```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> #####portfolio c3 setting 2
> #####Consider porfolios on derivatives based on 10 underlying uncorrelated assets
> #####investigate the loss probability, which is critical to estimating VAR
> install.packages("rootSolve")
Installing package into ‘C:/Users/s1155058334/Documents/R/win-library/3.5’
(as ‘lib’ is unspecified)
--- Please select a CRAN mirror for use in this session ---
trying URL 'https://mirror-hk.koddos.net/CRAN/bin/windows/contrib/3.5/rootSolve_1.7.zip'
Content type 'application/zip' length 787735 bytes (769 KB)
downloaded 769 KB

package ‘rootSolve’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\s1155058334\AppData\Local\Temp\RtmpqwW4UQ\downloaded_packages
> library(rootSolve)
Warning message:
package ‘rootSolve’ was built under R version 3.5.2
> install.packages("gtools")
Installing package into ‘C:/Users/s1155058334/Documents/R/win-library/3.5’
(as ‘lib’ is unspecified)
trying URL 'https://mirror-hk.koddos.net/CRAN/bin/windows/contrib/3.5/gtools_3.8.1.zip'
Content type 'application/zip' length 325812 bytes (318 KB)
downloaded 318 KB

package ‘gtools’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\s1155058334\AppData\Local\Temp\RtmpqwW4UQ\downloaded_packages
> library(gtools)
Warning message:
package ‘gtools’ was built under R version 3.5.2
> install.packages("Matrix")
Installing package into ‘C:/Users/s1155058334/Documents/R/win-library/3.5’
(as ‘lib’ is unspecified)
trying URL 'https://mirror-hk.koddos.net/CRAN/bin/windows/contrib/3.5/Matrix_1.2-17.zip'
Content type 'application/zip' length 4475329 bytes (4.3 MB)
downloaded 4.3 MB

package ‘Matrix’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\s1155058334\AppData\Local\Temp\RtmpqwW4UQ\downloaded_packages
> library(Matrix)
Warning message:
package ‘Matrix’ was built under R version 3.5.3
> rm(list=ls())
>
> set.seed(1000)
> ##consider exchange option that exchange the ith asset FOR the (i+5)th
> S0<-rep(100,10)
> ##the initial price of the five five assets
> U0<-S0[1:5]
```

```
> ##the initial price of the last five assets
> V0<-S0[6:10]
> T<-0.1
> sigma<-0.3
> r<-0.05
> K<-S0
> dt<-0.04
> ####################################################################################
> #since V0 is occupied herem, we name value of of the portfolio at time 0 as Value0#
> ####################################################################################
>
> #####define a function for calculating the value of a unit of exchange option (long position
), under risk neutral framework
> EXVU<-function(V,U,T,t,sigma,r){
+ d1<-(log(V/U)+(sigma^2)*(T-t))/sqrt(2)/sigma/sqrt(T-t)
+ d2<-d1-sqrt(2)*sigma*sqrt(T-t)
+ exvu<-V*pnorm(d1)-U*pnorm(d2)
+ return(exvu)
+ }
> #####All the partial derivatives, thus the greeks, are evaluated at t=0
> #####Calculating the greeks of exchange options on each assets pair (exchange the ith asset
FOR the (i+5)th)
> ##the initial price of the five five assets
> U0<-S0[1:5]
> ##the initial price of the last five assets
> V0<-S0[6:10]
> d1<-(log(V0/U0)+(sigma^2)*(T))/sqrt(2)/sigma/sqrt(T)
> d2<-d1-sqrt(2)*sigma*sqrt(T)
>   #####caluculating theta of the exchange options on each asset pair
>   (thetaexvu<-(-sigma)/sqrt(2)*V0*dnorm(d1)*T^(-1/2))
[1] -26.70172 -26.70172 -26.70172 -26.70172 -26.70172
>   #####caluculating delta of the exchange options on each asset pair
>   (deltav<-pnorm(d1))
[1] 0.5267418 0.5267418 0.5267418 0.5267418 0.5267418
>   (deltau<-(-pnorm(d2)))
[1] -0.4732582 -0.4732582 -0.4732582 -0.4732582 -0.4732582
>   #####caluculating gamma of the exchange options on each asset pair
>   (gammavv<-1/V0/sqrt(2)/sigma/sqrt(T)*dnorm(d1))
[1] 0.02966857 0.02966857 0.02966857 0.02966857 0.02966857
>   (gammauu<-1/U0/sqrt(2)/sigma/sqrt(T)*dnorm(d2))
[1] 0.02966857 0.02966857 0.02966857 0.02966857 0.02966857
>   ##gammavv and gammauu happens to be the same as V0=U0 in this portfolio
>   (gammavu<-(-1)/V0/U0/sqrt(2)/sigma/sqrt(T)*U0*dnorm(d2))
[1] -0.02966857 -0.02966857 -0.02966857 -0.02966857 -0.02966857
>   (gammauv<-(-1)/V0/U0/sqrt(2)/sigma/sqrt(T)*V0*dnorm(d1))
[1] -0.02966857 -0.02966857 -0.02966857 -0.02966857 -0.02966857
> #####FDM
> (EXVU(V0,U0,T-1/250,0,sigma,r)-EXVU(V0,U0,T,0,sigma,r))/(1/250)
[1] -26.97543 -26.97543 -26.97543 -26.97543 -26.97543
> (EXVU(V0+0.01,U0,T,0,sigma,r)-EXVU(V0,U0,T,0,sigma,r))/0.01
[1] 0.5268901 0.5268901 0.5268901 0.5268901 0.5268901
> (EXVU(V0,U0+0.01,T,0,sigma,r)-EXVU(V0,U0,T,0,sigma,r))/0.01
[1] -0.4731099 -0.4731099 -0.4731099 -0.4731099 -0.4731099
> (EXVU(V0+0.01,U0,T,0,sigma,r)-2*EXVU(V0,U0,T,0,sigma,r)+EXVU(V0-0.01,U0,T,0,sigma,r))/0.01/0
.01
[1] 0.02966857 0.02966857 0.02966857 0.02966857 0.02966857
> (EXVU(V0,U0+0.01,T,0,sigma,r)-2*EXVU(V0,U0,T,0,sigma,r)+EXVU(V0,U0-0.01,T,0,sigma,r))/0.01/0
.01
[1] 0.02966857 0.02966857 0.02966857 0.02966857 0.02966857
>
>
> #####define functions for the caluculating the value of a unit of the option(long position),
 under risk neutral framework
> Call<-function(S,T,t,sigma,r,K){
+ d1<-(log(S/K)+(r+0.5*sigma^2)*(T-t))/sigma/sqrt(T-t)
+ d2<-d1-sigma*sqrt(T-t)
+ c<-S*pnorm(d1)-K*exp(-r*(T-t))*pnorm(d2)
+ return(c)
+ }
> Put<-function(S,T,t,sigma,r,K){
+ d1<-(log(S/K)+(r+0.5*sigma^2)*(T-t))/sigma/sqrt(T-t)
+ d2<-d1-sigma*sqrt(T-t)
+ p<-(-S)*pnorm(-d1)+K*exp(-r*(T-t))*pnorm(-d2)
+ return(p)
```

```
+ }
> #####All the partial derivatives, thus the greeks, are evaluated at t=0
> #####Calculating the European call options' and European put options' greeks and thus, the p
ortfolio's greeks
> d1<-(log(S0/K)+(r+0.5*sigma^2)*T)/sigma/sqrt(T)
> d2<-d1-sigma*sqrt(T)
>   #####caluculating theta of the European Call option on the 10 assets
>   (thetac<-(-S0)*dnorm(d1)*sigma/2/sqrt(T)-r*K*exp(-r*T)*pnorm(d2))
 [1] -21.32684 -21.32684 -21.32684 -21.32684 -21.32684 -21.32684 -21.32684 -21.32684 -21.32684
 -21.32684
>   #####caluculating theta of the European Put options on the 10 assets
>   (thetap<-(-S0)*dnorm(d1)*sigma/2/sqrt(T)+r*K*exp(-r*T)*pnorm(-d2))
 [1] -16.35178 -16.35178 -16.35178 -16.35178 -16.35178 -16.35178 -16.35178 -16.35178 -16.35178
 -16.35178
>   #####caluculating delta of the European Call option on the 10 assets
>   (deltac<-pnorm(d1))
 [1] 0.5398829 0.5398829 0.5398829 0.5398829 0.5398829 0.5398829 0.5398829 0.5398829 0.5398829
 0.5398829
>   #####caluculating delta of the European Put option on the 10 assets
>   (deltap<-pnorm(d1)-1)
 [1] -0.4601171 -0.4601171 -0.4601171 -0.4601171 -0.4601171 -0.4601171 -0.4601171 -0.4601171 -
0.4601171 -0.4601171
>   #####caluculating delta of the European Call option on the 10 assets
>   (gammac<-dnorm(d1)/S0/sigma/sqrt(T))
 [1] 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0
.04184189 0.04184189
>   #####caluculating delta of the European Put option on the 10 assets
>   (gammap<-dnorm(d1)/S0/sigma/sqrt(T))
 [1] 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0.04184189 0
.04184189 0.04184189
>
>
> #####characterize the portfolio(e.g. short 10 exchange options on each of the asset pair, sh
ort 10 ATM calls and short 5 ATM puts on each asset)
> weightexvu<-rep(-10,5)
> weightc<-rep(-10,10)
> weightp<-rep(-5,10)
> #####calculate the initial value of the portfolio (value at time 0)
> ##the initial price of the five five assets
> U0<-S0[1:5]
> ##the initial price of the last five assets
> V0<-S0[6:10]
> ##the initial price of the portfolio
> Value0<-sum(weightexvu*EXVU(V0,U0,T,0,sigma,r))+sum(weightc*Call(S0,T,0,sigma,r,K)+weightp*P
ut(S0,T,0,sigma,r,K))
> Value0
[1] -846.7491
> #####In order to suimulate the loss, we have to be able to samples the change in asset price
s(assumed to follow multivariate normal),
> #####Hence we need to approximate the SIGMA, given the asset price are uncorrelated, SIGMA i
s diagonal
> #####notice that SIGMA is about the underlying assets itself, not the derivatives based on t
hem
> sigmadum<-S0^2*exp(2*r*dt)*(exp(sigma^2*dt)-1)
> SIGMA<-diag(sigmadum,10)
>
>
> #####consider Delta-GAMMA approximation of the portfolio loss, calculating the greeks of the
 portfolio
> #####caluculating theta of the portfolio consisting of exchange options on 5 asset pairs
> THETA<-sum(weightexvu*thetaexvu)+sum(weightc*thetac+weightp*thetap)
> #####caluculating delta of the portfolio(by assets) consisting of exchange options on 5 asse
t pairs
> dumdelta<-c(weightexvu*deltau,weightexvu*deltav)+weightc*deltac+weightp*deltap
> delta<-matrix(dumdelta,10,1)
> #####caluculating gamma of the portfolio(by pairs of assets) consisting of exchange options
on 5 asset pairs, Given exchange option is multiasset, GAMMA is not diagonal
> GAMMA<-matrix(0,10,10)
> dumgamma<-c(weightexvu*gammauu,weightexvu*gammavv)+weightc*gammac+weightp*gammap
> diag(GAMMA)<-dumgamma
> #for(i in 1:5){
> #GAMMA[i,i+5]<-weightexvu[i]*gammauv[i]
> #GAMMA[i+5,i]<-weightexvu[i]*gammavu[i]
> #}
```

```
> #####Caluculating parameters for the Delta-Gamma approximatino on the portfolio loss
> a0=-THETA*dt
> a=-delta
> A=-1/2*GAMMA
>
>
> #---------------------------------------------------------------------------------------------------------------------------------------#
>
>
> #####
> #####step1: Express Q in diagonal form
> Ct<-t(chol(SIGMA))
> ED<-eigen(t(Ct)%*%A%*%Ct)
> U<-ED$vectors
> LAMBDA<-diag(ED$values,10)
> C<-Ct%*%U
> b<-t(C)%*%a
> #define a function to calculate Q
> Q<-function(Z){t(b)%*%Z+t(Z)%*%LAMBDA%*%Z}
>
>
> #####
> #####step2: Identify the IS distribution Z~N(thetax*B(thetax)%*%b,B(thetax)), B(thetax)=solve(I-2thetax*LAMBDA)
> ###Given x, find thetax that makes E[Q]=(x-a0) under the IS chagne of measure (assume D-G approximation is exact)
> ###The x is adjusted so that the loss probability is close to 1.1%, xstd=2.7 under the original distribution of Z
> vecb<-as.vector(b)
> veclambda<-diag(LAMBDA)
> #xstd<-2.7
> #x<-(a0+sum(veclambda))+xstd*sqrt(sum(vecb^2)+2*sum(veclambda^2))
> ######################################################################
> ###Caution: x in setting 2 should be the same as that in setting 1###
> ######################################################################
> x<-371.3361
> ###To identify thetax, we numerically solve psipithetax=(x-a0), notice that E[Q]=psipitheta for general theta
> psipithetax<-function(thetax){
+ (thetax*vecb[1]^2*(1-thetax*veclambda[1])/(1-2*thetax*veclambda[1])^2 + veclambda[1]/(1-2*thetax*veclambda[1])
+ +thetax*vecb[2]^2*(1-thetax*veclambda[2])/(1-2*thetax*veclambda[2])^2 + veclambda[2]/(1-2*thetax*veclambda[2])
+ +thetax*vecb[3]^2*(1-thetax*veclambda[3])/(1-2*thetax*veclambda[3])^2 + veclambda[3]/(1-2*thetax*veclambda[3])
+ +thetax*vecb[4]^2*(1-thetax*veclambda[4])/(1-2*thetax*veclambda[4])^2 + veclambda[4]/(1-2*thetax*veclambda[4])
+ +thetax*vecb[5]^2*(1-thetax*veclambda[5])/(1-2*thetax*veclambda[5])^2 + veclambda[5]/(1-2*thetax*veclambda[5])
+ +thetax*vecb[6]^2*(1-thetax*veclambda[6])/(1-2*thetax*veclambda[6])^2 + veclambda[6]/(1-2*thetax*veclambda[6])
+ +thetax*vecb[7]^2*(1-thetax*veclambda[7])/(1-2*thetax*veclambda[7])^2 + veclambda[7]/(1-2*thetax*veclambda[7])
+ +thetax*vecb[8]^2*(1-thetax*veclambda[8])/(1-2*thetax*veclambda[8])^2 + veclambda[8]/(1-2*thetax*veclambda[8])
+ +thetax*vecb[9]^2*(1-thetax*veclambda[9])/(1-2*thetax*veclambda[9])^2 + veclambda[9]/(1-2*thetax*veclambda[9])
+ +thetax*vecb[10]^2*(1-thetax*veclambda[10])/(1-2*thetax*veclambda[10])^2 + veclambda[10]/(1-2*thetax*veclambda[10]))-(x-a0)
+ }
> curve(psipithetax)
> abline(h=0,v=0)
> uni<-uniroot.all(psipithetax,c(0,0.05))
> uni
[1] 0.01072071 0.04227634
>
> ###choose the thetax that makes a valid change of measure
> k<-0
> for(i in 1:length(uni)){
+ if(sum(sign(1-2*uni[i]*veclambda))==length(veclambda)){
+ k<-i
+ break}
+ }
```

```
> (thetax<-uni[k])
[1] 0.01072071
> psipithetax(thetax)
[1] -0.2858664
> ax<-thetax-0.0001
> bx<-thetax+0.0001
> while(abs(psipithetax(thetax))>0.0000001){
+ thetax<-(ax+bx)/2
+ ifelse(sign(psipithetax(thetax))==sign(bx),bx<-(ax+bx)/2,ax<-(ax+bx)/2)
+ }
> thetax
[1] 0.01072521
> psipithetax(thetax)
[1] 7.965878e-08
>
> ###identify the IS distribution
> Bthetax<-solve(diag(10)-2*thetax*LAMBDA)
> muthetax<-thetax*Bthetax%*%b
>
> ###generate 5000000 samples of Q under IS change of measure, check whether E[Q] approximatel
y equals (x-a0)
> Qsamples<-rep(0,5000000)
> for(j in 1:5000000){
+ Z<-muthetax+chol(Bthetax)%*%matrix(rnorm(10),10,1)
+ Qsamples[j]<-Q(Z)
+ }
> ###check whether E[Q] approximately equals (x-a0) under the importance sampling change of me
asure
> mean(Qsamples)
[1] 542.7429
> x-a0
[1] 542.7505
> a0
[1] -171.4144
> x
[1] 371.3361
> thetax
[1] 0.01072521
>
>
> ###display the parameters
>  SIGMA
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 36.20943  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [2,]  0.00000 36.20943  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [3,]  0.00000  0.00000 36.20943  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [4,]  0.00000  0.00000  0.00000 36.20943  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [5,]  0.00000  0.00000  0.00000  0.00000 36.20943  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [6,]  0.00000  0.00000  0.00000  0.00000  0.00000 36.20943  0.00000  0.00000  0.00000  0.0000
0
 [7,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 36.20943  0.00000  0.00000  0.0000
0
 [8,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 36.20943  0.00000  0.0000
0
 [9,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 36.20943  0.0000
0
[10,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 36.2094
3
>  THETA
[1] 4285.359
>  a0
[1] -171.4144
>  delta
          [,1]
 [1,]  1.634338
 [2,]  1.634338
 [3,]  1.634338
 [4,]  1.634338
```

```
 [5,]  1.634338
 [6,] -8.365662
 [7,] -8.365662
 [8,] -8.365662
 [9,] -8.365662
[10,] -8.365662
>  a
           [,1]
 [1,] -1.634338
 [2,] -1.634338
 [3,] -1.634338
 [4,] -1.634338
 [5,] -1.634338
 [6,]  8.365662
 [7,]  8.365662
 [8,]  8.365662
 [9,]  8.365662
[10,]  8.365662
>  GAMMA
            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
      [,9]      [,10]
 [1,] -0.9243141  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [2,]  0.0000000 -0.9243141  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [3,]  0.0000000  0.0000000 -0.9243141  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [4,]  0.0000000  0.0000000  0.0000000 -0.9243141  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [5,]  0.0000000  0.0000000  0.0000000  0.0000000 -0.9243141  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [6,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.9243141  0.0000000  0.0000000
 0.0000000  0.0000000
 [7,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.9243141  0.0000000
 0.0000000  0.0000000
 [8,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.9243141
 0.0000000  0.0000000
 [9,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
-0.9243141  0.0000000
[10,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000 -0.9243141
>  A
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 0.462157 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [2,] 0.000000 0.462157 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [3,] 0.000000 0.000000 0.462157 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [4,] 0.000000 0.000000 0.000000 0.462157 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [5,] 0.000000 0.000000 0.000000 0.000000 0.462157 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [6,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.462157 0.000000 0.000000 0.000000 0.00000
0
 [7,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.462157 0.000000 0.000000 0.00000
0
 [8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.462157 0.000000 0.00000
0
 [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.462157 0.00000
0
[10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.46215
7
>
>  b
           [,1]
 [1,] 50.339759
 [2,] 50.339759
 [3,] 50.339759
 [4,] 50.339759
 [5,] 50.339759
 [6,] -9.834509
 [7,] -9.834509
```

```
 [8,] -9.834509
 [9,] -9.834509
[10,] -9.834509
>  LAMBDA
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 16.73444  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [2,]  0.00000 16.73444  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [3,]  0.00000  0.00000 16.73444  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [4,]  0.00000  0.00000  0.00000 16.73444  0.00000  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [5,]  0.00000  0.00000  0.00000  0.00000 16.73444  0.00000  0.00000  0.00000  0.00000  0.0000
0
 [6,]  0.00000  0.00000  0.00000  0.00000  0.00000 16.73444  0.00000  0.00000  0.00000  0.0000
0
 [7,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 16.73444  0.00000  0.00000  0.0000
0
 [8,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 16.73444  0.00000  0.0000
0
 [9,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 16.73444  0.0000
0
[10,]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000 16.7344
4
>  muthetax
            [,1]
 [1,]  0.8422331
 [2,]  0.8422331
 [3,]  0.8422331
 [4,]  0.8422331
 [5,]  0.8422331
 [6,] -0.1645409
 [7,] -0.1645409
 [8,] -0.1645409
 [9,] -0.1645409
[10,] -0.1645409
>  Bthetax
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 1.559967 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [2,] 0.000000 1.559967 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [3,] 0.000000 0.000000 1.559967 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [4,] 0.000000 0.000000 0.000000 1.559967 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [5,] 0.000000 0.000000 0.000000 0.000000 1.559967 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [6,] 0.000000 0.000000 0.000000 0.000000 0.000000 1.559967 0.000000 0.000000 0.000000 0.00000
0
 [7,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.559967 0.000000 0.000000 0.00000
0
 [8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.559967 0.000000 0.00000
0
 [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.559967 0.00000
0
[10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.55996
7
>
>
> #----------------------------------------------------------------------------------------------
----#
>
> ###define a function that calculate the Loss for a generated ds (ds=C%*%Z)
> L<-function(dS){
+ Sdt<-S0+as.vector(dS)
+ Udt<-Sdt[1:5]
+ Vdt<-Sdt[6:10]
+         Valuedt<-sum(weightexvu*EXVU(Vdt,Udt,T,dt,sigma,r))+sum(weightc*Call(Sdt,T,dt,sigma,
r,K)+weightp*Put(Sdt,T,dt,sigma,r,K))
+ return(Value0-Valuedt)
```

```
+ }
> ###define a function that calculate the likelihood ratio for a generated Z
> likelihood<-function(Z){
+ p1<-sum((1/2)*((thetax*vecb)^2/(1-2*thetax*veclambda)-log(1-2*thetax*veclambda)))
+ p2<-thetax*Q(Z)
+ return(exp(p1-p2))
+ }
>
> #-----------------------------------------------------------------------------------------
----#
>
>
> #####
> #####step3: Define k strata
> ###plot the empirical CDF
> Qsamples<-sort(Qsamples,decreasing=FALSE)
> ECDFc3s2<-ecdf(Qsamples)
> #plot.ecdf(Qsamples)
> ###mimics the quantiles of Q using the 5000000 samples of Q generated in the previous step
> stratabyQ<-rep(0,40-1)
> for(i in 1:39){stratabyQ[i]<-quantile(Qsamples,0.025*i)}
> stratabyQ
 [1]  122.0814  173.6091  209.5636  238.6229  263.7031  286.0594  306.5822  325.8003  343.9173
  361.2370  377.9924  394.2412
[13]  410.1301  425.7534  441.2190  456.6033  471.9944  487.3507  502.8460  518.4016  534.1820
  550.1862  566.5185  583.2056
[25]  600.4092  618.2435  636.6330  655.9316  676.2258  697.8037  720.8077  745.7018  772.9482
  803.2715  837.7614  878.2164
[37]  928.1321  994.7630 1101.7621
> ECDFc3s2(stratabyQ)
 [1] 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275 0.300 0.325 0.350 0.375
 0.400 0.425 0.450 0.475 0.500
[21] 0.525 0.550 0.575 0.600 0.625 0.650 0.675 0.700 0.725 0.750 0.775 0.800 0.825 0.850 0.875
 0.900 0.925 0.950 0.975
>
> ###calculate the optimal alocation of samples size for each strata
> options(warn=-1)
> bins<-c(stratabyQ,.Machine$double.xmax)
> vars<-matrix(0,40,10000)
> counts<-rep(0,40)
> while(sum(counts)!=400000){
+ Z<-muthetax+chol(Bthetax)%*%matrix(rnorm(10),10,1)
+ k<-tail(binsearch(function(y) bins[y]-(Q(Z)), range=c(1, length(bins)))$where,1)
+ if(counts[k]<10000){
+ counts[k]<-counts[k]+1
+ vars[k,counts[k]]<-ifelse(L(C%*%Z)>x,1,0)*likelihood(Z)
+ }
+ else{}
+ }
> (dumvar<-apply(vars,1,var))
 [1] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+0
0 0.000000e+00 0.000000e+00
[10] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 1.044646e-05 2.290717e-05 1.064347e-0
4 4.384541e-04 1.168155e-03
[19] 2.014660e-03 2.501296e-03 2.474097e-03 1.985581e-03 1.394996e-03 8.750505e-04 4.969679e-0
4 2.660372e-04 1.158448e-04
[28] 5.108442e-05 1.934809e-05 7.660848e-06 3.231088e-06 1.168354e-06 6.651659e-07 3.830814e-0
7 2.412077e-07 1.490428e-07
[37] 8.543893e-08 4.395304e-08 1.906759e-08 4.380973e-09
> #Since we assume equiprobable strata, pj=1/k, where k is the number of strata, which is 40 i
n the case
> (qj<-sqrt(dumvar)/sum(sqrt(dumvar)))
 [1] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.000000000
0 0.0000000000 0.0000000000
[10] 0.0000000000 0.0000000000 0.0000000000 0.0000000000 0.0080972855 0.0119905934 0.025846194
2 0.0524586340 0.0856258927
[19] 0.1124490668 0.1252960686 0.1246129912 0.1116345641 0.0935710353 0.0741090895 0.055849475
1 0.0408626087 0.0269645496
[28] 0.0179060242 0.0110198060 0.0069341520 0.0045032841 0.0027079591 0.0020432426 0.001550602
4 0.0012304118 0.0009671869
[37] 0.0007322896 0.0005252298 0.0003459416 0.0001658213
>
>
> #####
```

```
> #####step4: Perform the simulation
> ###define a function to generate estimates of P{L>xp} using three methods: SMC, IS, ISSQ, IS
SQO
> options(warn=-1)
> run<-function(n,strata){
+
+ results<-rep(0,4)
+ SMC<-0
+ IS<-0
+ ISSQ<-0
+ ISSQO<-0
+
+ bins<-c(stratabyQ,.Machine$double.xmax)
+ binscount<-rep(0,strata)
+ binscountpi<-rep(0,strata)
+
+ nj<-round(n*qj)
+ nj[match(max(nj),nj)]<-nj[match(max(nj),nj)]+(n-sum(nj))
+
+
+ for(i in 1:n){
+ Z1<-matrix(rnorm(10),10,1)
+ #Standard Monte Carlo
+ dS1<-C%*%Z1
+ L1<-L(dS1)
+ SMC<-SMC+(ifelse(L1>x,1,0)*(1/n))
+
+ Z2<-muthetax+chol(Bthetax)%*%Z1
+ #Monte Carlo (IS)
+ dS2<-C%*%Z2
+ L2<-L(dS2)
+ IS<-IS+(ifelse(L2>x,1,0)*likelihood(Z2)*(1/n))
+ kthbins<-tail(binsearch(function(y) bins[y]-(Q(Z2)), range=c(1, length(bins)))$where,1)
+ #Monte Carlo (IS and Stratification)
+ if(binscount[kthbins]<(n/strata)){
+ binscount[kthbins]<-binscount[kthbins]+1
+ ISSQ<-ISSQ+(ifelse(L2>x,1,0)*likelihood(Z2)*(1/n))
+ }
+ else{
+ }
+ #Monte Carlo (IS and Stratification with optimized smaple size for each strata)
+ if(binscountpi[kthbins]<nj[kthbins]){
+ binscountpi[kthbins]<-binscountpi[kthbins]+1
+ ISSQO<-ISSQO+(ifelse(L2>x,1,0)*likelihood(Z2)*(1/nj[kthbins])*(1/strata))
+ }
+ else{
+ }
+ }
+ results[1]<-SMC
+ results[2]<-IS
+
+
+ while(sum(binscount)<n){
+ Z2<-muthetax+chol(Bthetax)%*%matrix(rnorm(10),10,1)
+ kthbins<-tail(binsearch(function(y) bins[y]-(Q(Z2)), range=c(1, length(bins)))$where,1)
+ #Monte Carlo (IS and Stratification) continue...
+ if(binscount[kthbins]<(n/strata)){
+ binscount[kthbins]<-binscount[kthbins]+1
+ ISSQ<-ISSQ+(ifelse(L(C%*%Z2)>x,1,0)*likelihood(Z2)*(1/n))
+ }
+ else{
+ }
+ #Monte Carlo (IS and Stratification with optimized sample size for each strata) continue...
+ if(binscountpi[kthbins]<nj[kthbins]){
+ binscountpi[kthbins]<-binscountpi[kthbins]+1
+ ISSQO<-ISSQO+(ifelse(L(C%*%Z2)>x,1,0)*likelihood(Z2)*(1/nj[kthbins])*(1/strata))
+ }
+ else{
+ }
+ }
+ results[3]<-ISSQ
+
+ while(sum(binscountpi)<n){
+ Z2<-muthetax+chol(Bthetax)%*%matrix(rnorm(10),10,1)
```

```
+ kthbins<-tail(binsearch(function(y) bins[y]-(Q(Z2)), range=c(1, length(bins)))$where,1)
+ #Monte Carlo (IS and Stratification with optimized sample size for each strata) continue...
+ if(binscountpi[kthbins]<nj[kthbins]){
+ binscountpi[kthbins]<-binscountpi[kthbins]+1
+ ISSQO<-ISSQO+(ifelse(L(C%*%Z2)>x,1,0)*likelihood(Z2)*(1/nj[kthbins])*(1/strata))
+ }
+ else{
+ }
+ }
+ results[4]<-ISSQO
+
+ return(results)
+ }
> run(1000,40)
[1] 0.00900000 0.01070954 0.01062695 0.01054714
> run(10000,40)
[1] 0.01100000 0.01104863 0.01113924 0.01099782
>
> ###define a function to generate the replications
> replication<-function(N,n,strata){
+ dum<-c(0,0,0,0)
+ for(i in 1:N){
+ dum<-rbind(dum,run(n,strata))
+ }
+ return(tail(dum,-1))
+ }
>
>
> #####
> #####Step5:evaluate the performance of the algorithm
> SAMPLES<-replication(10000,10000,40)
> (ISratio<-var(SAMPLES[,1])/var(SAMPLES[,2]))
[1] 19.23134
> (ISSQratio<-var(SAMPLES[,1])/var(SAMPLES[,3]))
[1] 31.79879
> (ISSQOratio<-var(SAMPLES[,1])/var(SAMPLES[,4]))
[1] 109.6501
>
> n<-10000
> strata<-40
> var(SAMPLES[,1])
[1] 1.09682e-06
> (sum(sqrt(dumvar)*(1/strata)))^2/n
[1] 9.957955e-09
> (theoreticalISSQOratio<-var(SAMPLES[,1])/((sum(sqrt(dumvar)*(1/strata)))^2/n))
[1] 110.1451
>
> save.image("C:\\Users\\s1155058334\\Desktop\\c32os5workspace")
> save.image("C:\\Users\\s1155058334\\Desktop\\portfolio c3 setting 2 (5.3) os5 pending\\c32os
5workspace")
>
```