Financial Risk Management and Regulation

Term Project

Yushan Zhang, yz4739
Xiaojing Zhang, xz3285
Yiwei Chen, yc4209

December 18, 2023

Contents

1 Executive Summary

This report details the development of an advanced risk assessment system that integrates parametric, historical, and Monte Carlo methods for risk management and regulation. Our focus is on in-depth analysis and precise calculation of Value at Risk (VaR) and Expected Shortfall (ES), providing a comprehensive and accurate tool for risk management.

Combining theoretical knowledge with practical software design, this system is capable of handling complex datasets of stocks and options and simulating risk dynamics under various market conditions. The key features of the system include:

1. Take a portfolio of stock and option positions as input
2. Calculate parametric VaR and ES, specifically:
    - Geometric Brownian Motion (GBM) model.
    - Normal distribution model.
3. Calculate Historical VaR and ES
    - Calculate with absolute changes for VaR and ES.
    - Calculate with relative changes for VaR and ES.
4. Calculate of Monte Carlo VaR and ES .
5. Backtest analysis of VaR results against history.

Our goal with this system is to merge theory and practice, providing an efficient, multidimensional solution for risk management and regulation in complex market environments.

2 Model Documentation

2.1 Introduction of model

This model is designed with a central focus on computing VaR (Value at Risk) and ES (Expected Shortfall) values, leveraging input portfolio data that includes both stock and options positions. A key feature of the system is its utilization of parametric data for precise calibration against historical market data. In addition to these risk metrics, the model also generates parametric, historical, and Monte Carlo VaR and ES values. By integrating these outcomes, investors can leverage the system to derive relevant risk indicators and values for making informed risk assessments in dynamic financial environments.

In the subsections, we will provide a detailed exploration of pertinent concepts and assumptions. This will be followed by an elucidation of the risk measurement system and calculation methods, encompassing Parametric VaR/ES, Historical VaR/ES, and Monte Carlo VaR/ES. Throughout this explanation, we will introduce our assumptions, mathematical equations, and the formula derivation process. This comprehensive presentation is aimed at assisting readers in grasping both our model's design philosophy and computational process.

2.2 General Value at Risk and Expected Shortfall

2.2.1 Value at Risk(VaR)

VaR [1] is a risk measurement technique based on statistical analysis. The p percentile VaR is X means that p percent of the time, losses will be less than or equal to X.

The general formula for VaR[1] is given by:

$$\text{VaR}(V, T, p) = G^{-1}(p) \tag{1}$$

where $G(X) = P[V_0 - V_T \leq X] = E[P(1_{V_0 - V_T \leq X})]$

2.2.2 Expected Shortfall (ES)

Expected Shortfall[2], also known as Conditional Value at Risk (C-VaR), is a financial risk measurement. Expected Shortfall refers to the expected value of losses over a period of

time T, given that the losses exceed the Xth percentile. Expected Shortfall provides an estimate of the average loss when adverse market conditions trigger losses. It provides information about the average loss when adversities occur.

The general formula for ES[2] is given by:

$$\begin{aligned} \mathrm{ES}(V, T, p) &= -\mathbb{E}_P \left[V_T - V_0 \,|\, V_T - V_0 < -\mathrm{VaR}(V_T, p) \right] \\ &= \mathbb{E}_P \left[V_0 - V_T \,|\, V_0 - V_T > \mathrm{VaR}(V_T, p) \right] \end{aligned} \tag{2}$$

In the following section, we will utilize three methods—parametric, historical, and Monte Carlo—to compute the corresponding Value at Risk (VaR) and Expected Shortfall (ES).

2.3 Parametric Value at Risk and Expected Shortfall

Parametric [8]Value at Risk (VaR) stands as a pivotal methodology within our risk management framework, offering a streamlined approach to estimate potential portfolio losses based on statistical parameters. It simplifies assumptions to derive a formula relying on approximate mean and variance calculations. Particularly effective in scenarios where positions and payoffs are linear, akin to a simple variance calculation, this methodology plays a crucial role in quantifying and managing portfolio risk.

Parametric Expected Shortfall: is a crucial method in finance for risk measurement and asset pricing. By parameterizing the distribution of asset losses, often assuming log-normal or normal distributions, this approach provides an estimate of the average loss at a given confidence level. Unlike traditional Value at Risk (VaR) methods, Parametric Expected Shortfall focuses on the average loss beyond the VaR level, offering a more comprehensive view of risk. This method allows investors to more accurately assess potential risks, supporting informed investment decisions.

2.3.1 Model Assumptions

Our model explores two parametric VaR and ES calculation approaches under specific assumptions:

GBM Assumption: One scenario assumes the entire portfolio follows GBM.

Normal Distribution Assumption: Another scenario assumes that individual stocks follow Geometric Brownian Motion (GBM), while the entire portfolio adheres to a normal distribution.

2.3.2 Formula Derivations for Parametric VaR and ES

The calculation method involves determining the mean (expected value) and standard deviation of the portfolio's returns over a specified look-back period. By leveraging probability theory, the approach analyzes historical price movements to estimate the maximum potential loss.

2.3.2.1 Entire Portfolio follows GBM[3]

In Long Portfolios

VaR [9]:

First, we will go through the computation of μ and σ:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \ln\left(\frac{S_i}{S_{i-1}}\right)$$

$$\sigma = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\mu_i - \hat{\mu})^2}}{\sqrt{dt}}$$

$$\mu = \frac{\hat{\mu}}{dt} + \frac{1}{2}\sigma^2$$

when the entire portfolio follows Geometric Brownian Motion (GBM), we have

$$S_T = S_0 e^{(\mu - \frac{\sigma^2}{2})T + \sigma W_T}$$

The probability of the stock price falling below a level X in this scenario is given by[4]:

$$\begin{aligned} & P(S_T < X) \\ & \quad = P\left[\log(S_T) < \log(X)\right] \\ & \quad = P\left[\log(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)T + \sigma W_T < \log(X)\right] \\ & \quad = P\left[W_T < \frac{\log\left(\frac{X}{S_0}\right) - \left(\mu - \frac{\sigma^2}{2}\right)T}{\sigma}\right] \end{aligned}$$

Therefore, VaR Formula:

$$\mathrm{VaR}(S, T, p) = S_0 - S_0 e^{\sigma\sqrt{T}\Phi^{-1}(1-p) + \left(\mu - \frac{\sigma^2}{2}\right)T} \tag{3}$$

ES[5]:
First, We derive the expected shortfall (ES) for a long position based on a parametric Value at Risk (VaR) model.

The Expected Shortfall at a certain time T and for a given confidence level p is defined as the negative expected value of the loss beyond the VaR threshold:

$$ES(S, T, p) = S_0 - \mathbb{E}[S_T \,|\, S_T < S_0 - VaR(S, T, p)]$$

where S_0 is the initial price The conditional expectation of S_T given that S_T is less than X is represented by:

$$E^P[S_T \mid S_T < X] = \frac{\int_{-\infty}^{X} S_T \, dP}{\int_{-\infty}^{X} dP}$$

Where, dP is the differential of the probability measure, and X is the VaR threshold.

The integral of $S_T \, dP$ from negative infinity to X is transformed into an integral with respect to a new variable Y, which satisfies a relationship involving the initial price S_0, expected return μ, volatility σ, and the Brownian motion W_T:

$$\int_{-\infty}^{X} S_T \, dP = \frac{1}{\sqrt{2\pi T}} \int_{-\infty}^{Y} S_0 e^{(\mu - \frac{\sigma^2}{2})T + \sigma W_T} e^{-\frac{W_T^2}{2T}} \, dW_T$$

By completing the square in the exponent of the integr and and solving for Y, substitute

$X = S_0 - VaR(S, T, p)$

Note that the denominator $\int_{-\infty}^{X} dP$ becomes $1-p$ when X is the VaR level, which normalizes the conditional expectation.

By evaluating the integral and performing the necessary algebraic manipulations, the final formula for Expected Shortfall is obtained:

$$ES = S_0 - \frac{1}{1-p} \cdot S_0 \cdot e^{(\mu \cdot T) \cdot \Phi(\Phi^{-1}(1-p) - \mu \cdot \sqrt{T})} \tag{4}$$

Here, $\Phi(\Phi^{-1}(p) - \sigma \cdot \sqrt{T})$ is related to the quantile corresponding to the chosen confidence level for VaR.

In Short Portfolios

VaR:

First, we will go through the computation of μ and σ:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \ln \left(\frac{S_i}{S_{i-1}} \right)$$

$$\sigma = \frac{\sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (\mu_i - \hat{\mu})^2}}{\sqrt{dt}}$$

$$\mu = \frac{\hat{\mu}}{dt} + \frac{1}{2}\sigma^2$$

when the entire portfolio follows Geometric Brownian Motion (GBM), we have

$$S = S_0 e^{(\mu - \frac{\sigma^2}{2})T + \sigma W_T}$$

The probability of the stock price falling above a level X in this scenario is given by:

$$\begin{aligned}
&P(S_T > X) \\
&\quad = P\left[\log(S_T) > \log(X)\right] \\
&\quad = P\left[\log(S_0) + \left(\mu - \frac{\sigma^2}{2}\right) T + \sigma W_T > \log(X)\right] \\
&\quad = P\left[W_T > \frac{\log\left(\frac{X}{S_0}\right) - \left(\mu - \frac{\sigma^2}{2}\right) T}{\sigma}\right]
\end{aligned}$$

Then, we use same calculation method as above. Therefore, VaR Formula:

$$\text{VaR}(S,T,p) = S_0 e^{\sigma\sqrt{T}\Phi^{-1}(p)+\left(\mu-\frac{\sigma^2}{2}\right)T} - S_0 \tag{5}$$

2.3.2.2 Entire Portfolio follows to a Normal Distribution, individual stocks follow GBM

In Long Portfolios:

VaR[9]: :

We consider a portfolio consisting of two assets, whose value at time t is given by a linear combination of the assets:

$$V_t = aS_{1,t} + bS_{2,t}$$

where a and b are the weights of the assets in the portfolio. Because the assets $S_{i,t}$ follow a Geometric Brownian Motion (GBM):

$$dS_i = \mu_i S_i dt + \sigma_i S_i dW_i$$

where μ_i represents the drift, σ_i represents the volatility, and dW_i represents the increment of a Wiener process. The correlation between the two Wiener processes is captured by:

$$dW_1 dW_2 = \rho dt$$

where ρ is the correlation coefficient. The logarithmic returns over a time period $[t_1, t_2]$ are:

$$\log\left(\frac{S_1(t_2)}{S_1(t_1)}\right) = \left(\mu_1 - \frac{\sigma_1^2}{2}\right)(t_2 - t_1) + \sigma_1(W_1(t_2) - W_1(t_1)),$$
$$\log\left(\frac{S_2(t_2)}{S_2(t_1)}\right) = \left(\mu_2 - \frac{\sigma_2^2}{2}\right)(t_2 - t_1) + \sigma_2(W_2(t_2) - W_2(t_1)).$$

Let R_1 and R_2 be the logarithmic returns:

$$R_1 = \log\left(\frac{S_1(t_2)}{S_1(t_1)}\right),$$
$$R_2 = \log\left(\frac{S_2(t_2)}{S_2(t_1)}\right).$$

The expected returns $E[R_1]$ and $E[R_2]$ are:

$$E[R_1] = \left(\mu_1 - \frac{\sigma_1^2}{2}\right)(t_2 - t_1),$$
$$E[R_2] = \left(\mu_2 - \frac{\sigma_2^2}{2}\right)(t_2 - t_1).$$

The expected value of the product of the deviations of the two returns from their expected values is:

$$\begin{aligned} E[(R_1 - E[R_1])(R_2 - E[R_2])] &= E[\sigma_1(W_1(t_2) - W_1(t_1))\sigma_2(W_2(t_2) - W_2(t_1))] \\ &= \sigma_1\sigma_2\rho(t_2 - t_1). \end{aligned}$$

The correlation coefficient ρ is therefore:

$$\rho = \frac{E[(R_1 - E[R_1])(R_2 - E[R_2])]}{\sigma_1\sigma_2(t_2 - t_1)}.$$

Assuming V_t is normally distributed, we can determine the distribution by its mean and variance. The expected value of V_t is:

$$E[V_t] = aE[S_{1,t}] + bE[S_{2,t}]$$

and the variance of V_t is:

$$\mathrm{var}[V_t] = a^2\mathrm{var}[S_{1,t}] + b^2\mathrm{var}[S_{2,t}] + 2ab\,\mathrm{cov}[S_{1,t}, S_{2,t}]$$

The expected value of the product of the two stochastic asset prices is:

$$E[S_{1,t}S_{2,t}] = S_{1,0}S_{2,0}e^{(\mu_1+\mu_2-\frac{1}{2}(\sigma_1^2+\sigma_2^2))t+\rho\sigma_1\sigma_2 t}$$

The expected value of the product of two correlated Wiener processes is:

$$E[W_{1,t}W_{2,t}] = \rho t$$

the Value at Risk for the portfolio is calculated as:

$$\mathrm{VaR}(V) = V_0 - \left(E[V_t] - \Phi^{-1}(1-p)\cdot\sqrt{\mathrm{var}[V_t]}\right) \tag{6}$$

where V_0 is the initial value of the portfolio, $E[V_t]$ is the expected value of the portfolio at time t, $\Phi^{-1}(1-p)$ is the inverse cumulative distribution function of the standard normal distribution at confidence level p, and $\sqrt{\text{var}[V_t]}$ is the standard deviation of the portfolio's value at time t.

ES:
The Expected Shortfall at confidence level α for a loss random variable X is the expected value of X given that X has exceeded the Value at Risk (VaR) at level p:

$$ES_\alpha(X) = E[X \mid X \geq VaR_\alpha(X)]$$

The Value at Risk $VaR_\alpha(X)$ for a normally distributed variable X at the confidence level α is given by:

$$VaR_\alpha(X) = -\sigma q_\alpha$$

where q_α is the upper 100α-th percentile of the standard normal distribution. The ES is calculated as the negative average of the tail distribution beyond the VaR threshold. For a normal distribution, this becomes an integral from $VaR_\alpha(X)$to infinity. Then Substituting $x = \sigma t$ and solving the integral, we obtain:

$$ES_\alpha(X) = -\frac{1}{\alpha\sigma\sqrt{2\pi}} \int_{VaR_\alpha(t)}^{\infty} xe^{-\frac{t^2}{2\sigma^2}}\, dt$$

The final formula for the Expected Shortfall at the confidence level α is[17]:

$$ES_\alpha(X) = \frac{\sigma}{\alpha}\left(\frac{1}{\sqrt{2\pi}} e^{-\frac{q_\alpha^2}{2}}\right) \tag{7}$$

2.4 Historical VaR and ES

Historical method [6]: also known as historical simulation or historical method, is a method for calculating VaR and ES based on historical market data. The fundamental idea is to estimate the risk of a portfolio by utilizing the price or return changes observed in historical data. In this approach, it is assumed that risk factors follow the actual historical distributions, meaning the distribution of market changes today equals the historical distribution of market changes. For each day in the historical dataset, the change for that day is applied to the current day.

2.4.1 Model Selection

Regarding the choice of the model, the following principles[7]: are generally considered:

- If a risk factor is believed to have constant absolute volatility (ABM), absolute changes are deemed more representative.
- If a risk factor is thought to have constant relative volatility (GBM), relative changes are considered more representative.
- Absolute changes are typically used for spreads in the data, while relative changes are applied to interest rates and prices in the data.

Note: The results of the historical simulation method are highly dependent on the selected historical data interval. Different intervals may lead to different VaR values. Therefore, when selecting historical data, one needs to consider the representativeness of the data and changes in market conditions.

2.4.2 Formula Derivations

In the following section, we provide a detailed explanation of how to calculate Historical Value-at-Risk under relative and absolute changes.

2.4.2.1 Relative Changes:

In long portfolios:
VaR:
1. Determine the Time Span (`t`):
Choose a lookback period `t`, representing the number of days over which you will calculate the log returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Log Returns of the Daily Portfolio Value:
For each day in your data, calculate the log return for a long position using the formula:

$$\text{log\_return}_t = \ln\left(\frac{V_T}{V_{T-t}}\right)$$

Here, V_T is the portfolio value on the current day, and V_{T-t} is the portfolio value t days before.

3. Calculate the loss:

$$\text{loss} = S_0 - \log\_ \text{return}_t * S_0$$

4. Rank the Loss:

Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.

5. Identify the Percentile Value for VaR:

Find the log return at the desired percentile, such as the 1th percentile for a 99% confidence level. This log return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level.

ES:

1. Determine the Time Span (`t`):

Choose a lookback period `t`, representing the number of days over which you will calculate the log returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Log Returns of the Daily Portfolio Value:

For each day in your data, calculate the log return for a long position using the formula:

$$\text{log\_return}_t = \ln\left(\frac{V_T}{V_{T-t}}\right)$$

Here, V_T is the portfolio value on the current day, and V_{T-t} is the portfolio value t days before.

3. Calculate the loss:

$$\text{loss} = S_0 - \log\_ \text{return}_t * S_0$$

4. Rank the Loss:

Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.

5. Identify the Percentile Value for VaR:
Find the log return at the desired percentile, such as the 1th percentile for a 99% confidence level. This log return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level.

6.Use the sorted historical sample to calculate ES by averaging the losses that exceed the VaR threshold. The calculation is given by:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

where the losses are calculated as the difference between the initial portfolio value (V_0) and the historical values exceeding the VaR.

In Short Portfolios
VaR:
1. Determine the Time Span (`t`):
Choose a lookback period `t`, representing the number of days over which you will calculate the log returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis. 2. Calculate the Log Returns of the Daily Portfolio Value:
For each day in your data, calculate the log return for a short position using the formula:

$$\text{log\_return}_t = -\ln\left(\frac{V_T}{V_{T-t}}\right)$$

Here, V_T is the portfolio value on the current day, and V_{T-t} is the portfolio value t days before.

3. Calculate the loss:

$$\text{loss } = S_0 - \text{log\_ return}_t * S_0$$

4. Rank the Loss:
Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.

5. Identify the Percentile Value for VaR:
Find the log return at the desired percentile, such as the 1th percentile for a 99% confidence

level. This log return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level.

ES:
1. Determine the Time Span (`t`):
Choose a lookback period `t`, representing the number of days over which you will calculate the log returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Log Returns of the Daily Portfolio Value:
For each day in your data, calculate the log return for a short position using the formula:

$$\text{log\_return}_t = -\ln\left(\frac{V_T}{V_{T-t}}\right)$$

Here, V_T is the portfolio value on the current day, and V_{T-t} is the portfolio value t days before. 3. Calculate the loss:

$$\text{loss } = -S_0 + \text{log\_ return}_t * S_0$$

3. Rank the loss:
Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.
4. Identify the Percentile Value for VaR:
Find the log return at the desired percentile, such as the 1th percentile for a 99% confidence level. This log return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level. 5.Use the sorted historical sample to calculate ES by averaging the losses that exceed the VaR threshold. The calculation is given by:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

where the losses are calculated as the difference between the initial portfolio value (V_0) and the historical values exceeding the VaR.

2.4.2.2 Absolute Changes:

In long position:

VaR:

1. Determine the Time Span (`t`):

Choose a lookback period `t`, representing the number of days over which you will calculate the absolute returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Absolute Changes of the Daily Portfolio Value:

Calculate the absolute change, which is the difference between the current and the previous day's portfolio value:

$$\text{abs\_return}_t = V_t - V_{t-1}$$

Here, V_t is the portfolio value on the current day, and V_{t-1} is the portfolio value on the previous day. 3. Calculate the loss:

$$\text{loss } = S_0 - \text{abs\_return}_t * S_0$$

4. Rank the Loss:

Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.

5. Identify the Percentile Value for VaR:

Find the absolute return at the desired percentile, such as the 1th percentile for a 99% confidence level. This absolute return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level.

ES:

1. Determine the Time Span (`t`):

Choose a lookback period `t`, representing the number of days over which you will calculate the absolute returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Absolute Changes of the Daily Portfolio Value:

Calculate the absolute change, which is the difference between the current and the previous day's portfolio value:

$$\text{abs\_return}_t = V_t - V_{t-1}$$

Here, V_t is the portfolio value on the current day, and V_{t-1} is the portfolio value on the previous day. 3. Calculate the loss:

$$\text{loss } = S_0 - \text{abs\_return}_t * S_0$$

4. Rank the Loss:
Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.
5. Identify the Percentile Value for VaR:
Find the absolute return at the desired percentile, such as the 1th percentile for a 99% confidence level. This absolute return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level. 6.Use the sorted historical sample to calculate ES by averaging the losses that exceed the VaR threshold. The calculation is given by:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

where the losses are calculated as the difference between the initial portfolio value (V_0) and the historical values exceeding the VaR.

In Short position:
VaR:
1. Determine the Time Span (`t`):
Choose a lookback period `t`, representing the number of days over which you will calculate the absolute returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Absolute Changes of the Daily Portfolio Value:

Calculate the absolute change, which is the difference between the current and the previous day's portfolio value:

$$\text{abs\_return}_t = V_{t-1} - V_t$$

Here, V_t is the portfolio value on the current day, and V_{t-1} is the portfolio value on the previous day. 3. Calculate the loss:

$$\text{loss } = S_0 - \text{abs\_return}_t * S_0$$

4. Rank the Loss:
Sort these loss in ascending order. This sorting will help in determining the percentile value

for VaR.
5. Identify the Percentile Value for VaR:
Find the absolute return at the desired percentile, such as the 1th percentile for a 99% confidence level. This absolute return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level.

ES:
1. Determine the Time Span (`t`):
Choose a lookback period `t`, representing the number of days over which you will calculate the absolute returns. This period should be sufficiently long to capture market volatility and the trends relevant to your analysis.

2. Calculate the Absolute Changes of the Daily Portfolio Value:
Calculate the absolute change, which is the difference between the current and the previous day's portfolio value:

$$\text{abs\_return}_t = V_{t-1} - V_t$$

Here, V_t is the portfolio value on the current day, and V_{t-1} is the portfolio value on the previous day.
3. Calculate the loss:

$$\text{loss } = S_0 - \text{abs\_return}_t * S_0$$

4. Rank the Loss:
Sort these loss in ascending order. This sorting will help in determining the percentile value for VaR.
5. Identify the Percentile Value for VaR:
Find the absolute return at the desired percentile, such as the 1th percentile for a 99% confidence level. This absolute return represents the VaR, indicating the maximum expected loss over the specified period at a 99% confidence level. 6. Use the sorted historical sample to calculate ES by averaging the losses that exceed the VaR threshold. The calculation is given by:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

where the losses are calculated as the difference between the initial portfolio value (V_0) and the historical values exceeding the VaR.

2.5 Monte Carlo VaR and ES

The Monte Carlo method[11], a VAR calculation approach based on stochastic simulation, plays a pivotal role in our project, directly utilizing asset prices for risk assessment. Noteworthy characteristics of this method include its intuitive computation, ability to accommodate nonlinear payoffs, slower convergence for achieving high accuracy, and equivalence to parametric VaR under specific conditions. In our project, Monte Carlo VaR serves as a robust and flexible tool, generating random paths to simulate future price movements of various financial instruments. This flexibility is crucial for a comprehensive risk assessment within our portfolio.

2.5.1 Formula Derivations

2.5.1.1 Distinguishing Long and Short Portfolios

Long Portfolio: For long positions, the loss calculation is based on the change between the initial portfolio value and the future portfolio value. The loss formula is $V_0 - V_t$, where V_t is the future portfolio value.

Short Portfolio: For short positions, the loss calculation is based on the change between the future portfolio value and the initial portfolio value. The loss formula is $V_t - V_0$, where V_t is the future portfolio value.

These distinctions in loss calculation are crucial, and the appropriate method should be selected based on the characteristics of the portfolio.

In long portfolios:
VaR:
1.Generate Monte Carlo Simulations:
For each simulation, generate forecasted asset prices using the Geometric Brownian Motion model. The formula for a single asset's price at time t is given by[16]:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}$$

where S_t represents the asset price at time t, S_0 is the initial price, μ is the drift rate, σ is the volatility, and W_t is the standard Brownian motion.

2.Calculate Portfolio Value for Each Simulation:
If the portfolio consists of multiple assets, calculate its value based on the simulated prices of these assets. For a portfolio with two assets, the value at time t can be calculated as:

$$\begin{aligned} V_t &= aS_{1,t} + bS_{2,t} \\ &= aS_{1,0}e^{(\mu_1 - \frac{\sigma_1^2}{2})t + \sigma_1 W_{1,t}} + bS_{2,0}e^{(\mu_2 - \frac{\sigma_2^2}{2})t + \sigma_2 W_{2,t}} \end{aligned}$$

Here, a and b are the weights or the number of units of each asset in the portfolio, and $S_{1,t}$, $S_{2,t}$ are the prices of the two assets at time t.
3. Calculate Loss for Each Path:
For long positions, the loss calculation is based on the change between the initial portfolio value and the future portfolio value. The loss formula is $V_0 - V_t$, where V_t is the future portfolio value. Calculate the loss for each simulated portfolio path using this formula.
4. Rank the Losses and Identify VaR:
Rank the calculated losses and identify the VaR at a specific percentile (P). For a 99% VaR, find the loss at the 1th percentile of the ranked losses.

ES:
1.Generate Monte Carlo Simulations:
For each simulation, generate forecasted asset prices using the Geometric Brownian Motion model. The formula for a single asset's price at time t is given by:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}$$

where S_t represents the asset price at time t, S_0 is the initial price, μ is the drift rate, σ is the volatility, and W_t is the standard Brownian motion.
2.Calculate Portfolio Value for Each Simulation:
If the portfolio consists of multiple assets, calculate its value based on the simulated prices of these assets. For a portfolio with two assets, the value at time t can be calculated as:

$$V_t = aS_{1,t} + bS_{2,t} = aS_{1,0}e^{(\mu_1 - \frac{\sigma_1^2}{2})t + \sigma_1 W_{1,t}} + bS_{2,0}e^{(\mu_2 - \frac{\sigma_2^2}{2})t + \sigma_2 W_{2,t}}$$

$$= aS_{1,0}e^{(\mu_1 - \frac{\sigma_1^2}{2})t + \sigma_1 W_{1,t}} + bS_{2,0}e^{(\mu_2 - \frac{\sigma_2^2}{2})t + \sigma_2 W_{2,t}}$$

Here, a and b are the weights or the number of units of each asset in the portfolio, and $S_{1,t}$, $S_{2,t}$ are the prices of the two assets at time t.
3. Calculate Loss for Each Path:

For long positions, the loss calculation is based on the change between the initial portfolio value and the future portfolio value. The loss formula is $V_0 - V_t$, where V_t is the future portfolio value. Calculate the loss for each simulated portfolio path using this formula
4. Rank the Losses and Identify VaR:
Rank the calculated losses and identify the VaR at a specific percentile (P). For a 99% VaR, find the loss at the 1th percentile of the ranked losses.
5.Calculate Expected Shortfall(ES):
Once the losses are ranked, calculate the Expected Shortfall (ES). ES is the average of the losses that are worse than the VaR. This can be calculated as:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

The ES provides a measure of the average loss in the worst-case scenarios, exceeding the VaR threshold.

In Short portfolios:
VaR:
1.Generate Monte Carlo Simulations:
For each simulation, generate forecasted asset prices using the Geometric Brownian Motion model. The formula for a single asset's price at time t is given by:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}$$

where S_t represents the asset price at time t, S_0 is the initial price, μ is the drift rate, σ is the volatility, and W_t is the standard Brownian motion.
2.Calculate Portfolio Value for Each Simulation:
If the portfolio consists of multiple assets, calculate its value based on the simulated prices of these assets. For a portfolio with two assets, the value at time t can be calculated as:

$$V_t = aS_{1,t} + bS_{2,t}$$

$$= aS_{1,0}e^{(\mu_1 - \frac{\sigma_1^2}{2})t + \sigma_1 W_{1,t}} + bS_{2,0}e^{(\mu_2 - \frac{\sigma_2^2}{2})t + \sigma_2 W_{2,t}}$$

Here, a and b are the weights or the number of units of each asset in the portfolio, and $S_{1,t}$, $S_{2,t}$ are the prices of the two assets at time t.
3. Calculate Loss for Each Path:

For short positions, the loss calculation is based on the change between the initial portfolio value and the future portfolio value. The loss formula is $V_t - V_0$, where V_t is the future portfolio value. Calculate the loss for each simulated portfolio path using this formula.
4. Rank the Losses and Identify VaR:
Rank the calculated losses and identify the VaR at a specific percentile (P). For a 99% VaR, find the loss at the 1th percentile of the ranked losses.

ES:
1.Generate Monte Carlo Simulations:
For each simulation, generate forecasted asset prices using the Geometric Brownian Motion model. The formula for a single asset's price at time t is given by:

$$S_t = S_0 e^{(\mu - \frac{\sigma^2}{2})t + \sigma W_t}$$

where S_t represents the asset price at time t, S_0 is the initial price, μ is the drift rate, σ is the volatility, and W_t is the standard Brownian motion.
2.Calculate Portfolio Value for Each Simulation:
If the portfolio consists of multiple assets, calculate its value based on the simulated prices of these assets. For a portfolio with two assets, the value at time t can be calculated as:

$$V_t = aS_{1,t} + bS_{2,t}$$

$$= aS_{1,0} e^{(\mu_1 - \frac{\sigma_1^2}{2})t + \sigma_1 W_{1,t}} + bS_{2,0} e^{(\mu_2 - \frac{\sigma_2^2}{2})t + \sigma_2 W_{2,t}}$$

Here, a and b are the weights or the number of units of each asset in the portfolio, and $S_{1,t}$, $S_{2,t}$ are the prices of the two assets at time t.
3. Calculate Loss for Each Path:
For short positions, the loss calculation is based on the change between the initial portfolio value and the future portfolio value. The loss formula is $V_t - V_0$, where V_t is the future portfolio value. Calculate the loss for each simulated portfolio path using this formula.
4. Rank the Losses and Identify VaR:
Rank the calculated losses and identify the VaR at a specific percentile (P). For a 99% VaR, find the loss at the 1th percentile of the ranked losses. 5.Calculate Expected Shortfall(ES):
Once the losses are ranked, calculate the Expected Shortfall (ES). ES is the average of the losses that are worse than the VaR. This can be calculated as:

$$\text{ES} = \frac{\sum(\text{losses beyond VaR})}{\text{number of losses beyond VaR}}$$

The ES provides a measure of the average loss in the worst-case scenarios, exceeding the VaR threshold.

3 Software Design Documentation

In this chapter, we begin by outlining the overall software design procedure in section 3.1. Following that, in section 3.2, we provide a detailed description of all the methods and the definition of the equations used in the analysis.

3.1 General Procedure

3.1.1 Package import

The packages we need to import include pandas, numpy, scipy.stats, scipy.integrate, matplotlib.pyplot, math.

3.1.2 Stock Data Preprocessing and Portfolio Dataframe Creation

First, we read csv files and preprocess stock price data and implied volatility. The csv file we take in contain stock prices and implied volatility of put / call option from oldest date to newest date with column name 'PX_LAST', '12MO_CALL_IMP_VOL', '12MO_PUT_IMP_VOL'. Then create a portfolio dataframe and read stock data for individual tickers into a dictionary.

We compute log returns for the closing prices and update the dataframe accordingly. In the case of multiple stocks, a new combined dataframe is generated, encompassing both stocks and options positions. Subsequently, we got the list of portfolio price.

3.1.3 Calculate mean and volatility of close prices

We use two methods to calculate the mean and volatility of portfolio.

3.1.3.1 Moving Average Method

The moving average method is calculated by adding up all the data points during a specific period and dividing the sum by the number of time periods. In the project, we are using the 5-year moving windows and 252 days a year. We simply define the `calculate_mu_sigma` in the RCS class. This method would calculate the estimated μ and σ (μ_{bar} and σ_{bar}) of the log return of the portfolio and finally get mean and volatility.

3.1.3.2 Exponential Weighted Method

The exponential weighted method is designed as such that older observations are given lower weights. The weights fall exponentially as the data point gets older. We simply define the `calculate_exp_mu_sigma` in RCS class. This method would be based on the lambda value. This lambda corresponds roughly to periods because the weight becomes 1/2 after approximately several days. This method would calculate the estimated μ and σ (μ_{bar} and σ_{bar}) of the log return of the portfolio and finally get mean and volatility.

3.1.4 Calculating VaR/ES

We determine the portfolio's value at risk (VaR) and expected shortfall (ES) using three methods: the parametric method (portfolio follows GBM and Normal distribution), historical method, and Monte Carlo method. Various parameters are essential for this calculation, such as the time periods, the specified percentage for VaR or ES, and the computed drift and volatility. After calculating, we define `plot_VaR_ES` to plot VaR and ES in our dataframe and compare them.

3.1.4.1 Parametric VaR/ES

We define `para_normal_dis_long_VaR_ES` for the long position of stock assuming the entire portfolio follows a normal distribution. Also, assuming the entire portfolio follows GBM, we define `para_long_VaR_ES` for the long position of stock and `para_short_VaR_ES` for the short position of the stock. This method assumes the input individual stocks have the Geometric Brownian Motion and calculates the portfolio VaR/ES according to this assumption. The calculated VaR and ES will be added into dataframe.

3.1.4.2 Historical VaR/ES

When computing historical VaR/ES, we delve into the distinctions between calculating VaR/ES using relative change and absolute change. For GBM, using relative changes. We define `hist_long_VaR_ES` for the long position of stock and `hist_short_VaR_ES` for short position of the stock. We use the estimated drift values and volatility values which using 5-year windows to calculate the 5-day historical VaR/ES. The calculated VaR and ES will be added into dataframe.

For ABM, using absolute changes. We define `abs_hist_long_VaR_ES` for the long position of stock and `abs_hist_short_VaR_ES` for short position of the stock. We use the estimated

drift values and volatility values which using 5-year windows to calculate the 5-day historical VaR/ES. The calculated VaR and ES will be added into dataframe.

3.1.4.3 Monte Carlo VaR/ES

We define `long MC_long_VaR_ES` for the long position of the stock and `MC_short_VaR_ES` for the short position of the stock. In this process, we generate a Monte Carlo sample of stock prices called `MC_sample`. We use the estimated drift values and volatility values which using 5-year windows to calculate the 5-day Monte Carlo VaR/ES. The calculated VaR and ES will be added into dataframe.

3.1.5 Portfolio with call/put option VaR

We also calculate the current VaR of our portfolio when add some options in it (call/put options). First, we need information about options, including option type (call/put), initial price, strike price, time to expiration, risk-free rate, dividend yield in the market to define black_scholes and get the option price. Then, we use Monte Carlo method to calculate VaR.

For put option, we generate a Monte Carlo sample called `mc_put_sample` with some stocks and put options in our portfolio. Then, we define `MC_long_put_VaR` to calculate Monte Carlo VaR for a long position with a put option and `MC_short_put_VaR` to calculate Monte Carlo VaR for a short position with a put option.

The procedure for call options mirrors that of put options, involving the creation of a Monte Carlo sample (`mc_call_sample`) and the definition of functions (`MC_long_call_VaR` and `MC_short_call_VaR`) to determine Monte Carlo VaR for long and short positions, respectively.

3.1.6 VaR Backtest

For parametric VaR, historical VaR and Monte Carlo VaR, we test the VaR estimates for long and short portfolios by counting the number of times the VaR on each date is exceeded by the subsequent 5-day change in each 1-year window. Do this for long and short position with two different series of drift and volatility (Moving Average Method and Exponential Weighted Method).

We define `long_VaR_backtest` and `short_VaR_backtest` for long and short portfolios, re-

spectively, to calculate daily return, VaR and the number of days that the loss is greater than VaR on each day.

3.2 Description of software design

The software is encoded into a class `Risk_Calculation_System` (called RCS below). It is used to achieve the above models. This section gives a brief introduction to all the methods this class has.

3.2.1 RCS.__init__

Initialize the Risk Calculation System with given parameters. Specially, we will not define parameters in other software because we have define them in this equation. When testing, we don not need to enter these parameters for several times.
- S0: Initial investment amount
- T: Time horizon in years
- p_var: Confidence level for Value at Risk (VaR)
- p_es: Confidence level for Expected Shortfall (ES)
- l: Lookback period for historical methods
- M: Number of Monte Carlo simulations
- Stocks: List of tuples containing file paths and positions for individual stocks in the portfolio

3.2.2 RCS.preprocess_data

Preprocess stock data and create a portfolio dataframe.
- Stocks: List of tuples containing file paths and positions for individual stocks in the portfolio

3.2.3 RCS.read_stocks_data

Read stock data for individual tickers into a dictionary.
- Stocks: List of tuples containing file paths and positions for individual stocks in the portfolio

3.2.4 RCS.calculate_mu_sigma

Calculate the estimated drift[13] and volatility[14] using rolling window.
- yr: Lookback period in years

3.2.5 RCS.calculate_exp_mu_sigma

Calculate the estimated drift and volatility using exponential weighting[15].
- lmbda: Lambda value for exponential weighting
- stock_df: DataFrame containing stock price data

3.2.6 RCS.plot_mu_sigma

Plot the estimated drift and volatility in the same plot.
- mu: Series containing estimated drift values
- sigma: Series containing estimated volatility values

3.2.7 RCS.para_normal_dis_long_VaR_ES

Calculate Parametric VaR and ES for a long position assuming normal distribution.
- lmbda: Lambda value for exponential weighting

3.2.8 RCS.para_long_VaR_ES

Calculate Parametric VaR and ES for a long position.
- mu: Series of estimated drift values
- sigma: Series of estimated volatility values

3.2.9 RCS.para_short_VaR_ES

Calculate Parametric VaR and ES for a short position.
- mu: Series of estimated drift values
- sigma: Series of estimated volatility values

3.2.10 RCS.historical_sample

Extract historical sample for a given time point.
- l: Lookback period in years
- i: Index for the time point

3.2.11 RCS.hist_long_VaR_ES

Calculate Historical VaR and ES for a long position.
- mu: Series of estimated drift values
- sigma: Series of estimated volatility values

3.2.12 RCS.hist_short_VaR_ES

Calculate Historical VaR and ES for a short position.
- mu: Series of estimated drift values
- sigma: Series of estimated volatility values

3.2.13 RCS.abs_hist_long_VaR_ES

Calculate Historical VaR and ES for a long position assuming absolute change.
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.14 RCS.abs_hist_short_VaR_ES

Calculate Historical VaR and ES for a short position assuming absolute change.
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.15 RCS.MC_sample

Generate a Monte Carlo sample of stock prices.
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

- M: Number of Monte Carlo simulations
- i: Index for the time point

3.2.16 RCS.MC_long_VaR_ES

Calculate Monte Carlo VaR and ES for a long position.
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.17 RCS.MC_short_VaR_ES

Calculate Monte Carlo VaR and ES for a short position.
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.18 RCS.plot_VaR_ES

Plot VaR and ES values for a specific risk calculation type and trading position.
- Type: Risk calculation type ('Parametric', 'Historical', or 'Monte Carlo')
- Trade: Trading position ('long' or 'short') - drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.19 RCS.black_scholes

Calculate the Black-Scholes option pricing formula.
- call_put: Option type ('c' for call or 'p' for put)
- S: Current stock price
- K: Option strike price
- r: Risk-free rate
- q: Dividend yield
- sigma: Volatility of the underlying stock
- T: Time to expiration (in years)

3.2.20 RCS.mc_put_sample

Generate a Monte Carlo sample for a put option.
- df: Dataframe for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values
- i: Index for the time point

3.2.21 RCS.MC_long_put_VaR

Calculate Monte Carlo VaR for a long position with a put option.
- kicker: Ticker for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.22 RCS.MC_short_put_VaR

Calculate Monte Carlo VaR for a short position with a put option.
- kicker: Ticker for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.23 RCS.mc_call_sample

Generate a Monte Carlo sample for a call option.
- df: Dataframe for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values
- i: Index for the time point

3.2.24 RCS.MC_long_call_VaR

Calculate Monte Carlo VaR for a long position with a call option.
- kicker: Ticker for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.25 RCS.MC_short_call_VaR

Calculate Monte Carlo VaR for a short position with a call option.
- kicker: Ticker for the stock data
- drift: Series of estimated drift values
- volatility: Series of estimated volatility values

3.2.26 RCS.long_VaR_backtest

Perform a backtest on long VaR.
- long_VaR: Series of long VaR values
- Type: Risk calculation type ('Parametric', 'Historical', or 'Monte Carlo')

3.2.27 RCS.short_VaR_backtest

Perform a backtest on short VaR.
- short_VaR: Series of short VaR values
- Type: Risk calculation type ('Parametric', 'Historical', or 'Monte Carlo')

4 Test Plan

4.1 Introduction

In this chapter, we show test plan on our financial risk management system by dividing into some sections below. For each testing, we will test on both long portfolio and short portfolio.

1. Creating an instance of `Risk_Calculation_System`
2. Preprocessing data and displaying the portfolio
3. Calculating μ and σ using the moving average method or exponential weighting method
4. Test: Parametric VaR / ES assuming normal distribution or GBM with moving average method or exponential weighting method
5. Test: Historical VaR / ES using relative change or absolute change
6. Test: Monte Carlo VaR / ES with moving average method or exponential weighting method
7. Test: Backtest Parametric VaR assuming normal distribution or GBM with moving average method or exponential weighting method
8. Test: Backtest Historical VaR using relative change or absolute change
9. Test: Backtest Monte Carlo VaR with moving average method or exponential weighting method
10. Test: Monte Carlo VaR with put/call option

4.2 Test Data

4.2.1 Test Data Introduction

We install stock data from Bloomberg for testing, focusing on The Coca-Cola Company (KO) and Netflix, Inc. (NFLX). The dataset spans from October 22, 2004, to December 8, 2023. On December 9, 2009, we invest $5000 in KO (173 shares), $5000 in NFLX (625 shares) in the portfolio. So the initial price for portfolio is $10000. KO will be specifically used for call/put options analysis.

4.2.2 Calculating μ and σ using the moving average method and exponential weighting method

We estimate μ and σ by two method. One is using moving window of 5 years , the other one is using the exponential weighting method with $\lambda = 0.9989003714$ by software and plot the drift and volatility of portfolio.

4.2.3 Parametric VaR / ES assuming normal distribution or GBM with moving average method or exponential weighting method

First, the assumption is that the entire portfolio is following normal distribution.

We test data and calculate VaR/ES by using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. 5-day 99% VaR and 97.5% ES for long position would be calculated.

Then, the assumption is that the entire portfolio is following Geometric Brownian Motion.

We test data and calculate VaR/ES by two methods: using moving window of 5 years or using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. 5-day 99% VaR and 97.5% ES for long and short position would be calculated.

4.2.4 Historical VaR / ES using relative change or absolute change

We assess the performance of historical VaR/ES by log returns. The calculation involves determining the 5-day 99% VaR and 97.5% ES for both long and short positions, consistently applying the moving average method with a 5-year window. We employ both relative or absolute change in our calculations to compare.

4.2.5 Monte Carlo VaR / ES with moving average method or exponential weighting method

We assess the performance of Monte Carlo VaR/ES by two methods: using moving window of 5 years or using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. The assumption here is that the portfolio is following Geometric Brownian Motion. The simple size we use on the test is 10000. 5-day 99% VaR and 97.5% ES for both long and short positions would be calculated.

4.2.6 Backtest Parametric VaR assuming normal distribution or GBM with moving average method or exponential weighting method

We plan to assess the efficacy of our parametric VaR backtest function, employing daily portfolio returns.

First, the parametric VaR relies on the assumption of a normal distribution and exponential-weighted drift and volatility. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

Then, the parametric VaR calculated by assuming that the entire portfolio follows GBM and using 5 years moving window or using exponential weight $\lambda = 0.9989003714$ and to estimate drift and volatility. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data. We backtest by counting the number of times the VaR on each date is exceeded by the subsequent 5 day change in each 1 year window.

4.2.7 Backtest Historical VaR using relative change or absolute change

We plan to assess the efficacy of our historical VaR backtest[12] function, employing daily portfolio returns. The parametric VaR calculated by using exponential weight $\lambda = 0.9989003714$ to estimate drift and volatility and using both relative change or absolute change. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

4.2.8 Backtest Monte Carlo VaR with moving average method or exponential weighting method

We plan to assess the efficacy of our Monte Carlo VaR backtest function, employing daily portfolio returns. The Monte Carlo VaR calculated by using 5 years moving window or using exponential weight $\lambda = 0.9989003714$ to estimate drift and volatility. The assumption here is that the portfolio is following Geometric Brownian Motion. The simple size we use on the test is 10000. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

4.2.9 Monte Carlo VaR with put/call option

We test our Monte Carlo Value at Risk (VaR) implementation for the European put/call option using actual market data for both the option and the underlying stocks in the portfolio. The put/call option constitutes 1% of the portfolio, with the remaining 99% allocated to stocks. Employing the current risk-free rate of 5%, time to expiration of 1 year and no dividend, we derive the implied volatility from the available option data. Our Monte Carlo sample size is 10,000. Then we can determine the 5-day 99% VaR for both long and short positions on the nearest trading day.

5 Test Result

5.1 Test data

We first creating an instance of `Risk_Calculation_System`. Then read two datasets and create a portfolio that contains 173 shares of KO and 625 shares of NFLX. We create a system for this portfolio and get the log returns using closing price 'PX_LAST'. Figure 1 shows how to we preprocess the test data.

```
In [45]: # Test: Creating an instance of Risk_Calculation_System
         # On 2009-12-9, $5000 in KO (173 shares), $5000 in NFLX (625 shares)
         file_path = [['KO.csv', 173], ['NFLX.csv', 625]]
         rms_instance = Risk_Calculation_System(10000, 5/252, 0.99, 0.975, 5, 10000, file_path)

In [46]: # Test: Preprocessing data and displaying the portfolio
         portfolio = rms_instance.preprocess_data(file_path)
         print("Processed Portfolio:")
         print(portfolio)

Processed Portfolio:
               PX_LAST  LogReturn
Dates
2004-10-22     875.000        NaN
2004-10-25     891.875   0.019102
2004-10-26     858.125  -0.038576
2004-10-27     864.375   0.007257
2004-10-28     856.250  -0.009444
...                ...        ...
2023-12-04  283687.500  -0.025751
2023-12-05  284468.750   0.002750
2023-12-06  279206.250  -0.018673
2023-12-07  282500.000   0.011728
2023-12-08  283600.000   0.003886

[4991 rows x 2 columns]
```

Figure 1: Read test data

5.2 Calculating μ and σ

Before testsing VaR/ES, we calculating μ and σ using two methods: moving average method (5 years moving window) and exponential weighting method($\lambda = 0.9989003714$). Figure 2 and Figure 3 show the magnitude and trends of drift and volatility by two methods.

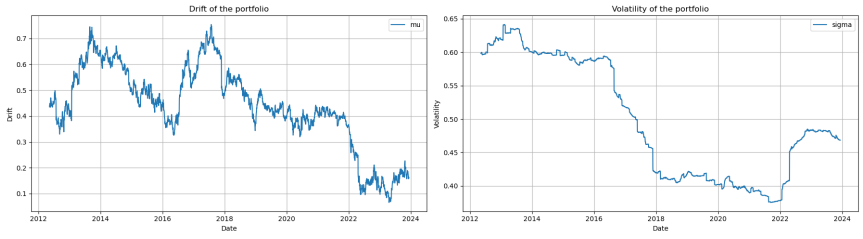

Figure 2: Moving Average Drift and Volatility

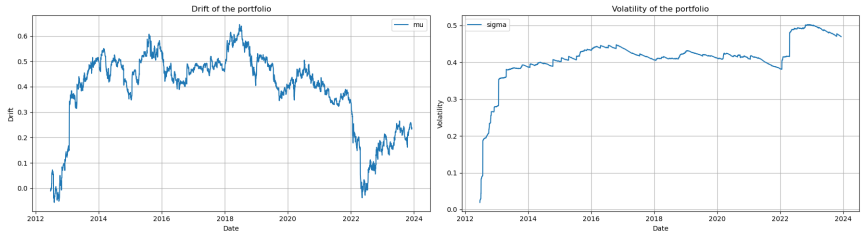

Figure 3: Exponential Weighted drift and Volatility

Observations:

The drift estimates are much more noisy than the volatility estimates.

Different choices of calculating method substantially different estimates for the drift and volatility.

There are large change in the drift and volatility in this portfolio using both methods.

5.3 Parametric VaR / ES

First, the assumption is that the entire portfolio is following normal distribution.

We test data and calculate VaR/ES by using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. 5-day 99% VaR and 97.5% ES for long position would be calculated and plotted in Figure 4.

Then, the assumption is that the entire portfolio is following Geometric Brownian Motion.

We test data and calculate VaR/ES by two methods: using moving window of 5 years or using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. 5-day 99% VaR and 97.5% ES for long and short position would be calculated.

Figure 5 and Figure 6 shows parametric VaR/ES assuming the entire portfolio follows GBM for long/short position using moving average drift and volatility.

Figure 7 and Figure 8 shows parametric VaR/ES assuming the entire portfolio follows GBM for long/short position using exponential weighted drift and volatility.

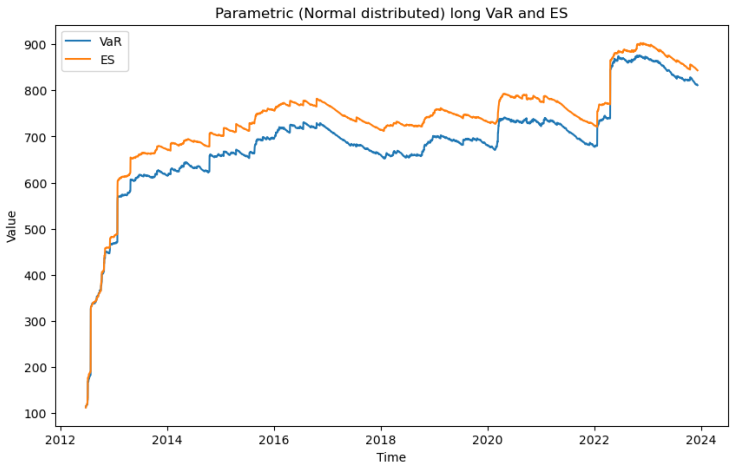

Figure 4: Parametric VaR and ES for long position(Normal distribution)

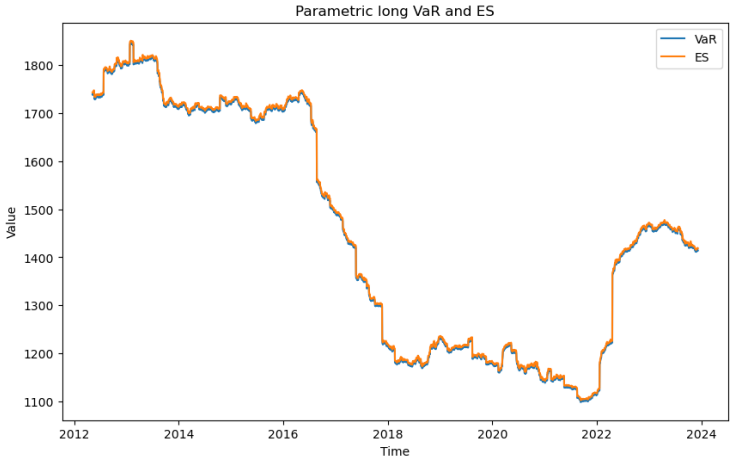

Figure 5: Parametric VaR and ES for long position using moving average drift and volatility

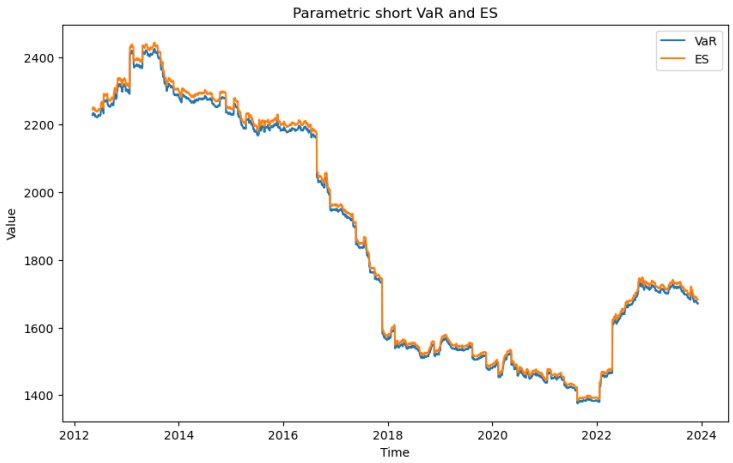

Figure 6: Parametric VaR and ES for short position using moving average drift and volatility

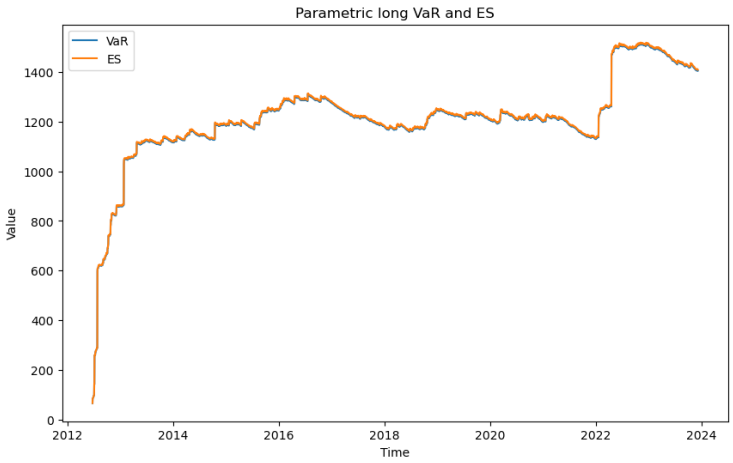

Figure 7: Parametric VaR/ES for long Position using exponential weighted drift and volatility

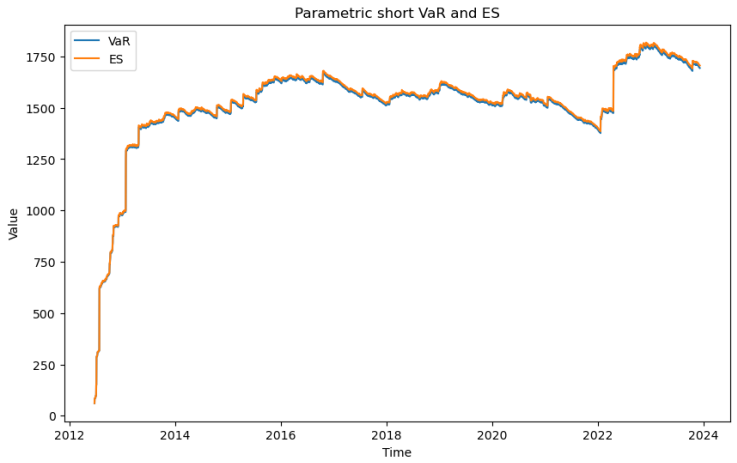

Figure 8: Parametric VaR/ES for short Position using exponential weighted drift and volatility

Observations:

In Figure 4, 5, 6, 7, 8, 97.5% ES is virtually identical to 99% VaR for both the normal case and GBM case. This observation holds true for both long and short positions, regardless of the method employed to calculate drift and volatility. Notely, in the normal case, 97.5% ES is slightly larger than 99% VaR for normal case.

In Figure 5, 6, 7, 8, when comparing the two methods for calculating drift and volatility, equivalent exponential weighting rolls in changes in VaR/ES faster and smoother than windowing does.

In Figure 5, 6, 7, 8, when comparing long and short position, the short positions have substantially larger downside, and hence larger VaRs/ESes than the long portfolio. It suggests that for most of the times, investing the short portfolio would incur greater loss.

In Figure 4 and 7, the GBM and normal based VaR/ES calculations for long position using exponential weighted drift and volatility are quite similar.

5.4 Historical VaR / ES

We assess the performance of historical VaR/ES by log returns. The calculation involves determining the 5-day 99% VaR and 97.5% ES for both long and short positions, consistently applying the moving average method with a 5-year window. We employ both relative or absolute change in our calculations to compare.

Figure 9 and Figure 10 shows historical VaR/ES for long/short position using relative change.

Figure 11 and Figure 12 shows historical VaR/ES for long/short position using absolute change.

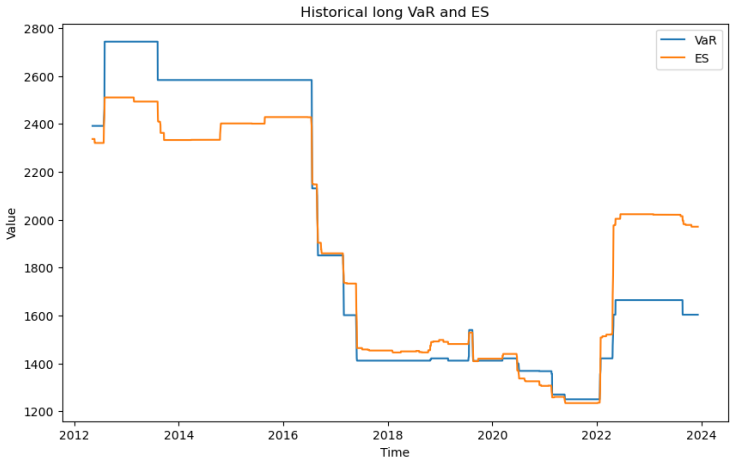

Figure 9: Historical VaR / ES for long position with relative change

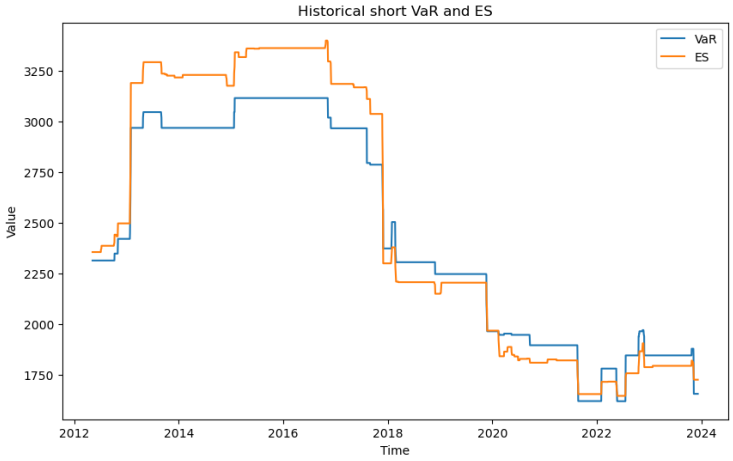

Figure 10: Historical VaR / ES for short position relative change

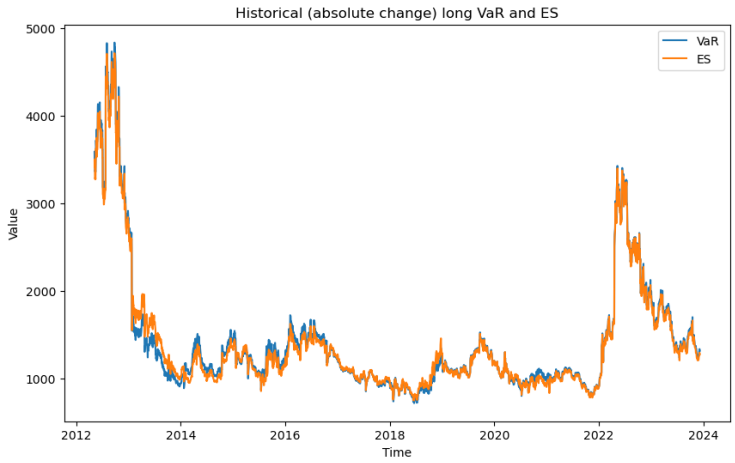

Figure 11: Historical VaR / ES for long position with absolute change

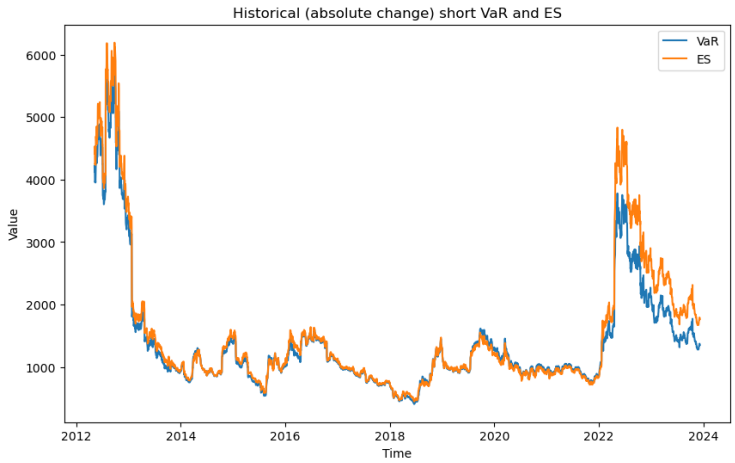

Figure 12: Historical VaR / ES for short position with absolute change

Observations:

In Figure 9 and 10, when using relative change , there are extended periods of flat historical VaRs and ESes. Notably, a spike in VaR occurs whenever an extreme scenario enters or exits the sliding window.

Back to Figure 5 and 6, the historical VaRs are mostly higher than the parametric VaRs, signaling that using GBM for VaR calculations tends to underestimate risks, particularly in accordance with the historical distribution.

In Figure 9 and 10, historical VaR and ES often exhibit proximity but also substantial

differences. Notably, the 99% VaR and the 97.5% ES align closely for both GBM and Normal VaR, indicating a departure from both normal and lognormal distributions in the historical data.

In Figure 11 and 12, when using absolute change, the historical VaR and ES for the portfolio are very different from using relative change. Rapid fluctuations occur due to the fixed value of the stock, leading to day-to-day variability as absolute stock price changes affect a different number of shares each day.

In Figure 9, 10, 11, 12, when comparing long and short position, the short positions have substantially larger downside, and hence larger VaRs/ESs than the long portfolio. It suggests that for most of the times, investing the short portfolio would incur greater loss.

5.5 Monte Carlo VaR / ES

We assess the performance of Monte Carlo VaR/ES by two methods: using moving window of 5 years or using the exponential weighting method with $\lambda = 0.9989003714$ to estimate drift and volatility of log returns. The assumption here is that the portfolio is following Geometric Brownian Motion. The sample size we use on the test is 10000. 5-day 99% VaR and 97.5% ES for both long and short positions would be calculated.

Figure 13 and Figure 14 shows Monte Carlo VaR/ES for long/short position using moving average drift and volatility.

Figure 15 and Figure 16 shows Monte Carlo VaR/ES for long/short position using exponential weighted drift and volatility.

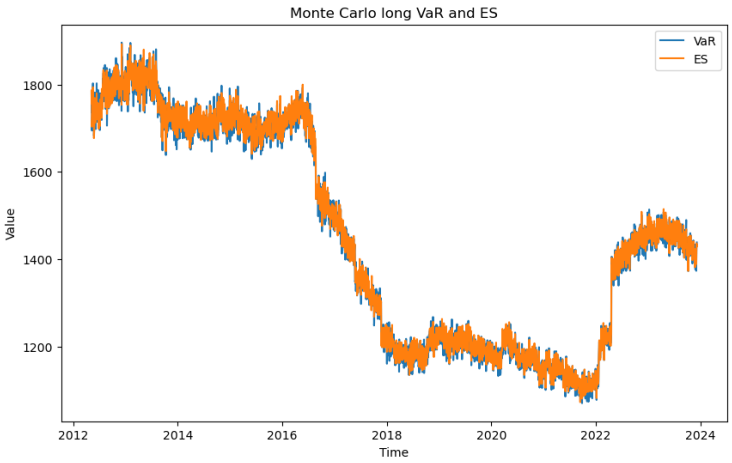

Figure 13: Monte Carlo VaR / ES for long position with moving average method

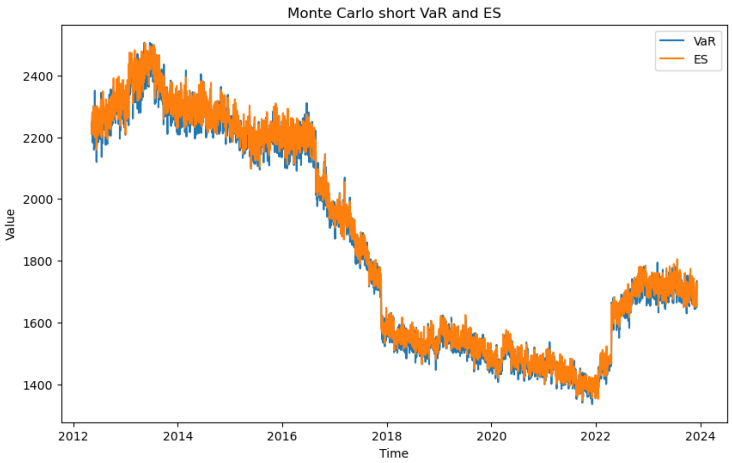

Figure 14: Monte Carlo VaR / ES for short position with exponential weighting method

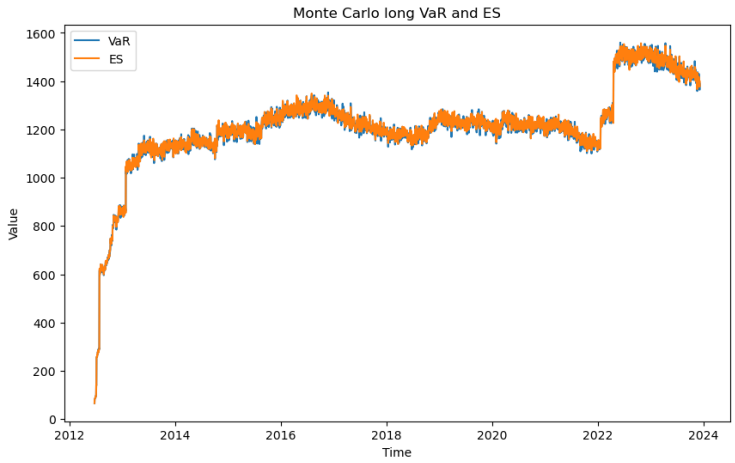

Figure 15: Monte Carlo VaR / ES for long position with exponential weighting method

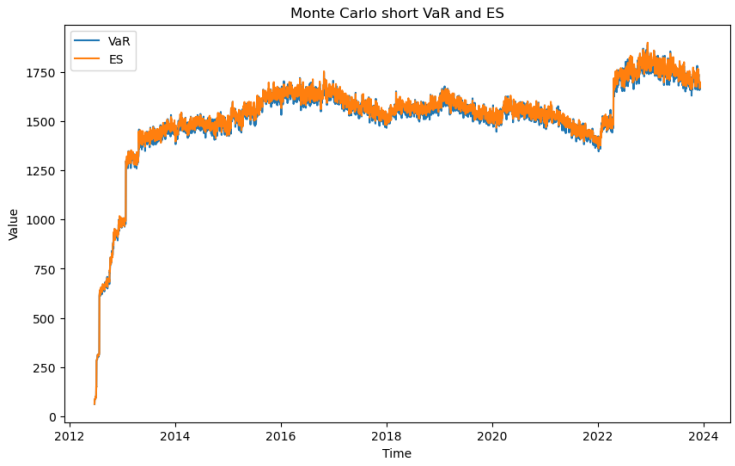

Figure 16: Monte Carlo VaR / ES for short position with exponential weighting method

Observations:

Comparing Figure 13, 14, 15, 16 (Monte Carlo VaR/ES) and Figure 5, 6, 7, 8 (Parametric GBM VaR/ES), the Monte Carlo VaR and ES sing the portfolio GBM parameters matches the results from parametric GBM VaR/ES for both two method of calculating drift and volatility. But they are noisier and takes longer to run. This is because of model validation of the two different sets of software.

In Figure 13, 14, 15, 16, Monte Carlo VaRs and ESes have very similar results, although we could find that the VaRs look slightly noisier.

In Figure 13, 14, 15, 16, when comparing long and short position, the short positions have substantially larger downside, and hence larger VaRs/ESes than the long portfolio. It suggests that for most of the times, investing the short portfolio would incur greater loss.

5.6 Backtest Parametric VaR

We assess the efficacy of our parametric VaR backtest function, employing daily portfolio returns. In the backtest, we count the number of the times VaR on each dates is exceeded by the subsequent 1 day change in each one year window.

Figure 17 shows that the parametric VaR relies on the assumption of a normal distribution and exponential-weighted drift and volatility. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

Figure 18, 19, 20, 21, the parametric VaR calculated by assuming that the entire portfolio follows GBM and using 5 years moving window (Figure 18 and 19) or using exponential weight $\lambda = 0.9989003714$ (Figure 20 and 21) and to estimate drift and volatility. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

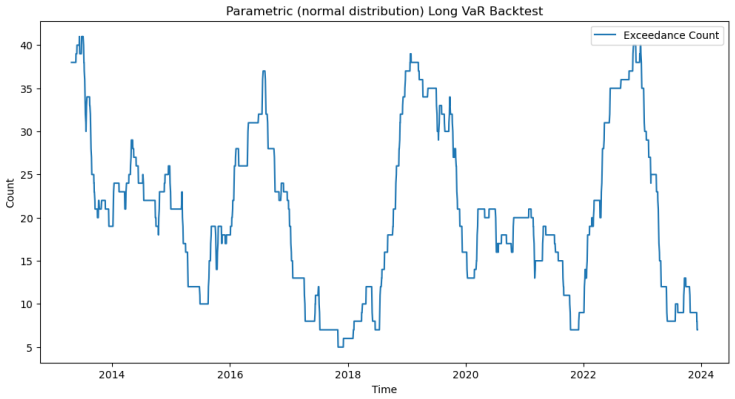

Figure 17: Parametric VaR for long position (normal distribution) Backtest

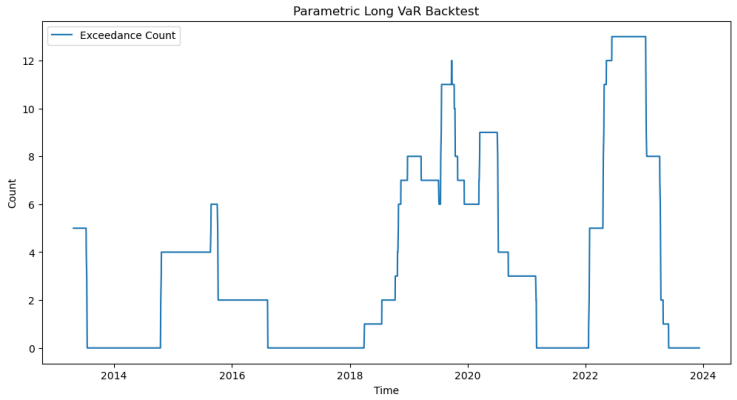

Figure 18: Parametric VaR for long position with moving average method Backtest

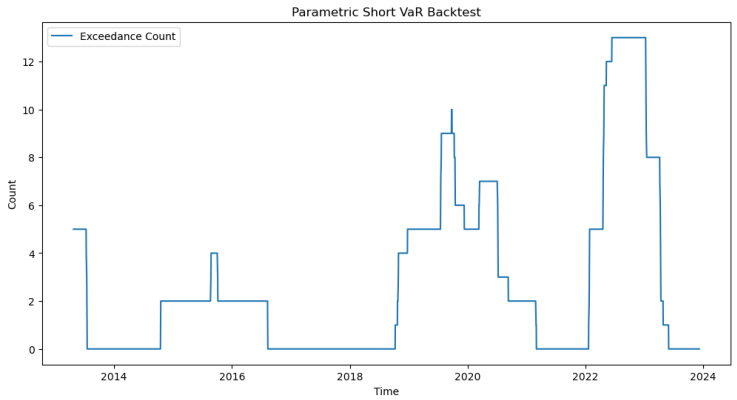

Figure 19: Parametric VaR for short position with moving average method Backtest

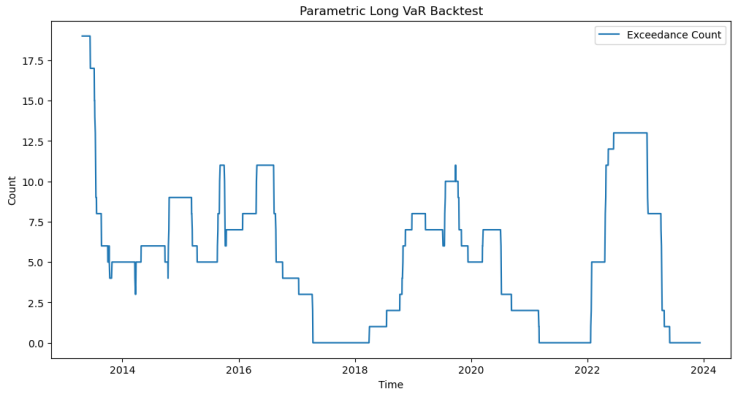

Figure 20: Parametric VaR for long position with exponential weighting method Backtest

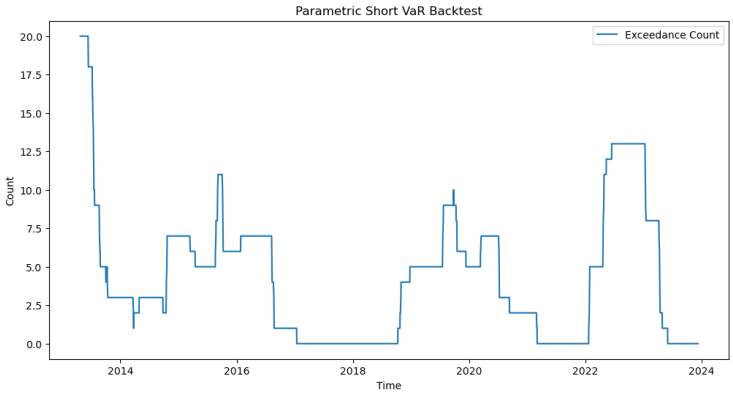

Figure 21: Parametric VaR for short position with exponential weighting method Backtest

Observations:

With a 99% VaR, we expect an exception 1% of the time, which translates to $(1 - 0.99) \times 252 = 2.52$ per year.

In Figure 17, under normal distribution, most of the values are larger than 2.52. We conclude that parametric VaR under normal distribution is not a good choice in calculating VaR for long portfolio.

In Figure 18 and 19, parametric GBM VaR using moving average method for long/short portfolio backtest plots look similarly. According to the plots, from 2014 to 2023, at most of the time the results match the expected number of exceptionsm except 2016, 2020 and 2022, which means they are smaller or equal to the expected number.

In Figure 20 and 21, parametric GBM VaR using exponential weighting method for long/short portfolio backtest plots look similarly. Most of the values are larger than 2.52 except 2018 and 2022. We conclude that parametric VaR under normal distribution is not a good choice in calculating VaR for long portfolio.

So, for long/short position, assuming the entire portfolio follows GBM and using moving average method for calculating drift and volatility tend to give the best results when calculating parametric VaR.

5.7 Backtest Historical VaR

We assess the efficacy of our historical VaR backtest function, employing daily portfolio returns. The parametric VaR calculated by using exponential weight $\lambda = 0.9989003714$ to

estimate drift and volatility and using both relative change or absolute change. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

Figure 22, 23, 24, 25, the historical VaR using moving average method for calculating drift and volatility and using relative change (Figure 22 and 23) or using absolute change (Figure 24 and 25). During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

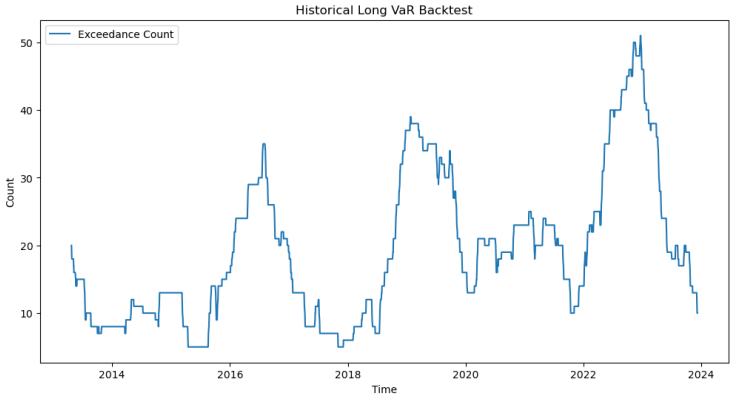

Figure 22: Historical VaR for long position with moving average method Backtest (relative change)

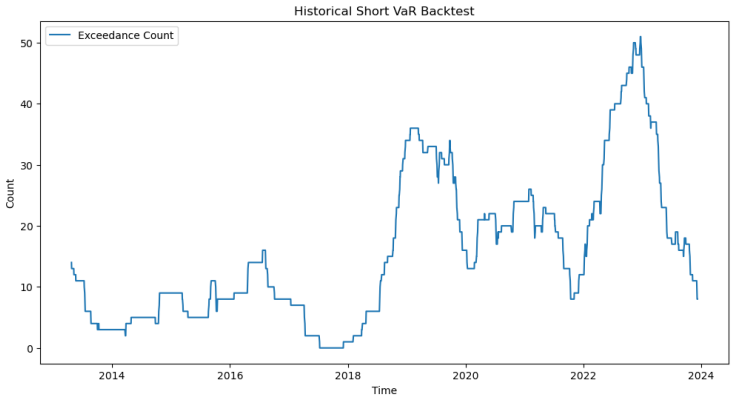

Figure 23: Historical VaR for short position with moving average method Backtest (relative change

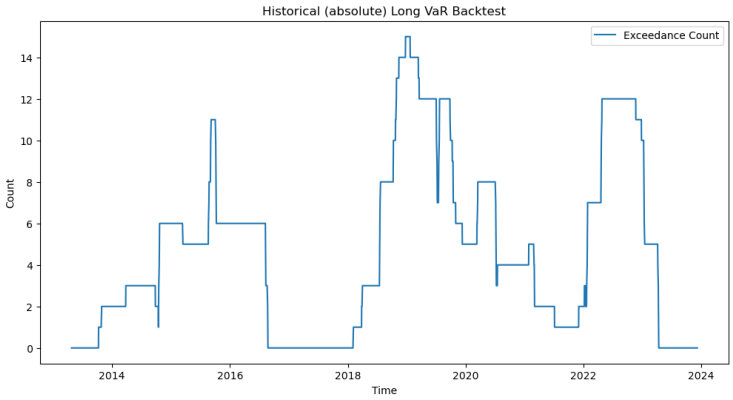

Figure 24: Historical VaR for long position with moving average method Backtest (absolute change)

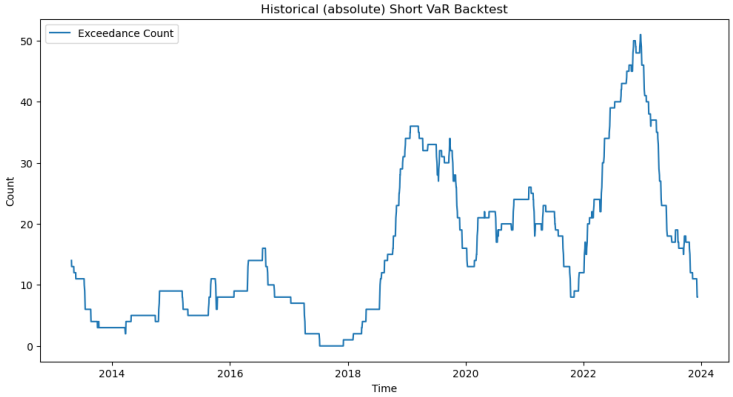

Figure 25: Historical VaR for short position with moving average method Backtest (absolute change)

Observations:

With a 99% VaR, we expect an exception 1% of the time, which translates to $(1 - 0.99) \times 252 = 2.52$ per year.

In Figure 22, 23, 24, 25, most of the values are larger than 2.52. We conclude that historical VaR is not a good choice in calculating VaR for long/short portfolio.

5.8 Backtest Monte Carlo VaR

We assess the efficacy of our Monte Carlo VaR backtest function, employing daily portfolio returns. The Monte Carlo VaR calculated by using 5 years moving window or using exponential weight $\lambda = 0.9989003714$ to estimate drift and volatility. The assumption here is that the portfolio is following Geometric Brownian Motion. The simple size we use on the test is 10000. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

Figure 26, 27, 28, 29, the Monte Carlo VaR calculated by assuming that the entire portfolio follows GBM and using 5 years moving window (Figure 26 and 27) or using exponential weight $\lambda = 0.9989003714$ (Figure 28 and 29) and to estimate drift and volatility. During the evaluation, we aim to contrast the 5-day 99% VaR for both long and short positions with the VaR derived from test data.

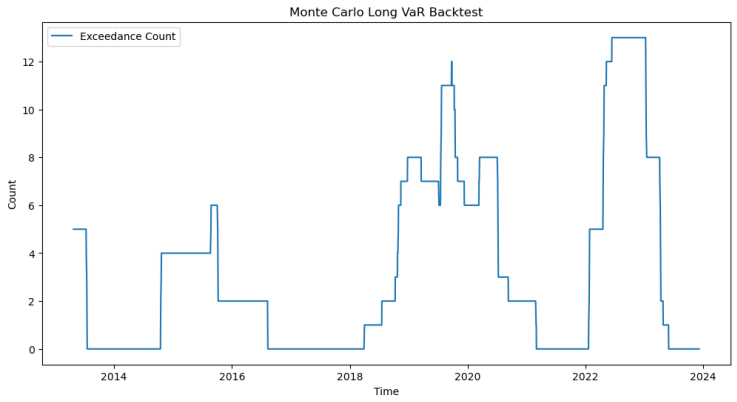

Figure 26: Monte Carlo VaR for long position with moving average method Backtest

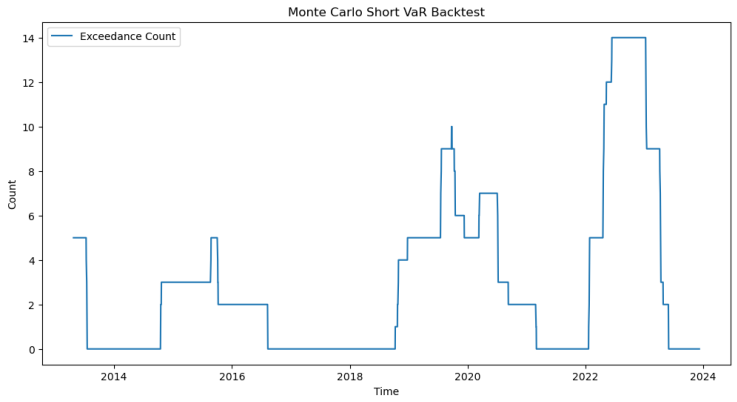

Figure 27: Monte Carlo VaR for short position with moving average method Backtest

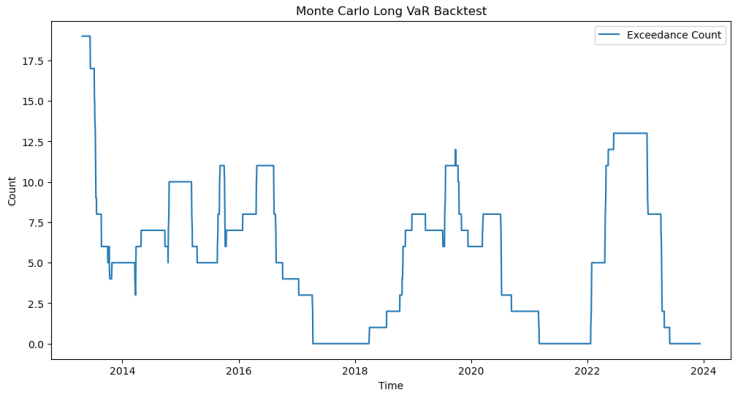

Figure 28: Monte Carlo VaR for long position with exponential weighting method Backtest

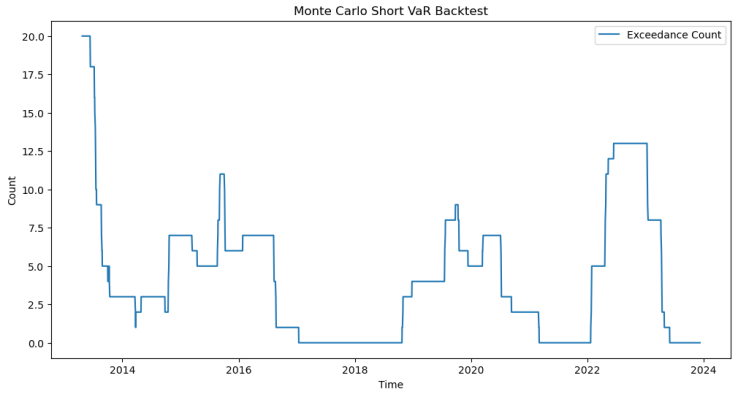

Figure 29: Monte Carlo VaR for short position with exponential weighting method Backtest

Observations:

With a 99% VaR, we expect an exception 1% of the time, which translates to $(1 - 0.99) \times 252 = 2.52$ per year.

In Figure 26 and 27, Monte Carlo VaR using moving average method for long/short portfolio backtest plots look similarly. According to the plots, from 2014 to 2023, at most of the time the results match the expected number of exceptionsm except 2016, 2020 and 2022, which means they are smaller or equal to the expected number.

In Figure 28 and 29, Monte Carlo VaR using exponential weighting method for long/short portfolio backtest plots look similarly. Most of the values are larger than 2.52 except 2018 and 2022. We conclude that parametric VaR under normal distribution is not a good choice in calculating VaR for long portfolio.

So, for long/short position, assuming the entire portfolio follows GBM and using moving average method for calculating drift and volatility tend to give the best results when calculating Monte Carlo VaR.

5.9 Monte Carlo VaR with put/call option

We test our Monte Carlo Value at Risk (VaR) implementation for the European put/call option using actual market data for both the option and the underlying stocks in the portfolio. The put/call option constitutes 1% of the portfolio, with the remaining 99% allocated to stocks. Employing the current risk-free rate of 5%, time to expiration of 1 year and no dividend, we derive the implied volatility from the available option data. Our Monte Carlo sample size is 10,000.

Figure 30 shows the results of the 5-day 99% VaR with 1% put/call options in the portfolio for both long and short positions on December 8, 2023.

Figure 31 shows the results of the 5-day 99% VaR without 1% put options in the portfolio for both long and short positions on December 8, 2023. And have a comparation by calculating percentage reduction.

Figure 32 shows the results of the 5-day 99% VaR without 1% call options in the portfolio for both long and short positions on December 8, 2023. And have a comparation by calculating percentage reduction.

```
In [65]: # Test: Monte Carlo VaR with Moving Average method (with put option)
         short_put_VaR = rms_instance.MC_short_put_VaR('KO', mu, sigma)
         print('Monte Carlo short put VaR: ')
         print(short_put_VaR)
         long_put_VaR = rms_instance.MC_long_put_VaR('KO', mu, sigma)
         print('Monte Carlo long put VaR: ')
         print(long_put_VaR)

         Monte Carlo short put VaR:
         1610.670474698396
         Monte Carlo long put VaR:
         1231.3569942426675

In [66]: # Test: Monte Carlo VaR with Moving Average method (with call option)
         short_call_VaR = rms_instance.MC_short_call_VaR('KO', mu, sigma)
         print('Monte Carlo short call VaR: ')
         print(short_call_VaR)
         long_call_VaR = rms_instance.MC_long_call_VaR('KO', mu, sigma)
         print('Monte Carlo long call VaR: ')
         print(long_call_VaR)

         Monte Carlo short call VaR:
         1871.7422261251631
         Monte Carlo long call VaR:
         1512.423760619551
```

Figure 30: Monte Carlo VaR with put/call option

```
In [98]: # Compare with VaR without put option
VaR_long = rms_instance.para_long_VaR_ES(mu, sigma)[0][-1]
print('Long VaR without put option: ',VaR_long)
per_redu = (VaR_long - long_put_VaR)/VaR_long * 100
print("percentage reduction: ",per_redu)
print('-'*60)

VaR_short = rms_instance.para_short_VaR_ES(mu, sigma)[0][-1]
print('Short VaR without put option: ',VaR_short)
s_per_redu = (VaR_short - short_put_VaR)/VaR_short * 100
print("percentage reduction: ",s_per_redu)
```

```
Long VaR without put option:  1413.958572233405
percentage reduction:  11.310933777561946
------------------------------------------------------------
Short VaR without put option:  1671.4802752692103
percentage reduction:  4.822555884971715
```

Figure 31: Compare Monte Carlo VaR with or without put option

```
In [99]: # Compare with VaR without call option
VaR_long = rms_instance.para_long_VaR_ES(mu, sigma)[0][-1]
print('Long VaR without call option: ',VaR_long)
per_redu = (VaR_long - long_call_VaR)/VaR_long * 100
print("percentage reduction: ",per_redu)
print('-'*60)

VaR_short = rms_instance.para_short_VaR_ES(mu, sigma)[0][-1]
print('Short VaR without call option: ',VaR_short)
s_per_redu = (VaR_short - short_call_VaR)/VaR_short * 100
print("percentage reduction: ",s_per_redu)
```

```
Long VaR without call option:  1413.958572233405
percentage reduction:  -4.866122544887263
------------------------------------------------------------
Short VaR without call option:  1671.4802752692103
percentage reduction:  -9.977936666056419
```

Figure 32: Compare Monte Carlo VaR with or without put option

Observations:

Long VaR without put option : 1413
Long VaR with put option : 1231
Percentage reduction : 11.31

Short VaR without put option : 1671
Short VaR with put option : 1611
Percentage reduction : 4.82

Long VaR without call option : 1413
Long VaR with call option : 1512
Percentage reduction: -4.87

Short VaR without call option : 1671
Short VaR with call option : 1872
Percentage reduction : -9.98

Thus we conclude in addition to the portfolio containing KO and NFLX, we should long this put KO option to reduce risk (reduce VaR).

Acknowledgements

We would like to express our heartfelt appreciation to Prof. Harvey J. Stein for exceptional guidance and unwavering support throughout the course of our group project. A special thank you goes to our TAs, Xiaoce Wang and Zexi Zhang, for the extremely helpful office hours.

References

[1] H. J. Stein. Lecture 3 risk measurement. *MATH GR 5320 Financial Risk Management and Regulation*, page 34, 2023.

[2] H. J. Stein. Lecture 3 risk measurement. *MATH GR 5320 Financial Risk Management and Regulation*, page 38, 2023.

[3] H. J. Stein. Lecture 3 risk measurement. *MATH GR 5320 Financial Risk Management and Regulation*, page 45, 2023.

[4] H. J. Stein. Lecture 3 risk measurement. *MATH GR 5320 Financial Risk Management and Regulation*, page 52, 2023.

[5] H. J. Stein. Lecture 3 risk measurement. *MATH GR 5320 Financial Risk Management and Regulation*, page 55, 2023.

[6] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 33, 2023.

[7] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 34, 2023.

[8] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 22, 2023.

[9] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 25, 2023.

[10] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 49, 2023.

[11] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 31, 2023.

[12] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 19, 2023.

[13] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 43, 2023.

[14] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 44, 2023.

[15] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 48, 2023.

[16] H. J. Stein. Lecture 4 market risk. *MATH GR 5320 Financial Risk Management and Regulation*, page 33, 2023.

[17] Yasuhiko Yamai, Toshinao Yoshiba. Value-at-risk versus expected shortfall: A practical perspective. *Journal of Banking & Finance*, 29(2005) 997–1015. Available online September 2004.

Appendix

```
import pandas as pd
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
import matplotlib.pyplot as plt
import math

class Risk_Calculation_System:

    def __init__(self, S0, T, p_var, p_es, l, M, Stocks=None):
        """
        Initialize the Risk Calculation System with given parameters.

        Parameters:
        - S0: Initial investment amount
        - T: Time horizon in years
        - p_var: Confidence level for Value at Risk (VaR)
        - p_es: Confidence level for Expected Shortfall (ES)
        - l: Lookback period for historical methods
        - M: Number of Monte Carlo simulations
        - Stocks: List of tuples containing file paths and positions for individual
          stocks in the portfolio
        """

        self.S0 = S0
        self.T = T
        self.p_var = p_var
        self.p_es = p_es
        self.l = l
        self.M = M

        self.portfolio = None
```

```
    self.stock_data_dict = None

    if (Stocks != None):
        self.preprocess_data(Stocks)
        self.read_stocks_data(Stocks)
        self.position1 = Stocks[0][1]
        self.position2 = Stocks[1][1]



def preprocess_data(self,Stocks):

    """
    Preprocess stock data and create a portfolio dataframe.

    Parameters:
    - Stocks: List of tuples containing file paths and positions for individual
      stocks in the portfolio
    """

    self.portfolio = pd.DataFrame()

    for stock in Stocks:
        file_path, position = stock
        try:
            new_stock = pd.read_csv(file_path)
        except:
            print("Error reading data")
            return

    new_stock['Dates'] = pd.to_datetime(new_stock['Dates'])
    new_stock.set_index('Dates', inplace=True)
    new_stock['PX_LAST'] *= position

    if self.portfolio.empty:
```

```
            self.portfolio = new_stock[['PX_LAST']].copy()
            self.portfolio.columns = ['PX_LAST']
        else:
            self.portfolio = pd.merge(self.portfolio, new_stock[['PX_LAST']],
                                      left_index=True, right_index=True, how='outer')

        self.portfolio.fillna(0.001, inplace=True)  # Handle missing values
        self.portfolio['PX_LAST'] = self.portfolio.sum(axis=1)
        self.portfolio['LogReturn'] = np.log(self.portfolio['PX_LAST'] /
                                             self.portfolio['PX_LAST'].shift(1))

        return self.portfolio



def read_stocks_data(self,Stocks):

    """
    Read stock data for individual tickers into a dictionary.

    Parameters:
    - Stocks: List of tuples containing file paths and positions for individual
      stocks in the portfolio
    """

    self.stock_data_dict = {}

    for stock in Stocks:
        file_path = stock[0]
        stock_ticker = file_path.split('/')[-1].split('.')[0]
        stock_df = pd.read_csv(file_path)
        stock_df['Dates'] = pd.to_datetime(stock_df['Dates'])
        stock_df.set_index('Dates', inplace=True)
        self.stock_data_dict[stock_ticker] = stock_df
```

```
    return self.stock_data_dict



def calculate_mu_sigma(self,yr):
    """
    Calculate the estimated drift and volatility using rolling window.

    Parameters:
    - yr: Lookback period in years
    """

    dt = 1/252
    results_mu = pd.Series(index=self.portfolio.index)
    results_sigma = pd.Series(index=self.portfolio.index)

    mu_bar = self.portfolio['LogReturn'].rolling(window=yr*252).mean()
    var_bar = ((self.portfolio['LogReturn']**2).rolling(window=yr*252).mean())
               -mu_bar**2
    sigma_bar = np.sqrt(var_bar)

    results_sigma = sigma_bar/np.sqrt(dt)
    results_mu = mu_bar/dt+results_sigma**2/2

    self.portfolio['sigma'] = results_sigma
    self.portfolio['mu'] = results_mu

    return results_mu, results_sigma



def calculate_exp_mu_sigma(self, lmbda, stock_df):
    """
    Calculate the estimated drift and volatility using exponential weighting.
```

```
Parameters:
- lmbda: Lambda value for exponential weighting
- stock_df: DataFrame containing stock price data
"""

df_copy = stock_df.sort_index(ascending=False)
prices = df_copy['PX_LAST']

rtn = -np.diff(np.log(prices))
rtnsq = rtn * rtn
windowLen = min(int(np.ceil(np.log(0.01)/np.log(lmbda))),2000)

w = np.array([lmbda ** i for i in range (windowLen)])
w /= w.sum ()

n = len(prices)-windowLen+1
sigma = pd.Series(index=prices.index)
mu = pd.Series(index=prices.index)

for x in range(n):
    end = min(x+windowLen,n)
    w_adj = w[:end-x]
    rtn_seg = rtn[x:end]
    mubar = np.sum(rtn_seg*w_adj)
    rtnsq_seg = rtnsq[x:end]
    x2bar = np.sum(rtnsq_seg*w_adj)

    var = x2bar-mubar**2
    sigmabar = np.sqrt(np.maximum(var,0))
    Sigma = sigmabar/np.sqrt(1/252)
    Mu = mubar/(1/252)+Sigma**2/2

    sigma[x] = Sigma
    mu[x] = Mu
sigma = sigma.sort_index()
```

```
    mu = mu.sort_index()

    return mu, sigma



def plot_mu_sigma(self, mu, sigma):
    """
    Plot the estimated drift and volatility in the same plot.

    Parameters:
    - mu: Series containing estimated drift values
    - sigma: Series containing estimated volatility values
    """

    sigma = sigma[len(sigma)-12*252:]
    mu = mu[len(mu)-12*252:]
    plt.figure(figsize=(18, 5))

    plt.subplot(1, 2, 1)
    plt.plot(mu, label='mu', linestyle='-')
    plt.title('Drift of the portfolio')
    plt.xlabel('Date')
    plt.ylabel('Drift')
    plt.grid(True)
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(sigma, label='sigma', linestyle='-')
    plt.title('Volatility of the portfolio')
    plt.xlabel('Date')
    plt.ylabel('Volatility')
    plt.grid(True)
    plt.legend()
```

```
    plt.tight_layout()
    plt.show()



def para_normal_dis_long_VaR_ES(self, lmbda):
    '''
    Calculate Parametric VaR and ES for a long position assuming normal
    distribution.

    Parameters:
    - lmbda: Lambda value for exponential weighting
    '''

    a = self.position1
    b = self.position2
    S_H0 = self.S0 / 2 / a
    S_U0 = self.S0 / 2 / b
    t = self.T

    long_VaR_normal = pd.Series(index=self.portfolio.index)
    long_ES_normal = pd.Series(index=self.portfolio.index)

    first_stock_key = list(self.stock_data_dict.keys())[0]
    stock_1 = self.stock_data_dict[first_stock_key]
    second_stock_key = list(self.stock_data_dict.keys())[1]
    stock_2 = self.stock_data_dict[second_stock_key]

    stock_1['Return'] = np.log(stock_1['PX_LAST'] / stock_1['PX_LAST'].shift(1))
    stock_2['Return'] = np.log(stock_2['PX_LAST'] / stock_2['PX_LAST'].shift(1))
    cov = np.cov(stock_1['Return'][len(stock_1)-12*252:],
                 stock_2['Return'][len(stock_2)-12*252:])[0, 1]

    exp_mu_1, exp_sigma_1 = self.calculate_exp_mu_sigma(lmbda, stock_1)
    exp_mu_2, exp_sigma_2 = self.calculate_exp_mu_sigma(lmbda, stock_2)
```

```
def mu_sigma(mu_H, sigma_H, mu_U, sigma_U):
    rho = cov / (sigma_H * sigma_U * self.T)
    m_s_r = mu_H + mu_U + rho * sigma_H * sigma_U
    E_S1S2 = S_H0 * S_U0 * np.exp(m_s_r)
    E_V = a * S_H0 * np.exp(mu_H * t) + b * S_U0 * np.exp(mu_U * t)
    E_V2 = a**2 * S_H0**2 * np.exp(2 * mu_H * t + sigma_H**2 * t)
           + b**2 * S_U0**2 * np.exp(2 * mu_U * t + sigma_U**2 * t)
           + 2 * a * b * S_H0 * S_U0 * np.exp(m_s_r * t)
    sd_V = np.sqrt(E_V2 - E_V**2)
    return E_V, sd_V

def long_var_normal(mu, sigma, p):
    z = norm.ppf(p)
    return self.S0 - (mu - z * sigma)

def long_es_normal(sigma, p):
    Z = norm.ppf(p)
    ES = (np.exp(-0.5 * Z**2)) / ((1 - p) * np.sqrt(2 * math.pi)) * sigma
    return ES

for date in exp_mu_1.index:
    mu_H = exp_mu_1[date]
    sigma_H = exp_sigma_1.loc[date]
    mu_U = exp_mu_2.loc[date]
    sigma_U = exp_sigma_2.loc[date]
    mu, sigma = mu_sigma(mu_H, sigma_H, mu_U, sigma_U)
    var = long_var_normal(mu, sigma, self.p_var)
    es = long_es_normal(sigma, self.p_es)

    long_VaR_normal.loc[date] = var
    long_ES_normal.loc[date] = es

return long_VaR_normal, long_ES_normal
```

```
def para_normal_dis_short_VaR_ES(self, lmbda):
    '''
    Calculate Parametric VaR and ES for a short position assuming normal
    distribution.

    Parameters:
    - lmbda: Lambda value for exponential weighting
    '''

    a = self.position1
    b = self.position2
    S_H0 = self.S0 / 2 / a
    S_U0 = self.S0 / 2 / b
    t = self.T

    short_VaR_normal = pd.Series(index=self.portfolio.index)
    short_ES_normal = pd.Series(index=self.portfolio.index)

    first_stock_key = list(self.stock_data_dict.keys())[0]
    stock_1 = self.stock_data_dict[first_stock_key]
    second_stock_key = list(self.stock_data_dict.keys())[1]
    stock_2 = self.stock_data_dict[second_stock_key]

    stock_1['Return'] = np.log(stock_1['PX_LAST'].shift(1) / stock_1['PX_LAST'])
    stock_2['Return'] = np.log(stock_2['PX_LAST'].shift(1) / stock_2['PX_LAST'])
    cov = np.cov(stock_1['Return'][len(stock_1)-12*252:],
                                    stock_2['Return'][len(stock_2)-12*252:])[0, 1]

    exp_mu_1, exp_sigma_1 = self.calculate_exp_mu_sigma(lmbda, stock_1)
    exp_mu_2, exp_sigma_2 = self.calculate_exp_mu_sigma(lmbda, stock_2)

    def mu_sigma(mu_H, sigma_H, mu_U, sigma_U):
        rho = cov / (sigma_H * sigma_U * self.T)
```

```
        m_s_r = mu_H + mu_U + rho * sigma_H * sigma_U
        E_S1S2 = S_H0 * S_U0 * np.exp(m_s_r)
        E_V = a * S_H0 * np.exp(mu_H * t) + b * S_U0 * np.exp(mu_U * t)
        E_V2 = a**2 * S_H0**2 * np.exp(2 * mu_H * t + sigma_H**2 * t)
               + b**2 * S_U0**2 * np.exp(2 * mu_U * t + sigma_U**2 * t)
               + 2 * a * b * S_H0 * S_U0 * np.exp(m_s_r * t)
        sd_V = np.sqrt(E_V2 - E_V**2)
        return E_V, sd_V

    def short_var_normal(mu, sigma, p):
        z = norm.ppf(p)
        return -self.S0 + (mu - z * sigma)

    def short_es_normal(sigma, p):
        Z = norm.ppf(p)
        ES = (np.exp(-0.5 * Z**2)) / ((1 - p) * np.sqrt(2 * math.pi)) * sigma
        return -ES

    for date in exp_mu_1.index:
        mu_H = exp_mu_1[date]
        sigma_H = exp_sigma_1.loc[date]
        mu_U = exp_mu_2.loc[date]
        sigma_U = exp_sigma_2.loc[date]
        mu, sigma = mu_sigma(mu_H, sigma_H, mu_U, sigma_U)
        var = short_var_normal(mu, sigma, self.p_var)
        es = short_es_normal(sigma, self.p_es)

        short_VaR_normal.loc[date] = var
        short_ES_normal.loc[date] = es

    return short_VaR_normal, short_ES_normal



def para_long_VaR_ES(self, mu, sigma):
```

```
    """
    Calculate Parametric VaR and ES for a long position.

    Parameters:
    - mu: Series of estimated drift values
    - sigma: Series of estimated volatility values
    """

    para_long_VaR = pd.Series(dtype=float)
    para_long_ES = pd.Series(dtype=float)

    def long_VaR(drift, volatility):
        z = norm.ppf(1 - self.p_var)
        VaR = self.S0 - self.S0 * np.exp(volatility * np.sqrt(self.T) * z
              + (drift - volatility**2/2) * self.T)
        return VaR
    def long_ES(drift, volatility):
        z_ES = norm.ppf(1 - self.p_es)
        ES = self.S0 - (self.S0 * np.exp(drift * self.T)
             * norm.cdf(z_ES - volatility * np.sqrt(self.T))) / (1 - self.p_es)
        return ES
    for j in range(len(self.portfolio)):
        drift = mu.iloc[j]
        volatility = sigma.iloc[j]
        para_long_VaR.loc[j] = long_VaR(drift, volatility)
        para_long_ES.loc[j] = long_ES(drift, volatility)

    para_long_VaR.index = self.portfolio.index
    para_long_ES.index = self.portfolio.index

    return para_long_VaR, para_long_ES



def para_short_VaR_ES(self, mu, sigma):
```

```
"""
Calculate Parametric VaR and ES for a short position.

Parameters:
- mu: Series of estimated drift values
- sigma: Series of estimated volatility values
"""

para_short_VaR = pd.Series(dtype=float)
para_short_ES = pd.Series(dtype=float)

def short_VaR(drift, volatility):
    z = norm.ppf(self.p_var)
    VaR = -self.S0+self.S0*np.exp(volatility*np.sqrt(self.T)*z
          +(drift-volatility**2/2)*self.T)
    return VaR
def short_ES(drift, volatility):
    z = norm.ppf(self.p_es)
    ES = -self.S0+(self.S0*np.exp(drift*self.T)
         *(1-norm.cdf(z-volatility*np.sqrt(self.T))))/(1-self.p_es)
    return ES

for j in range(len(self.portfolio)):
    drift = mu.iloc[j]
    volatility = sigma.iloc[j]
    para_short_VaR.loc[j] = short_VaR(drift, volatility)
    para_short_ES.loc[j] = short_ES(drift, volatility)

para_short_VaR.index = self.portfolio.index
para_short_ES.index = self.portfolio.index

return para_short_VaR, para_short_ES
```

```
def historical_sample(self,l,i):
    """
    Extract historical sample for a given time point.

    Parameters:
    - l: Lookback period in years
    - i: Index for the time point
    """

    log_rtn = self.portfolio['LogReturn'].rolling(5).sum()
    historical_sample =log_rtn.iloc[i-252*l:i]
    historical_sample = np.exp(historical_sample)*self.S0

    return historical_sample



def hist_long_VaR_ES(self, mu, sigma):
    """
    Calculate Historical VaR and ES for a long position.

    Parameters:
    - mu: Series of estimated drift values
    - sigma: Series of estimated volatility values
    """

    his_long_VaR = pd.Series(dtype=float)
    his_long_ES = pd.Series(dtype=float)

    def long_hVaR(historical_sample):
        sorted_historical_sample = np.sort(historical_sample)
        index = int(self.l*252 * (1 - self.p_var))
        return self.S0 - sorted_historical_sample[index]
    def long_hES(historical_sample):
        sorted_historical_sample = np.sort(historical_sample)
```

```
        index = int(self.l*252 * (1 - self.p_es))
        loss = self.S0 - sorted_historical_sample[:index]
        ES = sum(loss) / len(loss)
        return ES

    k = 0
    df = self.portfolio
    for i in range(len(df)-12*252,len(df)):
        k = k+1
        h_sample = self.historical_sample(self.l,i)
        his_long_VaR.loc[k] = long_hVaR(h_sample)
        his_long_ES.loc[k] = long_hES(h_sample)
    his_long_VaR.index = df.index[len(df)-12*252:len(df)]
    his_long_ES.index = df.index[len(df)-12*252:len(df)]
    return his_long_VaR, his_long_ES



def hist_short_VaR_ES(self, mu, sigma):
    """
    Calculate Historical VaR and ES for a short position.

    Parameters:
    - mu: Series of estimated drift values
    - sigma: Series of estimated volatility values
    """

    his_short_VaR = pd.Series(dtype=float)
    his_short_ES = pd.Series(dtype=float)

    def short_hVaR(historical_sample):
        sorted_historical_sample = np.sort(historical_sample)
        index = int(self.l*252 * self.p_var)
        return - self.S0 + sorted_historical_sample[index]
    def short_hES(historical_sample):
```

```
            sorted_historical_sample = np.sort(historical_sample)
            index = int(self.l*252 * self.p_es)
            loss = - self.S0 + sorted_historical_sample[index:]
            ES = sum(loss) / len(loss)
            return ES

        k = 0
        df = self.portfolio
        for i in range(len(df)-12*252,len(df)):
            k = k+1
            h_sample = self.historical_sample(self.l,i)
            his_short_VaR.loc[k] = short_hVaR(h_sample)
            his_short_ES.loc[k] = short_hES(h_sample)
        his_short_VaR.index = df.index[len(df)-12*252:len(df)]
        his_short_ES.index = df.index[len(df)-12*252:len(df)]
        return his_short_VaR, his_short_ES



def abs_hist_long_VaR_ES(self,drift,volatility):
    """
    Calculate Historical VaR and ES for a long position assuming absolute change.

    Parameters:
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    def abs_hist_sample(i):
        abs_return = self.portfolio['PX_LAST']-self.portfolio['PX_LAST'].shift(1)
        sum_rtn = abs_return.rolling(5).sum()
        his_sample = sum_rtn.iloc[i-252*5:i]
        his_sample = his_sample*self.S0/self.portfolio['PX_LAST'].iloc[i]
        return his_sample
```

```
    def abs_hVaR(sample):
        sorted_sample = np.sort(sample)
        index = int(5*252*self.p_var)
        return sorted_sample[index]
    def abs_hES(sample):
        sorted_sample = np.sort(sample)
        index = int(5*252*self.p_es)
        loss = sorted_sample[index:]
        ES = sum(loss) / len(loss)
        return ES

    VaR = pd.Series(dtype=float)
    ES = pd.Series(dtype=float)
    k = 0
    for i in range(len(self.portfolio)-12*252,len(self.portfolio)):
        k = k+1
        h_sample = abs_hist_sample(i)
        VaR.loc[k] = abs_hVaR(h_sample)
        ES.loc[k] = abs_hES(h_sample)
    VaR.index =self.portfolio.index[len(self.portfolio)-12*252:len(self.portfolio)]
    ES.index = self.portfolio.index[len(self.portfolio)-12*252:len(self.portfolio)]
    return VaR, ES



def abs_hist_short_VaR_ES(self,drift,volatility):
    """
    Calculate Historical VaR and ES for a short position assuming absolute change.

    Parameters:
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """
    def abs_hist_sample(i):
        abs_return = self.portfolio['PX_LAST'].shift(1) - self.portfolio['PX_LAST']
```

```
        sum_rtn = abs_return.rolling(5).sum()
        his_sample = sum_rtn.iloc[i-252*5:i]
        his_sample = his_sample*self.S0/self.portfolio['PX_LAST'].iloc[i]
        return his_sample

    def abs_hVaR(sample):
        sorted_sample = np.sort(sample)
        index = int(5*252*self.p_var)
        return sorted_sample[index]
    def abs_hES(sample):
        sorted_sample = np.sort(sample)
        index = int(5*252*self.p_es)
        loss = sorted_sample[index:]
        ES = sum(loss) / len(loss)
        return ES

    VaR = pd.Series(dtype=float)
    ES = pd.Series(dtype=float)
    k = 0
    for i in range(len(self.portfolio)-12*252,len(self.portfolio)):
        k = k+1
        h_sample = abs_hist_sample(i)
        VaR.loc[k] = abs_hVaR(h_sample)
        ES.loc[k] = abs_hES(h_sample)
    VaR.index =self.portfolio.index[len(self.portfolio)-12*252:len(self.portfolio)]
    ES.index = self.portfolio.index[len(self.portfolio)-12*252:len(self.portfolio)]
    return VaR, ES



def MC_sample(self, drift, volatility, M, i):
    """
    Generate a Monte Carlo sample of stock prices.

    Parameters:
```

```
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    - M: Number of Monte Carlo simulations
    - i: Index for the time point
    """
    mu = drift.iloc[i]
    sigma = volatility.iloc[i]
    W = np.random.normal(0, np.sqrt(self.T), M)
    S_t = self.S0 * np.exp((mu - (sigma ** 2) / 2) * self.T + sigma * W)
    return S_t



def MC_long_VaR_ES(self, drift, volatility):
    """
    Calculate Monte Carlo VaR and ES for a long position.

    Parameters:
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    MC_long_VaR = pd.Series(dtype=float)
    MC_long_ES = pd.Series(dtype=float)

    def long_mc_VaR(sample):
        loss = self.S0 - sample
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR
    def long_mc_ES(sample):
        loss = self.S0 - sample
        loss = np.sort(loss)
        index = int(self.M * self.p_es)
```

```
        loss = loss[index:]
        ES = sum(loss) / len(loss)
        return ES

    for i in range(len(self.portfolio)):
        p_sample = self.MC_sample(drift, volatility, self.M, i)
        MC_long_VaR.loc[i] = long_mc_VaR(p_sample)
        MC_long_ES.loc[i] = long_mc_ES(p_sample)

    MC_long_VaR.index = self.portfolio.index
    MC_long_ES.index = self.portfolio.index
    return MC_long_VaR, MC_long_ES



def MC_short_VaR_ES(self, drift, volatility):
    """
    Calculate Monte Carlo VaR and ES for a short position.

    Parameters:
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    MC_short_VaR = pd.Series(dtype=float)
    MC_short_ES = pd.Series(dtype=float)

    def short_mc_VaR(sample):
        loss = - self.S0 + sample
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR
    def short_mc_ES(sample):
        loss = - self.S0 + sample
```

```
            loss = np.sort(loss)
            index = int(self.M * self.p_es)
            loss = loss[index:]
            ES = sum(loss) / len(loss)
            return ES

        for i in range(len(self.portfolio)):
            p_sample = self.MC_sample(drift, volatility, self.M, i)
            MC_short_VaR.loc[i] = short_mc_VaR(p_sample)
            MC_short_ES.loc[i] = short_mc_ES(p_sample)

        MC_short_VaR.index = self.portfolio.index
        MC_short_ES.index = self.portfolio.index
        return MC_short_VaR, MC_short_ES



def plot_VaR_ES(self, Type, Trade, drift, volatility):
    """
    Plot VaR and ES values for a specific risk calculation type and trading
    position.

    Parameters:
    - Type: Risk calculation type ('Parametric', 'Historical', or 'Monte Carlo')
    - Trade: Trading position ('long' or 'short')
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    if Type == 'Parametric (Normal distributed)':
        lmbda = 0.9989003714
        if Trade == 'long':
            VaR, ES = self.para_normal_dis_long_VaR_ES(lmbda)
        elif Trade == 'short':
            VaR, ES = self.para_normal_dis_short_VaR_ES(lmbda)
```

```
        else:
            raise ValueError('Wrong Tradr. Please choose long or short.')
    else:
        if Type == 'Parametric':
            if Trade == 'long':
                f = self.para_long_VaR_ES
            elif Trade == 'short':
                f = self.para_short_VaR_ES
            else:
                raise ValueError('Wrong Tradr. Please choose long or short.')
        elif Type == 'Historical':
            if Trade == 'long':
                f = self.hist_long_VaR_ES
            elif Trade == 'short':
                f = self.hist_short_VaR_ES
            else:
                raise ValueError('Wrong Tradr. Please choose long or short.')
        elif Type == 'Historical (absolute change)':
            if Trade == 'long':
                f = self.abs_hist_long_VaR_ES
            elif Trade == 'short':
                f = self.abs_hist_short_VaR_ES
            else:
                raise ValueError('Wrong Tradr. Please choose long or short.')
        elif Type == 'Monte Carlo':
            if Trade == 'long':
                f = self.MC_long_VaR_ES
            elif Trade == 'short':
                f = self.MC_short_VaR_ES
            else:
                raise ValueError('Wrong Tradr. Please choose long or short.')
        else:
            raise ValueError('Wrong Type. Please choose Parametric, Historical,
                             or Monte Carlo.')
```

```
        VaR, ES = f(drift, volatility)

    VaR = VaR[len(VaR)-12*252:]
    ES = ES[len(ES)-12*252:]

    plt.figure(figsize=(10, 6))
    plt.plot(VaR, label='VaR')
    plt.plot(ES, label='ES')
    plt.title(f'{Type} {Trade} VaR and ES')
    plt.xlabel('Time')
    plt.ylabel('Value')
    plt.legend()
    plt.show()



def black_scholes(self, call_put, S, K, r, q, sigma, T):
    """
    Calculate the Black-Scholes option pricing formula.

    Parameters:
    - call_put: Option type ('c' for call or 'p' for put)
    - S: Current stock price
    - K: Option strike price
    - r: Risk-free rate
    - q: Dividend yield
    - sigma: Volatility of the underlying stock
    - T: Time to expiration (in years)
    """

    d1 = (np.log(S/K) + (r - q + 0.5*sigma**2)*T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    if call_put == 'c':
            # for call
```

```
        N1 = norm.cdf(d1)
        N2 = norm.cdf(d2)
        price = S * N1 * np.exp(-q * T) - K * np.exp(-r * T) * N2
    elif call_put == 'p':
            # for put
        N1 = norm.cdf(-d1)
        N2 = norm.cdf(-d2)
        price = K * np.exp(-r * T) * N2 - S * N1 * np.exp(-q * T)
    else:
        raise ValueError("Invalid option type. Use 'c' for call or 'p' for put.")

    return price



def mc_put_sample(self,df,drift,volatility,i):
    """
    Generate a Monte Carlo sample for a put option.

    Parameters:
    - df: Dataframe for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    - i: Index for the time point
    """

    stock_price = df.iloc[i]['PX_LAST']
    put_price = self.black_scholes('p', stock_price, stock_price, 0.005, 0,
                                   df.iloc[i]['12MO_PUT_IMP_VOL']/100, 1)
    num_put = self.S0 * 0.01 / put_price
    num_stock = self.S0 * 0.99 / stock_price
    mu = drift.iloc[i]
    sigma = volatility.iloc[i]

    W = np.random.normal(0, np.sqrt(self.T), self.M)
```

```
    S_t = stock_price * np.exp((mu - (sigma ** 2) / 2) * self.T + sigma * W)
    vol = df.iloc[i]['12MO_PUT_IMP_VOL'] / 100
    put = self.black_scholes('p', S_t, stock_price, 0.005, 0, vol, 1)

    return S_t * num_stock + put * num_put



def MC_long_put_VaR(self, kicker, drift, volatility):
    """
    Calculate Monte Carlo VaR for a long position with a put option.

    Parameters:
    - kicker: Ticker for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    df = self.stock_data_dict[kicker]

    def long_var(sample):
        loss = self.S0 - sample
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR

    long_VaR_put = long_var(self.mc_put_sample(df,drift,volatility, -1))

    return long_VaR_put



def MC_short_put_VaR(self, kicker, drift, volatility):
    """
```

```
    Calculate Monte Carlo VaR for a short position with a put option.

    Parameters:
    - kicker: Ticker for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    df = self.stock_data_dict[kicker]

    def short_var(sample):
        loss = sample - self.S0
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR

    short_VaR_put = short_var(self.mc_put_sample(df,drift,volatility, -1))

    return short_VaR_put



def mc_call_sample(self,df,drift,volatility,i):
    """
    Generate a Monte Carlo sample for a call option.

    Parameters:
    - df: Dataframe for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    - i: Index for the time point
    """

    stock_price = df.iloc[i]['PX_LAST']
```

```
    call_price = self.black_scholes('c',stock_price,stock_price,0.005,0,
                                    df.iloc[i]['12MO_CALL_IMP_VOL']/100,1)
    num_call = self.S0 * 0.01 / call_price
    num_stock = self.S0 * 0.99 / stock_price
    mu = drift.iloc[i]
    sigma = volatility.iloc[i]

    W = np.random.normal(0, np.sqrt(self.T), self.M)
    S_t = stock_price * np.exp((mu - (sigma ** 2) / 2) * self.T + sigma * W)
    vol = df.iloc[i]['12MO_CALL_IMP_VOL'] / 100
    call = self.black_scholes('c', S_t, stock_price, 0.005, 0, vol, 1)

    return S_t * num_stock + call * num_call



def MC_long_call_VaR(self, kicker, drift, volatility):
    """
    Calculate Monte Carlo VaR for a long position with a call option.

    Parameters:
    - kicker: Ticker for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    df = self.stock_data_dict[kicker]

    def long_var(sample):
        loss = self.S0 - sample
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR
```

```
    long_VaR_call = long_var(self.mc_call_sample(df,drift,volatility, -1))

    return long_VaR_call



def MC_short_call_VaR(self, kicker, drift, volatility):
    """
    Calculate Monte Carlo VaR for a short position with a put option.

    Parameters:
    - kicker: Ticker for the stock data
    - drift: Series of estimated drift values
    - volatility: Series of estimated volatility values
    """

    df = self.stock_data_dict[kicker]

    def short_var(sample):
        loss = sample - self.S0
        loss = np.sort(loss)
        index = int(self.M * self.p_var)
        VaR = loss[index]
        return VaR

    short_VaR_call = short_var(self.mc_call_sample(df,drift,volatility, -1))

    return short_VaR_call



def long_VaR_backtest(self,long_VaR,Type):
    """
    Perform a backtest on long VaR.
```

```
    Parameters:
    - long_VaR: Series of long VaR values
    - Type: Risk calculation type ('Parametric', 'Historical', or 'Monte Carlo')
    """

    day_return = self.portfolio['PX_LAST']/
                 self.portfolio['PX_LAST'].shift(int(self.T*252))
    day_return = day_return[len(day_return)-12*252:]
    long_VaR = long_VaR[len(long_VaR)-12*252:]
    long_Exceed = ((self.S0 - day_return * self.S0) > long_VaR).astype(int)
    count = long_Exceed.rolling(window=252).sum()

    plt.figure(figsize=(12, 6))
    plt.plot(count, label='Exceedance Count')
    plt.title(f'{Type} Long VaR Backtest')
    plt.xlabel('Time')
    plt.ylabel('Count')
    plt.legend()
    plt.show()



def short_VaR_backtest(self,short_VaR,Type):
    """
    Perform a backtest on short VaR.

    Parameters:
    - short_VaR: Series of short VaR values
    - Type: Risk calculation type ('Parametric','Historical','Monte Carlo')
    """

    day_return = self.portfolio['PX_LAST'].shift(int(self.T*252))/
                 self.portfolio['PX_LAST']
    day_return = day_return[len(day_return)-12*252:]
    short_VaR = short_VaR[len(short_VaR)-12*252:]
```

```
short_Exceed = ((-self.S0 + day_return * self.S0) > short_VaR).astype(int)
count = short_Exceed.rolling(window=252).sum()

plt.figure(figsize=(12, 6))
plt.plot(count, label='Exceedance Count')
plt.title(f'{Type} Short VaR Backtest')
plt.xlabel('Time')
plt.ylabel('Count')
plt.legend()
plt.show()
```