UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE & ENGINEERING

MIE1622H
Computational Finance and Risk Management

Assignment 3

Credit Risk Modeling and Simulation

Professor: Dr. Oleksandr Romanko

Student name: Jianhui Li

Student number:1002116907

Date: March 25th, 2020

Introduction

The objective of this assignment is to model a credit-risky portfolio of corporate bonds for the data of 100 counterparties, simulate 1-year losses for each corporate bond. In this assignment, 3 sets of scenarios was generate:

- Monte Carlo approximation 1 : 5000 in-sample scenarios (N = 1000 · 5 = 5000 (1000 systemic scenarios and 5 idiosyncratic scenarios for each systemic), non-Normal distribution of losses)
- Monte Carlo approximation 2 : 5000 in-sample scenarios (N = 5000 (5000 systemic scenarios and 1 idiosyncratic scenario for each systemic), non-Normal distribution of losses)
- True distribution: 100000 out-of-sample scenarios (N = 100000 (100000 systemic scenarios and 1 idiosyncratic scenario for each systemic), non-Normal distribution of losses)

The out-of-sample scenarios represent true distribution of portfolio losses. Two in-sample non-Normal datasets are used for evaluating sampling error and performing portfolio optimization. Calculation of VaR and CVaR was implemented by non-Normal distribution model and Normal distribution model separately. In the in-sample Normal model, mean loss and standard deviation of losses for each corporate bond is calculated from the 3 scenario sets (N=1000×5, 5000, 100000). There are two different portfolios:

(1) one unit invested in each of 100 bonds

(2) equal value (dollar amount) is invested in each of 100 bonds

For each portfolio, the structural models for portfolio credit risk is used to infer the future credit state from a continuous random variable. Then, VaR and CVaR is calculated at quantile levels 99% and 99.9% for each of the six datasets (non-Normal with N=1000×5, 5000, 100000; and Normal with mean/standard deviation computed from N=1000×5, 5000, 100000). To better evaluate sampling and model errors, 100 times of experiment generate in-sample datasets and compute VaR and CVaR. Keep the out-of-sample scenario set unchanged. In the end of this report, I will compare each in-sample VaR and CVaR to out-of- sample VaR and CVaR, and explain the effects of sampling error and model error.

Implementation of in-Sample Experiment and out-Sample

True distribution

In this assignment, there are six dataset in total, non-Normal with N=1000×5, 5000, 100000, and Normal with mean/standard deviation computed from N=1000×5, 5000, 100000. N represents the sample number. However, the out-of-sample scenarios is used to represent true distribution of portfolio losses. Every VaR and CVaR of in-Sample model needs to compare with the true distribution results out-of- sample VaR and CVaR. And then the effects of sampling error and model error of each model of a credit-risky portfolio can be found. First, this out-sample scenario is considered to have a 100000 systemic scenarios and 1 idiosyncratic scenario for each systemic N=100000. Then 100000 random number is generated for Yj by the normal distribution N (0,1), which means mean is equal to 0, and standard deviation to be 1. Yj, systemic risk credit driver factor correlated credit drivers, represents the number of systemic scenarios. Yj is calculated by Cholesky decomposition of correlations for credit drivers multiply the random number sets. Next, Zj is generated by using the same method as generating Yj, with the same number N=100000. At last, using the structural model $w_j = \beta_j y_{j(k)} + \sigma_j z_j$ of portfolio credit risk to infer a counterparty's future credit state from a continuous random variable called a creditworthiness index. The beta is given from the credit_driver_corr.csv, and alpha is square root of (1-beta ^2). After the creditworthiness index Wj is computed from the equation, each of counterparties future credit state from a continuous random variable can be found through comparing the value obtained from the credit_driver_corr.csv. Then the each counterparty's loss distribution of out-sample scenario is obtained.

```
if Path('Losses_out_1.npy').is_file():
    Losses_out = np.load('Losses_out_1.npy')

else:

#Calculated out-of-sample losses (100000 x 100)

    normal_random_vector=np.random.randn(Ndriver,Nout)
    y=(sqrt_rho @ normal_random_vector).T

    for s in range( Nout):
        z=np.random.randn(K,1)
        for k in range(K):
            cd =int(driver[k])
            w[s,k] = beta[k] * y[s,cd] +math.sqrt(1-beta[k]**2) * z[k]
            temp = np.append(w[s,k],CS_Bdry[k,:])
            temp = np.sort(temp)
            cs_index=np.where(temp==w[s,k])
            Lossess_out[s,k] = exposure[k,cs_index]
    Losses_out=Lossess_out
    np.save('Losses_out_1.npy',Lossess_out)
```

Figure 1 code for out-sample losses

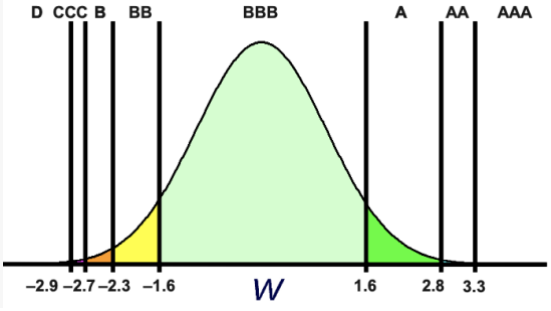

Figure 2 creditworthiness index

The VaR and CVaR is calculation from the non-Normal distribution. Once the loss value of each portfolio is obtain as a list, which is calculation by using the loss distribution and asset units.

```
losses_out_wtd.append(np.sort(Losses_out @ x0[0].T))  # (100000, 100) @ (2,100)
losses_out_wtd.append(np.sort(Losses_out @ x0[1].T))
losses_out_wtd=np.array(losses_out_wtd)  # (2, 100000)
```

Figure 3: code: total loss value

After sorting the list, I can use the historical distribution in lecture note to get the VaR and CVaR. The code shows in figure 5

$$\mathrm{VaR}_{\alpha,N} = \ell_{(\lceil N\alpha \rceil)}$$

$$\mathrm{CVaR}_{\alpha,N} = \frac{1}{N(1-\alpha)} \left[(\lceil N\alpha \rceil - N\alpha)\, \ell_{(\lceil N\alpha \rceil)} + \sum_{k=\lceil N\alpha \rceil + 1}^{N} \ell_{(k)} \right]$$

Figure 4 calculation equation for non-normal

```
VaRout[portN, q] = losses_out_wtd[portN][int(math.ceil(Nout*alf))-1]

CVaRout[portN, q] = (1 / (Nout*(1-alf))) * ((math.ceil(Nout*alf) - Nout*alf) *  VaRout[portN, q] + sum (losses_out _wtd
[portN,(int(math.ceil(Nout*alf))-1):]))
```

Figure 5: code for calculation equation for non-normal

In-Sample N=100000, Normal distribution

First, this in-sample scenario is considered to have a 100000 systemic scenarios and 1 idiosyncratic scenario for each systemic N=100000. Same as calculating true distribution of portfolio losses, 100000 random number is generated for Yj by the normal distribution N (0,1), which means mean is equal to 0, and standard deviation to be 1. Yj, systemic risk credit driver factor correlated credit drivers, represents the number of systemic scenarios. Yj is calculated by Cholesky decomposition of correlations for credit drivers multiply the random number sets. Next, Zj is generated by using the same method as generating Yj, with the same number N=100000. At last, using the structural model $w_j = \beta_j y_{j(k)} + \sigma_j z_j$ of portfolio credit risk infers a counterparty's future credit state from a continuous random variable called a creditworthiness index. The beta and alpha is same as calculating in out-sample model. After the creditworthiness index Wj is computed from the equation, each of counterparties future credit state from a continuous random variable can be found through comparing the value obtained from the

credit_driver_corr.csv. Then the counterparties loss distribution of out-sample scenario is obtained. Overall, this model is using the same calculation process. But the VaR and CVaR was calculated using normal distribution model, Therefore, the mean and standard deviation are needed.

Second, the loss value of each portfolio was generated by multiply loss distribution and asset units of each portfolio. The code shows in figure 4.

Third, the VaR, and CVaR is calculated by mean and standard deviation of the total loss value of each portfolio. The code shows in figure 6 below.

```
VaRinN[portN, q] = np.mean(losses_out_wtd[portN]) + scs.norm.ppf(alf) * np.std(losses_out_wtd[portN])
CVaRinN[portN, q] = np.mean(losses_out_wtd[portN]) + (scs.norm.pdf(scs.norm.ppf(alf))\
                                                      / (1 - alf)) * np.std(losses_out_wtd[portN])
```

Figure 6: code for normal distribution

In-sample MC1

First, this in-sample scenario is considered to have 5000 in-sample scenarios with 1000 systemic scenarios and 5 idiosyncratic scenarios for each systemic. 1000 set groups of random numbers is generated for Yj by the normal distribution N (0,1), and each group has 100 random number. Yj, systemic risk credit driver factor correlated credit drivers, represents the number of systemic scenarios. Yj is calculated by Cholesky decomposition of correlations for credit drivers multiply the random number sets. Next, Zj is generated by using the same method as generating Yj, has a 5000 set groups of random numbers. At last, using the structural model $w_j = \beta_j y_{j(k)} + \sigma_j z_j$ of portfolio credit risk infers a counterparty's future credit state from a continuous random variable. After the creditworthiness index Wj is computed from the equation, each of counterparties future credit state from a continuous random variable can be found through comparing the value obtained from the credit_driver_corr.csv. Then the counterparties loss distribution of in-sample scenario is obtained.

```
for tr in range(N_trials):
    # Monte Carlo approximation 1
    normal_random_vector=np.random.randn(Ndriver,np.int(np.ceil(Nin / Ns)))
    y_inMC1=(sqrt_rho @ normal_random_vector).T
    for s in range(np.int(np.ceil(Nin / Ns))): # systemic scenarios s =1:1000.   5000/5
        for si in range( Ns): # idiosyncratic scenarios for each systemic  #5
            z_inMC1= np.random.randn(K,1)
            Losses_inMC1_prep=np.zeros((K,1))
            for k in range(K):  ##100 size
                cd =int(driver[k])
                w_inMC1 = beta[k] * y_inMC1[s,cd] +math.sqrt(1-beta[k]**2) * z_inMC1[k]

                temp_inMC1 = np.append(w_inMC1,CS_Bdry[k,:])
                temp_inMC1 = np.sort(temp_inMC1)
                cs_index=np.where(temp_inMC1==w_inMC1)
                Losses_inMC1_prep[k] = exposure[k,cs_index]
                # Losses_inMC1_prep (100,1)
            Losses_inMC1.append(Losses_inMC1_prep)
```

Figure7 : code for MC1

Second, the loss value of each portfolio was generated by multiply loss distribution and asset units of each portfolio.

```
portf_loss_inMC1 = np.sort(Losses_inMC1@x0[portN].T)
portf_loss_inMC2 =np.sort(Losses_inMC2@x0[portN].T)
```

Figure 9: code for MC1, MC2 loss value

Third, the VaR and CVaR of in-sample MC1 is calculated by the equation in figure 4.

```
VaRinMC1[portN, q][tr] =portf_loss_inMC1[int(math.ceil(Nin*alf))-1]
VaRinMC2[portN, q][tr] = portf_loss_inMC2[int(math.ceil(Nin*alf))-1]
CVaRinMC1[portN, q][tr] = (1 / (Nin*(1-alf))) * \
((math.ceil(Nin*alf) - Nin*alf) * VaRinMC1[portN, q][tr] + sum(portf_loss_inMC1[int(math.ceil(Nin*alf)):]))
CVaRinMC2[portN, q][tr] = (1 / (Nin*(1-alf))) * \
((math.ceil(Nin*alf) - Nin*alf) * VaRinMC2[portN, q][tr] + sum(portf_loss_inMC2[int(math.ceil(Nin*alf)):]))

mu_p_MC1 = np.mean(portf_loss_inMC1)
sigma_p_MC1 = np.std(portf_loss_inMC1)
mu_p_MC2 = np.mean(portf_loss_inMC2)
sigma_p_MC2 = np.std(portf_loss_inMC2)
VaRinN1[portN, q][tr] = np.mean(portf_loss_inMC1) + scs.norm.ppf(alf) * np.std(portf_loss_inMC1)
VaRinN2[portN, q][tr] =  np.mean(portf_loss_inMC2) + scs.norm.ppf(alf) * np.std(portf_loss_inMC2)
CVaRinN1[portN, q][tr] =  np.mean(portf_loss_inMC1) + (scs.norm.pdf(scs.norm.ppf(alf)) / \
                                                       (1 - alf)) * np.std(portf_loss_inMC1)
CVaRinN2[portN, q][tr] =  np.mean(portf_loss_inMC2) + (scs.norm.pdf(scs.norm.ppf(alf)) / \
                                                       (1 - alf)) * np.std(portf_loss_inMC2)
```

Figure 8: code for VaR, and CVaR

In-sample N1

Same as calculating in-Sample N1, 5000 in-sample scenarios with 1000 systemic scenarios and 5 idiosyncratic scenarios for each systemic. The process of generating loss distribution is same except VaR and CVaR calculation. In figure 8, it shows how to calculate VaR, and CVaR is

calculated by mean and standard deviation of the total loss value of each portfolio. The code shows in figure 6 below.

In-sample MC2

First, this in-sample scenario is considered to have 5000 in-sample scenarios with 5000 systemic scenarios and 1 idiosyncratic scenario for each systemic. 5000 set groups of random numbers is generated for Yj by the normal distribution N (0,1), and each group has 100 random number. Yj, systemic risk credit driver factor correlated credit drivers, represents the number of systemic scenarios. Yj is calculated by Cholesky decomposition of correlations for credit drivers multiply the random number sets. Next, Zj is generated by using the same method as generating Yj, has a 5000 set groups of random numbers. At last, using the structural model $w_j = \beta_j y_{j(k)} + \sigma_j z_j$ of portfolio credit risk infers a counterparty's future credit state from a continuous random variable. After the creditworthiness index Wj is computed from the equation, each of counterparties future credit state from a continuous random variable can be found through comparing the value obtained from the credit_driver_corr.csv. Then the counterparties loss distribution of in-sample scenario is obtained. The code shows in figure 7 above. The loss value of each portfolio was generated by multiply loss distribution and asset units of each portfolio shows in figure 9. The VaR and CVaR of in-sample MC2 shows in figure 8.

In-sample N2

Same as calculating in-Sample N2, 5000 in-sample scenarios with 5000 systemic scenarios and 1 idiosyncratic scenarios for each systemic. The process of generating loss distribution is same except VaR and CVaR calculation. In figure 8, it shows how to calculate VaR, and CVaR is calculated by mean and standard deviation of the total loss value of each portfolio. The code shows in figure 6.

Result and Discussion

```
Portfolio 1:

Out-of-sample: VaR 99.0% = $38368624.46, CVaR 99.0% = $46289827.34
In-sample MC1: VaR 99.0% = $38366196.94, CVaR 99.0% = $46210996.89
In-sample MC2: VaR 99.0% = $38526916.75, CVaR 99.0% = $46508915.51
In-sample No: VaR 99.0% = $26750259.97, CVaR 99.0% = $29722820.20
In-sample N1: VaR 99.0% = $26770523.07, CVaR 99.0% = $29744343.38
In-sample N2: VaR 99.0% = $26833359.87, CVaR 99.0% = $29816435.13

Out-of-sample: VaR 99.9% = $56488685.76, CVaR 99.9% = $64628062.28
In-sample MC1: VaR 99.9% = $55619881.08, CVaR 99.9% = $63901492.14
In-sample MC2: VaR 99.9% = $56200564.27, CVaR 99.9% = $64271501.70
In-sample No: VaR 99.9% = $33451109.76, CVaR 99.9% = $35879726.26
In-sample N1: VaR 99.9% = $33474213.37, CVaR 99.9% = $35903859.37
In-sample N2: VaR 99.9% = $33557913.02, CVaR 99.9% = $35995120.42


Portfolio 2:

Out-of-sample: VaR 99.0% = $27814460.45, CVaR 99.0% = $33696520.67
In-sample MC1: VaR 99.0% = $27842078.57, CVaR 99.0% = $33961829.63
In-sample MC2: VaR 99.0% = $27819468.40, CVaR 99.0% = $34070384.34
In-sample No: VaR 99.0% = $21336568.31, CVaR 99.0% = $23537334.38
In-sample N1: VaR 99.0% = $21388356.28, CVaR 99.0% = $23594480.48
In-sample N2: VaR 99.0% = $21398197.62, CVaR 99.0% = $23607274.22

Out-of-sample: VaR 99.9% = $40911849.11, CVaR 99.9% = $48112687.77
In-sample MC1: VaR 99.9% = $41548099.82, CVaR 99.9% = $47809926.02
In-sample MC2: VaR 99.9% = $41621210.49, CVaR 99.9% = $48409405.06
In-sample No: VaR 99.9% = $26297612.57, CVaR 99.9% = $28095664.21
In-sample N1: VaR 99.9% = $26361479.01, CVaR 99.9% = $28163908.30
In-sample N2: VaR 99.9% = $26377975.77, CVaR 99.9% = $28182817.21
```

Figure 10: VaR and CVaR

Overall Data Analysis

From figure 10, the out-sample value is considered to be true distribution, therefore, it is the true value to compare and evaluate the sample error and model error of each model. There are two conclusion getting from the overall results. First, for both VaR and CVaR value, the Monte Carlo approximation give the most accurate value. Second, all the normal distribution models give the lower value of VaR and CVaR. Therefore, I think the Normal approximation model underestimate the portfolio loss, and Monte Carlo approximation model is more accurate and reliable. Then I will compare them in plots.

Out of Sample vs. In Sample

For VaR 99% of both portfolio, according to figure 10 and the result printed out, the Monte Carlo simulation in sample results are so closed to the true distribution result. Monte Carlo simulation results may be larger or lower than the true distribution result. It is random happening after I run my code several times. It is also same for portfolio 2. Therefore, this is the sample error.

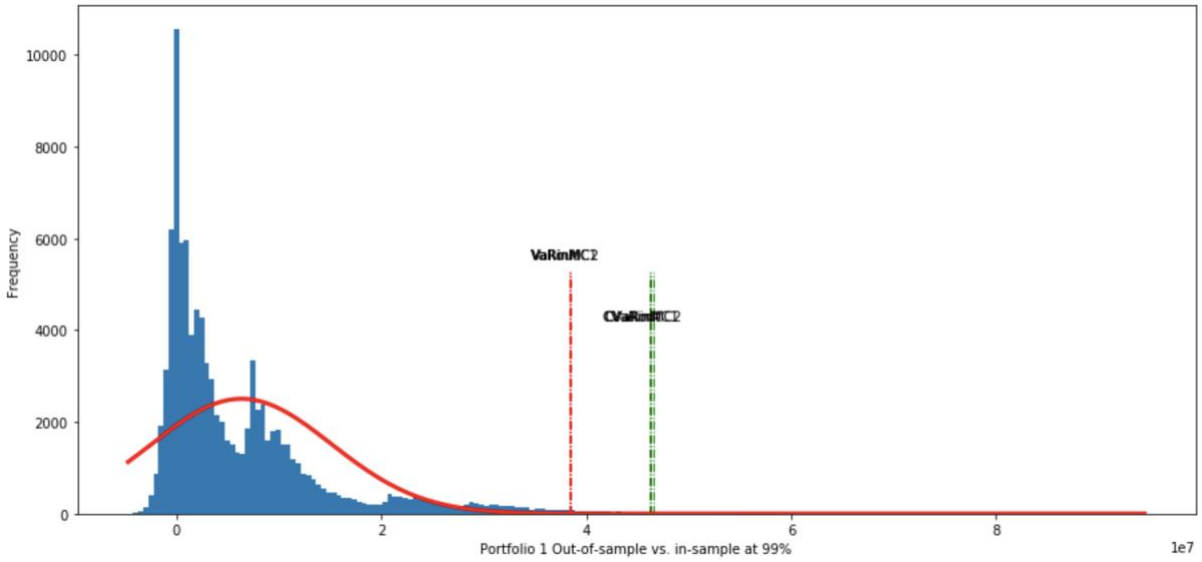

Figure 11

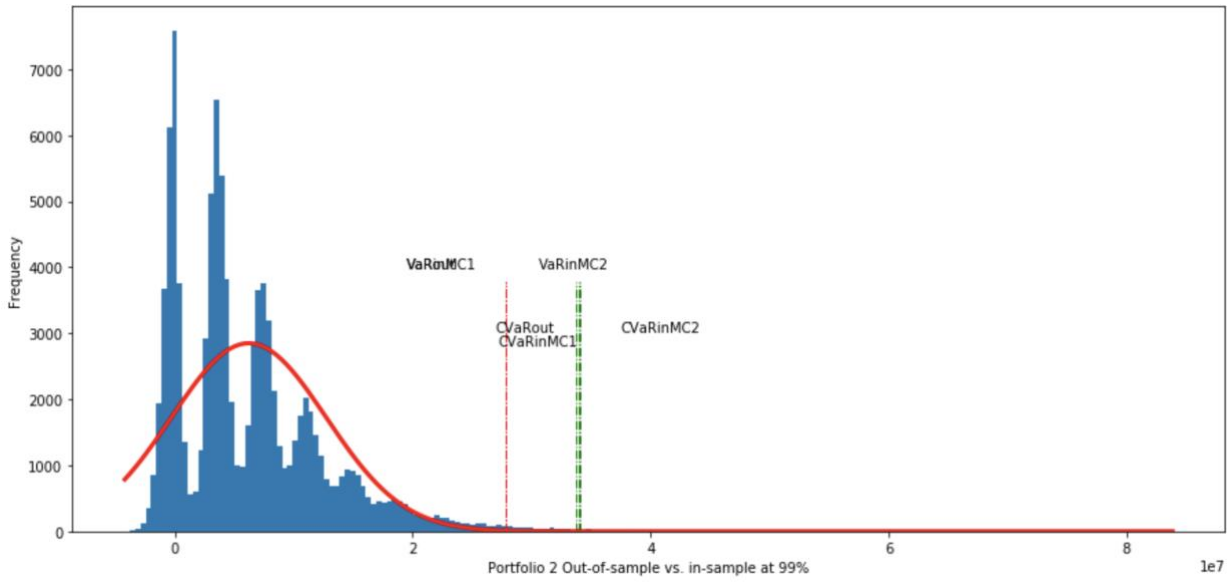

Figure 12

Out of Sample vs. Normal distribution

For VaR 99% of both portfolio, according to figure 3 ,4 and the result printed out, all the normal approximations seems to underestimate the 99% VaR loss. The true 99%VaR value is significantly larger than the normal approximation values. The trend shows in both portfolio 1 and portfolio 2. We can see the model error here.

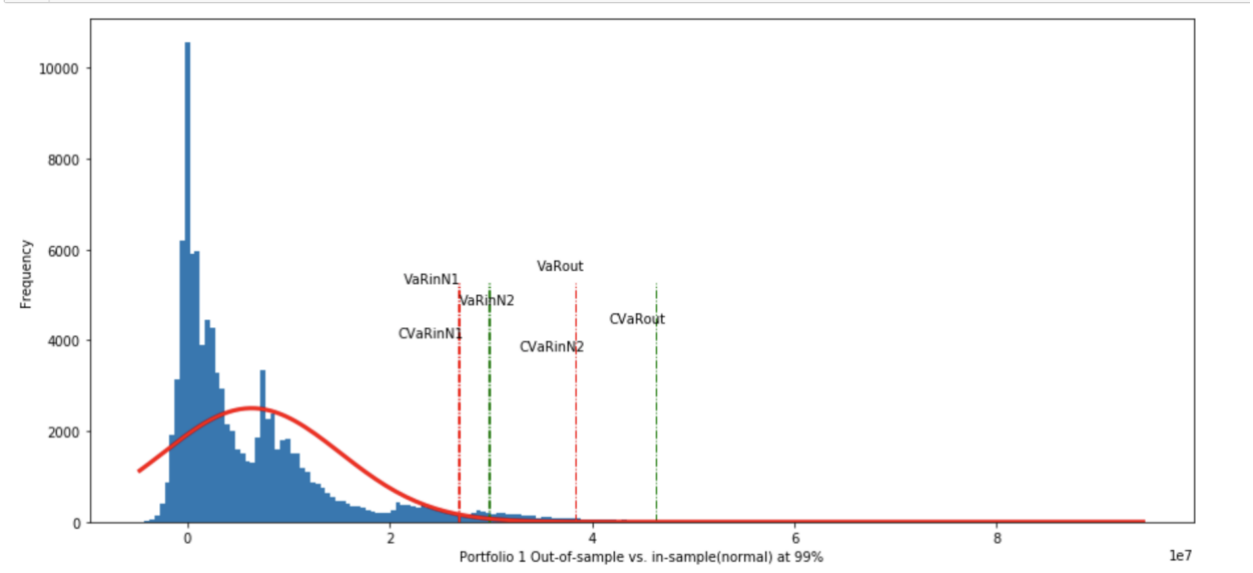

Figure 13

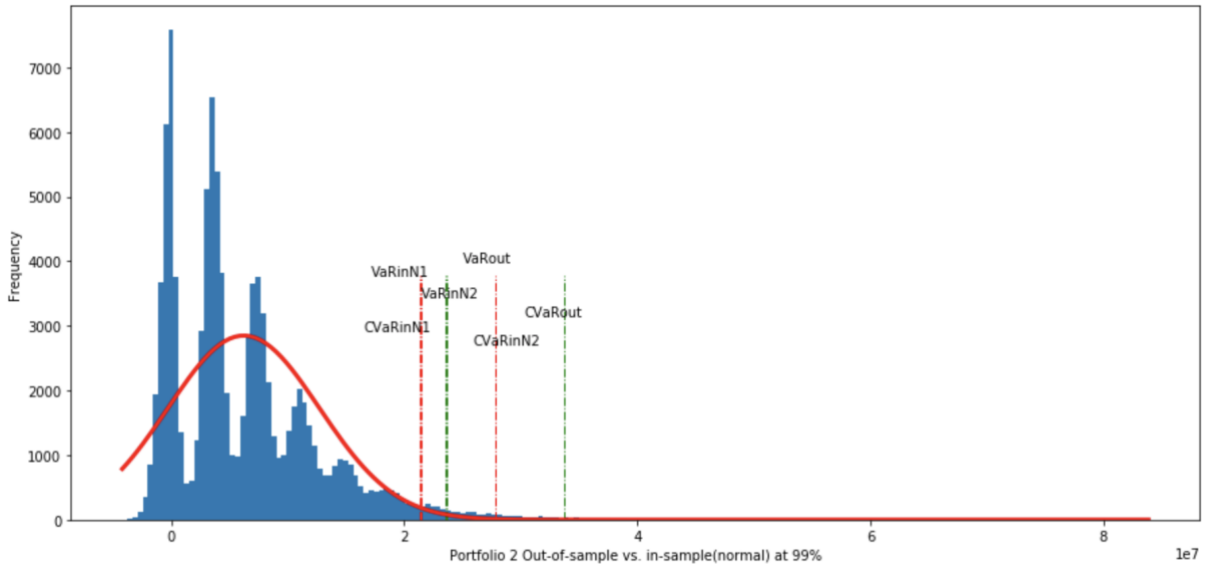

Figure 14

VaR and CVaR Comparison, Effect of Sampling Error and Model Error

From figure 11 and figure 12, all the VaR and CVaR of the in-sample results compare to those of out of sample results, we can see that in sample results may overestimate or underestimate to the 99% VaR loss. This further implies sampling error will affect predict the loss.

From figure 13 and 14, all the VaR and CVaR of Normal distribution results compare to those of out of sample results, we can tell that Normal distribution always underestimates the 99% VaR loss. This further implies that the model error will underestimate the loss.

In conclusion, in-sample normal assumption for VaR and CVaR will underestimate the loss. If just report the in-sample VaR and CVaR to decision makers in the bank could cause serious trouble, potentially leading to bankruptcy. Because both normal in-sample values have significantly large sampling error so both values are extremely under estimated. If the under estimated values are reported, the decision makers might not prepare enough capital to overcome unexpected financial risks. Therefore, the Monte Carlo approximation is more suitable for us to predict the loss.

Possible Strategies

To minimize modeling error：

1. Obtain model from historical data. Because the true historical data can use to test the model correct or not.
2. Choose Monte Carlo approximation better than Normal distribution approximation. Because, the normal distribution model always underestimate the loss.
3. Use Basel III BackTesting to test the model, and make sure the model works well

To minimize sampling error：

4. Increase the simulation times to capture the general characteristics of loss distribution. Therefore, we need a powerful computation tools.
5. Generate multiple simulation for the same portfolio and then take average of the multiple simulations

6. Use different methods to generate random number in the simulation