Risk Management and Regulation

- Final Project Report -

Project Report

Duanhong Gao/UNI: dg2896
Xuxu Tang/UNI: xt2172

Columbia University
Fall 2016

Contents

Chapter 1

Introduction

1.1 Objective

The objective of this project is to construct a risk calculation system which includes parametric VaR, historical VaR, and Monte Carlo VaR estimations of a portfolio consisting of stocks and options. This model also include the back test results of each estimated VaR.

1.2 General Description

The input of this system is historical data of stock prices. 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. A detailed description of each VaR calculation method is included in second 2. A detailed description of software design is also included in the project so that users can have an in depth understanding of this system.

To test the risk calculation system, we take historical data of a portfolio as input. The portfolio contains long and short positions of stocks and options. The output of this example is the estimated daily parametric VaR, historical VaR, and Monte Carlo VaR in a time window. Back test results of each method are also included in this project to evaluate and compare each method.

1.3 Advantages and Limitations

The major advantage of this system is flexibility. There is no constraints on the long shot positions of each stock and option in the portfolio. The number of stocks and options in the portfolio is also not limited. Users are able to obtain estimated VaR of all three methods and select the most suitable one.

Each method in this system has different advantages and limitations. A detailed comparison of each method is included in section 2. A common limitation of this system is that it does not take market option prices. Instead, it uses the Black-Scholes model for option pricing. In parametric VaR estimation, the system uses a delta approach approximation for option VaR calculation. This approximation will not be accurate if change of underlying stock price is large.

Chapter 2

Model Description

In this project, we are calculating the value at risk of a portfolio using three different methods. The methods we use are parametric VaR, historical VaR, and Monte Carlo VaR. Each of them is based on different assumptions and different calculation. In this section, we are going to demonstrate the assumptions, input variables, derivations, and pros and cons of each type of VaR calculation. Since this model can be used for portfolios containing options, a section of option VaR approximation is also included.

2.1 Model Assumptions

2.1.1 Parametric VaR

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.

2.1.2 Historical VaR

There is no assumption of distribution needed for calculating historical VaR for a portfolio, a stock, or an option. In this case, we simply forecast VaR based on the asset's historical performance. Therefore, we need to assume that asset performance in the future is consistent with the asset performance in the past. The major assumption of historical VaR is that historical price distribution can represent

future price distribution accurately. This assumption holds for all historical VaR calculations including stocks, options, and portfolios.

There two methods to calculate historical changes in historical VaR. The methods are absolute changes and relative changes. The assumption under absolute changes is that the asset's value follows an Arithmetic Brownian Motion with a constant absolute volatility. The assumption under relative changes is that the asset value follows a Geometric Brownian Motion with constant relative volatility.

2.1.3 Monte Carlo VaR

Monte Carlo VaR is a numerical method for VaR calculation. When calculating Monte Carlo VaR, we need to make an assumption that the value of the asset follows certain distribution so that we can simulate future values based on that assumption. In most cases, stock prices are assumed to follow a Geometric Brownian Motion and option values can be calculated using the Black-Scholes model and the simulated underlying stock prices. When assuming portfolio values follow a normal distribution, Monte Carlo VaR is approximately the same as parametric VaR.

2.2 Model Inputs

2.2.1 Parametric VaR

To calculate the parametric VaR of portfolio by assuming portfolio value follows a normal distribution, the mean and standard deviation of portfolio returns are needed. To calculate the mean and standard deviation of portfolio returns, we need to obtain mean returns and covariance matrix of all returns of components in the portfolio.

2.2.2 Historical VaR

To calculate the historical VaR of a portfolio, only historical values of each component in the portfolio is needed.

2.2.3 Monte Carlo VaR

To calculate the Monte Carlo VaR of a portfolio, the parameters of each component distribution are needed and a correlation matrix of all components is also needed in Monte Carlo simulation. Parameters and correlation matrix are usually obtained

from historical performances of all components of the portfolio.

2.3 Mathematical Description

2.3.1 Parametric VaR

In this section, we assume there are two stock in this portfolio. Let V_t denote the portfolio value at time t, let a and b denote the number of shares of each stock, and let $S_{1,t}$ and $S_{2,t}$ denote stock prices at time t. Then the value of the portfolio at time t is:

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

By assuming each stock in this portfolio follows a Geometric Brownian Motion with parameters μ_i and σ_i, we get:

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

$dW_1 dW_2 = \rho dt$, where ρ is the correlation between the two stocks.

At time t, we can get the following information about the portfolio V_t:

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

$E[V_t{}^2] = E[a^2 S_{1,t}{}^2 + b^2 S_{2,t}{}^2 + 2abS_{1,t}S_{2,t}]$,

$var[V_t] = E[V_t{}^2] - E[V_t]^2$, and

$sd[V_t] = \sqrt{var[V_t]}$.

Since both stocks follow Geometric Brownian Motion, we can get that:

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

To compute $E[S_{1,t}S_{2,t}]$, we know that:

$S_{1,t}S_{2,t} = S_{1,0}S_{2,0}exp((\mu_1 + \mu_2 - (\sigma_1{}^2 + \sigma_2{}^2)/2)t + \sigma_1 W_{1,t} + \sigma_2 W_{2,t})$

Since $E[\sigma_1 W_{1,t} + \sigma_1 W_{1,t}] = 0$,

$var[\sigma_1 W_{1,t} + \sigma_1 W_{1,t}] = E[(\sigma_1 W_{1,t} + \sigma_1 W_{1,t})^2] = E[\sigma_1{}^2 W_{1,t}{}^2 + \sigma_2{}^2 W_{2,t}{}^2 + 2\sigma_1\sigma_2 W_{1,t}W_{2,t}]$

Then $var[\sigma_1 W_{1,t} + \sigma_1 W_{1,t}] = \sigma_1{}^2 t + \sigma_2{}^2 t + 2\sigma_1\sigma_2 E[W_{1,t}W_{2,t}]$

By Itos formula, we can get:

$dW_1W_2 = W_1dW_2 + W_2dW_1 + \rho dt$

Then we can get:

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

$E[(\sigma_1 W_{1,t} + \sigma_1 W_{1,t})^2] = (\sigma_1{}^2 + \sigma_2{}^2 + 2\sigma_1\sigma_2\rho)t$

From derivations above, we can get:

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

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

Then, we can get $sd[V_t] = \sqrt{var[V_t]} = \sqrt{E[V_t{}^2] - E[V_t]^2}$

After obtaining $E[V_t]$ and $sd[V_t]$, we can get:

$VaR[V] = V_0 - (E[V_t] - \phi^{-1}(1-\alpha)sd[V_t])$

2.3.2 Historical VaR

1. After determining the timespan of VaR, calculate a series of changes of values of each components in the portfolio in that timespan and obtain a series of losses.

2. Then, obtain a series of losses of the portfolio by multiplying the losses of each component by their number of shares in the portfolio.

3. Rank the series of losses of the portfolio from high to low.

4. The portfolio loss with rank $n(1-\alpha)\%$ is the $(1-\alpha)\%$ VaR of the portfolio.

2.3.3 Monte Carlo VaR

1. After calculating the parameters of component's distributions, use these parameters to simulate a large number of outcomes of each component by generating correlated random variables using correlation matrix. For each stock S in the portfolio, simulate $S_t = S_0 * exp((\mu - \sigma^2/2)t + \sigma W_t)$ for n times. For each option in the portfolio, simulate its underlying stock for n times, and use Black-Scholes formula to calculate option price. Black-Scholes equations for option calculation can

be found in section 2.5 Option VaR Approximation.

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 Vt from V0 for each outcome.

4. Rank the losses from high to low and obtain the n*(1-α)$\%rankedlossasthe(1-\alpha)\%VaR$.

2.4 Advantages and Limitations

2.4.1 Parametric VaR

The major advantage of parametric VaR is that calculation of VaR becomes very simple and fast since it can be calculated in several equations by inputting several variables. It is also very simple to understand. If portfolio payoffs are linear and normal, parametric VaR is accurate.

The disadvantage of parametric VaR is that it has a strict assumption of normality. In reality, portfolio values are not necessarily normally distributed. The actual distributions of assets sometimes follow a fat tail distribution in which extreme events are more likely to happen compared to the normal distribution. In this case, parametric VaR can be underestimated by using the normality assumption.

2.4.2 Historical VaR

The advantage of historical VaR is that it makes no assumption on the distribution and linearity of portfolio values. It will not underestimate VaR even if the distribution is fat-tailed instead of normal.

Since historical VaR depends highly on historical values. If there is not sufficient historical data, historical VaR will become less accurate. Since it makes the assumption that historical performance can correctly represent current asset performance, historical VaR will not be accurate if the assumption does not hold. Due to complexity of this method, it will take longer time to obtain historical VaR than parametric VaR.

2.4.3 Monte Carlo VaR

The advantage of Monte Carlo VaR is that it is not sensitive to linearity. If the payoff of the portfolio is not linear by including options, Monte Carlo VaR will generate a better result than parametric Var.

Like parametric VaR, Monte Carlo VaR assumes that portfolio values follow a normal distribution. Therefore, when the actual distribution of portfolio is not normal, Monte Carlo VaR will not be accurate. Since it needs to simulate a large number of random outcomes, Monte Carlo VaR has the lowest speed of computation among these three VaR calculations.

2.5 Option VaR Approximation

Unlike stock payoffs, option payoffs are not linear. Therefore, to calculate VaR of an option with a parametric method, we need to obtain an approximation. In our model, we obtain the option value using the Black-Scholes model. The Black-Scholes model of call option and put option prices are:

$C = S_t N(d_1) - Ke^{-r(T-t)} N(d_2)$, and

$P = Ke^{-r(T-t)} N(-d_2) - S_t N(-d_1)$

where $d_1 = (\sigma\sqrt{T-t})^{-1}[ln(S_t/K) + (r + \sigma^2/2)(T-t)]$

and $d_2 = d_1 - \sigma\sqrt{T-t}$.

An option Δ measures the sensitivity of option value to the change of value of the underlying stock. It is calculated by taking the first derivative of the option by stock price. Assume the underlying stock price changes by x amount, then the approximate change of option value is going to be Δx. Δ of call and put options are:

$\Delta_c = N(d_1)$ and,

$\Delta_p = N(d_1) - 1$.

Since the price change of an option can be approximated by δ multiplied by the change of underlying stock price, option VaR can also be approximated by δ multiplied by stock VaR. This VaR approximation can be improved by adding more Taylor's expansion terms in the calculation. However, in this project, we will only

use Δ for option VaR approximation.

Chapter 3

Test plan

While model risk management includes elements of model development and applications, 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 paper. Robust model validation can help provide internal and external stakeholders a level of confidence that a model framework is sound and that results, at some level, can be relied upon to inform decisions. The primary purpose of model validation is ultimately to help address the management of model risk. In Chapter 3 , we want to those parts below.

- Validation Steps
  - Description of the settings test models performed (seek to be comprehensive as possible in terms of model inputs and instruments tested, and the intended model usage)
  - Outputs reviewed (e.g., VaR, ES)
  - Other data and position reviewed such as GE long positions, GS short positions, AMD long positions.
- The Scope of the Validation
  - Inputs and format

To further test our system's validity, we here choose two sets of data to see the effectiveness of our system.

3.1 Scope of the validation

Here we choose three stocks and two options in our portfolio to test the validity of our system. The stocks are AMD, MCD and GS respectively. And the options

are INTC and GE respectively. To make the results more widely applicable, we set the MCD and AMD in long position, and the GS in short position. And for option part, we set the GE as call option and INTC as put option.

We give examples of two setting parameters:
Parameters Set 2
Window: 5 years
Horizon:15 years
Day: 5 days
Confidence Level: 99%
Test period length: 1 year
Risk-free Interest Rate: 0.01

Table 3.1: Set 1

Stocks	Shares	Position
AMD US Equity	3000	long
MCD US Equity	2000	long
GS US Equity	-500	short

Options	Shares	Position
GE US Equity	200	call
INTC US Equity	-100	put

Parameters Set 2
Window: 2 years
Horizon:10 years
Day: 5 days
Confidence Level: 95%
Test period length: 1 year
Risk-free Interest Rate: 0.05

3.1.1 Assumption

- Stock share and option share represent how many shares of each asset we hold.
- Monte Carlo VaR and Parametric VaR will assume that each stock and each underlying follows GBM. For option part, we model the underlying, then calculate the price.

- We assume that the options we bought have the ever-lasting properties. In other words, the strike price and the time to maturity is unchanged on every day.
- The strike price of options we choose the S0, which is the initial price of underlying stocks.

3.1.2 Data Source

Sample Stocks Options AMD; INTC; GE; GS; MCD
Index: Historical call implied volatility; Historical put implied volatility; Last price; Adjusted close price
Time Period:12/5/1986 - 12/5/2016, daily data
Source: Bloomberg; Yahoo Finance

3.2 The Scope of the Validation

We input the file with the format of csv. And the stock file consists of 3 stocks (AMD, MCD, and GS). Date ranges from 1986/12/5 to 2016/12/5. The stock file contains their historical prices. The option file consists of 2 options (INTC, GE). Date ranges from 1986/12/5 to 2016/12/5 as well. The option file contains their underlying?s historical prices and implied volatility.
We uploaded the stocks in the following format:

Table 3.2: Stocks Set

AMD US Equity		MCD US Equity		GS US Equity	
Date	PX_LAST	Date	PX_LAST	Date	PX_LAST
12/5/16	8.68	12/5/16	119.29	12/5/16	228.55
12/2/16	8.53	12/2/16	118.24	12/2/16	223.36
12/1/16	8.39	12/1/16	118.47	12/1/16	226.63
11/30/16	8.91	11/30/16	119.27	11/30/16	219.29
11/29/16	8.93	11/29/16	120.68	11/29/16	211.75
11/28/16	8.83	11/28/16	121.82	11/28/16	210.35
11/25/16	8.77	11/25/16	120.66	11/25/16	211.38
11/23/16	8.8	11/23/16	120.14	11/23/16	212.31
11/22/16	8.69	11/22/16	119.69	11/22/16	211.11
11/21/16	8.94	11/21/16	119.5	11/21/16	211.08
11/18/16	8.71	11/18/16	120	11/18/16	210.35
11/17/16	8.46	11/17/16	119.45	11/17/16	209.63
11/16/16	7.67	11/16/16	119.21	11/16/16	206.26

We uploaded the option in the following format:

Table 3.3: Option Set

INTC US Equity			GE US Equity		
Date	hist Put imp vol	PX_LAST	Date	hist call imp vol	PX_LAST
12/5/16	19.67	34.39	12/5/16	17.235	31.11
12/2/16	20.366	34.16	12/2/16	17.359	31.34
12/1/16	21.235	33.76	12/1/16	17.634	31.39
11/30/16	18.714	34.7	11/30/16	16.692	30.76
11/29/16	18.049	35.31	11/29/16	16.984	31.05
11/28/16	18.009	35.51	11/28/16	16.717	31.25
11/25/16	17.853	35.44	11/25/16	15.801	31.44
11/23/16	18.156	35.2	11/23/16	16.289	31.34
11/22/16	17.988	35.48	11/22/16	16.487	31.18
11/21/16	18.648	34.98	11/21/16	16.109	30.87
11/18/16	19.491	34.95	11/18/16	16.531	30.67
11/17/16	20.297	35.02	11/17/16	16.505	30.79
11/16/16	20.657	34.84	11/16/16	15.384	30.74

3.2.1 Validation Results

After using our software to conduct the test plan, we get the following figures:
From the figures, we can find out that when we calibrate our VaR within 10000, the different approaches of calculating VaR are generally similar. Then we can take a look at the 2 sets? results respectively. Timeline ranges 15 years, from 2001 Dec 5 to 2016 Dec 5.

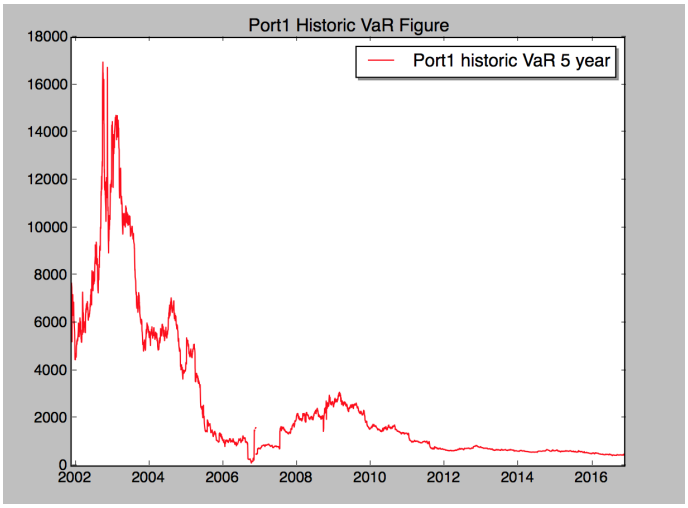

Figure 3.1: Portfolio VaR

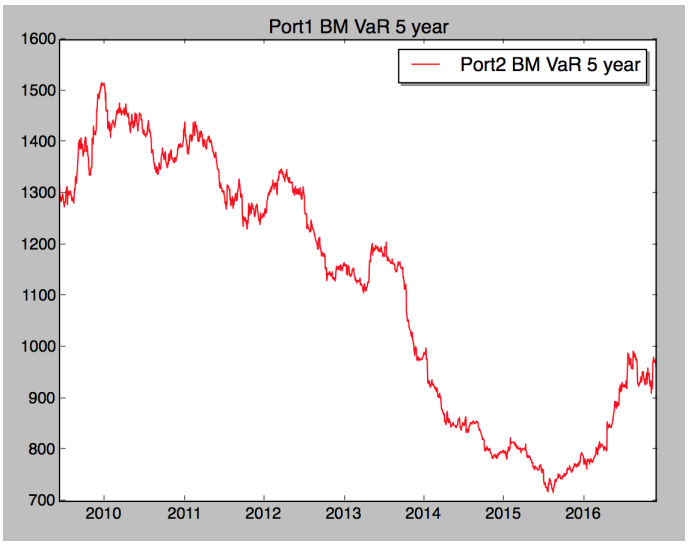

Figure 3.2: Portfolio VaR

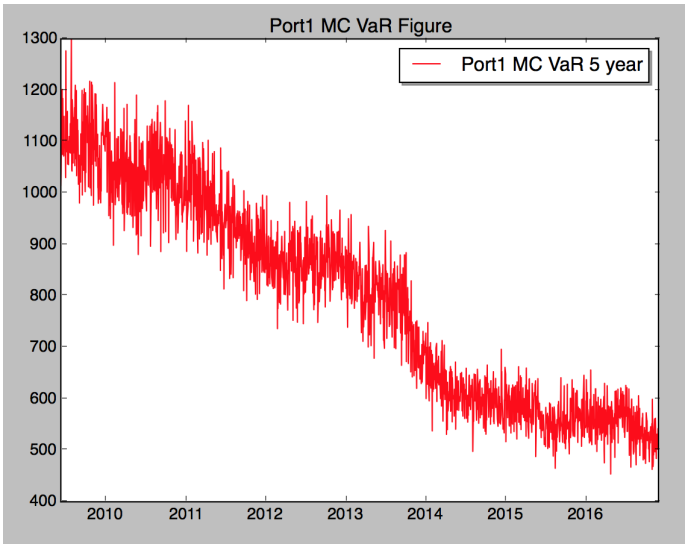

Figure 3.3: Portfolio VaR

3.2.2 Back Test Results

Through back test, we observe periods of time when there are no exceptions and periods of time with a substantial number of exceptions. Since the Historical VaR used the historical data to calculate the VaR, it doesn't need to apply the Back test. So the Backtest here only consists of Parametric VaR and Monte Carlo VaR. First, we can take a look at the graph we generate:

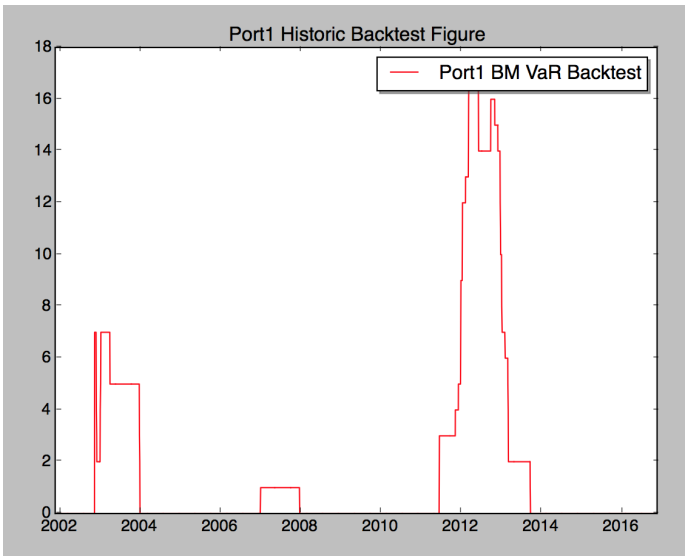

Figure 3.4: Backtest Portfolio

Remark: The Backtest Results of Test Set1. It is the test back result from the portfolio1 historic VaR, where horizon equals 5 days, and window len is 1 year. Data ranges 15 years, from 2001 Dec 5 to 2016 Dec 5.

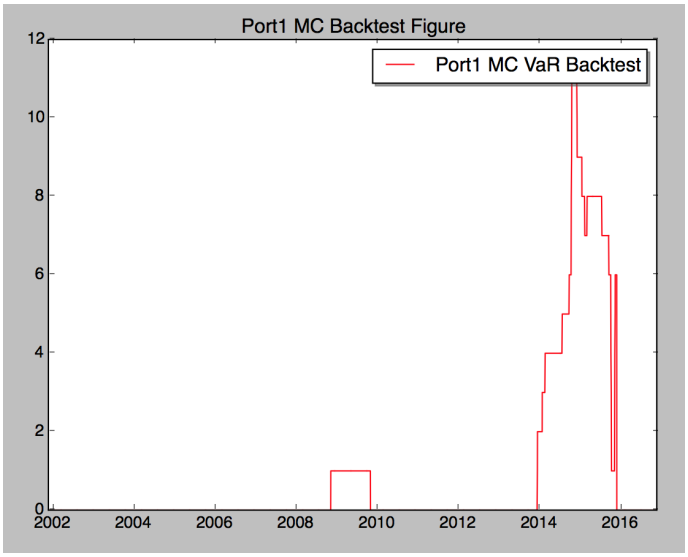

Figure 3.5: Portfolio Backtest

We observe that exceptions become more when volatility increases, as indicated by the range of the jump s in actual losses. The VaR starts to rise, but doesn?t rise fast enough to account for the increased market volatility. The VaR then falls when the volatility drops, but takes a long time to deflate to the new market behavior. We suspect that the overlapping periods contribute to the large number of exceptions. Since we are looking at 5 day VaR, there is a high correlation between the 5 day loss on a given day and the 5 day loss on the following day.

Chapter 4

Software

The software we designed includes three functions to do VaR calculation. There are five files in this program. Here we give a simple description of them.

4.1 Simple Description

Here we choose three stocks and two options in our portfolio to test the validity of our system. The stocks are AMD, MCD and GS respectively. And the options are INTC and GE respectively. To make the results more widely applicable, we set the MCD and AMD in long position, and the GS in short position. And for option part, we set the GE as call option and INTC as put option.

4.1.1 Software Composition

The software we designed includes three functions to do VaR calculation. There are five files in this program.

- BS Model calculation: define a function to calculate the delta. The output will be delta.

- Input_ComputeVaR_Output$_{B}acktest$:the Python script will have the following calculation. First it input parameters and data. Then it calculate the Parametric VaR, Monte Carlo VaR and Historical VaR respectively. Finally it back tests the result and plot both graphs of VaRs and back-test exceptions.
- Stocks.csv: files are named as stocks.csv, which includes time series data of stock prices.
- Options.csv: files are named as options.csv, which includes time series data of underlying stock prices and implied volatillity.

- Parameters.csv: includes all the parameters of the script. All the parameters are changeable inputs. The user of this software can change them according to their needs.

4.1.2 Software Assumption

- Stock share and option share represents how many shares of each asset we hold.
- Monte Carlo VaR and Parametric VaR will assume that each stock and each underlying follows GBM. For option part, we model the underlying, then calculate the price.
- We assume that the options we bought have the ever-lasting properties. In other words, the strike price and the time to maturity is unchanged on every day.
- The strike price of options we choose the S0, which is the initial price of underlying stocks.

4.1.3 Software Flexibility

Our software is a highly flexible calculation system. Users can construct any portfolio consisting of any numbers of stocks and options. The number of stocks or options can be 0 to infinity ($1 \rightarrow \infty$).
Moreover, our system is strongly flexible for users to use because all the parameters in Parameters.csv can be changed according to users? requirements. The stock shares, option shares, window, horizon, and length of input price data, the option maturity, option strike price, the put or call option type, the confidence level, and the risk free interest rate are all inputs input by users.

4.2 Detalied Programming Flow Code

4.2.1 Description of System Product

We use Python. We can calculate four kinds of VaRs: GBM VaR, parametric BM VaR, Monte Carlo VaR, historical VaR, respectively. And after we uploaded the three csv. file: sto_opt.csv, we can get the VaR to our portfolio. If users want to add or remove any stocks or options to the portfolio, just add or remove required data in input part of our program.

4.2.2 Design documentation

The control procedure and flow
The code may differ in the four approaches of VaR calculation, but the core logic is consistent with this control flow.

Step1: Enter stock prices in sto_opt.csv, underlying stock price of certain options and option implied volatility data, and construct dataframe for each of them. Those data are time series.

Step2: Enter the corresponding parameters in the defined class Portfolio1, Portfolio2, such as shares of each stock or option, Strike Price, Option Type (1 for call option, -1 for put option), Window, Horizon, Day, Risk-free rate, and Confidence Level. The Parameters are fixed for the whole calculation time interval and are all changeable input made by the users.

Step3: Calculate various VaRs of stocks and options. Here, the daily return we use log return. The formula should be $log(price/price - 1)$for each stock and the underlying stock of each option.
For every stock, they have a_j positions. For every option, they have b_j positions. We need to enter the positions of each stock to calculate portfolio value. $P = \sum_{i=1}^{n} a_i S_i + \sum_{j=1}^{m} b_i BS(S_j, vol, K, r, T)$ is the portfolio value.
The option price $BS(S_j, vol, K, r, T) = \Theta + \Delta S$, where we will leave Θ out.
Then we can reach the new formula:

$$P = \sum_{i=1}^{n} a_i S_i + \sum_{j=1}^{m} b_i \Delta_j S_j$$

Here, we need to pay attention that in calculating Δ , there are two branches. Because the Δ is different for call options and put options.

For call option, the $\Delta = N(d_1)$; for the put option, the $\Delta = N(d_1) - 1$, where $d_1 = \frac{ln(S_0/K)+(r+\sigma^2/2)T}{\sigma T}$, we need to notice that the d_1 will change every day. vol σ should be our time series input. r, t should be set by users and fixed for thewhole calculation time interval.

Step4: Calculate stock sample mean and variance. Calculate sample μ and σ for all the S_i, S_j, and Based on the results to calculate the drift and volatility parameters. The formulas are shown as following:

$$\sigma = \sqrt{252}\bar{\sigma}, \mu = \bar{\mu} * 252 + \sigma^2/2$$

Step5: Calculate four different kinds of VaR. In this part, we need to divide the calculation procedure into several parts. So in the computational level, we can do

it step by step. And we put the code parts in calculating the fourVaRs.

4.2.3 VaR calculation Code

- **Historical VaR**
  $> for\ i = 1 : 252 * windowlen$
  $>$ $his\_VaR\_list = sorted(A\_hist\_log\_rtn[i : i + 252 * windowlen])$
  $>$ $threshold = ceil((1 - p) * length(his\_VaR\_list))$
  $>$ $his\_VaR = (-S0) * his\_VaR\_list[threshold]$

- **GBM VaR**
  $> for\ i = 1 : 252 * windowlen$
  $>$ $vol\_years = std(log\_rtn[i : i + 252 * windowlen]) * sqrt(252)$
  $>$ $mu\_years = mean(log\_rtn[i : i + 252 * windowlen]) * 252 + (vol\_years * *2)/2$
  $>$ $GBM\_VaR = S0 - S0 * exp(vol\_years * horizon * *(0.5) * ss.norm.ppf(1 - p) + (mu\_years - pow(vol\_years, 2)/2) * horizon)$

- **Parametric BM VaR**
  $> for\ i = 1 : 252 * windowlen$
  $>$ $\#sigma = Volatility foreachstock.$
  $>$ $\#rho = Correlationmatrixofdriving Brownianmotions.$
  $>$ $cov\_mat = sigma. * rho. * sigma'$
  $>$ $\#a = Positionvector(numberofshares of eachstock).$
  $>$ $\#s = Initialstockpricevector.$
  $>$ $positionvalues : as = a. * s$
  $>$ $\#V\_0$
  $>$ $v0 = sum(as)$
  $>$ $\#E[V\_t]$
  $>$ $evt = as * exp(mu * t)'$
  $>$ $\#E[V\_t^2]$
  $>$ $evt2 = exp(mu * t) * (exp(covm * t). * (as' * as)) * exp(mu * t)';$
  $>$ $\#sd[V\_t]$
  $>$ $sqrt(evt2 - evt^2)$
  $>$ $\#ParametricVaR$
  $>$ $VaR = v0 - (evt + norminv(1 - p) * sdvt)$

- **Weighted GBM VaR**
  $> list\_lambda = [\lambda^i]$

$> wgt\_log\_return = log\_return * list\_lambda[i]$
$> for\ jinrange(len(self.price) - 252 * windowlen)$
$>$ $wgt\_mu\_lambda0 = sum(wgt\_log\_rtn[j : j + 252 * windowlen])/sum(list\_lambda[j : j + 252 * windowlen])$
$>$ $wgt\_vol\_lambda = sqrt(252) * np.sqrt(sum(wgt\_log\_rtn_sq[j : j + 252 * windowlen])/sum(list\_lambda[j : j + 252 * windowlen]) - wgt\_mu\_lambda0 * *2)$
$>$ $wgt\_mu\_lambda = 252 * wgt\_mu\_lambda0 + (wgt\_vol\_lambda * *2)/2$
$>$ $wgt\_VaR = S0 - S0 * exp(wgt\_vol\_lambda * horizon * *(0.5) * ss.norm.ppf(1 - p) + (wgt\_mu\_lambda - pow(wgt\_vol\_lambda, 2)/2) * horizon)$

- **Monte_Carlo_VaR**
  $> for\ jinrange(len(self.price) - 252 * windowlen)$
  $>$ $vol\_years = std(log\_rtn[i : i + 252 * windowlen]) * sqrt(252)$
  $>$ $mu\_years = mean(log\_rtn[i : i + 252 * windowlen]) * 252 + (vol\_years * *2)/2$
  $>$ $MC\_VaR = S0 - S0 * np.exp(vol_years * horizon * *(0.5) * ss.norm.ppf(1 - p) + (mu\_years + random - pow(vol\_years, 2)/2) * horizon)$

Step5: Back test the exception of each kind of VaR.
In this part, after we have computed the N' day VaR on each date, we need to compare it to the loss occurring on the following N' days. We also count the number of exceptions that occur in each M year window. This is straightforward to compute, because we just need to count the number that the loss exceeds the VaR and sum up the exception numbers.

- **Back Test**
  $> except\_cases = [\ ]$
  $> for\ j\ inrange(len(VaR\_list))$
  $>$ $window\_data = self.price[(len(VaR\_list) - i - 252 * windowlen) : (len(VaR\_list) - i - 1)]$
  $>$ $excep = 0$
  $>$ $for\ jinrange(len(window_data) - 5)$
  $>$ $if\ loss > VaR\_list[len(VaR\_list) - i - 252 + j]$
  $>$ $excep = excep + 1$
  $>$ $except\_cases = except\_cases + [excep]$

Chapter 5

Conclusion

5.1 conclusion

In this project, we construct a portfolio VaR calculation model with three methods. The three methods are Parametric VaR, historical VaR, and Monte Carlo VaR. Due to difference assumptions and derivations of each method, each method has its own advantages and limitations. The advantage of this model is its portfolio flexibility since both stocks and options can be included in the portfolio. There is also no limitation on the long short positions of each element and the number of elements in a portfolio. Since there are three methods in this model, users can choose the most suitable model based on different assumptions.

When choosing a particular VaR calculation method, user need to understand characteristics of each VaR calculation method. For parametric VaR of a portfolio with normality assumption, the primary advantage is that calculation is fast and easy to understand. The major limitation is that if the actual distribution of portfolio value is not normal or the portfolio payoff is not linear, parametric VaR will not be accurate. Especially when the actual distribution is a fat-tailed, parametric VaR is underestimated. Historical VaR has the advantage that it does not have any assumption of distribution. When portfolio distribution is not normal and historical performance can accurately forecast future performance, historical VaR can generate an accurate VaR estimation. However, historical VaR demands a larger amount of data and its calculation time is longer than parametric VaR. The advantage of Monte Carlo VaR is that it is not sensitive to linearity. Thus, it is useful when options are included in the portfolio. Since Monte Carlo VaR needs to generate a large number of simulations, computation speed is very low.

Another limitation of this model is due to its approximation of option VaR.

Since we use the Black-Scholes model to calculate option values, when assumptions of the Black-Scholes model do not hold in reality, option VaR in historical and Monte Carlo methods will not be accurately estimated. Since we use delta approach estimation in parametric VaR, estimation will become less accurate when underlying stock price changes by a large amount. This approximation can be improved by adding more elements of the Taylors expansion in option VaR estimation.

Appendix A

Reference

A.1 Paper Reference

1. Stein, H. J. (2016).Financial Risk Management and Regulation Lecture 4: Market Risk [PowerPoint slides].
2. Chen, R. (2013). Global Risk Management: A Quantitative Guide. Global Social Science Institute.
3. Coleman, T. S. (2011). A practical guide to risk management. Research Foundation of CFA Institute.
4. Duffie, D., Pan, J. (1997). An Overview of Value at Risk.
5. Harvey J. Stein. Model Validation Municipal Bonds. 2014.
6. Harvey J. Stein. Model Validation Report Template. 2014.

A.2 Code Reference

The code is programed by Python language. You can run it in ipynotebook or other Python compilers.

Risk_Mngt_Final_Project_Code

December 22, 2016

In [ ]:
```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec 10 14:50:01 2016

@author: Duanhong
"""
import numpy as np
import scipy.stats as ss
import pandas as pd
import math
import matplotlib.pyplot as plt
import datetime as dt
import operator

#----- Option Delta Calculation -----------#
def Option_Delta(opt_type, S0, K, r, sigma, T):
        """opt_type: put or call """
        d1 = (np.log(S0/K) + (r + sigma**2 / 2) * T)/(sigma * np.sqrt(T))
        if opt_type=="Call":
            return(ss.norm.cdf(d1))
        else:
            return(-ss.norm.cdf(-d1))

#----- Compute the delta value for options -------#


class Portfolio1(object):
    """ An US Stock we randomly choose. Stocks have the
    following properties:

    Attributes:
        name: A string representing the stock's name.
        price: Adj Close price representing the stock's price
                    from 12/5/1996 to 12/5/2016 .
        date: Ranging from 12/5/1996 to 12/5/2016
    """

    def __init__(self, name, price, date):
        """Return a stock object whose name is *name* and close
        price is *price*."""
        self.name = name
        self.price = price
        self.date = date
```

```
#-----Type I VaR model
def Historic_VaR(self, windowlen, horizon, p, S0):
    """ Return a stock object's Historic VaR after setting
    a stock VaR's parameters, e.g. windowlen(2,5,10 years),
    horizon = 5/252, p = 0.99, portfolio value S0 = 10000. """
    A_hist_rtn = []
    for i in range(1, len(self.price)- 5):
        hist_return =  self.price[i]- self.price[i+5]
        A_hist_rtn.append(hist_return)
    # Calculate Historic VaR
    hist_VaR = []
    for i in range(len(self.price)-252*windowlen):
        his_VaR_list = sorted(A_hist_rtn[i:i+252*windowlen])
        his_VaR_val =  S0* his_VaR_list[252*5-13+1]/self.price[i+5]
        hist_VaR.append(his_VaR_val)
    return(hist_VaR)


#------Type IIa VaR model
def GBM_VaR(self, windowlen, horizon, p, S0):
    """ Return a stock object's Historic VaR after setting
    a stock VaR's parameters, e.g. windowlen(2,5,10 years),
    horizon = 5/252, p = 0.99, portfolio value S0 = 10000. """
    log_rtn = []
    log_rtn_sq = []
    for i in range(1 ,len(self.price)-1):
        log_return = math.log( self.price[i]/self.price[i+1] )
        log_rtn.append(log_return)
        log_rtn_sq.append(log_return**2)
    # Calculate GBM VaR
    GBM_VaR = []
    for i in range(len(self.price)-252*windowlen):
        vol_years = np.std(log_rtn[i:i+252*windowlen]) * np.sqrt(252)
        mu_years = np.mean(log_rtn[i:i+252*windowlen])*252 + (vol_years**2)/2
        GBM_VaR_val = S0 - S0 * np.exp(vol_years * horizon**(0.5)* ss.norm.ppf(1-p)+(mu_yea
        GBM_VaR.append(GBM_VaR_val)
    return(GBM_VaR)

#-----Type IIb VaR model
def Wgt_GBM_VaR(self, lambda_wgt, windowlen, horizon, p, S0):
    """wgt is the exponential weight value like 0.99^n. """
    list_lambda = []
    for i in range(len(self.price)):
        list_lambda.append(lambda_wgt**i)

    wgt_log_rtn = []
    wgt_log_rtn_sq = []
    for i in range(len(self.price)-1 ):
        log_return = math.log( self.price[i]/self.price[i+1] )
        wgt_log_return = log_return *  list_lambda[i]
        wgt_log_rtn.append(wgt_log_return)
        log_return_sq = log_return ** 2
        wgt_log_rtn_sq.append( log_return_sq * list_lambda[i] )
```

```
        # Calculate Weighted VaR under GBM model
        wgt_VaR = []
        for j in range(len(self.price)-252*windowlen):
            wgt_mu_lambda0 =  sum(wgt_log_rtn[j:j+252*windowlen])/sum(list_lambda[j:j+252*windo
            wgt_vol_lambda = np.sqrt(252) * np.sqrt(sum(wgt_log_rtn_sq[j:j+252*windowlen])/sum(
            wgt_mu_lambda =252 * wgt_mu_lambda0 + (wgt_vol_lambda**2)/2
            wgt_VaR_val = S0 - S0 * np.exp( wgt_vol_lambda * horizon **(0.5)* ss.norm.ppf(1-p)
            wgt_VaR.append(wgt_VaR_val)
        return(wgt_VaR)

    #-----Type III VaR model
    def Monte_Carlo_VaR(self, windowlen, horizon, p, S0):
        """"  This is Monte Carlo VaR under GBM assumption  """
        log_rtn = []
        log_rtn_sq = []
        for i in range(1 ,len(self.price)-1):
            log_return = math.log(self.price[i]/self.price[i+1] )
            log_rtn.append(log_return)
            log_rtn_sq.append(log_return**2)
        # Calculate GBM MC VaR
        MC_VaR = []
        for i in range(len(self.price)-252*windowlen):
            vol_years = np.std(log_rtn[i:i+252*windowlen]) * np.sqrt(252)
            mu_years = np.mean(log_rtn[i:i+252*windowlen])*252 + (vol_years**2)/2
            random = vol_years * ss.norm.ppf(np.random.rand())
            MC_VaR_val =S0-S0 * np.exp(vol_years*horizon**(0.5)*ss.norm.ppf(1-p)+(mu_years + ra
            MC_VaR.append(MC_VaR_val)
        return(MC_VaR)

    #-----BackTest
    def BackTest(self,VaR_list, windowlen, horizon, S0):
        """Back test the VaR_list(eg.GBM VaR) based VaR estimates for long and
        short portfolios by counting the number of times the VaR on each date
        is exceeded by the subsequent horizon(eg. 5/252 ) change in each 1 year window. """
        nexcep = [ ]
        for i in range(len(VaR_list)):
            window_data = self.price[(len(VaR_list)-i-252*windowlen):(len(VaR_list)-i-1)]
            excep = 0
            for j in range( len(window_data) - 5):
                share = S0/window_data[j]
                pricet = window_data[j+5]
                loss = S0 - pricet * share
                if loss > VaR_list[len(VaR_list)-i-252 + j]:
                    excep = excep + 1
                else:
                    excep = excep
            nexcep.append(excep)
        return(nexcep)



class Portfoio2(object):
    def __init__(self, name, price1, price2, share1, share2, date):
        """Return a stock object whose name is *name* and close
```

```
        price is *price*."""
        self.name = name
        self.price1 = price1
        self.price2 = price2
        self.share1 = share1
        self.share2 = share2
        self.date = date

    def Port_BM_VaR(self, windowlen, horizon, p, S0):
        """ share is a vector containing stocks' shares in a portfolio.  """
        A_log_rtn = []
        A_log_rtn_sq = []
        for i in range(1 , len(self.price1)-1):
            log_return = math.log(self.price1[i] / self.price1[i+1])
            A_log_rtn.append(log_return)
            A_log_rtn_sq.append(log_return**2)

        I_log_rtn = []
        I_log_rtn_sq = []
        for i in range(1 , len(self.price1)-1):
            log_return = math.log(self.price2[i] /self.price2[i+1])
            I_log_rtn.append(log_return)
            I_log_rtn_sq.append(log_return**2)

        A_vol_years = []
        A_mu_years = []
        I_vol_years = []
        I_mu_years = []
        corr = []

        for i in range(len(self.price1)-252 * windowlen):
             vol_years1= np.std(A_log_rtn[i:i+252*windowlen]) * np.sqrt(252)
             mu_years1 = np.mean(A_log_rtn[i:i+252*windowlen])*252 + (vol_years1**2)/2

             vol_years2= np.std(I_log_rtn[i:i+252*windowlen]) * np.sqrt(252)
             mu_years2 = np.mean(I_log_rtn[i:i+252*windowlen])*252 + (vol_years2**2)/2

             x = np.asarray(A_log_rtn[i:i+252*windowlen])
             y = np.asarray(I_log_rtn[i:i+252*windowlen])
             X = np.vstack((x, y))
             cov = np.cov(X)
             corr_val = cov[0,1] * 252 / (vol_years1 * vol_years2)
             A_vol_years.append(vol_years1)
             I_vol_years.append(vol_years2)
             A_mu_years.append(mu_years1)
             I_mu_years.append(mu_years2)
             corr.append(corr_val)

        ###################### Window VaR and ES #################
        Port_VaR = []

        for i in range(len(self.price1)-252*windowlen):
            sigma = np.matrix([A_vol_years[i], I_vol_years[i]])
            rho = np.matrix([[1, corr[i]],[corr[i],1]] )
```

```
                x0 = np.dot(sigma, rho).T
                cov_mat = np.dot(x0, sigma)

                s0 = np.matrix([ self.price1[i], self.price2[i] ])
                pos = np.matrix([self.share1, self.share2])
                IndexVal = np.dot(pos, s0.T)
                portPos = pos/IndexVal * S0

                two_sto_val_split = np.array([[portPos[0,0]*s0[0,0]],[portPos[0,1]*s0[0,1]]])
                E_V_t = two_sto_val_split[0,0] * np.exp(A_mu_years[i] * 5/252) + two_sto_val_split[

                x4 = np.exp(cov_mat*5/252)
                x1 = np.matrix([np.exp(A_mu_years[i]*5/252), np.exp(I_mu_years[i]*5/252)])
                x2 = np.dot(two_sto_val_split, two_sto_val_split.T)
                x3 = np.matrix([[x4[0,0]*x2[0,0],x4[0,1]*x2[0,1] ] , [x4[1,0]*x2[1,0], x4[1,1]*x2[1

                E_V_t_sq = x1 * x3 * x1.T
                std_V_t = np.sqrt( E_V_t_sq - E_V_t ** 2)
                VaR_val = sum(two_sto_val_split) - (E_V_t + ss.norm.ppf(1-p) * std_V_t)
                Port_VaR.append(VaR_val.item(0))

            return(Port_VaR)
```

In [ ]:
```
path = '~/Desktop/stos_opts_data.csv'
st_opt =  pd.read_csv(path, header = 1)
data = st_opt.values
```

In [ ]:
```
###-------------Portfolio BM VaR-----------------###
AMD_close = list(data[:,2])
INTC_close = list(data[:,6])
MCD_close = list(data[:,10])

port2 = Portfoio2('port1',AMD_close,MCD_close, 1000, 200, data[:,0] )
BM_port2 = port.Port_BM_VaR(5,5/252, 0.99, 10000)
##-----BM PORT2
fig, ax = plt.subplots()
ax.plot(timeline, BM_port2[1:252*15],'r-', label='Port2 BM VaR 5 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port1 BM VaR 5 year')
plt.show()
```

In [ ]:
```
###-----------Port 2 Historic VaR
list1 = [x*1000 for x in AMD_close]
list2 = [x*200 for x in MCD_close]

portfolio2 = [sum(x) for x in zip(list1, list2)]
port2 = Portfolio1('portfolio1',portfolio2, date)

Hist_VAR_port2 = port2.Historic_VaR(5, 5/252, 0.99, 10000)
timeline = data[1:252*15,0]
timeline = [dt.datetime.strptime(d,'%m/%d/%y').date() for d in timeline]
```

```
fig, ax = plt.subplots()
ax.plot(timeline, Hist_VAR_port2[1:252*15],'r-', label='Port1 historic VaR 5 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port1 Historic VaR Figure')
plt.show()
```

In [ ]:
```
###-----------Port 2 MC VaR
MC_VAR_port2 = port2.Monte_Carlo_VaR(5, 5/252, 0.99, 10000)

timeline = data[1:252*15,0]
timeline = [dt.datetime.strptime(d,'%m/%d/%y').date() for d in timeline]
fig, ax = plt.subplots()
ax.plot(timeline, MC_VAR_port2[1:252*15],'r-', label='Port1 MC VaR 5 year')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port1 MC VaR Figure')
plt.show()
```

In [ ]:
```
##---Port2 Backtest Hist VaR
Hist_backtest2 = port2.BackTest(Hist_VAR_port2, 1, 5, 10000)
fig, ax = plt.subplots()
ax.plot(timeline, Hist_backtest2[1:252*15],'r-', label='Port1 BM VaR Backtest')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port1 Historic Backtest Figure')
plt.show()
```

In [ ]:
```
###---Port2 BM VaR----######
BM_backtest2 = port2.BackTest(BM_port2, 1, 5, 10000)
fig, ax = plt.subplots()
ax.plot(timeline, BM_backtest2[1:252*15],'r-', label='Port1 BM VaR Backtest')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port2 BM Backtest Figure')
plt.show()
```

In [ ]:
```
###---Port2 MC VaR
MC_backtest2 = port2.BackTest(MC_VAR_port2, 1, 5, 10000)
MC_backtest2 = MC_backtest2[::-1]
fig, ax = plt.subplots()
ax.plot(timeline, MC_backtest2[1:252*15],'r-', label='Port1 MC VaR Backtest')
legend = ax.legend(loc='upper right', shadow=True)
ax.set_title('Port1 MC Backtest Figure')
plt.show()
```