```
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 b6
> #####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\RtmpIj4iYj\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\RtmpIj4iYj\downloaded_packages
> library(gtools)
Warning message:
package ‘gtools’ was built under R version 3.5.2
> rm(list=ls())
>
> set.seed(4000)
> S0<-rep(100,10)
> T<-rep(0.1,10)
> sigma<-rep(0.3,10)
> r<-rep(0.05,10)
> K<-rep(100,10)
> H<-rep(95,10)
> dt<-0.04
> #####for the cash or nothing put options
> cash<-K
>
>
> #####define a function for calculating the value of a unit of down-and-out call option (long
 position), under risk neutral framework
> #####notice that H<K, refers to P.579 of Options, Futures, and Other Derivatives(8 ed.) for
the pricing formula of down-and-out call option.
> #####MUST take into account the situation that the call options are knocked out at time t+de
ltat
> Cdo<-function(S,T,t,sigma,r,K,H){
```

```
+ lam<-(r+0.5*sigma^2)/(sigma^2)
+ y<-log((H^2)/S/K)/sigma/sqrt(T-t)+lam*sigma*sqrt(T-t)
+ cdi<-S*(H/S)^(2*lam)*pnorm(y)-K*exp(-r*(T-t))*(H/S)^(2*lam-2)*pnorm(y-sigma*sqrt(T-t))
+
+ 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)
+
+ value<-(c-cdi)*(S>H)
+ return(value)
+ }
> #####All the partial derivatives, thus the greeks, are evaluated at t=0
> #####Calculating the greeks of down-and-out call options on each assets
> #####Since the theoretical greeks is complecated to derive, we approximate them with the Fin
ite Difference Method at the moment
> dS<-0.01
>   #####approximating delta of the Cdo's on each asset using FDM
>   deltacdo<-(Cdo(S0+dS,T,0,sigma,r,K,H)-Cdo(S0,T,0,sigma,r,K,H))/dS
>   #####approximating gamma of the cdo's on each asset using FDM
>   gammacdo<-(Cdo(S0+dS,T,0,sigma,r,K,H)-2*Cdo(S0,T,0,sigma,r,K,H)+Cdo(S0-dS,T,0,sigma,r,K,H)
)/dS/dS
>   #####approximating theta of the cdo's on each asset by the relation among theta, delta and
 gamma of an derivative at time 0 (OFAOD ed.10 P.393)
>   thetacdo<-r*Cdo(S0,T,0,sigma,r,K,H)-r*S0*deltacdo-0.5*sigma^2*S0^2*gammacdo
>
>
> #####define a function for caluculating the value of a cash-or-nothing put option(long posit
ion), under risk neutral framework
> Pcon<-function(S,T,t,sigma,r,K,cash){
+ d1<-(log(S/K)+(r+0.5*sigma^2)*(T-t))/sigma/sqrt(T-t)
+ d2<-d1-sigma*sqrt(T-t)
+ pcon<-cash*exp(-r*(T-t))*pnorm(-d2)
+ return(pcon)
+ }
> #####All the partial derivatives, thus the greeks, are evaluated at t=0
> #####Calculating the greeks of cash-or-nothing put options on each assets
> d1<-(log(S0/K)+(r+0.5*sigma^2)*T)/sigma/sqrt(T)
> d2<-d1-sigma*sqrt(T)
>   #####caluculating theta of the cash or nothing put options on each asset
>   thetapcon<-r*cash*exp(-r*T)*pnorm(-d2)-cash*exp(-r*T)*dnorm(-d2)*(log(S0/K)/2/sigma*T^(-3/
2)-(r-0.5*sigma^2)/2/sigma*T^(-1/2))
>   #####caluculating delta of the cash or nothing put option on each asset
>   deltapcon<-(-cash)*exp(-r*T)/S0/sigma/sqrt(T)*dnorm(-d2)
>   #####caluculating gamma of the cash or nothing put option on each asset
>   gammapcon<-(r*Pcon(S0,T,0,sigma,r,K,cash)-thetapcon-r*S0*deltapcon)*2/(sigma^2)/(S0^2)
> ######Cross validate against approximated greeks using FDM
> ##dS<-0.01
> ##deltapconpi<-(Pcon(S0+dS,T,0,sigma,r,K,cash)-Pcon(S0,T,0,sigma,r,K,cash))/dS
> ##gammapconpi<-(Pcon(S0+dS,T,0,sigma,r,K,cash)-2*Pcon(S0,T,0,sigma,r,K,cash)+Pcon(S0-dS,T,0,
sigma,r,K,cash))/dS/dS
> ##thetapconpi<-r*Pcon(S0,T,0,sigma,r,K,cash)-r*S0*deltapconpi-0.5*sigma^2*S0^2*gammapconpi
> #####Checked
>
>
> #####characterize b6(short 10 down-and-out calls and short certain cash-or-nothing put optio
ns on each assets to make the portfolio delta hedged)
> weightcdo<-rep(-10,10)
> weightpcon<-(-weightcdo*deltacdo)/deltapcon
> #####calculate the initial value of the portfolio (value at time 0)
> V0<-sum(weightcdo*Cdo(S0,T,0,sigma,r,K,H)+weightpcon*Pcon(S0,T,0,sigma,r,K,cash))
> V0
[1] -1146.089
> #####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
> 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 mix of the options on the 10 assets
> THETA<-sum(weightcdo*thetacdo+weightpcon*thetapcon)
```

```
> #####caluculating delta of the portfolio(by assets) consisting of mix of the options on the
10 assets
> delta<-matrix(weightcdo*deltacdo+weightpcon*deltapcon,10,1)
> #####caluculating gamma of the portfolio(by pairs of assets) consisting of mix of the option
s on the 10 assets.
> GAMMA<-diag(weightcdo*gammacdo+weightpcon*gammapcon,10)
> #####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)=solv
e(I-2thetax*LAMBDA)
> ###Given x, find thetax that makes E[Q]=(x-a0) under the IS chagne of measure (assume D-G ap
proximation is exact)
> ###The x is adjusted so that the loss probability is close to 1%, xstd=9 under the original
distribution of Z
> vecb<-as.vector(b)
> veclambda<-diag(LAMBDA)
> xstd<-9
> x<-(a0+sum(veclambda))+xstd*sqrt(sum(vecb^2)+2*sum(veclambda^2))
> ###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*th
etax*veclambda[1])
+ +thetax*vecb[2]^2*(1-thetax*veclambda[2])/(1-2*thetax*veclambda[2])^2 + veclambda[2]/(1-2*th
etax*veclambda[2])
+ +thetax*vecb[3]^2*(1-thetax*veclambda[3])/(1-2*thetax*veclambda[3])^2 + veclambda[3]/(1-2*th
etax*veclambda[3])
+ +thetax*vecb[4]^2*(1-thetax*veclambda[4])/(1-2*thetax*veclambda[4])^2 + veclambda[4]/(1-2*th
etax*veclambda[4])
+ +thetax*vecb[5]^2*(1-thetax*veclambda[5])/(1-2*thetax*veclambda[5])^2 + veclambda[5]/(1-2*th
etax*veclambda[5])
+ +thetax*vecb[6]^2*(1-thetax*veclambda[6])/(1-2*thetax*veclambda[6])^2 + veclambda[6]/(1-2*th
etax*veclambda[6])
+ +thetax*vecb[7]^2*(1-thetax*veclambda[7])/(1-2*thetax*veclambda[7])^2 + veclambda[7]/(1-2*th
etax*veclambda[7])
+ +thetax*vecb[8]^2*(1-thetax*veclambda[8])/(1-2*thetax*veclambda[8])^2 + veclambda[8]/(1-2*th
etax*veclambda[8])
+ +thetax*vecb[9]^2*(1-thetax*veclambda[9])/(1-2*thetax*veclambda[9])^2 + veclambda[9]/(1-2*th
etax*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(0,0.5))
> uni
[1] 0.09823881 0.12261366
>
> ###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] 0.09823881
> psipithetax(thetax)
[1] -0.1393996
> 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] 0.09825543
> psipithetax(thetax)
[1] 2.145109e-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] 204.8486
> x-a0
[1] 204.8193
> a0
[1] -38.23277
> x
[1] 166.5866
> thetax
[1] 0.09825543
>
> ###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] 955.8192
>  a0
[1] -38.23277
>  delta
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
```

```
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
>  a
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
>  GAMMA
            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]       [,7]       [,8]
      [,9]      [,10]
 [1,] -0.2251386  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [2,]  0.0000000 -0.2251386  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [3,]  0.0000000  0.0000000 -0.2251386  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [4,]  0.0000000  0.0000000  0.0000000 -0.2251386  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [5,]  0.0000000  0.0000000  0.0000000  0.0000000 -0.2251386  0.0000000  0.0000000  0.0000000
 0.0000000  0.0000000
 [6,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.2251386  0.0000000  0.0000000
 0.0000000  0.0000000
 [7,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.2251386  0.0000000
 0.0000000  0.0000000
 [8,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.2251386
 0.0000000  0.0000000
 [9,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
-0.2251386  0.0000000
[10,]  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 0.0000000 -0.2251386
>  A
           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]      [,9
]     [,10]
 [1,] 0.1125693 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
0 0.0000000
 [2,] 0.0000000 0.1125693 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
0 0.0000000
 [3,] 0.0000000 0.0000000 0.1125693 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
0 0.0000000
 [4,] 0.0000000 0.0000000 0.0000000 0.1125693 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
0 0.0000000
 [5,] 0.0000000 0.0000000 0.0000000 0.0000000 0.1125693 0.0000000 0.0000000 0.0000000 0.000000
0 0.0000000
 [6,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.1125693 0.0000000 0.0000000 0.000000
0 0.0000000
 [7,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.1125693 0.0000000 0.000000
0 0.0000000
 [8,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.1125693 0.000000
0 0.0000000
 [9,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.112569
3 0.0000000
[10,] 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.000000
0 0.1125693
>
>  b
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
```

```
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
>  LAMBDA
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 4.076069 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [2,] 0.000000 4.076069 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [3,] 0.000000 0.000000 4.076069 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [4,] 0.000000 0.000000 0.000000 4.076069 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [5,] 0.000000 0.000000 0.000000 0.000000 4.076069 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [6,] 0.000000 0.000000 0.000000 0.000000 0.000000 4.076069 0.000000 0.000000 0.000000 0.00000
0
 [7,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 4.076069 0.000000 0.000000 0.00000
0
 [8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 4.076069 0.000000 0.00000
0
 [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 4.076069 0.00000
0
[10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 4.07606
9
>  muthetax
      [,1]
 [1,]    0
 [2,]    0
 [3,]    0
 [4,]    0
 [5,]    0
 [6,]    0
 [7,]    0
 [8,]    0
 [9,]    0
[10,]    0
>  Bthetax
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]     [,9]    [,10
]
 [1,] 5.024922 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [2,] 0.000000 5.024922 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [3,] 0.000000 0.000000 5.024922 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [4,] 0.000000 0.000000 0.000000 5.024922 0.000000 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [5,] 0.000000 0.000000 0.000000 0.000000 5.024922 0.000000 0.000000 0.000000 0.000000 0.00000
0
 [6,] 0.000000 0.000000 0.000000 0.000000 0.000000 5.024922 0.000000 0.000000 0.000000 0.00000
0
 [7,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 5.024922 0.000000 0.000000 0.00000
0
 [8,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 5.024922 0.000000 0.00000
0
 [9,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 5.024922 0.00000
0
[10,] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 5.02492
2
>
>
> #---------------------------------------------------------------------------------------------
----#
>
> ###define a function that calculate the Loss for a generated ds (ds=C%*%Z)
> L<-function(dS){
+ Sdt<-S0+as.vector(dS)
+         Vdt<-sum(weightcdo*Cdo(Sdt,T,dt,sigma,r,K,H)+weightpcon*Pcon(Sdt,T,dt,sigma,r,K,cash
))
+ return(V0-Vdt)
+ }
```

```
> ###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)
> ECDFb6<-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]  66.45522  80.69768  91.08739  99.68429 107.25029 114.12012 120.50163 126.58171 132.38953
 137.98615 143.47628 148.85744
[13] 154.17724 159.45411 164.67748 169.94793 175.20711 180.48489 185.86814 191.32063 196.90732
 202.61197 208.50716 214.55050
[25] 220.81746 227.32861 234.13893 241.35449 248.95425 257.07661 265.84607 275.39372 285.90648
 297.78415 311.31268 327.43093
[37] 347.60888 375.01458 419.58745
> ECDFb6(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] 2.446494e-01 1.886294e-01 5.822600e-02 1.576925e-02 3.906004e-03 9.985826e-04 2.562586e-0
4 6.804087e-05 1.898965e-05
[10] 5.342248e-06 1.655400e-06 5.284886e-07 1.674825e-07 5.502376e-08 1.678033e-08 5.632341e-0
9 1.858479e-09 6.238552e-10
[19] 2.005053e-10 6.641922e-11 2.207922e-11 6.607769e-12 2.092863e-12 6.462941e-13 1.898675e-1
3 5.079054e-14 1.399196e-14
[28] 3.647095e-15 8.990458e-16 1.999976e-16 4.123437e-17 7.810917e-18 1.281976e-18 1.760511e-1
9 1.865285e-20 1.462344e-21
[37] 7.098301e-23 1.524870e-24 7.003540e-27 8.669064e-31
> #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] 3.473949e-01 3.050396e-01 1.694767e-01 8.819763e-02 4.389527e-02 2.219440e-02 1.124322e-0
2 5.793438e-03 3.060624e-03
[10] 1.623355e-03 9.036554e-04 5.105864e-04 2.874327e-04 1.647504e-04 9.098120e-05 5.271035e-0
5 3.027821e-05 1.754257e-05
[19] 9.945218e-06 5.723983e-06 3.300223e-06 1.805423e-06 1.016066e-06 5.646333e-07 3.060390e-0
7 1.582861e-07 8.307890e-08
[28] 4.241553e-08 2.105922e-08 9.932619e-09 4.510047e-09 1.962919e-09 7.952276e-10 2.946936e-1
0 9.592329e-11 2.685816e-11
[37] 5.917369e-12 8.672970e-13 5.877738e-14 6.539394e-16
>
>
> #####
> #####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 0.01677659 0.01712133 0.01025851
> run(10000,40)
[1] 0.00880000 0.01036914 0.01074471 0.01074417
>
> ###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] 0.7597851
> (ISSQratio<-var(SAMPLES[,1])/var(SAMPLES[,3]))
[1] 0.8003481
> (ISSQOratio<-var(SAMPLES[,1])/var(SAMPLES[,4]))
[1] 8.146823
>
> n<-10000
> strata<-40
> var(SAMPLES[,1])
[1] 1.068985e-06
> (sum(sqrt(dumvar)*(1/strata)))^2/n
[1] 1.267002e-07
> (theoreticalISSQOratio<-var(SAMPLES[,1])/((sum(sqrt(dumvar)*(1/strata)))^2/n))
[1] 8.437121
>
> save.image("C:\\Users\\s1155058334\\Desktop\\portfolio b6  (5.3) os5 pending\\b6os5workspace
")
>
```