google_TUFF <- tuff_backtest(GoogleReturns, VaR_
google)

# For the Portfolio
tuff_results$Portfolio_TUFF <- tuff_backtest(portfolio_returns,
VaR_portfolio)

# Print the TUFF backtesting results
for (stock in names(tuff_results)) {
  cat("TUFF-Backtest for", stock, ":\n")
  cat("TLT:", tuff_results[[stock]]$TLT, "\n")
  cat("UCT:", tuff_results[[stock]]$UCT, "\n")
  cat("FT:", tuff_results[[stock]]$FT, "\n")
  cat("FR:", tuff_results[[stock]]$FR, "\n\n")
}

TUFF backtesting for Monte Carlo Simulation for last 500
observations
#TUFF backtesting
# Function to perform TUFF backtesting

```

```

# Slice the last 500 observations
actual_returns <- tail(actual_returns, 500)
var_estimates <- tail(var_estimates, 500)
tuff_backtest <- function(actual_returns, VaR_estimates) {
  # Calculate hits
  hits <- actual_returns < -VaR_estimates

  # Traffic Light Test
  tlt <- sum(hits) / length(actual_returns)

  # Unconditional Coverage Test
  uct <- ifelse(sum(actual_returns < -VaR_estimates) != 0,
               sum(hits) / sum(actual_returns < -VaR_estimates)
               ,
               NA)

  # Frequency Test
  ft <- length(hits[hits == TRUE]) / length(hits)

  # Failure Rate
  fr <- ifelse(sum(hits) != 0,
               length(hits[hits == TRUE]) / sum(hits),
               NA)

  return(list(TLT = tlt, UCT = uct, FT = ft, FR = fr))
}

# Perform TUFF backtesting for each stock and the portfolio
tuff_results <- list()

# For Apple
tuff_results$Apple_TUFF <- tuff_backtest(AppleReturns, VaR_apple
)
```

```

# For Amazon
tuff_results$Amazon_TUFF <- tuff_backtest(AmazonReturns, VaR_
    amazon)

# For Netflix
tuff_results$Netflix_TUFF <- tuff_backtest(NetflixReturns, VaR_
    netflix)

# For Google
tuff_results$Google_TUFF <- tuff_backtest(GoogleReturns, VaR_
    google)

# For the Portfolio
tuff_results$Portfolio_TUFF <- tuff_backtest(portfolio_returns,
    VaR_portfolio)

# Print the TUFF backtesting results
for (stock in names(tuff_results)) {
  cat("TUFF-Backtest-for", stock, ":\n")
  cat("TLT:", tuff_results[[stock]]$TLT, "\n")
  cat("UCT:", tuff_results[[stock]]$UCT, "\n")
  cat("FT:", tuff_results[[stock]]$FT, "\n")
  cat("FR:", tuff_results[[stock]]$FR, "\n\n")
}

TUFF backtesting on Extreme Value Theory
#TUFF backtesting

# Define the TUFF backtesting function
tuff_backtest <- function(actual_losses, predicted_losses,
    confidence_level) {

```

```

# Calculate the number of observations
n <- length(actual_losses)

# Calculate the number of exceedances
exceedances <- sum(actual_losses > predicted_losses)

# Calculate the number of non-exceedances
non_exceedances <- n - exceedances

# Calculate the test statistic
tuff_statistic <- exceedances - (1 - confidence_level) * n

# Calculate the p-value
p_value <- 1 - pnorm(tuff_statistic)

# Return the p-value
return(p_value)
}

# Perform TUFF backtesting for each EVT model
tuff_backtest_results <- list()

# For 95% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_95) && all(!is.na(evt_
models$AppleReturns_EVT_95$VaR)) &&
    is.vector(evt_models$AppleReturns_EVT_95$VaR)) {
  tuff_backtest_results$AppleReturns_EVT_95 <- tuff_backtest(
    AppleReturns, evt_models$AppleReturns_EVT_95$VaR, 0.95)
} else {
  tuff_backtest_results$AppleReturns_EVT_95 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_95) && all(!is.na(evt_

```

```

models$AmazonReturns_EVT_95$VaR)) &&
  is.vector(evt_models$AmazonReturns_EVT_95$VaR)) {
tuff_backtest_results$AmazonReturns_EVT_95 <- tuff_backtest(
  AmazonReturns, evt_models$AmazonReturns_EVT_95$VaR, 0.95)
} else {
tuff_backtest_results$AmazonReturns_EVT_95 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_95) && all(!is.na(evt_
models$NetflixReturns_EVT_95$VaR)) &&
  is.vector(evt_models$NetflixReturns_EVT_95$VaR)) {
tuff_backtest_results$NetflixReturns_EVT_95 <- tuff_backtest(
  NetflixReturns, evt_models$NetflixReturns_EVT_95$VaR, 0.95)
} else {
tuff_backtest_results$NetflixReturns_EVT_95 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_95) && all(!is.na(evt_
models$GoogleReturns_EVT_95$VaR)) &&
  is.vector(evt_models$GoogleReturns_EVT_95$VaR)) {
tuff_backtest_results$GoogleReturns_EVT_95 <- tuff_backtest(
  GoogleReturns, evt_models$GoogleReturns_EVT_95$VaR, 0.95)
} else {
tuff_backtest_results$GoogleReturns_EVT_95 <- NA
}

# For 97.5% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_975) && all(!is.na(evt_
models$AppleReturns_EVT_975$VaR)) &&
  is.vector(evt_models$AppleReturns_EVT_975$VaR)) {
tuff_backtest_results$AppleReturns_EVT_975 <- tuff_backtest(
  AppleReturns, evt_models$AppleReturns_EVT_975$VaR, 0.95)
} else {

```



```

tuff_backtest_results$AppleReturns_EVT_975 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_975) && all(!is.na(evt_
models$AmazonReturns_EVT_975$VaR)) &&
is.vector(evt_models$AmazonReturns_EVT_975$VaR)) {
tuff_backtest_results$AmazonReturns_EVT_975 <- tuff_backtest(
AmazonReturns, evt_models$AmazonReturns_EVT_975$VaR, 0.95)
} else {
tuff_backtest_results$AmazonReturns_EVT_975 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_975) && all(!is.na(evt_
models$NetflixReturns_EVT_975$VaR)) &&
is.vector(evt_models$NetflixReturns_EVT_975$VaR)) {
tuff_backtest_results$NetflixReturns_EVT_975 <- tuff_backtest(
NetflixReturns, evt_models$NetflixReturns_EVT_975$VaR,
0.95)
} else {
tuff_backtest_results$NetflixReturns_EVT_975 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_975) && all(!is.na(evt_
models$GoogleReturns_EVT_975$VaR)) &&
is.vector(evt_models$GoogleReturns_EVT_975$VaR)) {
tuff_backtest_results$GoogleReturns_EVT_975 <- tuff_backtest(
GoogleReturns, evt_models$GoogleReturns_EVT_975$VaR, 0.95)
} else {
tuff_backtest_results$GoogleReturns_EVT_975 <- NA
}

# For 99% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_99) && all(!is.na(evt_

```

```

models$AppleReturns_EVT_99$VaR)) &&
  is.vector(evt_models$AppleReturns_EVT_99$VaR)) {
tuff_backtest_results$AppleReturns_EVT_99 <- tuff_backtest(
  AppleReturns, evt_models$AppleReturns_EVT_99$VaR, 0.95)
} else {
tuff_backtest_results$AppleReturns_EVT_99 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_99) && all(!is.na(evt_
models$AmazonReturns_EVT_99$VaR)) &&
  is.vector(evt_models$AmazonReturns_EVT_99$VaR)) {
tuff_backtest_results$AmazonReturns_EVT_99 <- tuff_backtest(
  AmazonReturns, evt_models$AmazonReturns_EVT_99$VaR, 0.95)
} else {
tuff_backtest_results$AmazonReturns_EVT_99 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_99) && all(!is.na(evt_
models$NetflixReturns_EVT_99$VaR)) &&
  is.vector(evt_models$NetflixReturns_EVT_99$VaR)) {
tuff_backtest_results$NetflixReturns_EVT_99 <- tuff_backtest(
  NetflixReturns, evt_models$NetflixReturns_EVT_99$VaR, 0.95)
} else {
tuff_backtest_results$NetflixReturns_EVT_99 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_99) && all(!is.na(evt_
models$GoogleReturns_EVT_99$VaR)) &&
  is.vector(evt_models$GoogleReturns_EVT_99$VaR)) {
tuff_backtest_results$GoogleReturns_EVT_99 <- tuff_backtest(
  GoogleReturns, evt_models$GoogleReturns_EVT_99$VaR, 0.95)
} else {
tuff_backtest_results$GoogleReturns_EVT_99 <- NA
}

```

```

}

# Print the TUFF backtest results
tuff_backtest_results

TUFF backtesting on Extreme Value Theory for last 500
  observations

# Define the TUFF backtesting function
# Ensure the last 500 observations are considered

tuff_backtest <- function(actual_losses , predicted_losses ,
  confidence_level) {
  actual_losses <- tail(actual_losses , 500)
  predicted_losses <- tail(predicted_losses , 500)
  # Calculate the number of observations
  n <- length(actual_losses)

  # Calculate the number of exceedances
  exceedances <- sum(actual_losses > predicted_losses)

  # Calculate the number of non-exceedances
  non_exceedances <- n - exceedances

  # Calculate the test statistic
  tuff_statistic <- exceedances - (1 - confidence_level) * n

  # Calculate the p-value
  p_value <- 1 - pnorm(tuff_statistic)

  # Return the p-value
  return(p_value)
}

```

```

# Perform TUFF backtesting for each EVT model
tuff_backtest_results <- list()

# For 95% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_95) && all(!is.na(evt_
models$AppleReturns_EVT_95$VaR)) &&
    is.vector(evt_models$AppleReturns_EVT_95$VaR)) {
  tuff_backtest_results$AppleReturns_EVT_95 <- tuff_backtest(
    AppleReturns, evt_models$AppleReturns_EVT_95$VaR, 0.95)
} else {
  tuff_backtest_results$AppleReturns_EVT_95 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_95) && all(!is.na(evt_
models$AmazonReturns_EVT_95$VaR)) &&
    is.vector(evt_models$AmazonReturns_EVT_95$VaR)) {
  tuff_backtest_results$AmazonReturns_EVT_95 <- tuff_backtest(
    AmazonReturns, evt_models$AmazonReturns_EVT_95$VaR, 0.95)
} else {
  tuff_backtest_results$AmazonReturns_EVT_95 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_95) && all(!is.na(evt_
models$NetflixReturns_EVT_95$VaR)) &&
    is.vector(evt_models$NetflixReturns_EVT_95$VaR)) {
  tuff_backtest_results$NetflixReturns_EVT_95 <- tuff_backtest(
    NetflixReturns, evt_models$NetflixReturns_EVT_95$VaR, 0.95)
} else {
  tuff_backtest_results$NetflixReturns_EVT_95 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_95) && all(!is.na(evt_

```

```

models$GoogleReturns_EVT_95$VaR)) &&
  is.vector(evt_models$GoogleReturns_EVT_95$VaR)) {
tuff_backtest_results$GoogleReturns_EVT_95 <- tuff_backtest(
  GoogleReturns, evt_models$GoogleReturns_EVT_95$VaR, 0.95)
} else {
tuff_backtest_results$GoogleReturns_EVT_95 <- NA
}

# For 97.5% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_975) && all(!is.na(evt_
models$AppleReturns_EVT_975$VaR)) &&
  is.vector(evt_models$AppleReturns_EVT_975$VaR)) {
tuff_backtest_results$AppleReturns_EVT_975 <- tuff_backtest(
  AppleReturns, evt_models$AppleReturns_EVT_975$VaR, 0.975)
} else {
tuff_backtest_results$AppleReturns_EVT_975 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_975) && all(!is.na(evt_
models$AmazonReturns_EVT_975$VaR)) &&
  is.vector(evt_models$AmazonReturns_EVT_975$VaR)) {
tuff_backtest_results$AmazonReturns_EVT_975 <- tuff_backtest(
  AmazonReturns, evt_models$AmazonReturns_EVT_975$VaR, 0.975)
} else {
tuff_backtest_results$AmazonReturns_EVT_975 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_975) && all(!is.na(evt_
models$NetflixReturns_EVT_975$VaR)) &&
  is.vector(evt_models$NetflixReturns_EVT_975$VaR)) {
tuff_backtest_results$NetflixReturns_EVT_975 <- tuff_backtest(
  NetflixReturns, evt_models$NetflixReturns_EVT_975$VaR,
  0.975)
}

```

```

} else {
  tuff_backtest_results$NetflixReturns_EVT_975 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_975) && all(!is.na(evt_
  models$GoogleReturns_EVT_975$VaR)) &&
  is.vector(evt_models$GoogleReturns_EVT_975$VaR)) {
  tuff_backtest_results$GoogleReturns_EVT_975 <- tuff_backtest(
    GoogleReturns, evt_models$GoogleReturns_EVT_975$VaR, 0.975)
} else {
  tuff_backtest_results$GoogleReturns_EVT_975 <- NA
}

# For 99% threshold quantile
if (!is.na(evt_models$AppleReturns_EVT_99) && all(!is.na(evt_
  models$AppleReturns_EVT_99$VaR)) &&
  is.vector(evt_models$AppleReturns_EVT_99$VaR)) {
  tuff_backtest_results$AppleReturns_EVT_99 <- tuff_backtest(
    AppleReturns, evt_models$AppleReturns_EVT_99$VaR, 0.99)
} else {
  tuff_backtest_results$AppleReturns_EVT_99 <- NA
}

if (!is.na(evt_models$AmazonReturns_EVT_99) && all(!is.na(evt_
  models$AmazonReturns_EVT_99$VaR)) &&
  is.vector(evt_models$AmazonReturns_EVT_99$VaR)) {
  tuff_backtest_results$AmazonReturns_EVT_99 <- tuff_backtest(
    AmazonReturns, evt_models$AmazonReturns_EVT_99$VaR, 0.99)
} else {
  tuff_backtest_results$AmazonReturns_EVT_99 <- NA
}

if (!is.na(evt_models$NetflixReturns_EVT_99) && all(!is.na(evt_

```

```

models$NetflixReturns_EVT_99$VaR)) &&
  is.vector(evt_models$NetflixReturns_EVT_99$VaR)) {
tuff_backtest_results$NetflixReturns_EVT_99 <- tuff_backtest(
  NetflixReturns, evt_models$NetflixReturns_EVT_99$VaR, 0.99)
} else {
  tuff_backtest_results$NetflixReturns_EVT_99 <- NA
}

if (!is.na(evt_models$GoogleReturns_EVT_99) && all(!is.na(evt_
models$GoogleReturns_EVT_99$VaR)) &&
  is.vector(evt_models$GoogleReturns_EVT_99$VaR)) {
tuff_backtest_results$GoogleReturns_EVT_99 <- tuff_backtest(
  GoogleReturns, evt_models$GoogleReturns_EVT_99$VaR, 0.99)
} else {
  tuff_backtest_results$GoogleReturns_EVT_99 <- NA
}

# Print the TUFF backtest results
tuff_backtest_results

```