MATH 5320 - Final Project

December 20, 2018

MATH 5320 - Financial Risk Management and Regulation

Authors

Yifan Wu (yw3101)

Yiding Xie (yx2443)

Qianfeng Ying (qy2187)

1 Table of Contents

2 Executive Summary

This document serves as the term project of **MATH 5320 - Financial Risk Management and Regulation**. This document is a collaborative effort among Yifan Wu, Yiding Xie, and Qianfeng Ying.

In this document, we constructed a risk calculation system using the various pieces of knowledge we learned from Dr. Harvey J. Stein in class.

The risk calculation system we developed has the following capabilities:

- Take a portfolio of stock and option positions as input
- System is calibrated to historical data and take parameters as input
- The calculation includes:
  - Parametric VaR
  - Historical VaR
  - Monte Carlo VaR
  - Historical ES
  - Monte Carlo ES
- The computed VaRs are backtested against history

A vast majority of the codes used in our term projects came from our previous homeworks. In addition, we have chosen models and methodologies, document and justify the choices and write and test the code.

A web-based risk management calculation system is also produced and the link is attached below: Program Link

Series of plots were attached for reference later in this report. A sample portfolio was constructed using AAPL, BAC, GE, and M.

Detailed code used in our project is listed in the Appendix at the end of this document.

3 Introduction

This document serves as the term project of MATH 5320 - Financial Risk Management and Regulation. In this document, we constructed a risk calculation system using the various pieces of knowledge we learned in class.

We cover the computations of Parametric VaR, Historical VaR, Monte Carlo VaR, Historical ES, and Monte Carlo ES for any portfolio of stock and option positions as input. In addition, we have chosen models and methodologies, document and justify the choices and write and test the code.

In parametric VaR and Monte Carlo VaR estimations, the system computes relevant parameters from historical data. In historical VaR estimation, the system uses historical data directly. Parametric VaR, historical VaR, and Monte Carlo Var estimations take different assumptions and compute VaR by standard equations, ranking historical losses, and ranking simulated outcomes respectively. Similar calculation is also done for historical ES and Monte Carlo ES. A detailed description of all these calculations is shown in the following sections.

3.1 Risk Measure

Coherence is really important when defining a risk measurement. If the measure is not coherent, it will not give us adequate information about the risk of our portfolio. For that reason, there are usually four properties required by any risk measure. In the lecture, we learned five properties.

1. Monotonic: if a portfolio achieves higher returns than another in every state of the world, then it will have lower risk, or $V \leq V' \implies \rho(V) \geq \rho(V')$
2. Positive homogeneity: maintaining the weights, if the size of a portfolio is increased by a factor, the risk will be multiplied by the same factor, or $\rho(\alpha V) = \alpha\rho(V)$ when $\alpha \geq 0$
3. Translation invariance: if an amount of cash is added to our portfolio, the risk will be reduced by that amount. For a constant a, $\rho(V + a) = \rho(V) - a$
4. Subadditive: the risk measure of two merged portfolios should be lower than the sum of their risk measures individually, or $\rho(V + V') \leq \rho(V) + \rho(V')$

3.2 Advantages and Disadvantages

In terms of the entire system, the greatest advantage is flexibility. As discussed earlier, the input of this risk calculation system is a portfolio of stock and option positions. Specifically, there is no constraint on the total number of stocks or options, and there's no requirement on the long/short positions.

The document shall also disclose the pros and cons for each type of methodology in later sections.

4 Model Description

4.1 Value at Risk (VaR) & Expected Shortfall (ES)

VaR and ES are the two main metrics used in risk management. Value at Risk (VAR) is defined as the most amount of money you are willing to lose given a certain confidence interval and over a defined period of time. In other words, for a given portfolio, time horizon, and probability p, the p VaR can be defined informally as the maximum possible loss during the time if we exclude worse outcomes whose probability is less than p. Expected Shortfall is defined as the average of all losses which are greater or equal than VaR, i.e. the average loss in the worst (1-p)% cases, where p is the confidence level.

Following the lecture notes, we can also mathematically define them as below:

VaR:

$$VaR(V, T, P) = G^{-1}(P)$$

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

ES:

$$ES(V, T, P) = -E^p[V_T - V_0 | V_T - V_0 < -VaR(V_T, p)] = E^p[V_0 - V_T | V_0 - V_T > VaR(V_T, p)]$$

Refering the risk measure back to the previous section, the Value at Risk measure always satisfies the first three properties but it will only satisfy the fourth one if portfolio returns follow a normal distribution. On the other hand, the Expected Shortfall measure satisfies the four properties in any circumstance.

4.2 Parametric VaR

4.2.1 Model Assumption

The parametric method VAR (also known as Variance/Covariance VAR) calculation is the most common form used in practice with hedge fund managers. This method is the popular because the only variables you need to do the calculation are the mean and standard deviation of the portfolio.

For parametric VaR of a single stock, the basic assumption is that the stock price follows a Geometric Brownian Motion. This means that at a given time, the distribution of stock price is a log normal distribution. When calculating parametric VaR of an option, we assume that the underlying stock price follows a Geometric Brownian Motion. When calculating parametric VaR of a portfolio, we assume that the value of the portfolio is under a normal distribution.

4.2.2 Model Input

As discussed in section above, due to the simplicity of this model, only three inputs are needed: mean and standard deviation of the portfolio returns and the time horizons. In other words, both the mean returns and the variance/covariance matrix of the portfolio returns need to be provided as inputs.

4.2.3 Model Definition

Following lecture 4 slide (Page 25 - 27), we can define the following (Given there are only two assets in one portfolio):

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

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

$$dW_1 dW_2 = \rho dt$$

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

$$E[V_t] = aS_{1,0}e^{\mu_1 t} + bS_{2,0}e^{\mu_2 t}$$

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

$$var[V_t] = E[V_t^2] - E[V_t]^2$$

When there are more than two assets in the portfolio, we can certainly easily expand above equations to multi-variate space.

4.2.4 Advantage & Disadvantage

The strengths of this method are the simplicity of the calculations and the fact that the data for the inputs is very easy to obtain. Since mean and standard deviation of the portfolio are the only input parameters, this method becomes very popular.

The biggest weakness of this method is the assumption of normality. In practice, this assumption of return normality has proven to be extremely risky. Indeed, this was the biggest mistake that LTCM made it gravely underestimating their portfolio risks.

4.3 Historical VaR

4.3.1 Model Assumption

Historical VAR is a better methodology to use if you cannot determine the distribution of your return series. This calculation is much easier than even the Parametric VAR calculation in that all you are doing is literally ranking all of your past historical returns in terms of lowest to highest and computing with a predetermined confidence rate what your lowest return historically has been.

That is, there is no explicit assumption for Historical VaR calculation. However, since we are using historical data of a portfolio to predict its own future, we are implicitly assuming the asset's past performance is consistent with its future performance. This assumption holds for all historical VaR calculations including stocks, options, and portfolios.

According to the **CFA Program Curriculum Textbook**, there are two formal ways to approach a Historical VaR calculation.

- Absolute change, which follows Arithmetic Brownian Motion with constant absolute volatility.
- Relative change, which follows Geometric Brownian Motion with constant relative volatility.

In this class, we have been using the second approach.

4.3.2 Model Input

This calculation is the easiest among all models, because the only required input parameter is the historical data of the portfolio and time horizons.

4.3.3 Model Definition

1. Calculate the returns (or price changes) of all the assets in the portfolio between each time interval.
2. Apply the price changes calculated to the current mark-to-market value of the assets and re-value your portfolio.
3. Sort the series of the portfolio-simulated P&L from the lowest to the highest value.
4. Read the simulated value that corresponds to the desired confidence level, and the portfolio loss with rank $n(1-\alpha)\%$ is the $(1-\alpha)\%$ VaR of the portfolio.

4.3.4 Advantage and Disadvantage

One advantage of historical VaR is that it is extremely simple to calculate, which made it easy to explain to non-risk professionals. Many professionals consider there is no explicit assumption under the Historical VaR model, which means this approach is non-parametric.

The biggest downside for this approach is that it implicitly assumes the past will exactly replicate the future, which seems to be very unlikely in reality. The other disadvantage of the historical simulation approach is that it can be very slow to react to changing market environments. Therefore, professionals adapt to a modified approach called "hybrid" method, which involves the use of decay factor.

4.4 Monte Carlo VaR

4.4.1 Model Assumption

The Monte Carlo approach involves developing a model for future stock price returns and running multiple hypothetical trials through the model. According to *Investopedia*, A Monte Carlo simulation refers to any method that randomly generates trials, but by itself does not tell us anything about the underlying methodology.

When calculating Monte Carlo VaR, we typically need to make assumptions on the asset distribution. Most of the times, stock prices are assumed to follow a Geometric Brownian Motion and option values are calculated using the Black Scholes model.

4.4.2 Model Input

The inputs for Monte Carlo VaR is often very similar to those for Parametric VaR. Specifically, the portfolio returns, variance/covariance matrix, and time horizons are required.

4.4.3 Model Definition

Stock A list of parameters, such as return and standard deviation, can be obtained after calculating the parameters of component's distributions. Then we can use this list of parameters to simulate the outcomes of the entire portfolio n times using the following equation:

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

Option We use Black Scholes equations to simulate the option prices. Black Scholes equations for option calculation are shown below:

$$C = N(d_1)S_t - N(d_2)Ke^{-rt}$$

$$P = N(-d_2)Ke^{-rt} - N(-d_1)S_t$$

$$d_1 = (\ln(S_t/k) + (r + \sigma^2/2))/\sigma\sqrt{t}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

After simulating the prices, we can then compute the value of portfolio at time t by combining all outcomes, which is simply the sum of each composite value multiplied by its number of shares. Then the loss of the portfolio can be calculated by substracting V_t from V_0 for each outcome. Lastly, rank the series of losses of the portfolio from high to low, and the portfolio loss with rank $n(1-\alpha)\%$ is the $(1-\alpha)\%$ VaR of the portfolio.

4.4.4 Advantage and Disadvantage

According to *Treasury Today*, using Monte Carlo methods brings the following pros and const to the software:

Advantage:

- It can consider non-linear instruments.
- It does not require significant numbers of historical observations.
- It can consider a wider range of possible outcomes.

Disadvantage:

- The outcome of the simulation is dependent on how the simulation is built.
- The calculation is quite complex and requires significant computational power. Thus, the calculation method is considered to be very expensive.

4.5 Historical ES

4.5.1 Model Assumption

Expected shortfall is also called conditional value at risk (CVaR), which is another risk measure, supplement to the VaR value. The Historical ES bears the same assumption as in Historical VaR, which assumes the asset's past performance is consistent with its future performance. This assumption holds for all historical VaR calculations including stocks, options, and portfolios.

4.5.2 Model Input

Same as the Historical VaR calculation, this calculation is the easiest among all models, because the only required input parameter is the historical data of the portfolio and time horizons.

4.5.3 Model Definition

1. Calculate the returns (or price changes) of all the assets in the portfolio between each time interval.
2. Apply the price changes calculated to the current mark-to-market value of the assets and re-value your portfolio.
3. Sort the series of the portfolio-simulated P&L from the lowest to the highest value.
4. Read the simulated value that corresponds to the desired confidence level, the Historical ES at α significance level is the average of the worst losses $n(1-\alpha)\%$ in the tail.

4.5.4 Advantage and Disadvantage

Again, the advantage and disadvantage of using Historical ES is very similar to that of Historical VaR. The greatest advantage of using Historical ES is its simplicity.

The biggest downside for this approach is that it implicitly assumes the past will exactly replicate the future, which seems to be very unlikely in reality. The other disadvantage of the historical simulation approach is that it can be very slow to react to changing market environments.

4.6 Monte Carlo ES

4.6.1 Model Assumption

Similar to Monte Carlo VaR, users typically need to make assumptions on the asset distribution, depending on the specific requirement and market conditions. Most of the times, stock prices are assumed to follow a Geometric Brownian Motion and option values are calculated using the Black Scholes model.

4.6.2 Model Input

The inputs for Monte Carlo VaR is often very similar to those for Monte Carlo VaR. Specifically, the portfolio returns, variance/covariance matrix and time horizons are required.

4.6.3 Model Definition

1. Simulate each asset in the portfolio based on its model assumption.
2. Calculate the value of portfolio at time t for all outcomes by summing up each composite value multiplied by its number of shares.
3. Calculate the loss by subtracting V_t from V_0 for each simulation outcome.
4. Rank the losses from high to low, the Monte Carlo ES at α significance level is the average of the worst losses $n(1-\alpha)\%$ in the tail.

4.6.4 Advantage and Disadvantage

Again, the pros and cons of using Monte Carlo ES are very similar to Monte Carlo VaR. Specifically, when assuming stock prices to follow a Geometric Brownian Motion and option values to follow the Black Scholes model, if any of the assumption under those two motion/model fails, the Monte Carlo ES result is not going to be valid.

Financial Risk Management System

Authors: Yifan Wu, Yiding Xie, Qianfeng Ying

The calculation system has the capability of taking a portfolio with any given stocks or options.

Please click on bottoms below to enter the calculation system.

Individual Stock Only

Portfolio Only

Portfolio With Option

Image1: User Interface

5 Risk Modeling

5.1 Test Software

The code is programed by Python language and a User Interface is created using Python Dash.

This web-based risk calculation system aims to provide a user-friendly software for users getting known of risk indicators for an individual stock or portfolio. It allows users to enter basic attributes regarding their investment choice, and outputs graphs of VaR/ES with respect to time.

A screenshot of the main panel is shown above:

The link to our online user interface system is here: Program Link

5.2 Test Plan

Model Risk Management typically consists two parts: Model Development and Model Validation. Developers in the Model Development team is in charge of building the platform and algorithms to make sure the underlying model is doing what it's supposed to be doing. On the other hand, model validation is a key area of research that can help mitigate model risk, and its important role in model risk management is the focus of this document. Detailed validation test plan step is listed below:

1. Collect raw data and define testing criteria
2. Pre-process raw data
3. Define model assumptions
4. Build and test different models (Parametric VaR, Historical VaR, Monte Carlo VaR, Historical ES, and Monte Carlo ES)

5. Analyze result

The ultimate goal of our software is for any portfolio, with any number of stocks and options.

5.3 Data Collection

5.3.1 Data

User has the option to type in any combinations of the stocks and options.

5.3.2 Window size

User has the option to pick 1 year, 5 year, and 10 year as the window size. A large enough window should be chosen to yield stable VaR and ES calculations. However, if the window is too large, the model is then tuned to be very slowly adjusted to market condition changes, which is not very ideal. On the other hand, if the window is too small, the VaR and ES calculation will be accurate on average, but the daily changes in VaR and ES will a high variance.

According to our homework assignment, we have determined the window size of 5 year to be the most optimal value.

5.4 System Requirements

- Users are responsible to enter valid, reasonable inputs
  - Tickers must be consistent with Yahoo finance
  - For portfolio parts, weights must sum up to exactly 1
- Users should make sure there is enough historical price data available from Yahoo finance, since the system will automatically retrieve data there. Enough data means that it has price history back until at least 10 years before the start date.

5.5 System Instruction

1. The system has 3 features: Individual Stock Only, Stock Portfolio, and Portfolio with Options
2. For each of the feature, the following information needs to be entered:
   - Ticker
   - Starting/End Dates
   - Window
   - VaR/ES Percentage
   - Weights
   - Horizon
   - Risk Method
3. After hitting "submit" button, plots will be generated:
   - Historical Price
   - Parameter Mu
   - Parameter Sigma
   - VaR/ES
   - Backtested Results VaR/Actual Loss

6 Results Analysis (Using Sample Data)

6.1 Data

For visualization purposes, we have selected the following stocks and options.

- AAPL (Apple)
- BAC (Bank of America)
- GE (General Electric)
- M (Macy's Inc)

Here, we purposely selected four stocks from four different sectors. All the data came from Yahoo Finance.

In terms of time period, since the inception of each stock varies, we have also aligned the data by joining the dates. The processed data ranges from Feb. 5th, 1992 to Dec. 10th, 2018.

```
In [149]: mergetable.head(10)
```

Out[149]:

Date	AAPL	BAC	GE	M
1992-02-05	2.361607	11.00000	6.458333	8.6250
1992-02-06	2.290179	11.34375	6.458333	8.4375
1992-02-07	2.285714	11.34375	6.437500	7.8125
1992-02-10	2.254464	11.28125	6.447917	7.5625
1992-02-11	2.245536	11.18750	6.416667	7.6875
1992-02-12	2.330357	11.37500	6.489583	7.9375
1992-02-13	2.294643	11.40625	6.447917	8.0000
1992-02-14	2.290179	11.34375	6.427083	7.8125
1992-02-18	2.241071	11.15625	6.427083	7.8125
1992-02-19	2.214286	11.00000	6.531250	8.0000

```
In [150]: mergetable.tail(10)
```

Out[150]:

Date	AAPL	BAC	GE	M
2018-11-26	174.619995	27.559999	7.58	32.560001
2018-11-27	174.240005	27.740000	7.44	33.889999
2018-11-28	180.940002	28.430000	7.74	34.419998
2018-11-29	179.550003	28.040001	7.94	33.740002
2018-11-30	178.580002	28.400000	7.50	34.220001
2018-12-03	184.820007	28.540001	7.81	34.349998
2018-12-04	176.690002	26.990000	7.28	32.419998
2018-12-06	174.720001	26.280001	7.35	32.369999
2018-12-07	168.490005	25.430000	7.01	31.690001
2018-12-10	169.600006	24.760000	6.93	31.889999

6.2 Assumptions

- Monte Carlo VaR and Parametric VaR calculations are under the assumption that each stock or each underlying follows GBM. For options, we model its corresponding underlying and then calculate the price.
- We also assume options to be under fictionless scenario. In other words, the strike price and the time to maturity are unchanged from day to day.
- Similar to what we did in homework, we assumed all options are ATM.

6.3 Sample Results

6.3.1 Individual Stock Only

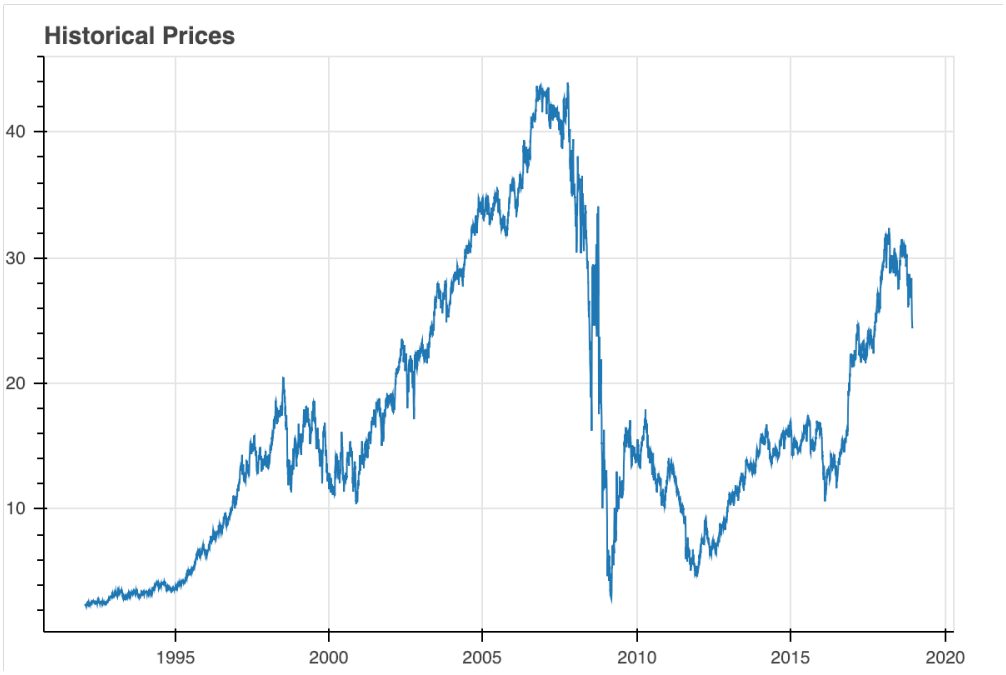

Image2: Stock Historical Price - BAC

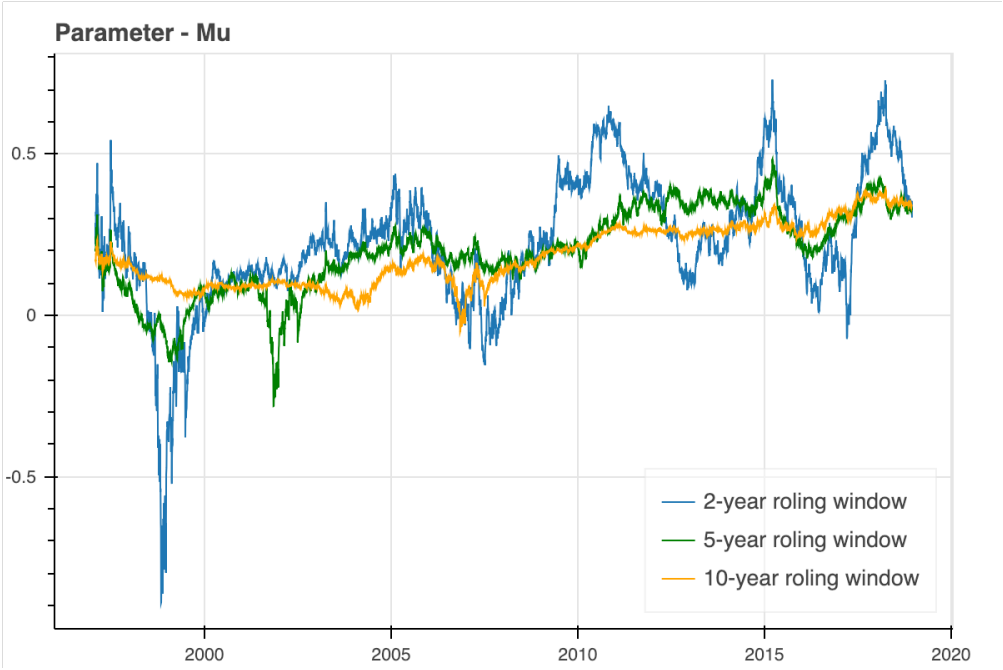

Image3: Stock Mu - BAC

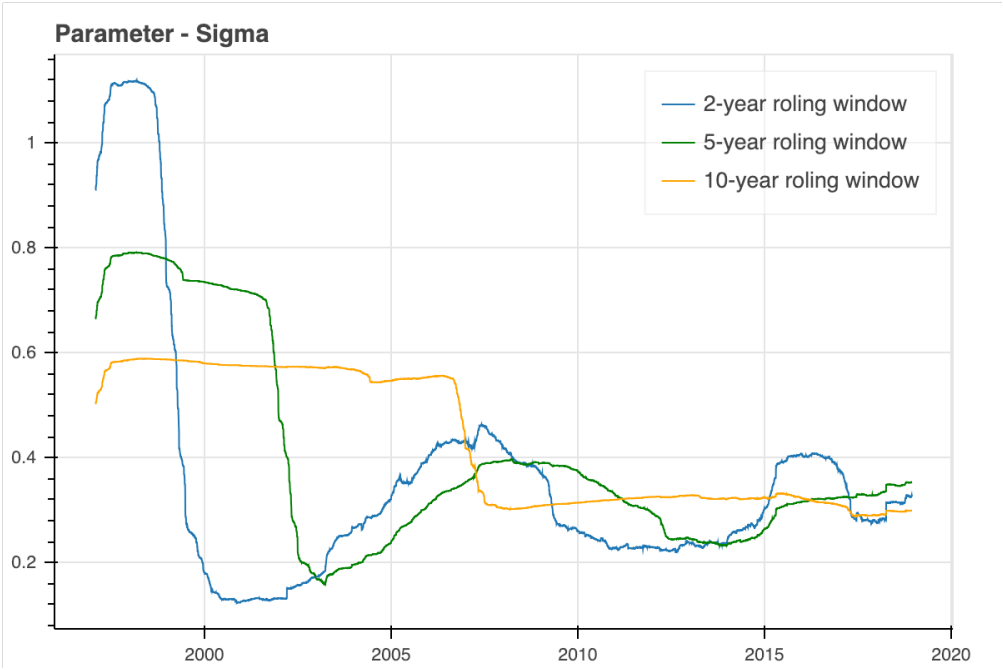

Image4: Stock Sigma - BAC

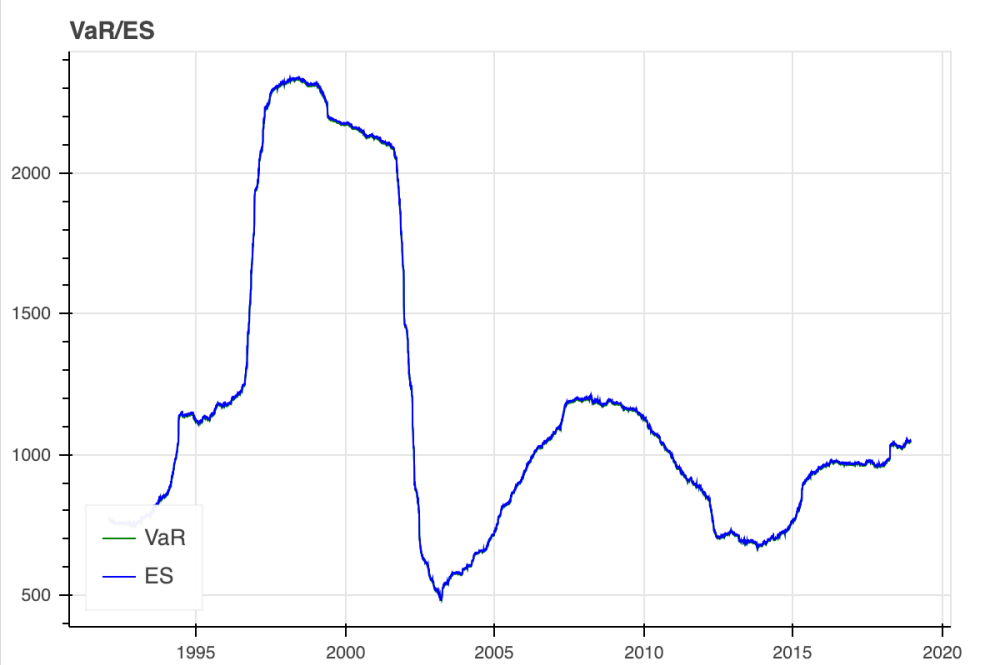

Image5: Stock VaR/ES - BAC

6.3.2 Portfolio (With Stocks Only)

Image6: Portfolio Historical Price

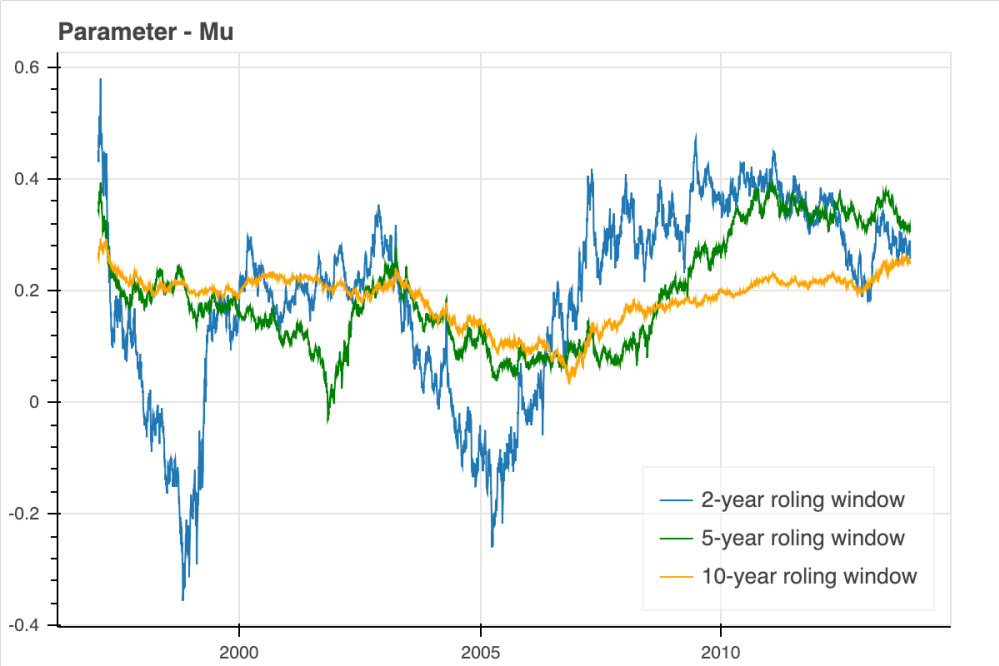

Image7: Portfolio Mu (Parametric Method)

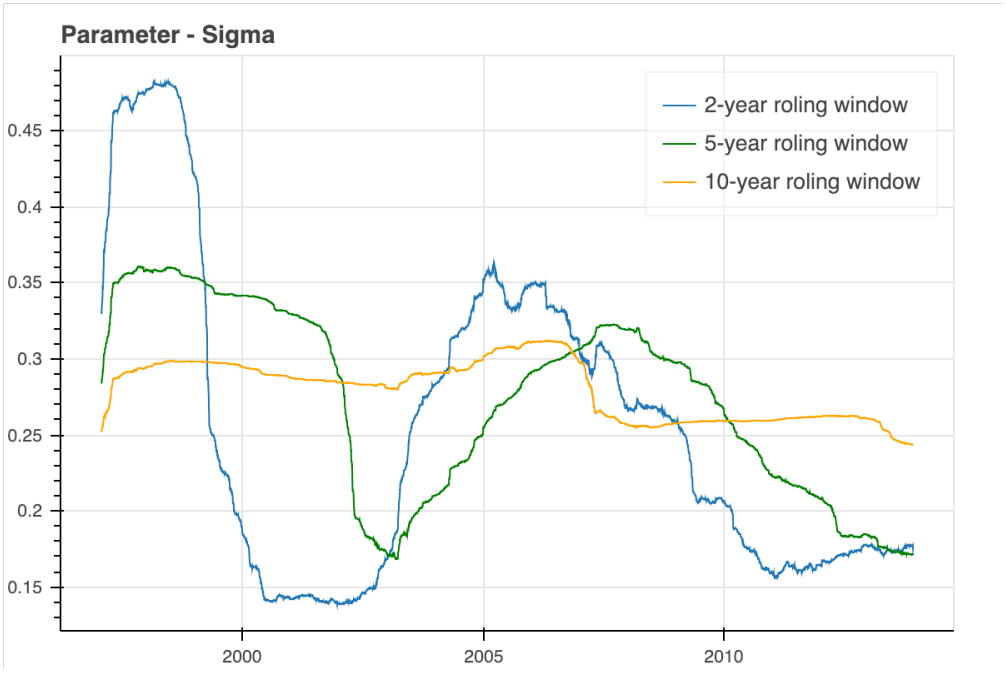

Image8: Portfolio Sigma (Parametric Method)

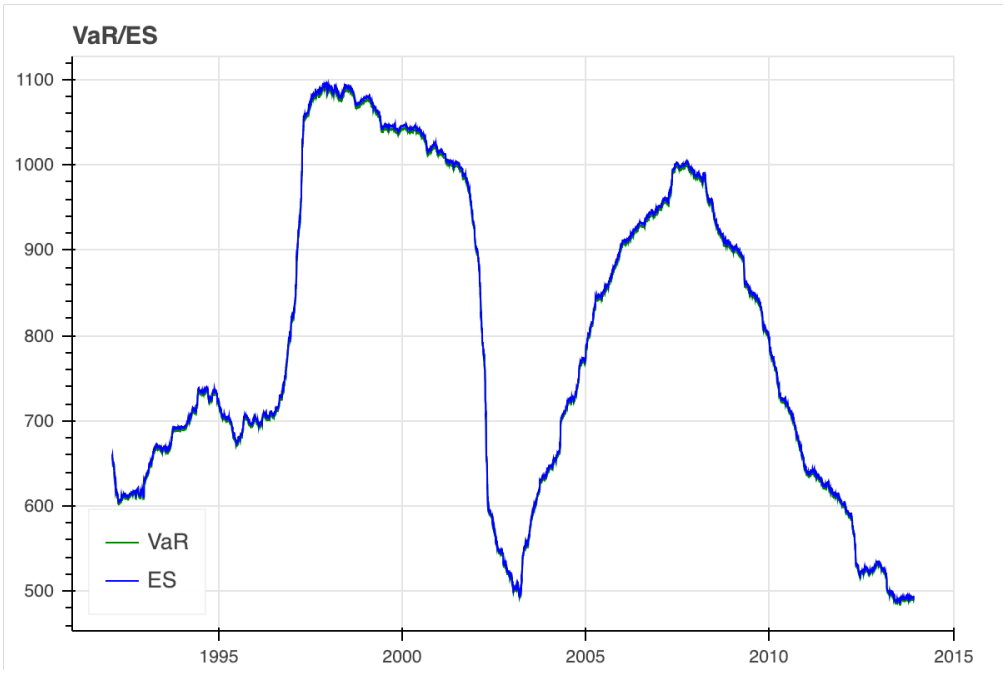

Image9: Portfolio VaR/ES (Parametric Method)

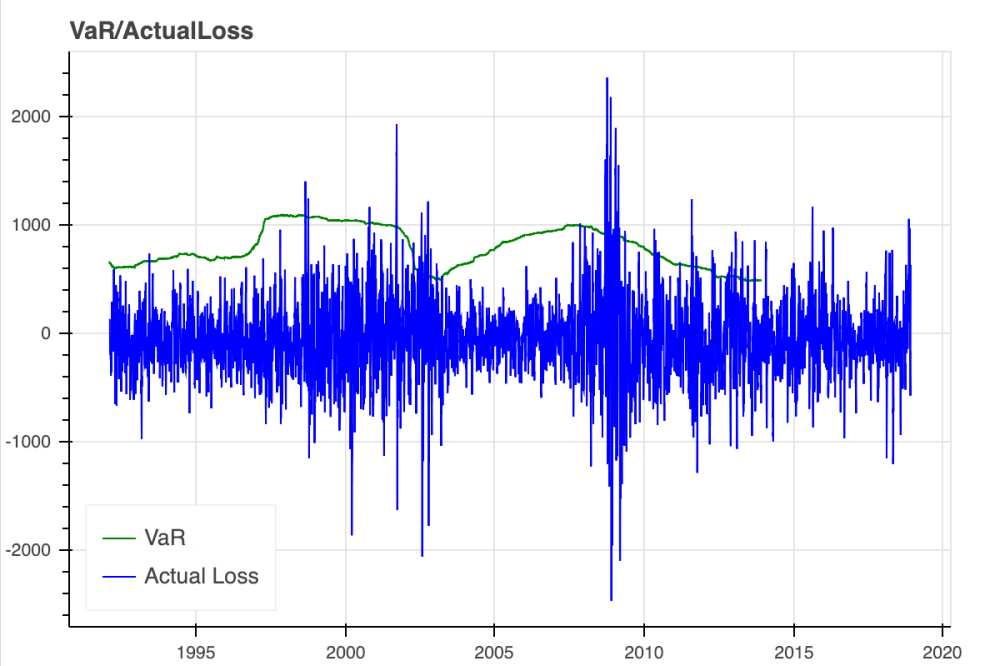

Image10: Portfolio VaR/Actual Loss (Parametric Method)

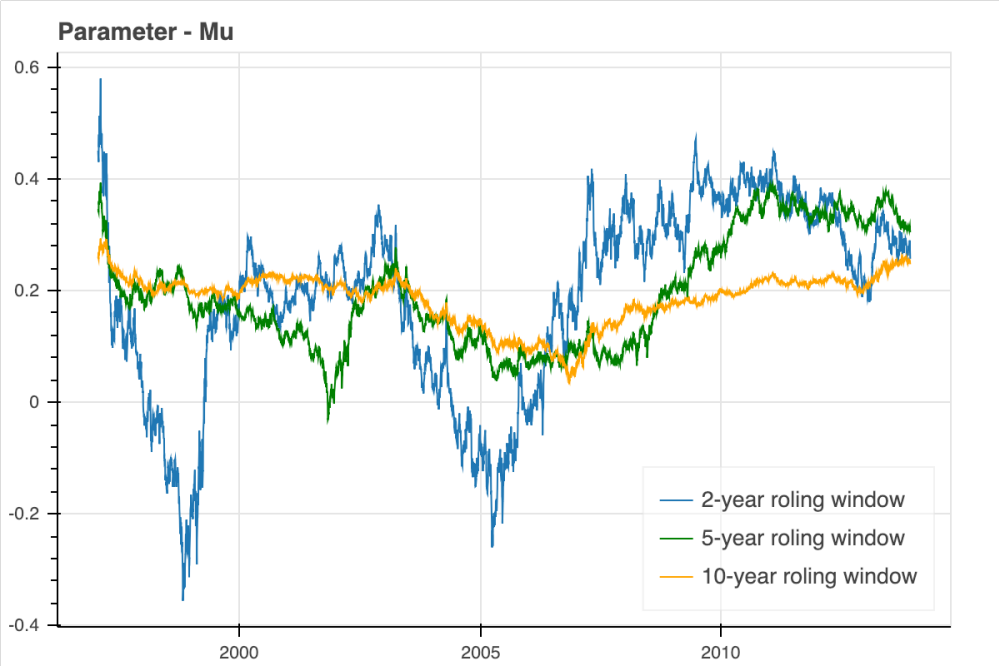

Image11: Portfolio Mu (Historical Method)

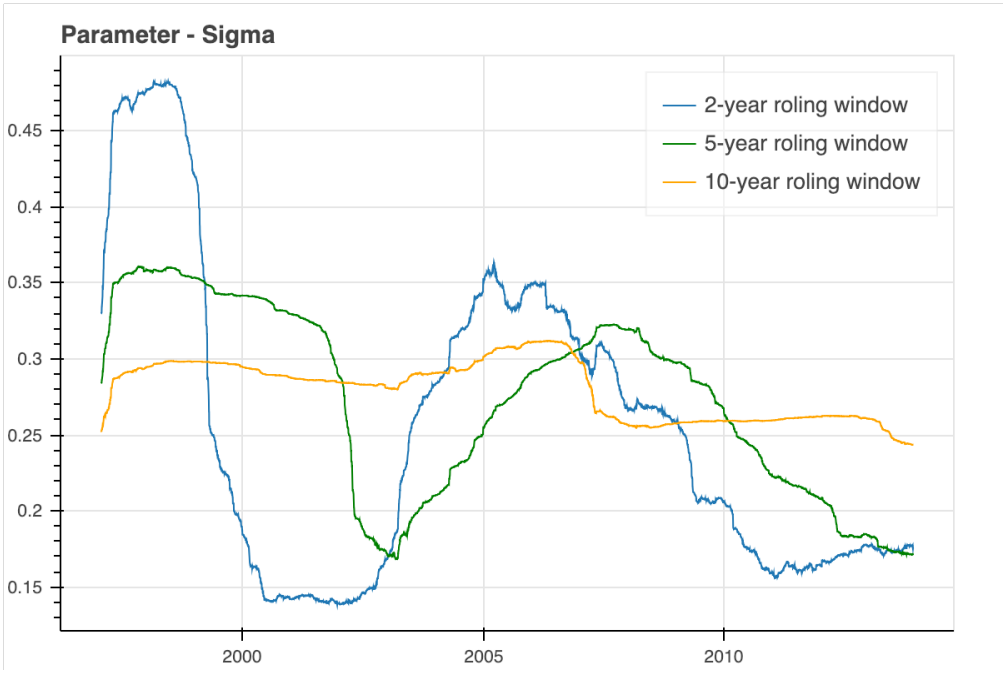

Image12: Portfolio Sigma (Historical Method)

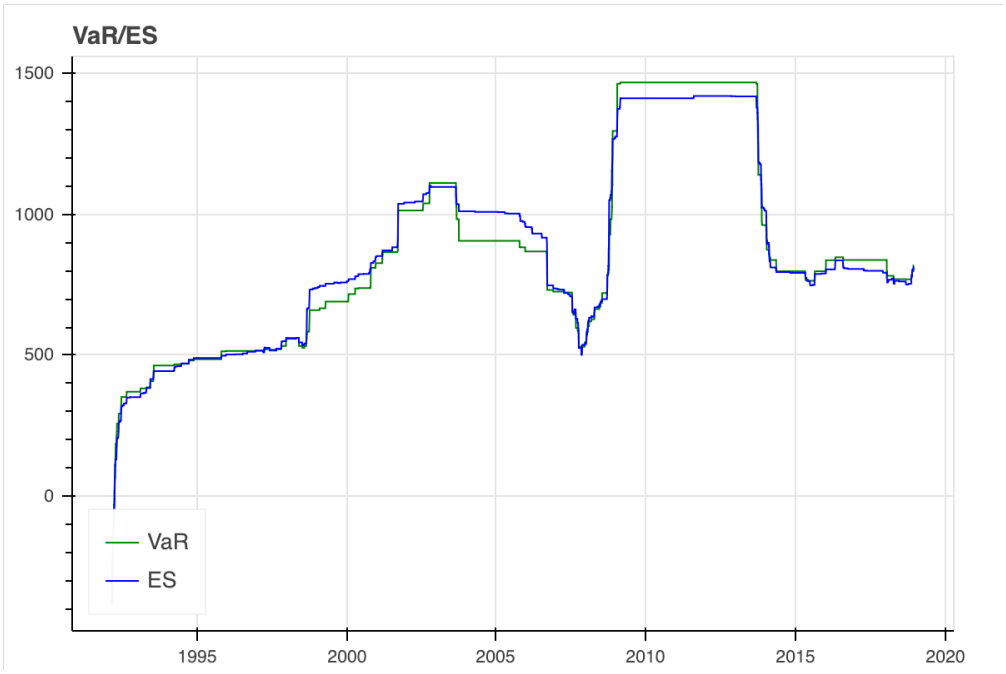

Image13: Portfolio VaR/ES (Historical Method)

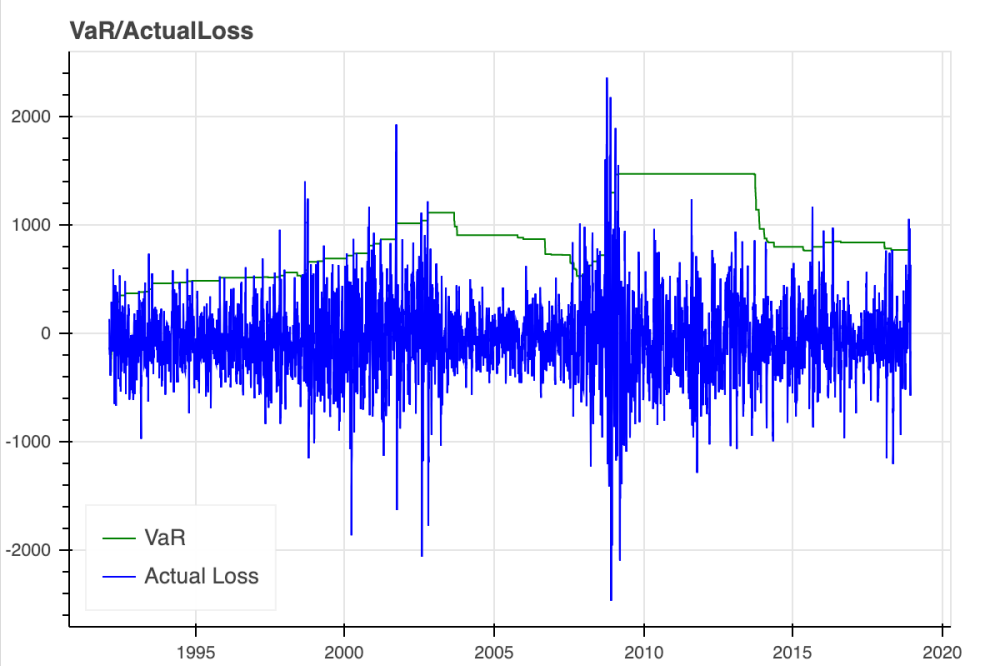

Image14: Portfolio VaR/Actual Loss (Historical Method)

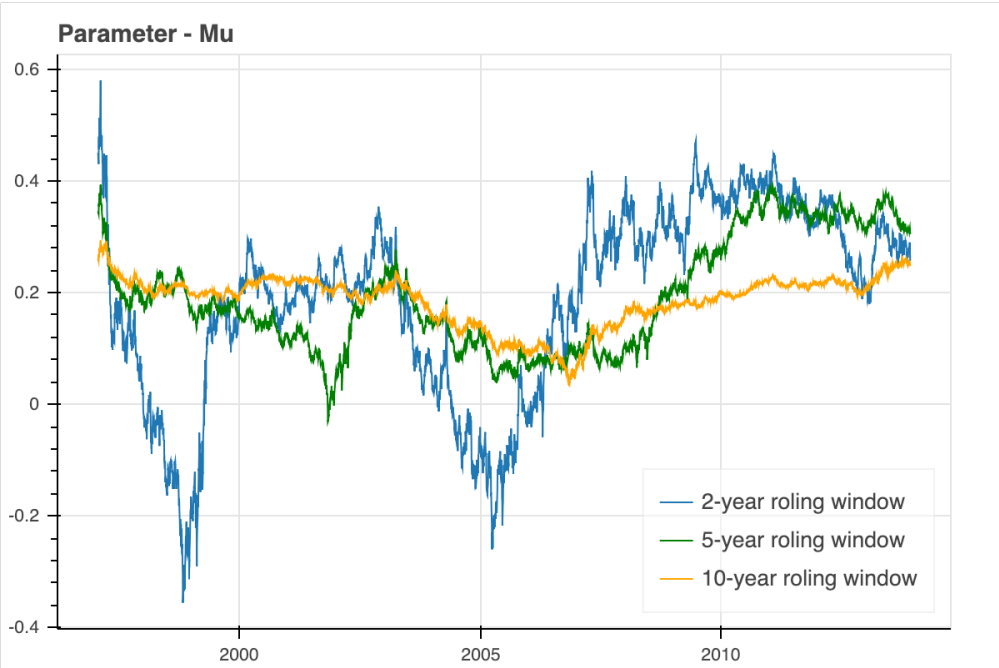

Image15: Portfolio Mu (Monte Carlo Method)

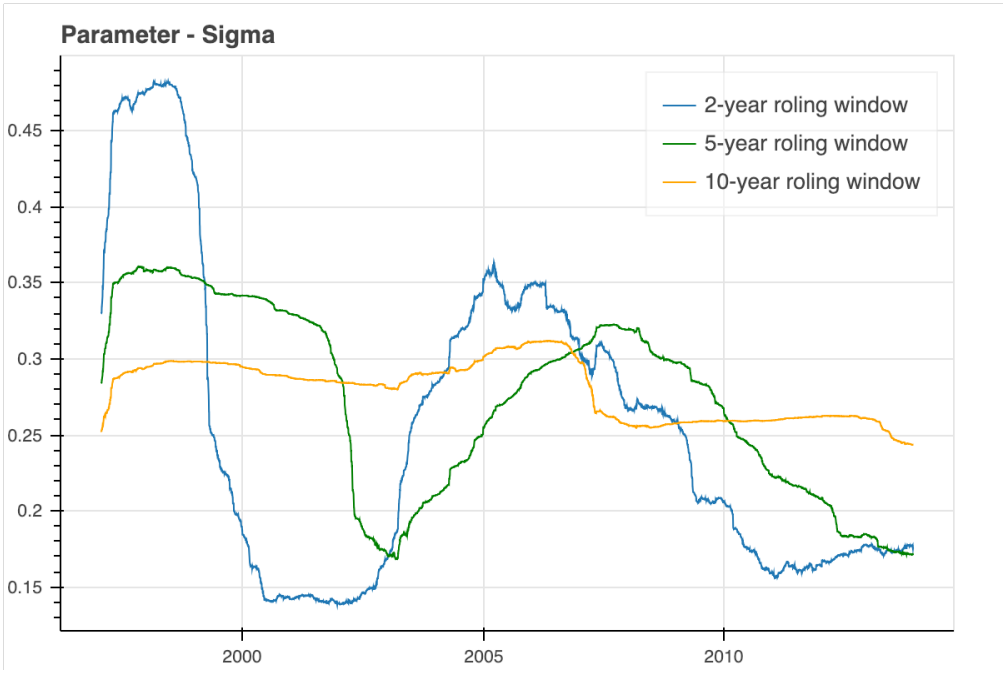

Image16: Portfolio Sigma (Monte Carlo Method)

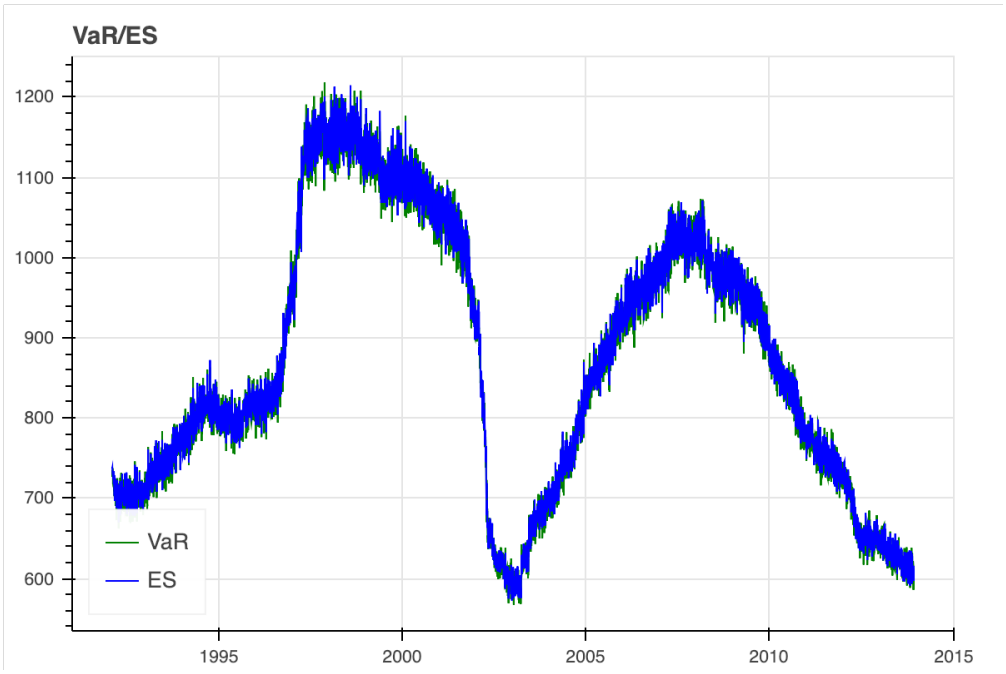

Image17: Portfolio VaR/ES (Monte Carlo Method)

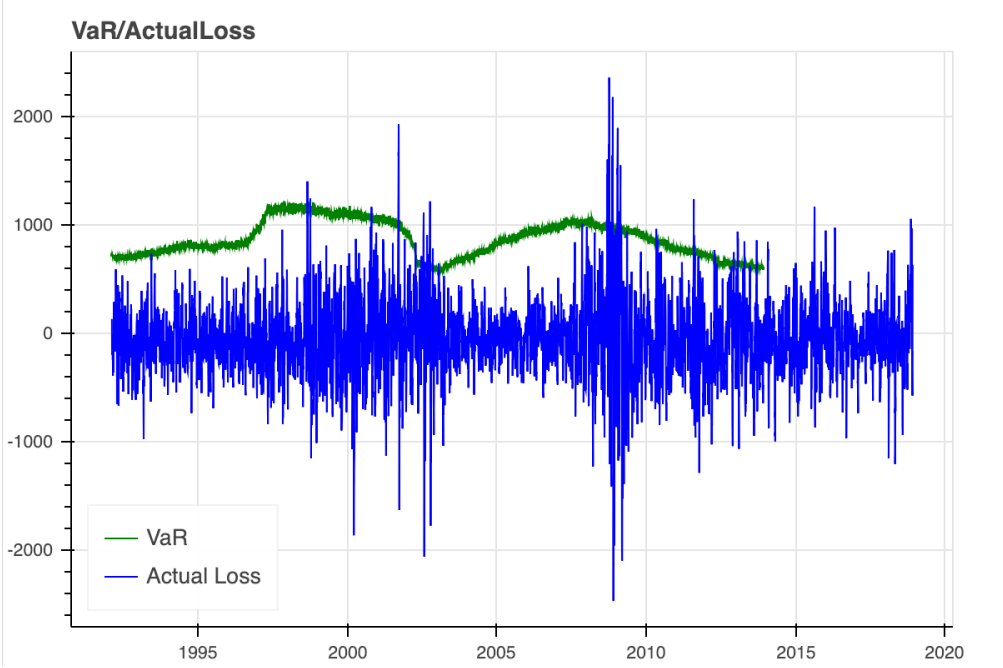

Image18: Portfolio VaR/Actual Loss (Monte Carlo Method)

6.3.3 Portfolio (Stocks & Options)

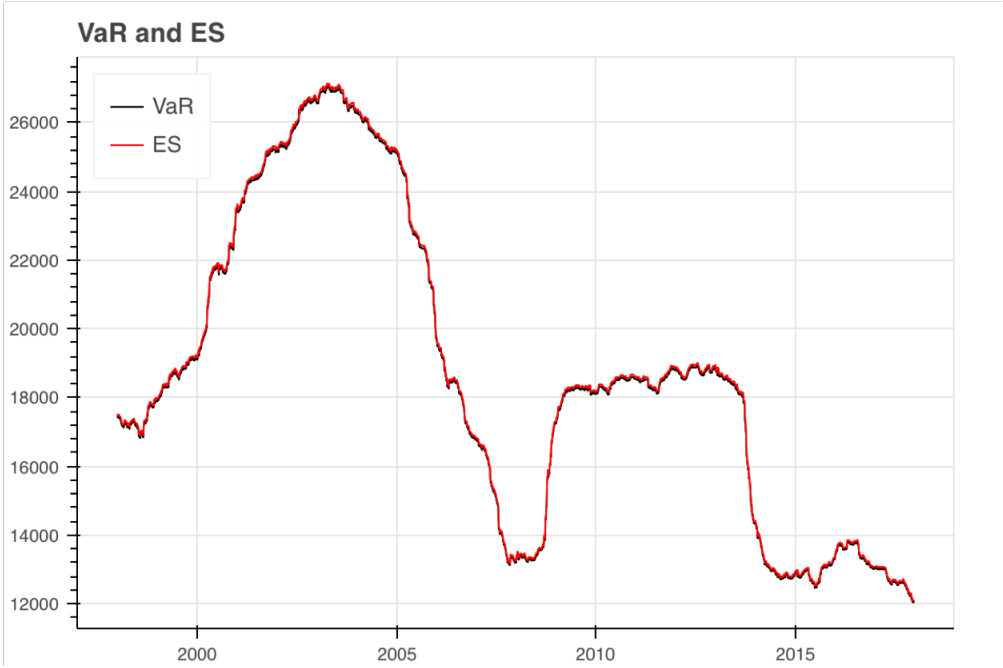

Image19: Portfolio-Stock&Option VaR/ES (Parametric)

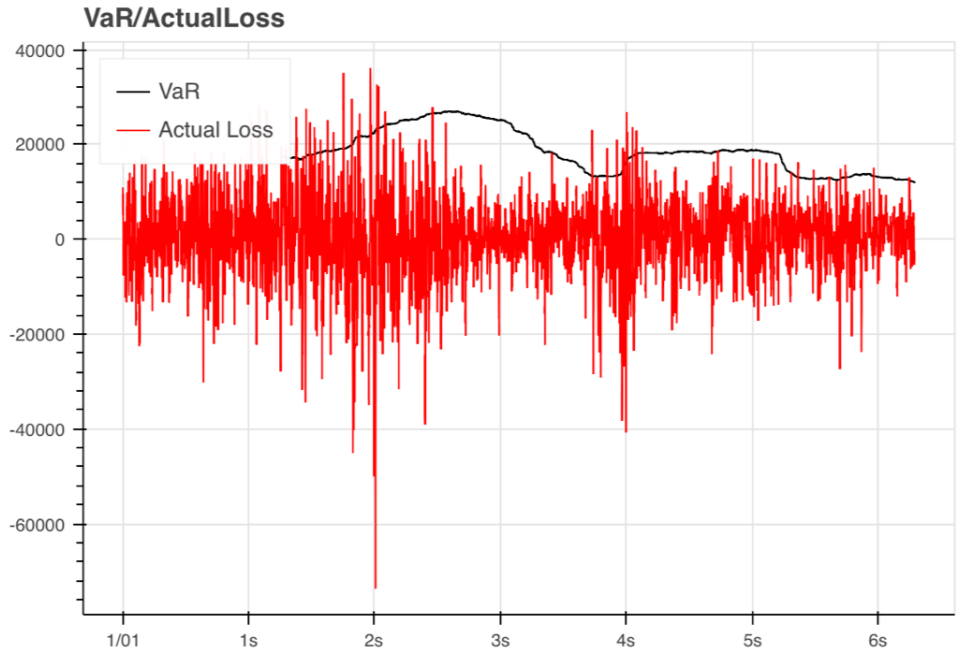

Image20: Portfolio-Stock&Option VaR/Actual Loss (Parametric)

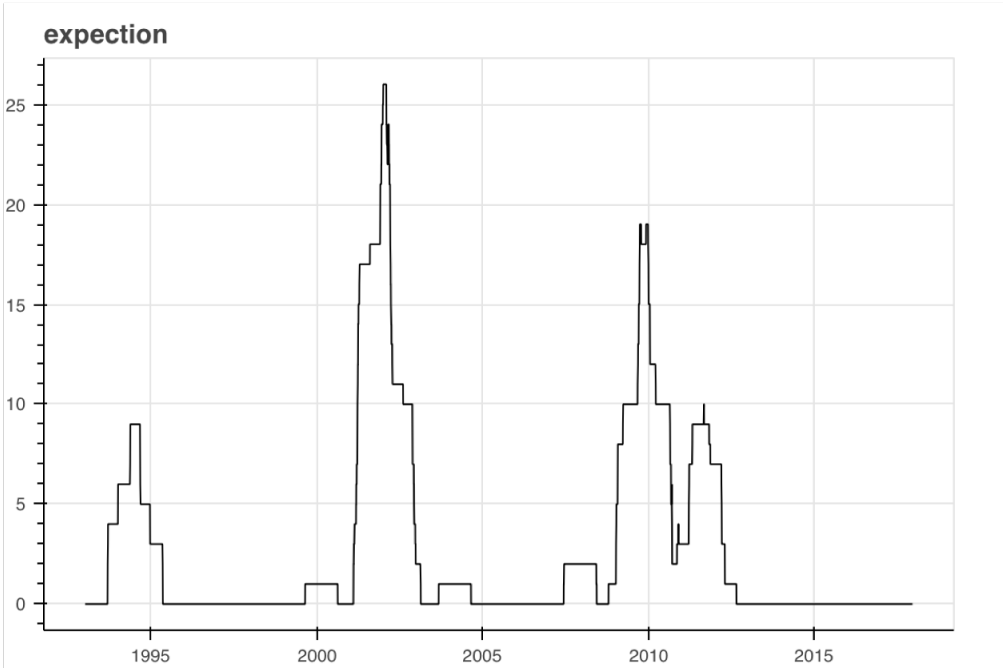

Image21: Portfolio-Stock&Option Backtest (Parametric)

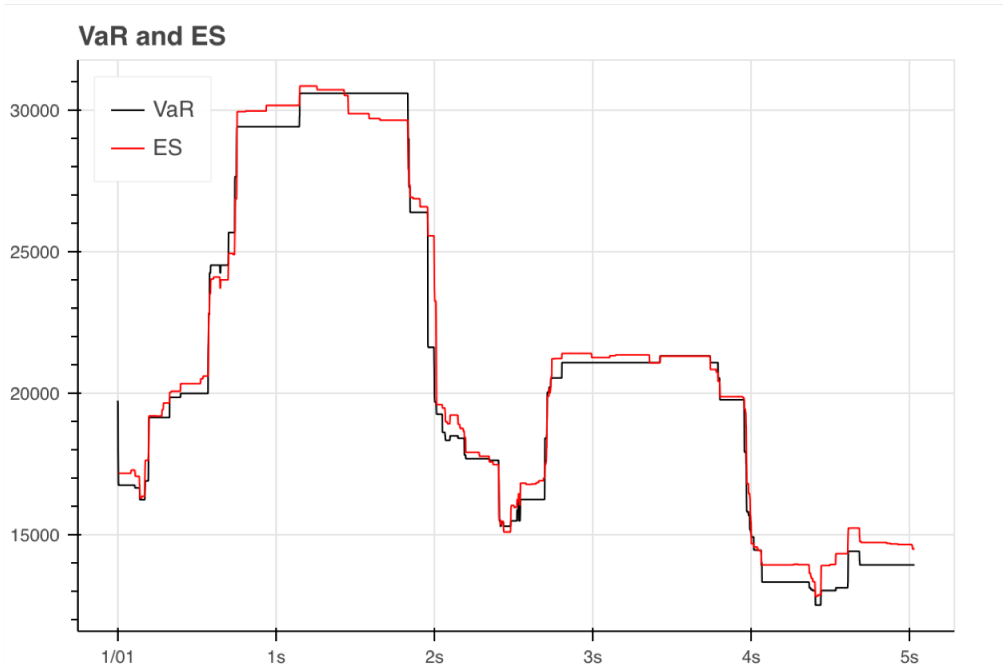

Image22: Portfolio-Stock&Option VaR/ES (Historic)

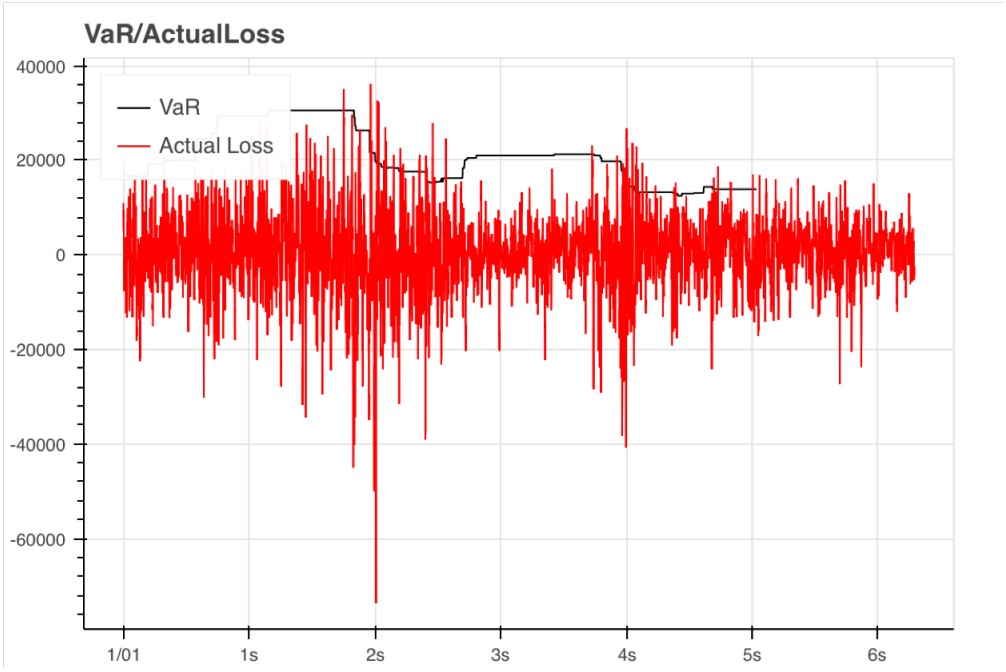

Image23: Portfolio-Stock&Option VaR/Actual Loss (Historical)

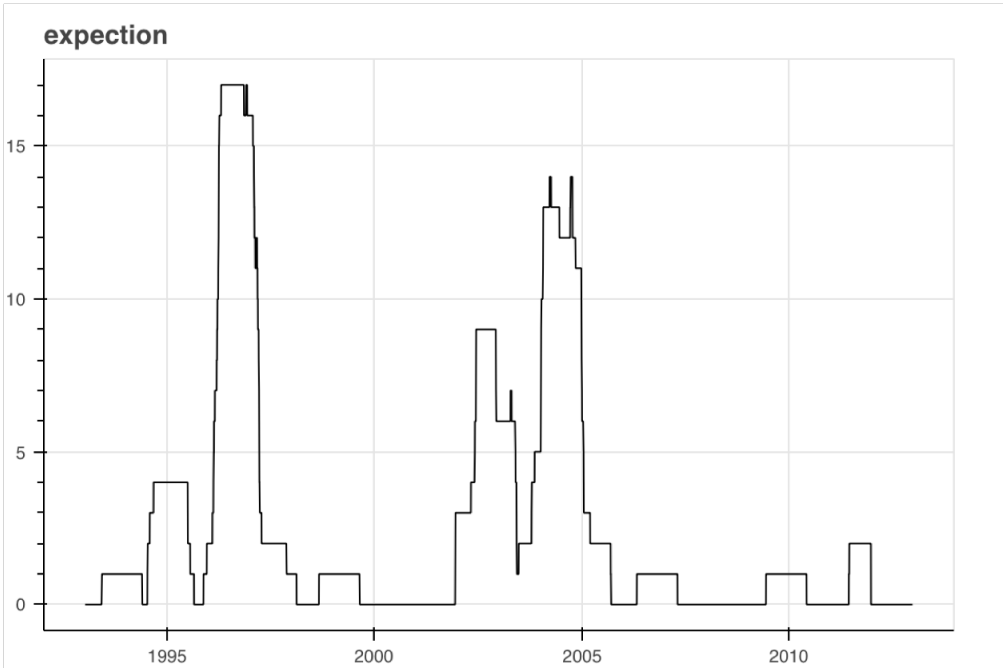

Image24: Portfolio-Stock&Option Backtest (Historical)

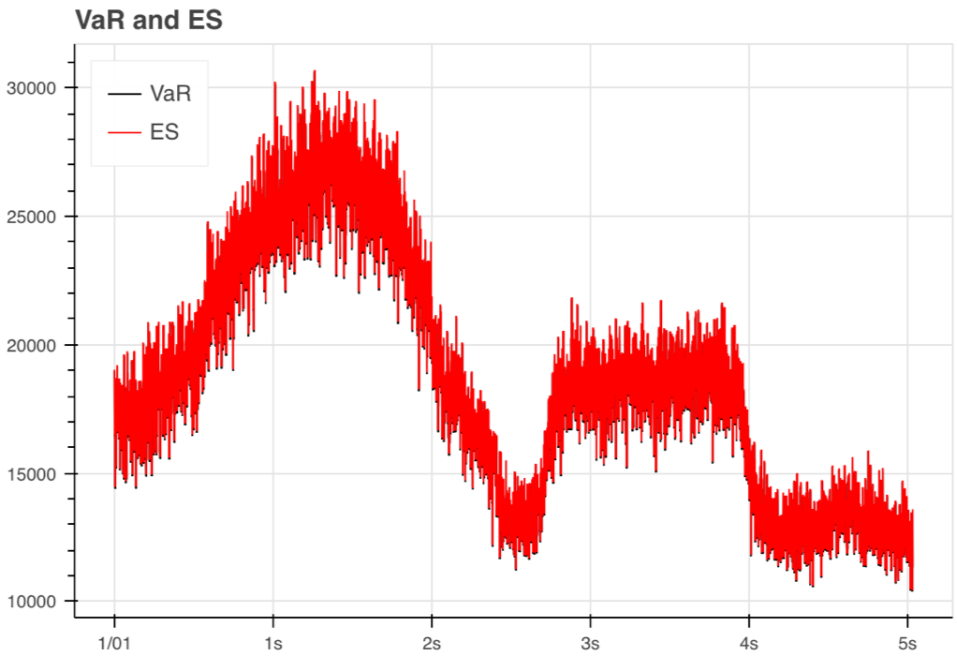

Image25: Portfolio-Stock&Option VaR/ES (Monte Carlo)

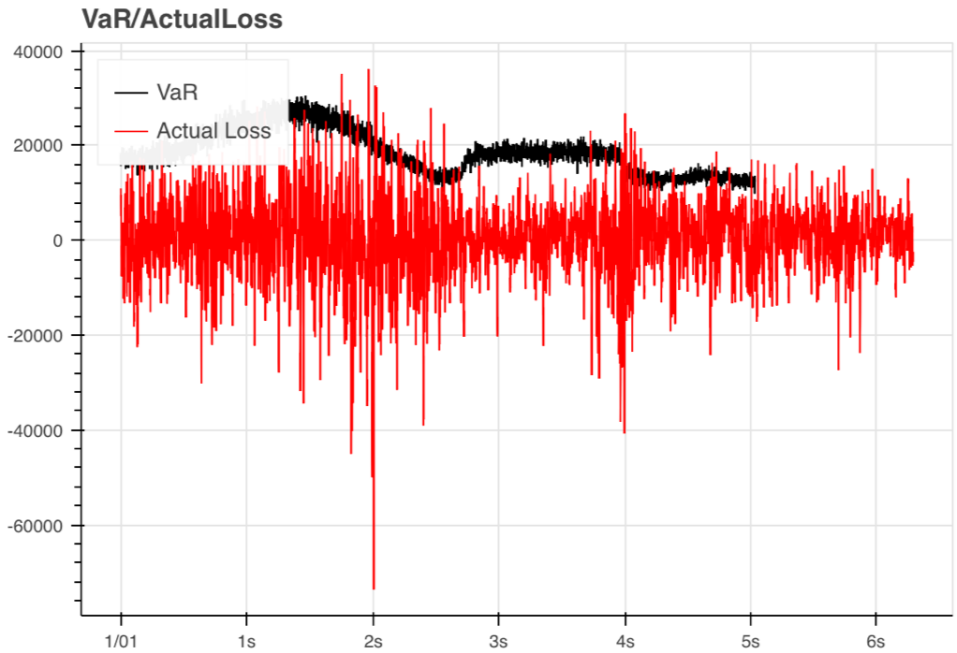

Image26: Portfolio-Stock&Option VaR/Actual Loss (Monte Carlo)

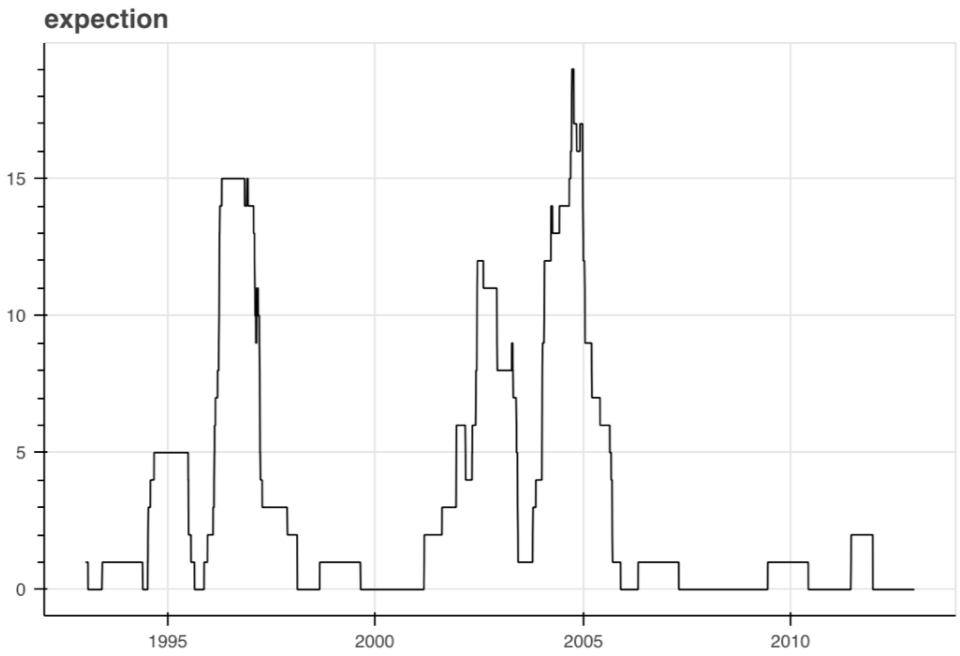

Image27: Portfolio-Stock&Option Backtest (Monte Carlo)

6.3.4 Homework Version (For Secondary Verification)

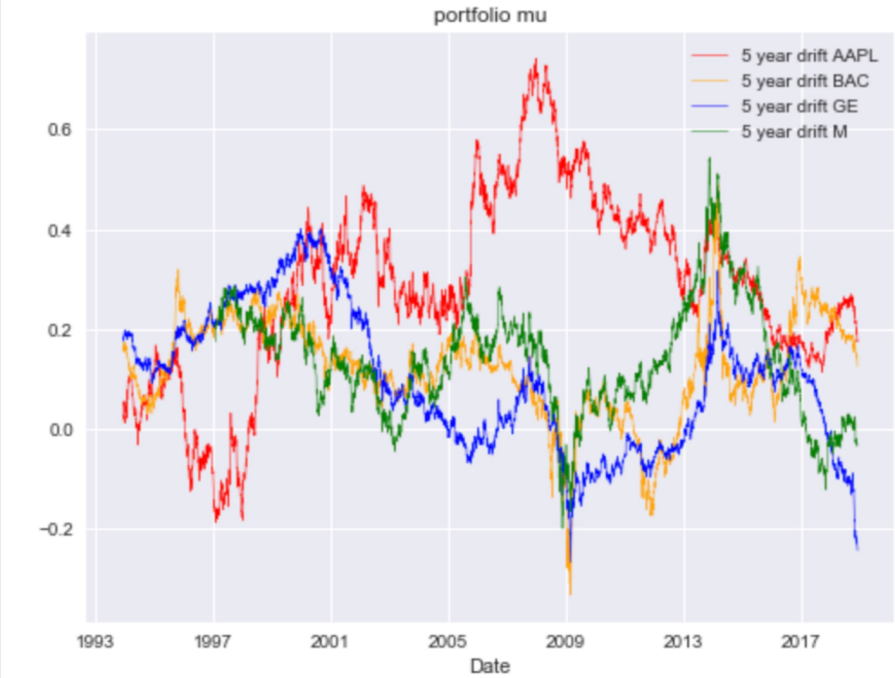

Image28: Stock Portfolio

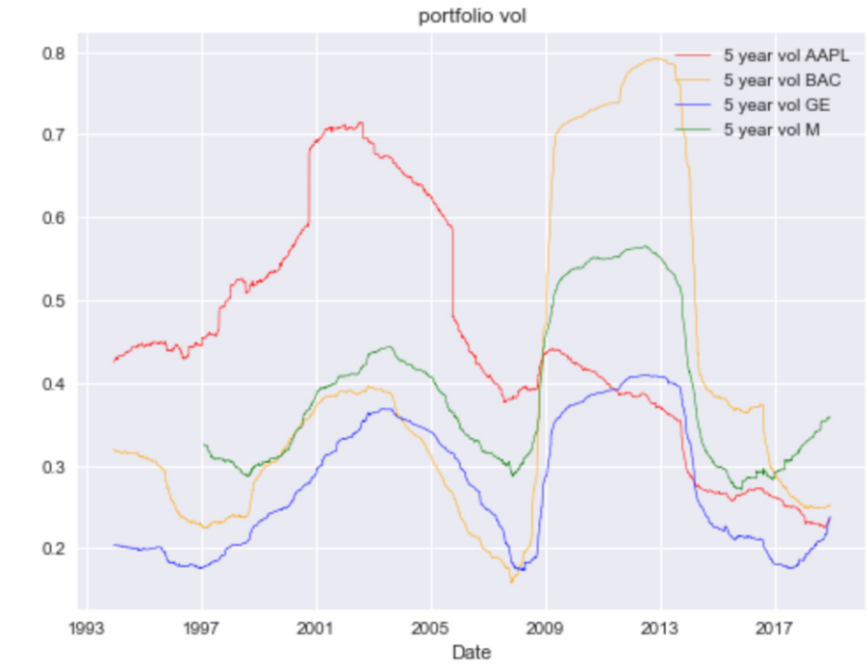

Image29: Stock Portfolio

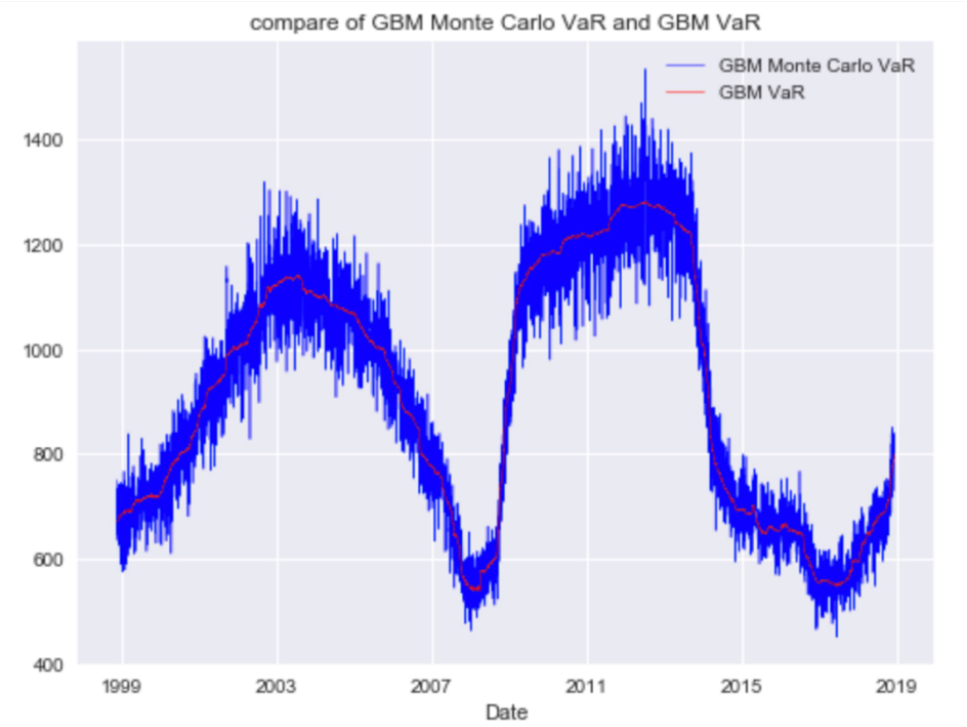

Image30: Stock Portfolio

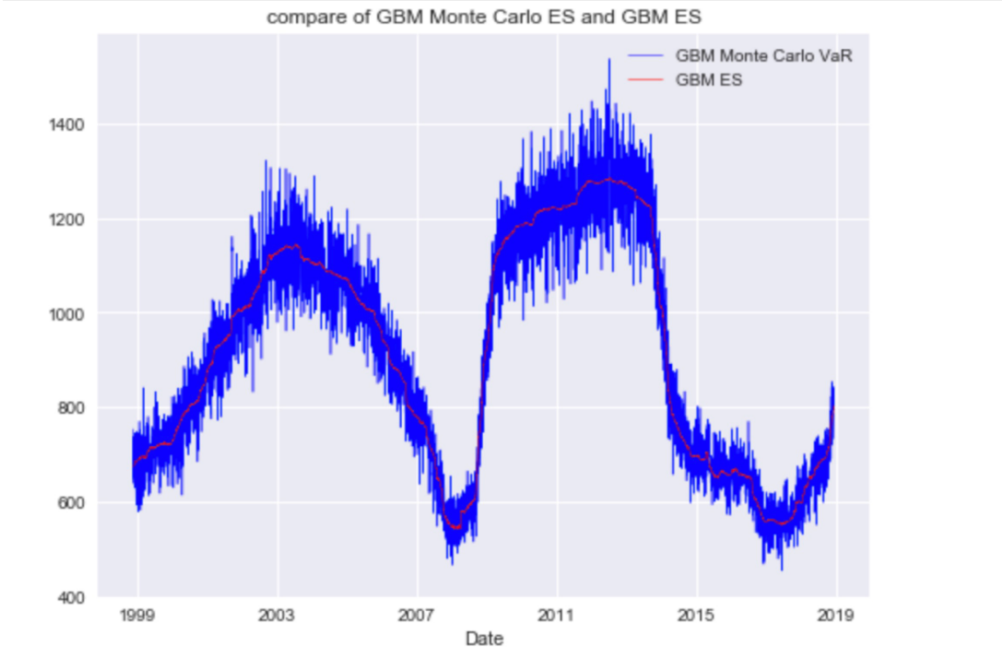

Image31: Stock Portfolio

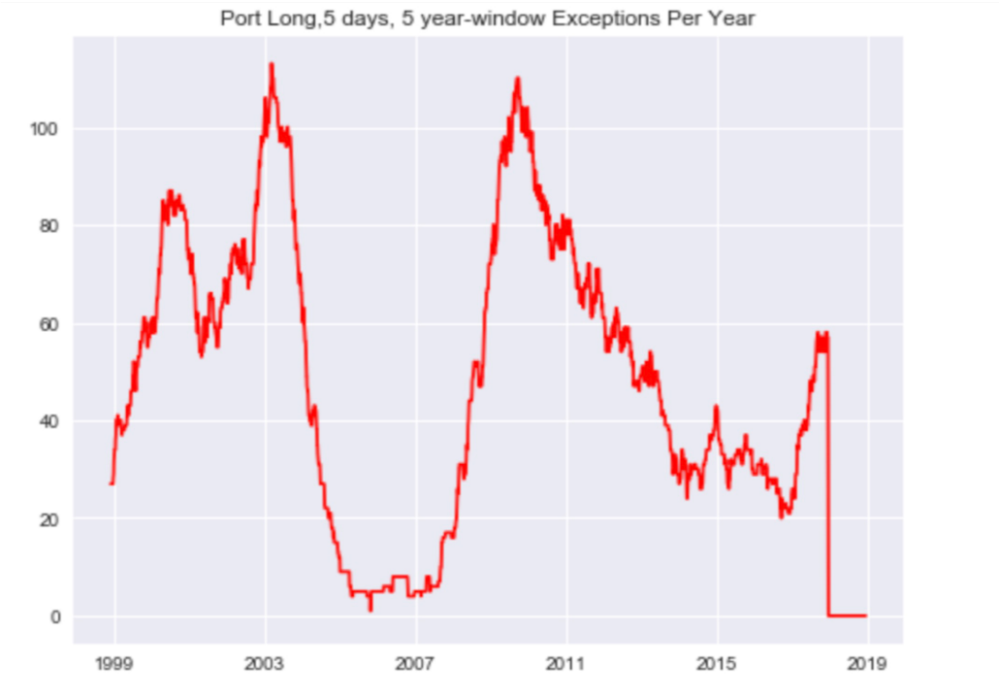

Image32: Stock Portfolio

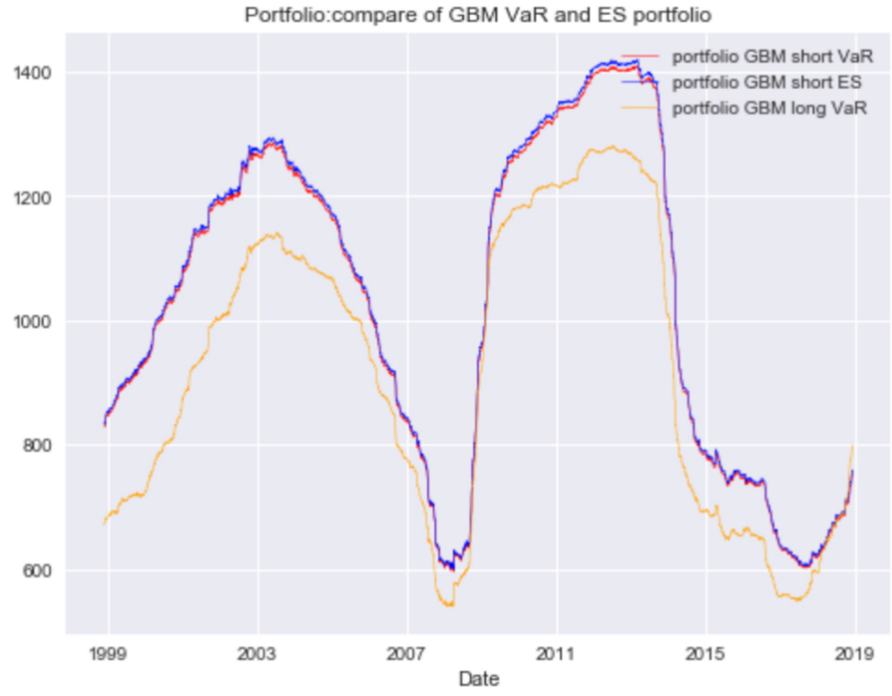

Image33: Stock Portfolio

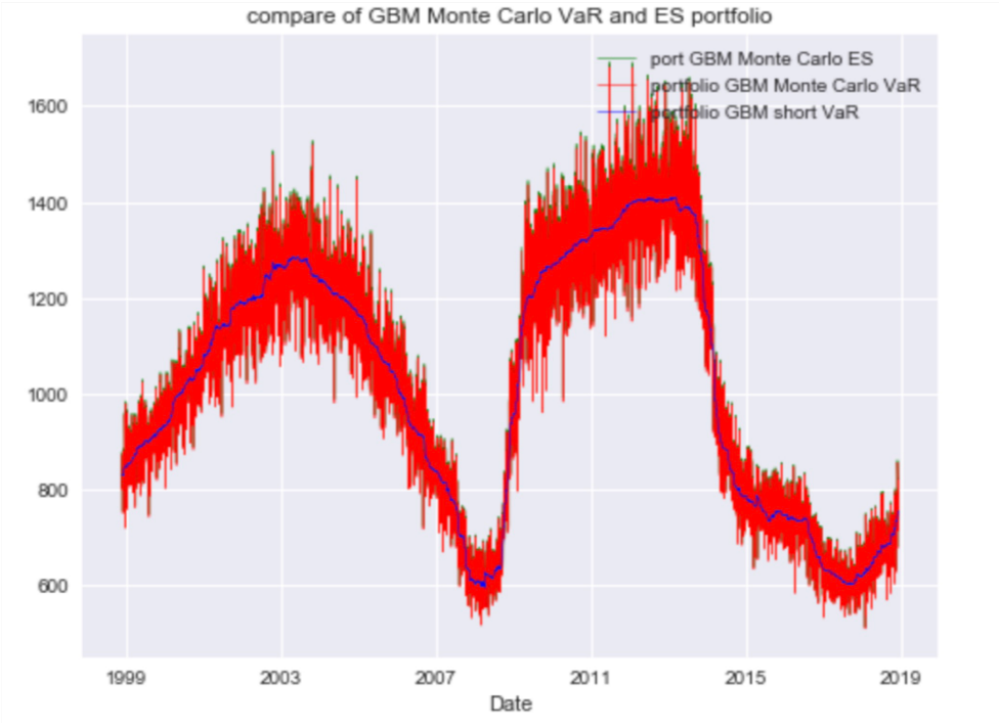

Image34: Stock Portfolio

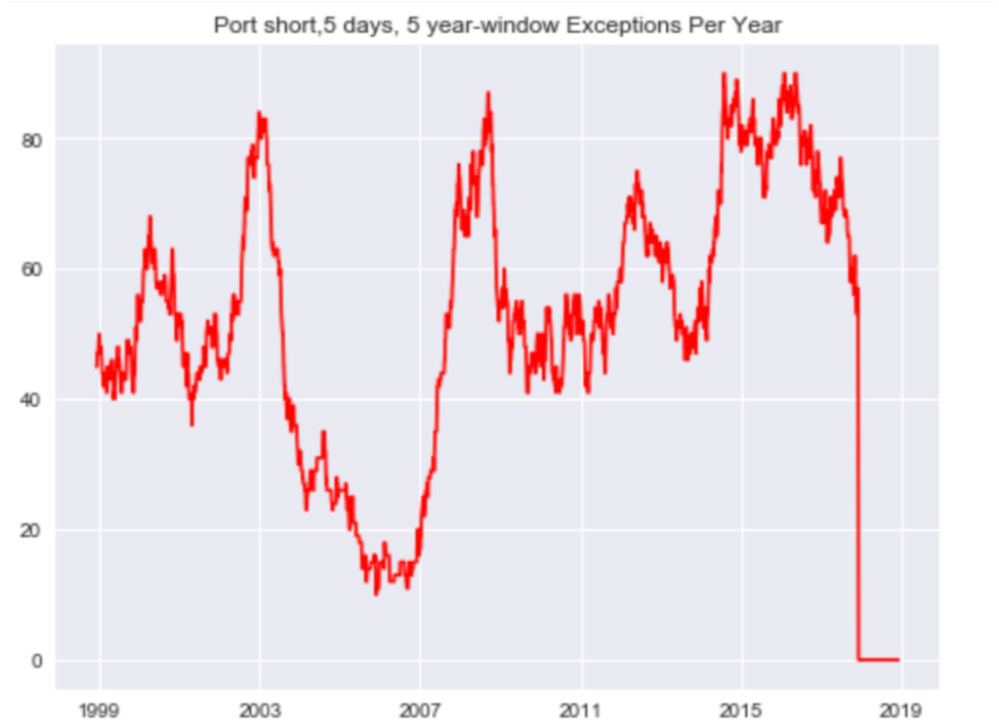

Image35: Stock Portfolio

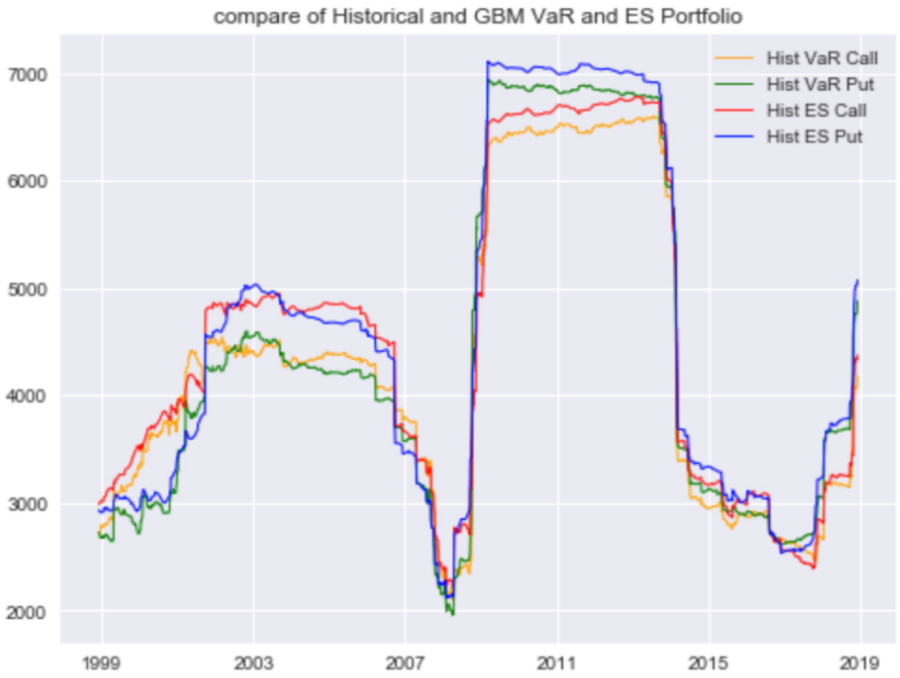

Image36: Stock Portfolio

7 Conclusion

Our portfolio VaR calculation model uses three different methods, which are Parametric VaR, historical VaR, and Monte Carlo VaR. As we know, there are difference assumptions and derivations, so each method has advantages and limitations. The advantage of this model is its portfolio flexibility since both stocks and options can be included in the portfolio. Additionally, there is no limitation on the long short positions of each element and also on the number of elements in a portfolio. Everyone can choose the best model based on different assumptions.

Users have to understand the details of each VaR calculation method to select their ideal VaR Calculation method. For parametric VaR, the advantages are fast calculation and easy to understand. The major limitation is that when the actual distribution of portfolio value is not normal or the portfolio payoff is not linear, this method will not be accurate. When the actual distribution is fat-tailed, parametric VaR is underestimated. The advantage of historical VaR is that it does not have any assumption of distribution. When portfolio distribution is not normal and historical performance correctly predicts the future performance, the historical VaR will give accurate VaR estimation. However, historical VaR method requires a larger amount of data and the calculation time is longer than parametric VaR method. Lastly, the advantage of Monte Carlo VaR is its insensitivity to linearity. Therefore, it is useful when options are included in the portfolio. Nevertheless, computation speed of Monte Carlo VaR is very low because it needs to generate a large number of simulations. Additionally, the approximation of option VaR is also another limitation of this model.

When the assumptions of the Black-Scholes model do not hold, historical and Monte Carlo methods will not accurate estimate the option VaR. And due to delta approach in parametric VaR, if the underlying stock price changes by a large amount, estimation will be less accurate. This can be improved by applying the Taylors expansion in option VaR estimation.

8 References

1. Dash User Guide and Documentation - Dash by Plotly, dash.plot.ly/getting-started.
2. Harper, David R. "An Introduction to Value at Risk (VAR)." Investopedia, Investopedia, 20 Nov. 2018, www.investopedia.com/articles/04/092904.asp.
3. Stein, H. J. (2018).Financial Risk Management and Regulation Lecture 1: Market Risk [PowerPoint slides]
4. Stein, H. J. (2018).Financial Risk Management and Regulation Lecture 2: Market Risk [PowerPoint slides]
5. Stein, H. J. (2018).Financial Risk Management and Regulation Lecture 3: Market Risk [PowerPoint slides]
6. Stein, H. J. (2018).Financial Risk Management and Regulation Lecture 4: Market Risk [PowerPoint slides]
7. Stein, H. J. (2018).Financial Risk Management and Regulation Lecture 5: Market Risk [PowerPoint slides]
8. "Value at Risk or Expected Shortfall." Quantdare, 21 Sept. 2018, www.quantdare.com/value-at-risk-or-expected-shortfall/.
9. "Value at Risk – Monte Carlo Simulation." Edited by Treasury Today 2018, Calculating WACC – an Art Not a Science | Treasury Today, www.treasurytoday.com/2002/02/value-at-risk-monte-carlo-simulation.

9 Appendix

In [151]:
```
# import packages

import pandas_datareader.data as web
import datetime
import pandas as pd
import numpy as np
import scipy.stats as stat
import dateutil.relativedelta
import sys

from bokeh.io import output_notebook, show
from bokeh.plotting import figure
from bokeh.layouts import column
from bokeh.models import Legend
from __future__ import division

import dash
import dash_core_components as dcc
import dash_html_components as html
```

In [152]:
```
AAPL = pd.read_csv("/Users/timxie/Downloads/AAPL.csv", usecols = ['Date','Close'])
AAPL['Date'] = pd.to_datetime(AAPL.Date).dt.date
AAPL = AAPL.set_index('Date')
BAC = pd.read_csv("/Users/timxie/Downloads/BAC.csv", usecols = ['Date','Close'])
BAC['Date'] = pd.to_datetime(BAC.Date).dt.date
BAC = BAC.set_index('Date')
GE = pd.read_csv("/Users/timxie/Downloads/GE.csv", usecols = ['Date','Close'])
GE['Date'] = pd.to_datetime(GE.Date).dt.date
GE = GE.set_index('Date')
M = pd.read_csv("/Users/timxie/Downloads/M.csv", usecols = ['Date','Close'])
M['Date'] = pd.to_datetime(M.Date).dt.date
M = M.set_index('Date')
```

In [153]:
```
### windowed
def window(price, time, size):
    logreturn = np.log(price/price.shift(1))
    vol = logreturn.rolling(window=size).std()/np.sqrt(time)
    mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
    para = pd.concat([price, mu, vol], axis=1)
    para.columns = ['price', 'mu', 'vol']
    para = para[len(para)-252*25:]
    return para

AAPL5 = window(AAPL,1/252,5*252)
BAC5 = window(BAC,1/252,5*252)
GE5= window(GE,1/252,5*252)
```

```
M5 = window(M,1/252,5*252)


#fig,ax=plt.subplots(figsize=(8,6))
#AAPL5.iloc[:,1].plot(color='red',linewidth=0.5,label='5 year drift AAPL')
#BAC5.iloc[:,1].plot(color='orange',linewidth=0.5,label='5 year drift BAC')
#GE5.iloc[:,1].plot(color='blue',linewidth=0.5,label='5 year drift GE')
#M5.iloc[:,1].plot(color='green',linewidth=0.5,label='5 year drift M')

#fig,ax=plt.subplots(figsize=(8,6))
#AAPL5.iloc[:,2].plot(color='red',linewidth=0.5,label='5 year vol AAPL')
#BAC5.iloc[:,2].plot(color='orange',linewidth=0.5,label='5 year vol BAC')
#GE5.iloc[:,2].plot(color='blue',linewidth=0.5,label='5 year vol GE')
#M5.iloc[:,2].plot(color='green',linewidth=0.5,label='5 year vol M')
```

In [154]:
```
porttwostock = pd.concat([AAPL,BAC,GE,M], axis = 1, join = 'inner')
porttwostock['Portfolio'] =porttwostock.iloc[:, 0]*10000/AAPL.iloc[2520]+ porttwosto

### window
def portfolio2(price, time, size):
    logreturn = np.log(price/price.shift(1))
    vol = logreturn.rolling(window=size).std()/np.sqrt(time)
    mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
    para = pd.concat([price, mu, vol], axis=1)
    para.columns = ['price', 'mu', 'vol']
    para = para[len(para)-252*25:]
    return para

portfolio5Ywindow= portfolio2(porttwostock.iloc[:, 2], 1/252,5*252)

### mu of 2 5 10
#fig,ax=plt.subplots(figsize=(8,6))
#portfolio5Ywindow.iloc[:,1].plot(color = "red", linewidth = 0.4, label = "5 year wi

###  vol of 2 5 10
#fig,ax=plt.subplots(figsize=(8,6))
#portfolio5Ywindow.iloc[:,2].plot(color = "green", linewidth = 0.4, label = "5 year
```

In [155]:
```
def option_loss_para():
```

```
  File "<ipython-input-155-7d549553809a>", line 1
    def option_loss_para():
                           ^
SyntaxError: unexpected EOF while parsing
```

In [ ]:
```
### Long
######  Assuming the portfolio follows the GBM ,  para VAR AND ES
```

```
        import scipy.stats as ss
        ### GBM of VaR and ES
        def GBMVE(price, time, size,S0,T,p1,p2,r):
            logreturn = np.log(price/price.shift(1))
            vol = logreturn.rolling(window=size).std()/np.sqrt(time)
            mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
            St=S0*exp(mu-0.5*vol**2)*T
            d1=((r + 0.5 * vol0 ** 2) * T) / (vol0 * np.sqrt(T))
            var = S0-S0*np.exp(vol*T**(0.5)*ss.norm.ppf(1-p1)+(mu-pow(vol,2)/2)*T)
            es = S0 * (1 - np.exp(mu *T)/(1-p2) * ss.norm.cdf(ss.norm.ppf(1-p2) - T**(0.5)*vol)
            para = pd.concat([price, mu, vol,var,es], axis=1)
            para.columns = ['price', 'mu', 'vol','var','es']
            return para

In [ ]: def Option_para(S0, r, sigma, T , opt_type):
            """opt_type: put or call """
            d1 = ((r + sigma**2 / 2) * T)/(sigma * np.sqrt(T))
            if opt_type=="Call":
                delta = ss.norm.cdf(d1)
                vega = ss.norm.pdf(d1)*S0*np.sqrt(T)
                return delta,vega
            else:
                delta = -ss.norm.cdf(-d1)
                vega = ss.norm.pdf(d1)*S0*np.sqrt(T)
            return delta,vega
        def VaR_option_para(p0,S0,delta,sigma0,mu,T,invest1,invest2,p):
            share1 = invest1/S0
            share2 = invest2/p0
            VaR= share2*p0+share1*S0+delta*share2*S0-share2*p0-(share1+delta*share2)*S0*np.exp
            return(VaR)
        def euro_vanilla_put(S, K, T, r, sigma):

            #S: spot price
            #K: strike price
            #T: time to maturity
            #r: interest rate
            #sigma: volatility of underlying asset

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

            put = (K * np.exp(-r * T) * ss.norm.cdf(-d2, 0.0, 1.0) - S * ss.norm.cdf(-d1, 0.0,

            return put

        def euro_vanilla_call(S, K, T, r, sigma):
            #S: spot price
```

```
        #K: strike price
        #T: time to maturity
        #r: interest rate
        #sigma: volatility of underlying asset

        d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        d2 = (np.log(S / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
        call = S * si.norm.cdf(d1, 0.0, 1.0)-K * np.exp(-r * T) * si.norm.cdf(d2, 0.0, 1.0)
        return call

    def est_para(price, windowlen,time):
        logreturn = np.log(price/price.shift(1))
        vol = logreturn.rolling(window=windowlen*252).std()/np.sqrt(time)
        mu = logreturn.rolling(window=windowlen*252).mean()/time + (vol**2)/2
        return logreturn, mu, vol

    def para_option(S0,r,T,windowlen,invest1,invest2,p,time,opt_type):
        logreturn,estmu,estvol = est_para(S0, windowlen,time)
        estdelta,estvega = Option_para(S0,r,estvol,T,opt_type)
        if opt_type == "Call":
            est_p0 = euro_vanilla_call(S0,S0,T,r,estvol)
        else:
            est_p0 = euro_vanilla_put(S0,S0,T,r,estvol)
        var = VaR_option_para(est_p0,S0,estdelta,estvol,estmu,T,invest1,invest2,0.99)
        return var

    paracallvar=para_option(portfolio5Ywindow['price'],0.005,5/252,5,40000,10000,0.99,5/252
    paraputvar=para_option(portfolio5Ywindow['price'],0.005,5/252,5,40000,10000,0.99,5/252
```

In [ ]:
```
def para_optionloss_his(v0,st,T,r,opt_type):
    sigma = portfolio5Ywindow['vol'][5*252:]
    #print(sigma)
    sigma0 = portfolio5Ywindow['vol'][5*252]
    #print(sigma0)
    mu = portfolio5Ywindow['mu']
    d1=((r + 0.5 * sigma0 ** 2) * T) / (sigma0 * np.sqrt(T))
    s0=portfolio5Ywindow['price'][5*252]
    #print(d1)
    if opt_type=="Call":
      delta = ss.norm.cdf(d1)
      vega=ss.norm.pdf(d1)*s0*np.sqrt(T)
    else:
      delta = -ss.norm.cdf(-d1)
      vega=ss.norm.pdf(-d1)*s0*np.sqrt(T)
    #print(delta)
    option_loss=v0*(delta*(st-s0)+vega*(sigma-sigma0))
    #print(st-s0)
    return(option_loss)
```

```
call_loss=para_optionloss_his(portfolio5Ywindow['price'][5*252],portfolio5Ywindow['pri
put_loss=para_optionloss_his(portfolio5Ywindow['price'][5*252],portfolio5Ywindow['price
```

In [ ]:
```
#### Call option VaR
logreturnp=pd.DataFrame(np.log(portfolio5Ywindow['price']/portfolio5Ywindow['price'].sh
A_his_VaR_yearspc = []
for i in range(len(logreturnp)-252*5):
    his_VaR_listp =logreturnp[i:i+252*5].sort_values(by='price')
    his_VaR_yearspc = 40000-40000*np.exp(his_VaR_listp.iloc[12])+min(call_loss.iloc[i:i
    A_his_VaR_yearspc.append(his_VaR_yearspc)
#A_his_VaR_yearspc

#### put option VaR
A_his_VaR_yearspp = []
for i in range(len(logreturnp)-252*5):
    his_VaR_listp =logreturnp[i:i+252*5].sort_values(by='price')
    his_VaR_yearspp =40000-40000*np.exp(his_VaR_listp.iloc[12])+min(put_loss.iloc[i:i+1
    A_his_VaR_yearspp.append(his_VaR_yearspp)
#A_his_VaR_yearspp
```

In [ ]:
```
#### call option es
logreturnp=pd.DataFrame(np.log(portfolio5Ywindow['price']/portfolio5Ywindow['price'].sh
A_his_ES_yearspc = []
for i in range(len(logreturnp)-252*5):
    his_VaR_listp =logreturnp[i:i+252*5].sort_values(by='price')
    hist_ES_listp =40000-40000*np.exp(his_VaR_listp[0:31])
    hist_ES_yearsp = np.mean(hist_ES_listp)+np.mean(call_loss.iloc[i:31+i])
    A_his_ES_yearspc.append(hist_ES_yearsp)
#A_his_ES_yearspc

#### put option es
A_his_ES_yearspp = []
for i in range(len(logreturnp)-252*5):
    his_VaR_listp =logreturnp[i:i+252*5].sort_values(by='price')
    hist_ES_listp =40000-40000*np.exp(his_VaR_listp[0:31])
    hist_ES_yearsp = np.mean(hist_ES_listp)+np.mean(put_loss.iloc[i:31+i])
    A_his_ES_yearspp.append(hist_ES_yearsp)
#A_his_ES_yearspp
```

In [ ]:
```
###  Monte Carlo
random=[]
for i in range(GBMp.shape[0]):
    mcmc=norm.ppf(np.random.rand())
    random.append(mcmc)
def GBMMTVE(price, time, size,S0,T,p1,p2,random):
    logreturn = np.log(price/price.shift(1))
    vol = logreturn.rolling(window=size).std()/np.sqrt(time)
```

```
        eps = vol*random
        mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
        var = S0-S0*np.exp(vol*T**(0.5)*ss.norm.ppf(1-p1)+(mu+eps-pow(vol,2)/2)*T)
        es = S0 * (1 - np.exp((mu+eps) *T)/(1-p2) * ss.norm.cdf(ss.norm.ppf(1-p2) - T**(0.5
        para = pd.concat([price, mu, vol,var,es], axis=1)
        para.columns = ['price', 'mu', 'vol','var','es']
        return para
    GBMMTp=GBMMTVE(portfolio5Ywindow['price'],1/252,5*252,10000,5/252,0.99,0.975,random)
```

In [ ]:
```
    ### Backtest

    ### GBM of VaR and ES
    def GBMVESHORT(price, time, size,S0,T,p1,p2):
        logreturn = np.log(price/price.shift(1))
        vol = logreturn.rolling(window=size).std()/np.sqrt(time)
        mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
        var = -(S0-S0*np.exp(vol*T**(0.5)*ss.norm.ppf(1-p1)+(mu-pow(vol,2)/2)*T))
        es=(0.975*S0 * (1 - np.exp(mu *T)/(1-0.025) * ss.norm.cdf(ss.norm.ppf(1-0.025) - T
        para = pd.concat([price, mu, vol,var,es], axis=1)
        para.columns = ['price', 'mu', 'vol','var','es']
        return para

    def GBMVELONG(price, time, size,S0,T,p1,p2):
        logreturn = np.log(price/price.shift(1))
        vol = logreturn.rolling(window=size).std()/np.sqrt(time)
        mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
        var = S0-S0*np.exp(vol*T**(0.5)*ss.norm.ppf(1-p1)+(mu-pow(vol,2)/2)*T)
        es = S0 * (1 - np.exp(mu *T)/(1-p2) * ss.norm.cdf(ss.norm.ppf(1-p2) - T**(0.5)*vol
        para = pd.concat([price, mu, vol,var,es], axis=1)
        para.columns = ['price', 'mu', 'vol','var','es']
        return para
```

In [ ]:
```
    ### Portfolio Long
    def portfolio(price, time, weight):
        logreturn = np.log(price/price.shift(1))
        vol = logreturn.ewm(alpha = 1 - weight).std()/np.sqrt(time)
        mu = logreturn.ewm(alpha = 1 - weight).mean()/time + (vol**2)/2
        para = pd.concat([price, mu, vol], axis=1)
        para.columns = ['price', 'mu', 'vol']
        para = para[len(para)-252*25:]
        para = para.loc[para.index >= pd.to_datetime('1997-08-07').date()]
        return para

    GBMLONG = GBMVELONG(portfolio5Ywindow.iloc[:,0],1/252,5*252,10000,5/252,0.99,0.975)
    gbmlongvar=GBMLONG['var'][252*5:]
```

In [ ]:
```
    S0=40000
    numberexception=[]
```

```
    for i in range(len(gbmlongvar)):
        window=portfolio5Ywindow['price'][(len(gbmlongvar)-i-252):(len(gbmlongvar)-i-1)]
        exception=0
        for j in range(len(window) - 4):
            share=S0/window[j]
            price0=window[j]
            pricet = window[j+4]
            loss = S0 - pricet * share
            if loss > gbmlongvar[len(gbmlongvar)-i-252 + j]:
                exception = exception + 1
            else:
                exception = exception

        numberexception.append(exception)
```

In [ ]:
```
#### Portolio vs realized loss
loss=[ ]
for i in range(len(gbmlongvar)):
    price0 = portfolio5Ywindow['price'][len(gbmlongvar)-i+5]
    pricet = portfolio5Ywindow['price'][len(gbmlongvar)-i+1]
    share = S0/price0
    loss.append((price0 - pricet)*share)
```

In [ ]:
```
### Short Position

### PORTFOLIO
GBMSHORTp=GBMVESHORT(portfolio5Ywindow['price'],1/252,5*252,10000,5/252,0.01,0.975)
GBMlongp=GBMVE(portfolio5Ywindow['price'],1/252,5*252,10000,5/252,0.99,0.975)
GBMlongvarp=GBMlongp.iloc[:,3]
GBMlongesp=GBMlongp.iloc[:4]
GBMVARSHORTp=GBMSHORTp.iloc[:,3]
GBMESSHORTp=GBMSHORTp.iloc[:,4]
```

In [ ]:
```
random=[]
for i in range(GBMSHORTp.shape[0]):
    mcmc=norm.ppf(np.random.rand())
    random.append(mcmc)

def GBMMTshort(price, time, size,S0,T,p1,p2,random):
    logreturn = np.log(price/price.shift(1))
    vol = logreturn.rolling(window=size).std()/np.sqrt(time)
    eps = vol*random
    mu = logreturn.rolling(window=size).mean()/time + (vol**2)/2
    var = -(S0-S0*np.exp(vol*T**(0.5)*ss.norm.ppf(1-p1)+(mu+eps-pow(vol,2)/2)*T))
    es = (0.975*S0*(1 - np.exp((mu+eps)*T)/(1-0.025) * ss.norm.cdf(ss.norm.ppf(1-0.025)
    para = pd.concat([price, mu, vol,var,es], axis=1)
    para.columns = ['price', 'mu', 'vol','var','es']
    return para
GBMMTp=GBMMTshort(portfolio5Ywindow['price'],1/252,5*252,10000,5/252,0.01,0.975,random)
```

```
In [ ]: ### backtest

        ### Portfolio short position
        gbmshortvar=GBMSHORTp['var'][252*5:]

        S0=40000
        numberexception2=[]
        for i in range(0,len(gbmshortvar)):
            window2=portfolio5Ywindow['price'][(len(gbmshortvar)-i-252):(len(gbmshortvar)-i)]
            exception2=0
            for j in range(1,len(window2) - 4):
                share=S0/window2[j]
                price0=window2[j]
                pricet = window2[j+4]
                loss = (pricet-price0)* share
                loss = -loss
                if loss < -gbmshortvar[i]:
                    exception2 = exception2 + 1
                else:
                    exception2 = exception2

            numberexception2.append(exception2)

In [ ]: #### port short vs realized loss
        loss=[ ]
        for i in range(0,len(gbmshortvar)):
            price0 = portfolio5Ywindow['price'][len(gbmshortvar)-i+5]
            pricet = portfolio5Ywindow['price'][len(gbmshortvar)-i+1]
            share = S0/price0
            loss.append(-(price0 - pricet)*share)

In [ ]: def para_optionloss(v0,st,T,r,opt_type):
            sigma = portfolio5Ywindow['vol'][5*252:]
            #print(sigma)
            sigma0 = portfolio5Ywindow['vol'][5*252]
            #print(sigma0)
            mu = portfolio5Ywindow['mu']
            d1=((r + 0.5 * sigma0 ** 2) * T) / (sigma0 * np.sqrt(T))
            s0=portfolio5Ywindow['price'][5*252]
            #print(d1)
            if opt_type=="Call":
              delta = ss.norm.cdf(d1)
              vega=ss.norm.pdf(d1)*s0*np.sqrt(T)
            else:
              delta = -ss.norm.cdf(-d1)
              vega=ss.norm.pdf(-d1)*s0*np.sqrt(T)
            #print(delta)
            option_loss=v0*(delta*(st-s0)+vega*(sigma-sigma0))
```

```
            #print(st-s0)
            return(option_loss)
```

In [ ]:
```
def euro_vanilla_call(S, K, T, r, sigma):
    #S: spot price
    #K: strike price
    #T: time to maturity
    #r: interest rate
    #sigma: volatility of underlying asset

    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = (np.log(S / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    call = S * si.norm.cdf(d1, 0.0, 1.0)-K * np.exp(-r * T) * si.norm.cdf(d2, 0.0, 1.0)
    return call

def euro_vanilla_put(S, K, T, r, sigma):

    #S: spot price
    #K: strike price
    #T: time to maturity
    #r: interest rate
    #sigma: volatility of underlying asset

    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = (np.log(S / K) + (r - 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    put = (K * np.exp(-r * T) * si.norm.cdf(-d2, 0.0, 1.0) - S * si.norm.cdf(-d1, 0.0,
    return put

def VaR_monte(mu,vol,option_type):
    T=5/252
    w1 = norm.ppf(np.random.rand(10000))
    w1 = w1*(T**0.5)
    St = portfolio5Ywindow['price'][-1]*np.exp((mu-(vol**2)/2)*T + vol*w1)
    if option_type == "Call":
        Vt = euro_vanilla_call(portfolio5Ywindow['price'][-1],St,1,0.005,portfolio5Ywi

        loss = 30000/portfolio5Ywindow['price'][-1]*(portfolio5Ywindow['price'][-1]-St)

        VaR = np.percentile(loss,99)
    else:
        Vt = euro_vanilla_put(portfolio5Ywindow['price'][-1],St,1,0.005,portfolio5Ywinc
        loss = 30000/portfolio5Ywindow['price'][-1]*(portfolio5Ywindow['price'][-1]-St)
        VaR = np.percentile(loss,99)

    return VaR
VaR_mc = list(map(VaR_monte,portfolio5Ywindow['mu'][5*252:],portfolio5Ywindow['vol'][5*
VaR_mp = list(map(VaR_monte,portfolio5Ywindow['mu'][5*252:],portfolio5Ywindow['vol'][5*
```

In [ ]:
```
# Price plot
```

```
        def plot_price(price, length):
            data = price[:length]
            output_notebook()
            plot = figure(width=600, height=400,
                          title = "Historical Prices", x_axis_type="datetime")
            plot.line(data.index, data)
            plot.title.text_font_size = '11pt'
            show(plot)
```

```
In [ ]: # Calculate estimated parameters for GBM based on x year (in days) rolling windows
```

```
        def gbm_est(prices, window_days):
            rtn = -np.diff(np.log(prices))
            rtnsq = rtn * rtn
            mubar = list(reversed(np.convolve(rtn,
                          np.ones((window_days,))/window_days, mode='valid')))
            x2bar = list(reversed(np.convolve(rtnsq,
                          np.ones((window_days,))/window_days, mode='valid')))
            var = x2bar - np.square(mubar)
            sigmabar = np.sqrt(np.maximum(var, np.zeros(len(var))))
            sigma = sigmabar / np.sqrt(1/252)
            mu = np.array(mubar)*252 + np.square(sigma)/2
            return rtn, mu, sigma, np.array(mubar), sigmabar
```

```
In [ ]: # Parameters plot
```

```
        def plot_parameters(price):
            rtn_2, mu_2, sigma_2, mubar_2, sigmabar_2 = gbm_est(price, 2*252)
            rtn_5, mu_5, sigma_5, mubar_5, sigmabar_5 = gbm_est(price, 5*252)
            rtn_10, mu_10, sigma_10, mubar_10, sigmabar_10 = gbm_est(price, 10*252)
            length = min(len(mu_2), len(mu_5), len(mu_10),
                          len(sigma_2), len(sigma_5), len(sigma_10))
            mu = pd.DataFrame({'Mu_2': mu_2[:length],
                               'Mu_5': mu_5[:length],
                               'Mu_10': mu_10[:length]},
                              index = price.index[:length])
            sigma = pd.DataFrame({'Sigma_2': sigma_2[:length],
                                  'Sigma_5': sigma_5[:length],
                                  'Sigma_10': sigma_10[:length]},
                                 index = price.index[:length])
            output_notebook()
            pmu = figure(width=600, height=400,
                          title = "Parameter - Mu", x_axis_type="datetime")
            pmu.line(mu.index, mu['Mu_2'], legend = '2-year roling window')
            pmu.line(mu.index, mu['Mu_5'],
                     color = 'green', legend = '5-year roling window')
            pmu.line(mu.index, mu['Mu_10'],
```

```
                color = 'orange', legend = '10-year roling window')
        pmu.title.text_font_size = '11pt'
        pmu.legend.location = 'bottom_right'
        pmu.legend.background_fill_alpha = 0.5
        psigma = figure(width=600, height=400,
                        title = "Parameter - Sigma", x_axis_type="datetime")
        psigma.line(mu.index, sigma['Sigma_2'],
                    legend = '2-year roling window')
        psigma.line(mu.index, sigma['Sigma_5'],
                    color = 'green', legend = '5-year roling window')
        psigma.line(mu.index, sigma['Sigma_10'],
                    color = 'orange', legend = '10-year roling window')
        psigma.title.text_font_size = '11pt'
        psigma.legend.location = 'top_right'
        psigma.legend.background_fill_alpha = 0.5
        plot = column(pmu, psigma)
        show(plot)
```

In [ ]:
```
# Calculate VaR and ES using parametric method

def parametric(v0, mu, sigma, VaR_prob, ES_prob, t):
    VaR = v0 - v0 * np.exp(
        sigma * np.sqrt(t) * stat.norm.ppf(1-VaR_prob) + (mu - np.square(sigma)/2) * t)
    ES = v0 * (1 - np.array(
        stat.norm.cdf(stat.norm.ppf(1-ES_prob) - np.sqrt(t)*sigma))
               * np.array(np.exp(mu*t)/(1-ES_prob)))
    return VaR, ES
```

In [ ]:
```
# Calculate VaR and ES using historical method

def historical(v0, price, VaR_prob, ES_prob, window_days, horizon_days):
    npaths = window_days - horizon_days
    ntrials = len(price) - window_days
    price_log = np.log(price)
    return_xdays = np.array(
        price_log[:(len(price_log)-horizon_days)]) - np.array(price_log[5:])
    price_res = v0 * np.exp(return_xdays)
    scenarios = np.zeros(shape=(npaths,ntrials))
    for i in range(ntrials):
        scenarios[0:npaths,i] = price_res[i:i+npaths]
    scenarios_sorted = np.sort(scenarios, axis=0)
    VaR = v0 - scenarios_sorted[np.ceil((1-VaR_prob)*npaths).astype(int) - 1]
    ES = v0 - np.mean(
        scenarios_sorted[0:(np.ceil((1-ES_prob)*npaths).astype(int))], axis=0)
    return VaR, ES
```

In [ ]:
```
# Calculate VaR and ES using Monte Carlo method
```

```
        def monte_carlo(v0, price, mu, sigma, VaR_prob, ES_prob, window_days, horizon):
            npaths = 5000
            ntrials = len(price) - window_days
            p1 = np.zeros(shape=(npaths,ntrials))
            for i in range(ntrials):
                tv = np.ones(shape =(npaths,1))*horizon
                bm = np.sqrt(horizon) * np.random.randn(npaths,1)
                y = v0 * np.exp(sigma[i] * bm - (mu[i] + sigma[i]*sigma[i]/2) * tv)
                p1[:,i] = y[:,0]
            p2 = np.sort(p1,axis = 0)
            VaR = v0 - p2[np.ceil((1-VaR_prob)*npaths).astype(int) - 1]
            ES = v0 - np.mean(p2[0:(np.ceil((1-ES_prob)*npaths).astype(int))], axis=0)
            return VaR, ES
```

```
In [ ]: # VaR/ES plot
```

```
        def plot_risk(v0, price, VaR_prob, ES_prob, method, window, horizon, plot_length):
            if method == 'Parametric Method':
                rtn, mu, sigma, mubar, sigmabar = gbm_est(price, window*252)
                VaR, ES = parametric(v0, mu, sigma, VaR_prob, ES_prob, horizon)
            elif method == 'Historical Method':
                VaR, ES = historical(v0, price, VaR_prob,
                                     ES_prob, int(window*252), int(horizon*252))
            elif method == 'Monte Carlo':
                rtn, mu, sigma, mubar, sigmabar = gbm_est(price, window*252)
                VaR, ES = monte_carlo(v0, price, mu, sigma,
                                      VaR_prob, ES_prob, window*252, horizon)
            else:
                sys.exit('Error!')

            length = min(len(VaR), len(ES), plot_length)
            VaR_ES = pd.DataFrame({'VaR': VaR[:length], 'ES': ES[:length]},
                                  index = price.index[:plot_length])
            plot = figure(width=600, height=400, title = "VaR/ES", x_axis_type="datetime")
            plot.line(VaR_ES.index, VaR_ES['VaR'], color = 'green', legend = 'VaR')
            plot.line(VaR_ES.index, VaR_ES['ES'], color = 'blue', legend = 'ES')
            plot.legend.location = 'bottom_left'
            plot.title.text_font_size = '11pt'

            share_change = np.divide(price[:(len(price)-int(horizon*252))],
                                     price[int(horizon*252):])
            loss = v0 - share_change * v0
            length_loss = min(len(loss), len(VaR), plot_length)
            test = pd.DataFrame({'VaR': VaR[int(horizon*252):length_loss],
                                 'Loss': loss[:(length_loss-int(horizon*252))]},
                                index = price.index[int(horizon*252):length_loss])
            plot_test = figure(width=600, height=400,
                               title = "VaR/ActualLoss", x_axis_type="datetime")
```

```
        plot_test.line(test.index, test['VaR'], color = 'green', legend = 'VaR')
        plot_test.line(test.index, test['Loss'], color = 'blue', legend = 'Actual Loss')
        plot_test.legend.location = 'bottom_left'
        plot_test.title.text_font_size = '11pt'
        output_notebook()
        show(column(plot,plot_test))
```

In [ ]:
```
# Black Scholes method to calculate put option price

def bs_put(stock, rf, sigma, strike, maturity):
    sigrt = 1/(sigma*np.sqrt(maturity))
    sig2 = sigma*sigma/2
    lsk = np.log(stock/strike)
    ert = np.exp(-rf*maturity)
    d1 = sigrt*(lsk+(rf+sig2)*maturity)
    d2 = sigrt*(lsk+(rf-sig2)*maturity)
    pr = stat.norm.cdf(-d2)*strike*ert-stat.norm.cdf(-d1)*stock
    return pr
```

In [ ]:
```
# Compute MC VaR for portfolio of a stock and a put option.
# % the stocks, assuming option implied vols are unchanged.

def option_mc(s0, mu, sigma, rf, iv, strike, mat, nstocks, nputs, VaR_prob, horizon):
    npaths = 1000000
    tv = np.ones(shape =(npaths,1))*horizon
    bm = np.sqrt(horizon) * np.random.randn(npaths,1)
    st = s0 * np.exp(sigma * bm - (mu + sigma*sigma/2) * tv)
    vtStock = st * nstocks
    v0Stock = s0 * nstocks
    putt = bs_put(st, rf, iv, strike, mat-horizon)
    vtPut = nputs * putt
    put0 = bs_put(s0, rf, iv, strike, mat)
    v0Put = nputs * put0
    loss = v0Stock + v0Put - (vtStock + vtPut)
    VaR = np.percentile(loss, 100*VaR_prob)
    return VaR
```

In [ ]:
```
# Option portfolio calculations

def options_cal(options, rf, mat, imp_vol, v0, liq_rate, VaR_prob, window, horizon):
    rtn, mu, sigma, mubar, sigmabar = gbm_est(options, window*252)
    mu = mu[0]
    sigma = sigma[0]
    VaR_1, ES_1 = parametric(v0, mu, sigma, VaR_prob, 0.975, horizon)
    s0 = options[0]
    strike = options[0]
    nstocks = v0 * (1-liq_rate) / s0
    put0 = bs_put(s0, rf, imp_vol, strike, mat)
```

```
nputs = v0 * liq_rate / put0
VaR_2 = option_mc(s0, mu, sigma, rf,
                  imp_vol, strike, mat, nstocks, nputs, VaR_prob, horizon)
reduction = 100*(1-VaR_2/VaR_1)
return s0, nstocks, put0, nputs, VaR_1, VaR_2, reduction
```