ImportantSampling

T Xu

```
N <- 1000 #number of loans.
PosNames <- paste("Loan",1:N,sep="")
beta <- rep(0.3,N) #factor loading
LGD <- rep(0.5,N)
PD <- rep(0.01,N)
EAD <- rep(100,N)

port <- data.frame(PosNames,PD,LGD,EAD,beta,stringsAsFactors = F)
head(port)
```

```
##   PosNames   PD LGD EAD beta
## 1    Loan1 0.01 0.5 100  0.3
## 2    Loan2 0.01 0.5 100  0.3
## 3    Loan3 0.01 0.5 100  0.3
## 4    Loan4 0.01 0.5 100  0.3
## 5    Loan5 0.01 0.5 100  0.3
## 6    Loan6 0.01 0.5 100  0.3
```

To simulate the Portfolio Loss Distribution, single factor model is used here, where the credit risk is determined by systematic factor$Z$, along with idiosyncratic risk ϵ

$$V_i = \beta_i Z + \sqrt{1-\beta_i^2}\epsilon_i$$

where Z ~ N(0,1), ϵ ~ N(0,1)

Recall that Expected Loss $\mathbb{E}[Loss] = PD * LGD * EAD$. During the simulation, for each scenario

$$Loss = LGD * EAD * 1_{Default}$$

$$1_{Default} = \begin{cases} 1, & \text{Default} \\ 0, & \text{Non-Default} \end{cases}$$

In Monte Carlo method, only $1_{Default}$ is simulated, and the loan will be Default if asset value V falls below than Default boundary that $V_i < b_i = N^{-1}(PD_i)$

Naive Monte Carlo

Systematic factor simulations are drawn from its original distribution Z ~$N(0,1)$, and Portfolio loss will be the sum of position losses, and all the scenario are equally weighted as $1/M$

```
###
simMC <- function(port,M){
  N <- nrow(port)
  Z <- (rnorm(M,mean=0))
  LossMC <- matrix(,M,1)
  for (m in 1:M){
    e <- rnorm(N,mean=0,sd=1)
    V <- beta*Z[m] + sqrt(1-beta^2)*e
    default_flag <- V < qnorm(PD)
    LossMC[m] <- sum(default_flag * port$EAD * port$LGD)
  }
  return (data.frame(Loss=LossMC,Weights=1/M))
}
```

Importance Sampling

Credit risk cares mostely about tail risk, that a portfolio can suffer extemre credit event with less than 1%. Naive Monte Carlo is inefficient as all scenario are equally weighted which requires significant number of simulation to reach a desired level of convegence. Importance Sampling can improve the variance convergence by shifting the weight more on tails during simulation.
In Importance Sapling Monte Carlo, simulations of Z is drawn from a shifted distribution $N(\mu, 1)$, so that mean of the distirubtion can be shifted to desired distribution. For example, $\mu = -3$ is used below representing about 99.86% level of loss.(the lower Z value, the higher probability that loss will occur.)

```
simIS <- function(port,M,mu){
  N <- nrow(port)
  Z <- (rnorm(M,mean=mu))
  LossIS <- matrix(,M,1)
  for (m in 1:M){
    e <- rnorm(N,mean=0,sd=1)
    V <- beta*Z[m] + sqrt(1-beta^2)*e
    #default_treshold <- (qnorm(PD)-Z*t(beta))/sqrt(1-beta^2)
    default_flag <- V < qnorm(PD)
    LossIS[m] <- sum(default_flag * port$EAD * port$LGD)
  }
  weights <- exp(-mu*Z+mu^2/2)/M
  return (data.frame(Loss=LossIS,Weights=weights))
}
```

```
library(Hmisc)
```

```
## Loading required package: grid
## Loading required package: lattice
## Loading required package: survival
## Loading required package: splines
## Loading required package: Formula
##
## Attaching package: 'Hmisc'
##
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
q = c(0.5,0.9,0.95,0.99,0.999,0.9999)
lossplot <- function(LossRes){
  LossAmt <- LossRes$Loss
  LossWgt <- LossRes$Weights
  Ecdf(LossAmt,weights = LossWgt,datadensity='density',xlim=c(0,10000),ylim=c(0.
9,1),q=q)
}

lossquantile <- function(LossRes){
  LossAmt <- LossRes$Loss
  LossWgt <- LossRes$Weights
  return(wtd.quantile(LossAmt,weights=LossWgt,probs=q,normwt=T,type='i/n'))
}

LossBench <- simMC(port,100000)
LossMC <- simMC(port,10000)
LossIS <- simIS(port,10000,-3)

Benchmark <- lossquantile(LossBench)
NaiveMC <- lossquantile(LossMC)
ImportanceSampling <- lossquantile(LossIS)
df <- data.frame(Benchmark,NaiveMC,ImportanceSampling)
```

Comparison

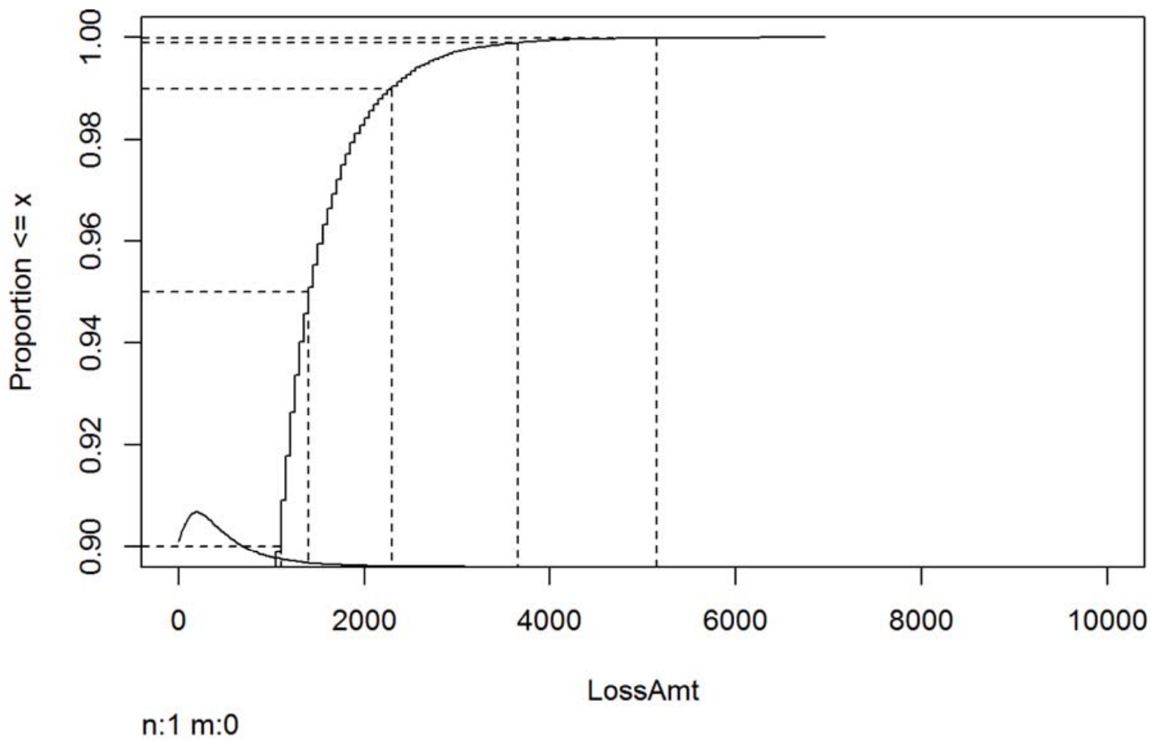
The following table shows the comparison of qifferent quantile using NaiveMC and ImportanceSampling, here the results from 100,000 simulations are used as a benchmark, and use 10,000 simulations for NaiveMC and ImportanceSampling respectively.

```
t(df)
```

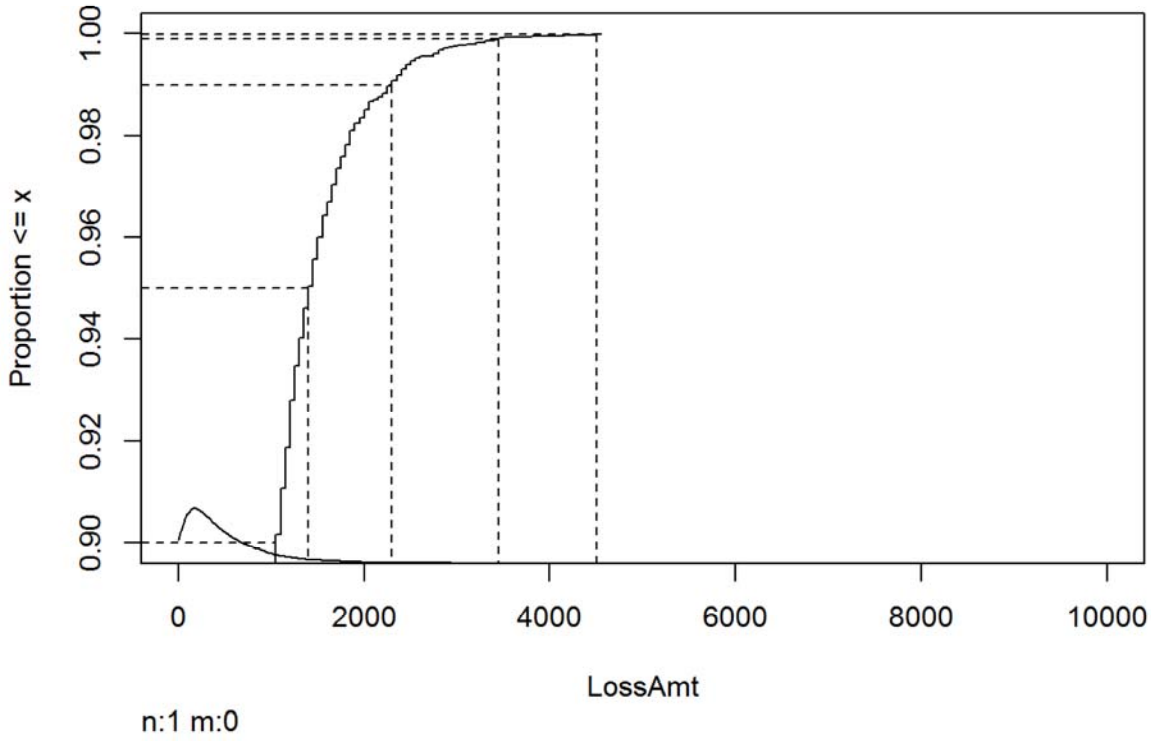
```
##                         50.00% 90.00% 95.00% 99.00% 99.90% 99.99%
## Benchmark                340.0   1055   1391   2268   3641   5117
## NaiveMC                  337.7   1043   1395   2264   3425   4500
## ImportanceSampling       369.9   1045   1400   2235   3684   5253
```

From 3 plots we can see that: 1) ImportanceSampling has better convergence than NaiveMC to Benchmark results on Loss Quantiles, espexially on quantiles above 99% 2) ImportanceSampling generate more samples at extreme losses above 2000.

```
lossplot(LossBench)
```




```
lossplot(LossMC)
```




```
lossplot(LossIS)
```