```
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]

> #porfolio a15
> 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\RtmpCiQNSl\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\RtmpCiQNSl\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\RtmpCiQNSl\downloaded_packages
> library(Matrix)
Warning message:
package ‘Matrix’ was built under R version 3.5.3
>
>
> #####Consider porfolios on derivatives based on 100 underlying correlated assets
> #####investigate the loss probability, which is critical to estimating VAR
> rm(list=ls())
> set.seed(1515)
> S0<-rep(100,100)
> T<-rep(0.1,100)
> dt<-0.04
```

```
> sigma<-c(rep(0.5,30),rep(0.3,40),rep(0.1,30))
> r<-rep(0.05,100)
> K<-rep(100,100)
> #####correlation between assets within the same group is 0.2, and is 0 between assets if the
y belong to different groups
>
>
> #####define functions for the caluculating the value of a unit of the call and put option(lo
ng 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 100 assets
>   thetac<-(-S0)*dnorm(d1)*sigma/2/sqrt(T)-r*K*exp(-r*T)*pnorm(d2)
>   #####caluculating theta of the European Put options on the 100 assets
>   thetap<-(-S0)*dnorm(d1)*sigma/2/sqrt(T)+r*K*exp(-r*T)*pnorm(-d2)
>   #####caluculating delta of the European Call option on the 100 assets
>   deltac<-pnorm(d1)
>   #####caluculating delta of the European Put option on the 100 assets
>   deltap<-pnorm(d1)-1
>   #####caluculating delta of the European Call option on the 100 assets
>   gammac<-dnorm(d1)/S0/sigma/sqrt(T)
>   #####caluculating delta of the European Put option on the 100 assets
>   gammap<-dnorm(d1)/S0/sigma/sqrt(T)
>
> #####characterize the portfolio (e.g. short 10 ATM alls and short 10 ATM puts on each assets
)
> weightc<-rep(-10,100)
> weightp<-rep(-10,100)
> #####calculating initial value of the portfolio (value at time 0)
> V0<-sum(weightc*Call(S0,T,0,sigma,r,K)+weightp*Put(S0,T,0,sigma,r,K))
> V0
[1] -7560.917
>
>
> ##############################################################################################
##############################################################################################
###
> #####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 correlated, SIGMA is
not diagonal
> #####sigmadum<-S0^2*exp(2*r*dt)*(exp(sigma^2*dt)-1)
> #####SIGMA<-diag(sigmadum,10)
>
> #####correlation between assets within the same group is 0.2, and is 0 between assets if the
y belong to different groups
> G1<-matrix(100*100*exp(2*0.05*0.04)*(exp(0.2*0.5^2*0.04)-1),10,10)
> diag(G1)<-rep(100*100*exp(2*0.05*0.04)*(exp(0.5^2*0.04)-1),10)
> G2<-matrix(100*100*exp(2*0.05*0.04)*(exp(0.2*0.3^2*0.04)-1),10,10)
> diag(G2)<-rep(100*100*exp(2*0.05*0.04)*(exp(0.3^2*0.04)-1),10)
> G3<-matrix(100*100*exp(2*0.05*0.04)*(exp(0.2*0.1^2*0.04)-1),10,10)
> diag(G3)<-rep(100*100*exp(2*0.05*0.04)*(exp(0.1^2*0.04)-1),10)
>
> SIGMA<-as.matrix(bdiag(list(G1,G1,G1,G2,G2,G2,G2,G3,G3,G3)))
> #sum(SIGMA[1:10,1:10]==SIGMA[11:20,11:20])
> #sum(SIGMA[11:20,11:20]==SIGMA[21:30,21:30])
> #sum(SIGMA[21:30,21:30]==SIGMA[31:40,31:40])
> #sum(SIGMA[31:40,31:40]==SIGMA[41:50,41:50])
> #sum(SIGMA[41:50,41:50]==SIGMA[51:60,51:60])
```

```
> #sum(SIGMA[51:60,51:60]==SIGMA[61:70,61:70])
> #sum(SIGMA[61:70,61:70]==SIGMA[71:80,71:80])
> #sum(SIGMA[71:80,71:80]==SIGMA[81:90,81:90])
> #sum(SIGMA[81:90,81:90]==SIGMA[91:100,91:100])
> ##############################################################################################
##############################################################################################
###
>
> #####consider Delta-GAMMA approximation of the portfolio loss
> #####caluculating theta of the portfolio consisting of mix of the options on the 100 assets
> THETA<-sum(weightc*thetac+weightp*thetap)
> #####caluculating delta of the portfolio(by assets) consisting of mix of the options on the
10 assets
> delta<-matrix(weightc*deltac+weightp*deltap,100,1)
> #####caluculating gamma of the portfolio(by pairs of assets) consisting of mix of the option
s on the 10 assets, given the assets are uncorrelated, GAMMA is diagonal
> GAMMA<-diag(weightc*gammac+weightp*gammap,100)
> #####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,100)
> 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=2.65 under the origin
al distribution of Z
> vecb<-as.vector(b)
> veclambda<-diag(LAMBDA)
> xstd<-2.65
> 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])
+
+ +thetax*vecb[11]^2*(1-thetax*veclambda[11])/(1-2*thetax*veclambda[11])^2 + veclambda[11]/(1-
2*thetax*veclambda[11])
```

```
+ +thetax*vecb[12]^2*(1-thetax*veclambda[12])/(1-2*thetax*veclambda[12])^2 + veclambda[12]/(1-
2*thetax*veclambda[12])
+ +thetax*vecb[13]^2*(1-thetax*veclambda[13])/(1-2*thetax*veclambda[13])^2 + veclambda[13]/(1-
2*thetax*veclambda[13])
+ +thetax*vecb[14]^2*(1-thetax*veclambda[14])/(1-2*thetax*veclambda[14])^2 + veclambda[14]/(1-
2*thetax*veclambda[14])
+ +thetax*vecb[15]^2*(1-thetax*veclambda[15])/(1-2*thetax*veclambda[15])^2 + veclambda[15]/(1-
2*thetax*veclambda[15])
+ +thetax*vecb[16]^2*(1-thetax*veclambda[16])/(1-2*thetax*veclambda[16])^2 + veclambda[16]/(1-
2*thetax*veclambda[16])
+ +thetax*vecb[17]^2*(1-thetax*veclambda[17])/(1-2*thetax*veclambda[17])^2 + veclambda[17]/(1-
2*thetax*veclambda[17])
+ +thetax*vecb[18]^2*(1-thetax*veclambda[18])/(1-2*thetax*veclambda[18])^2 + veclambda[18]/(1-
2*thetax*veclambda[18])
+ +thetax*vecb[19]^2*(1-thetax*veclambda[19])/(1-2*thetax*veclambda[19])^2 + veclambda[19]/(1-
2*thetax*veclambda[19])
+ +thetax*vecb[20]^2*(1-thetax*veclambda[20])/(1-2*thetax*veclambda[20])^2 + veclambda[20]/(1-
2*thetax*veclambda[20])
+
+ +thetax*vecb[21]^2*(1-thetax*veclambda[21])/(1-2*thetax*veclambda[21])^2 + veclambda[21]/(1-
2*thetax*veclambda[21])
+ +thetax*vecb[22]^2*(1-thetax*veclambda[22])/(1-2*thetax*veclambda[22])^2 + veclambda[22]/(1-
2*thetax*veclambda[22])
+ +thetax*vecb[23]^2*(1-thetax*veclambda[23])/(1-2*thetax*veclambda[23])^2 + veclambda[23]/(1-
2*thetax*veclambda[23])
+ +thetax*vecb[24]^2*(1-thetax*veclambda[24])/(1-2*thetax*veclambda[24])^2 + veclambda[24]/(1-
2*thetax*veclambda[24])
+ +thetax*vecb[25]^2*(1-thetax*veclambda[25])/(1-2*thetax*veclambda[25])^2 + veclambda[25]/(1-
2*thetax*veclambda[25])
+ +thetax*vecb[26]^2*(1-thetax*veclambda[26])/(1-2*thetax*veclambda[26])^2 + veclambda[26]/(1-
2*thetax*veclambda[26])
+ +thetax*vecb[27]^2*(1-thetax*veclambda[27])/(1-2*thetax*veclambda[27])^2 + veclambda[27]/(1-
2*thetax*veclambda[27])
+ +thetax*vecb[28]^2*(1-thetax*veclambda[28])/(1-2*thetax*veclambda[28])^2 + veclambda[28]/(1-
2*thetax*veclambda[28])
+ +thetax*vecb[29]^2*(1-thetax*veclambda[29])/(1-2*thetax*veclambda[29])^2 + veclambda[29]/(1-
2*thetax*veclambda[29])
+ +thetax*vecb[30]^2*(1-thetax*veclambda[30])/(1-2*thetax*veclambda[30])^2 + veclambda[30]/(1-
2*thetax*veclambda[30])
+
+ +thetax*vecb[31]^2*(1-thetax*veclambda[31])/(1-2*thetax*veclambda[31])^2 + veclambda[31]/(1-
2*thetax*veclambda[31])
+ +thetax*vecb[32]^2*(1-thetax*veclambda[32])/(1-2*thetax*veclambda[32])^2 + veclambda[32]/(1-
2*thetax*veclambda[32])
+ +thetax*vecb[33]^2*(1-thetax*veclambda[33])/(1-2*thetax*veclambda[33])^2 + veclambda[33]/(1-
2*thetax*veclambda[33])
+ +thetax*vecb[34]^2*(1-thetax*veclambda[34])/(1-2*thetax*veclambda[34])^2 + veclambda[34]/(1-
2*thetax*veclambda[34])
+ +thetax*vecb[35]^2*(1-thetax*veclambda[35])/(1-2*thetax*veclambda[35])^2 + veclambda[35]/(1-
2*thetax*veclambda[35])
+ +thetax*vecb[36]^2*(1-thetax*veclambda[36])/(1-2*thetax*veclambda[36])^2 + veclambda[36]/(1-
2*thetax*veclambda[36])
+ +thetax*vecb[37]^2*(1-thetax*veclambda[37])/(1-2*thetax*veclambda[37])^2 + veclambda[37]/(1-
2*thetax*veclambda[37])
+ +thetax*vecb[38]^2*(1-thetax*veclambda[38])/(1-2*thetax*veclambda[38])^2 + veclambda[38]/(1-
2*thetax*veclambda[38])
+ +thetax*vecb[39]^2*(1-thetax*veclambda[39])/(1-2*thetax*veclambda[39])^2 + veclambda[39]/(1-
2*thetax*veclambda[39])
+ +thetax*vecb[40]^2*(1-thetax*veclambda[40])/(1-2*thetax*veclambda[40])^2 + veclambda[40]/(1-
2*thetax*veclambda[40])
+
+ +thetax*vecb[41]^2*(1-thetax*veclambda[41])/(1-2*thetax*veclambda[41])^2 + veclambda[41]/(1-
2*thetax*veclambda[41])
+ +thetax*vecb[42]^2*(1-thetax*veclambda[42])/(1-2*thetax*veclambda[42])^2 + veclambda[42]/(1-
2*thetax*veclambda[42])
+ +thetax*vecb[43]^2*(1-thetax*veclambda[43])/(1-2*thetax*veclambda[43])^2 + veclambda[43]/(1-
2*thetax*veclambda[43])
+ +thetax*vecb[44]^2*(1-thetax*veclambda[44])/(1-2*thetax*veclambda[44])^2 + veclambda[44]/(1-
2*thetax*veclambda[44])
+ +thetax*vecb[45]^2*(1-thetax*veclambda[45])/(1-2*thetax*veclambda[45])^2 + veclambda[45]/(1-
2*thetax*veclambda[45])
+ +thetax*vecb[46]^2*(1-thetax*veclambda[46])/(1-2*thetax*veclambda[46])^2 + veclambda[46]/(1-
2*thetax*veclambda[46])
+ +thetax*vecb[47]^2*(1-thetax*veclambda[47])/(1-2*thetax*veclambda[47])^2 + veclambda[47]/(1-
```

```
2*thetax*veclambda[47])
+ +thetax*vecb[48]^2*(1-thetax*veclambda[48])/(1-2*thetax*veclambda[48])^2 + veclambda[48]/(1-
2*thetax*veclambda[48])
+ +thetax*vecb[49]^2*(1-thetax*veclambda[49])/(1-2*thetax*veclambda[49])^2 + veclambda[49]/(1-
2*thetax*veclambda[49])
+ +thetax*vecb[50]^2*(1-thetax*veclambda[50])/(1-2*thetax*veclambda[50])^2 + veclambda[50]/(1-
2*thetax*veclambda[50])
+
+ +thetax*vecb[51]^2*(1-thetax*veclambda[51])/(1-2*thetax*veclambda[51])^2 + veclambda[51]/(1-
2*thetax*veclambda[51])
+ +thetax*vecb[52]^2*(1-thetax*veclambda[52])/(1-2*thetax*veclambda[52])^2 + veclambda[52]/(1-
2*thetax*veclambda[52])
+ +thetax*vecb[53]^2*(1-thetax*veclambda[53])/(1-2*thetax*veclambda[53])^2 + veclambda[53]/(1-
2*thetax*veclambda[53])
+ +thetax*vecb[54]^2*(1-thetax*veclambda[54])/(1-2*thetax*veclambda[54])^2 + veclambda[54]/(1-
2*thetax*veclambda[54])
+ +thetax*vecb[55]^2*(1-thetax*veclambda[55])/(1-2*thetax*veclambda[55])^2 + veclambda[55]/(1-
2*thetax*veclambda[55])
+ +thetax*vecb[56]^2*(1-thetax*veclambda[56])/(1-2*thetax*veclambda[56])^2 + veclambda[56]/(1-
2*thetax*veclambda[56])
+ +thetax*vecb[57]^2*(1-thetax*veclambda[57])/(1-2*thetax*veclambda[57])^2 + veclambda[57]/(1-
2*thetax*veclambda[57])
+ +thetax*vecb[58]^2*(1-thetax*veclambda[58])/(1-2*thetax*veclambda[58])^2 + veclambda[58]/(1-
2*thetax*veclambda[58])
+ +thetax*vecb[59]^2*(1-thetax*veclambda[59])/(1-2*thetax*veclambda[59])^2 + veclambda[59]/(1-
2*thetax*veclambda[59])
+ +thetax*vecb[60]^2*(1-thetax*veclambda[60])/(1-2*thetax*veclambda[60])^2 + veclambda[60]/(1-
2*thetax*veclambda[60])
+
+ +thetax*vecb[61]^2*(1-thetax*veclambda[61])/(1-2*thetax*veclambda[61])^2 + veclambda[61]/(1-
2*thetax*veclambda[61])
+ +thetax*vecb[62]^2*(1-thetax*veclambda[62])/(1-2*thetax*veclambda[62])^2 + veclambda[62]/(1-
2*thetax*veclambda[62])
+ +thetax*vecb[63]^2*(1-thetax*veclambda[63])/(1-2*thetax*veclambda[63])^2 + veclambda[63]/(1-
2*thetax*veclambda[63])
+ +thetax*vecb[64]^2*(1-thetax*veclambda[64])/(1-2*thetax*veclambda[64])^2 + veclambda[64]/(1-
2*thetax*veclambda[64])
+ +thetax*vecb[65]^2*(1-thetax*veclambda[65])/(1-2*thetax*veclambda[65])^2 + veclambda[65]/(1-
2*thetax*veclambda[65])
+ +thetax*vecb[66]^2*(1-thetax*veclambda[66])/(1-2*thetax*veclambda[66])^2 + veclambda[66]/(1-
2*thetax*veclambda[66])
+ +thetax*vecb[67]^2*(1-thetax*veclambda[67])/(1-2*thetax*veclambda[67])^2 + veclambda[67]/(1-
2*thetax*veclambda[67])
+ +thetax*vecb[68]^2*(1-thetax*veclambda[68])/(1-2*thetax*veclambda[68])^2 + veclambda[68]/(1-
2*thetax*veclambda[68])
+ +thetax*vecb[69]^2*(1-thetax*veclambda[69])/(1-2*thetax*veclambda[69])^2 + veclambda[69]/(1-
2*thetax*veclambda[69])
+ +thetax*vecb[70]^2*(1-thetax*veclambda[70])/(1-2*thetax*veclambda[70])^2 + veclambda[70]/(1-
2*thetax*veclambda[70])
+
+ +thetax*vecb[71]^2*(1-thetax*veclambda[71])/(1-2*thetax*veclambda[71])^2 + veclambda[71]/(1-
2*thetax*veclambda[71])
+ +thetax*vecb[72]^2*(1-thetax*veclambda[72])/(1-2*thetax*veclambda[72])^2 + veclambda[72]/(1-
2*thetax*veclambda[72])
+ +thetax*vecb[73]^2*(1-thetax*veclambda[73])/(1-2*thetax*veclambda[73])^2 + veclambda[73]/(1-
2*thetax*veclambda[73])
+ +thetax*vecb[74]^2*(1-thetax*veclambda[74])/(1-2*thetax*veclambda[74])^2 + veclambda[74]/(1-
2*thetax*veclambda[74])
+ +thetax*vecb[75]^2*(1-thetax*veclambda[75])/(1-2*thetax*veclambda[75])^2 + veclambda[75]/(1-
2*thetax*veclambda[75])
+ +thetax*vecb[76]^2*(1-thetax*veclambda[76])/(1-2*thetax*veclambda[76])^2 + veclambda[76]/(1-
2*thetax*veclambda[76])
+ +thetax*vecb[77]^2*(1-thetax*veclambda[77])/(1-2*thetax*veclambda[77])^2 + veclambda[77]/(1-
2*thetax*veclambda[77])
+ +thetax*vecb[78]^2*(1-thetax*veclambda[78])/(1-2*thetax*veclambda[78])^2 + veclambda[78]/(1-
2*thetax*veclambda[78])
+ +thetax*vecb[79]^2*(1-thetax*veclambda[79])/(1-2*thetax*veclambda[79])^2 + veclambda[79]/(1-
2*thetax*veclambda[79])
+ +thetax*vecb[80]^2*(1-thetax*veclambda[80])/(1-2*thetax*veclambda[80])^2 + veclambda[80]/(1-
2*thetax*veclambda[80])
+
+ +thetax*vecb[81]^2*(1-thetax*veclambda[81])/(1-2*thetax*veclambda[81])^2 + veclambda[81]/(1-
2*thetax*veclambda[81])
+ +thetax*vecb[82]^2*(1-thetax*veclambda[82])/(1-2*thetax*veclambda[82])^2 + veclambda[82]/(1-
```

```
2*thetax*veclambda[82])
+ +thetax*vecb[83]^2*(1-thetax*veclambda[83])/(1-2*thetax*veclambda[83])^2 + veclambda[83]/(1-
2*thetax*veclambda[83])
+ +thetax*vecb[84]^2*(1-thetax*veclambda[84])/(1-2*thetax*veclambda[84])^2 + veclambda[84]/(1-
2*thetax*veclambda[84])
+ +thetax*vecb[85]^2*(1-thetax*veclambda[85])/(1-2*thetax*veclambda[85])^2 + veclambda[85]/(1-
2*thetax*veclambda[85])
+ +thetax*vecb[86]^2*(1-thetax*veclambda[86])/(1-2*thetax*veclambda[86])^2 + veclambda[86]/(1-
2*thetax*veclambda[86])
+ +thetax*vecb[87]^2*(1-thetax*veclambda[87])/(1-2*thetax*veclambda[87])^2 + veclambda[87]/(1-
2*thetax*veclambda[87])
+ +thetax*vecb[88]^2*(1-thetax*veclambda[88])/(1-2*thetax*veclambda[88])^2 + veclambda[88]/(1-
2*thetax*veclambda[88])
+ +thetax*vecb[89]^2*(1-thetax*veclambda[89])/(1-2*thetax*veclambda[89])^2 + veclambda[89]/(1-
2*thetax*veclambda[89])
+ +thetax*vecb[90]^2*(1-thetax*veclambda[90])/(1-2*thetax*veclambda[90])^2 + veclambda[90]/(1-
2*thetax*veclambda[90])
+
+ +thetax*vecb[91]^2*(1-thetax*veclambda[91])/(1-2*thetax*veclambda[91])^2 + veclambda[91]/(1-
2*thetax*veclambda[91])
+ +thetax*vecb[92]^2*(1-thetax*veclambda[92])/(1-2*thetax*veclambda[92])^2 + veclambda[92]/(1-
2*thetax*veclambda[92])
+ +thetax*vecb[93]^2*(1-thetax*veclambda[93])/(1-2*thetax*veclambda[93])^2 + veclambda[93]/(1-
2*thetax*veclambda[93])
+ +thetax*vecb[94]^2*(1-thetax*veclambda[94])/(1-2*thetax*veclambda[94])^2 + veclambda[94]/(1-
2*thetax*veclambda[94])
+ +thetax*vecb[95]^2*(1-thetax*veclambda[95])/(1-2*thetax*veclambda[95])^2 + veclambda[95]/(1-
2*thetax*veclambda[95])
+ +thetax*vecb[96]^2*(1-thetax*veclambda[96])/(1-2*thetax*veclambda[96])^2 + veclambda[96]/(1-
2*thetax*veclambda[96])
+ +thetax*vecb[97]^2*(1-thetax*veclambda[97])/(1-2*thetax*veclambda[97])^2 + veclambda[97]/(1-
2*thetax*veclambda[97])
+ +thetax*vecb[98]^2*(1-thetax*veclambda[98])/(1-2*thetax*veclambda[98])^2 + veclambda[98]/(1-
2*thetax*veclambda[98])
+ +thetax*vecb[99]^2*(1-thetax*veclambda[99])/(1-2*thetax*veclambda[99])^2 + veclambda[99]/(1-
2*thetax*veclambda[99])
+ +thetax*vecb[100]^2*(1-thetax*veclambda[100])/(1-2*thetax*veclambda[100])^2 + veclambda[100]
/(1-2*thetax*veclambda[100]))-(x-a0)
+ }
> curve(psipithetax)
> abline(h=0,v=0)
> uni<-uniroot.all(psipithetax,c(0,0.05))
> uni
[1] 0.004369673 0.007814166 0.009300676 0.012627193 0.017154521 0.024759461
[7] 0.032615084 0.041156823
>
> ###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] 0.004369673
> psipithetax(thetax)
[1] -6.628254
> 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] 0.004385429
> psipithetax(thetax)
[1] 9.180212e-08
>
> ###identify the IS distribution
> Bthetax<-solve(diag(100)-2*thetax*LAMBDA)
> muthetax<-thetax*Bthetax%*%b
>
> ###generate 5000000 samples of Q under IS change of measure,
> Qsamples<-rep(0,5000000)
```

```
> for(j in 1:5000000){
+ Z<-muthetax+chol(Bthetax)%*%matrix(rnorm(100),100,1)
+ Qsamples[j]<-Q(Z)
+ }
> ###check whether E[Q] approximately equals (x-a0) under the importance sampling change of me
asure
> mean(Qsamples)
[1] 2304.683
> x-a0
[1] 2305.08
> a0
[1] -1508.788
> x
[1] 796.2921
> thetax
[1] 0.004385429
>
> ###display the parameters
> #SIGMA
> #THETA
> #a0
> #delta
> #a
> #GAMMA
> #A
>
> #b
> #LAMBDA
> #muthetax
> #Bthetax
>
>
> ##
> #-----------------------------------------------------------------------------------------
----#
> ##
>
> ###define a function that calculate the Loss
> L<-function(dS){
+ Sdt<-S0+as.vector(dS)
+ Vdt<-sum(weightc*Call(Sdt,T,dt,sigma,r,K)+weightp*Put(Sdt,T,dt,sigma,r,K))
+ return(V0-Vdt)
+ }
> ###define a function that calculate the likelihood ratio
> 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)
> ECDFa15<-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] 1357.500 1459.228 1530.964 1589.006 1639.333 1685.131 1727.443 1767.297 1805.537 1842.396
 1878.413 1913.732 1948.500
[14] 1982.886 2017.197 2051.727 2086.533 2121.771 2157.571 2194.054 2231.347 2269.805 2309.431
 2350.694 2393.529 2438.508
[27] 2485.666 2535.741 2589.394 2646.778 2709.090 2777.975 2854.816 2941.582 3042.131 3163.239
 3316.715 3527.792 3879.118
> ECDFa15(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$double.xmax)
> vars<-matrix(0,40,10000)
> counts<-rep(0,40)
> while(sum(counts)!=400000){
+ Z<-muthetax+chol(Bthetax)%*%matrix(rnorm(100),100,1)
+ k<-tail(binsearch(function(y) bins[y]-(Q(Z)), range=c(1, length(bins)))$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 0.000000e+00 0.000000e+00 2.778878e-0
5 1.071040e-04 2.524403e-04
[19] 6.171770e-04 1.098700e-03 1.486410e-03 1.973416e-03 2.189296e-03 2.107863e-03 1.761202e-0
3 1.180003e-03 5.879861e-04
[28] 1.994633e-04 5.070671e-05 1.183757e-05 4.304907e-06 2.221730e-06 1.409456e-06 8.685069e-0
7 5.195781e-07 2.875883e-07
[37] 1.411966e-07 5.507155e-08 1.475309e-08 1.159745e-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] 0.000000e+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 0.000000e+00 0.000000e+00 1.324048e-0
2 2.599392e-02 3.990693e-02
[19] 6.239845e-02 8.325467e-02 9.683632e-02 1.115779e-01 1.175226e-01 1.153162e-01 1.054080e-0
1 8.628008e-02 6.090493e-02
[28] 3.547321e-02 1.788553e-02 8.641722e-03 5.211357e-03 3.743817e-03 2.981911e-03 2.340753e-0
3 1.810483e-03 1.346960e-03
[37] 9.438025e-04 5.894305e-04 3.050777e-04 8.553627e-05
>
>
> #####
> #####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(100),100,1)
+ #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)))$where,1)
+ #Monte Carlo (IS and Stratification)
+ if(binscount[kthbins]<(n/strata)){
+ binscount[kthbins]<-binscount[kthbins]+1
+ ISSQ<-ISSQ+(ifelse(L2>x,1,0)*likelihood(Z2)*(1/n))
+ }
+ else{
+ }
+ #Monte Carlo (IS and Stratification with optimized smaple size for each strata)
+ if(binscountpi[kthbins]<nj[kthbins]){
+ binscountpi[kthbins]<-binscountpi[kthbins]+1
+ ISSQO<-ISSQO+(ifelse(L2>x,1,0)*likelihood(Z2)*(1/nj[kthbins])*(1/strata))
+ }
+ else{
+ }
+ }
+ results[1]<-SMC
+ results[2]<-IS
+
+
+ while(sum(binscount)<n){
+ Z2<-muthetax+chol(Bthetax)%*%matrix(rnorm(100),100,1)
+ kthbins<-tail(binsearch(function(y) bins[y]-(Q(Z2)), range=c(1, length(bins)))$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(100),100,1)
+ 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] 0.005000000 0.009824072 0.009497076 0.009343650
> run(10000,40)
[1] 0.008300000 0.009446927 0.009450072 0.009522265
>
> ###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] 18.32693
> (ISSQratio<-var(SAMPLES[,1])/var(SAMPLES[,3]))
[1] 28.63735
> (ISSQOratio<-var(SAMPLES[,1])/var(SAMPLES[,4]))
[1] 98.13017
>
> n<-10000
> strata<-40
> var(SAMPLES[,1])
[1] 9.598731e-07
> (sum(sqrt(dumvar)*(1/strata)))^2/n
[1] 9.906998e-09
> (theoreticalISSQOratio<-var(SAMPLES[,1])/((sum(sqrt(dumvar)*(1/strata)))^2/n))
[1] 96.88839
>
> save.image("C:\\Users\\s1155058334\\Desktop\\portfolio a15  (5.3) os5 pending (1,1,40)\\a15o
s5workspace")
>
```