# Script to get Risk Margin for CSR model with four independent lines

# by Glenn Meyers

#

rm(list = ls()) # clear workspace")

t1=Sys.time()

#

# user inputs

#

grpcode="1538" #1538 1767 3240 5185 14176

line1.data="~/Dropbox/CAS Loss Reserve Database/comauto\_pos.csv"

line2.data="~/Dropbox/CAS Loss Reserve Database/ppauto\_pos.csv"

line3.data="~/Dropbox/CAS Loss Reserve Database/wkcomp\_pos.csv"

line4.data="~/Dropbox/CAS Loss Reserve Database/othliab\_pos.csv"

outfile=paste("Div-",grpcode,".csv")

#

# user inputs

#

fixed.rate=0.04

risky.rate=0.10

TVaR.Range=9701:10000

#

# user technical inputs

#

setwd("~/Dropbox/Risk Margin/Risk Margin - Latex") # Mac

library(rstan)

rstan\_options(auto\_write = TRUE)

options(mc.cores = parallel::detectCores())

library(parallel)

library(doParallel)

#

# Stan script for univariate models

#

scodeU = "

data{

real logprem[10];

real logloss[55];

int<lower=1,upper=10> w[55];

int<lower=1,upper=10> d[55];

}

parameters{

real r\_alpha[9];

real r\_beta[9];

real<lower=-1.5,upper=0.5> logelr;

real<lower=0,upper=1> a[10];

real gamma;

real delta;

}

transformed parameters{

real alpha[10];

real beta[10];

real speedup[10];

real sig2[10];

real sig[10];

real mu[55];

alpha[1]=0;

for (i in 2:10) alpha[i] = r\_alpha[i-1];

for (i in 1:9) beta[i] = 10\*r\_beta[i]-5;

beta[10] = 0;

speedup[1] = 1;

for (i in 2:10) speedup[i] = speedup[i-1]\*(1-gamma-(i-2)\*delta);

sig2[10] = a[10];

for (i in 1:9) sig2[10-i] = sig2[11-i]+a[i];

for (i in 1:10) sig[i] = sqrt(sig2[i]);

for (i in 1:55){

mu[i] = logprem[w[i]]+logelr+alpha[w[i]]+beta[d[i]]\*speedup[w[i]];

}

}

model {

for (i in 1:9) r\_alpha[i] ~ normal(0,3.162);

for (i in 1:9) r\_beta[i] ~ uniform(0,1);

for (i in 1:10) a[i] ~ uniform(0,1);

logelr ~ uniform(-1.5,0.5);

gamma ~ normal(0,0.05);

delta ~ normal(0,0.01);

for (i in 1:55) logloss[i] ~ normal(mu[i],sig[d[i]]);

}

"

#

# compile stan model

#

dummydata.uni=list(logprem = rep(8,10),

logloss = rep(8,55),

w = rep(1,55),

d = rep(1,55))

fitU = stan(model\_code=scodeU,chains=0,data=dummydata.uni)

#

# initialization function for scodeU

#

initU=function(chain\_id){

set.seed(12345+chain\_id)

list(r\_alpha=rnorm(9,0,0.2),r\_beta=runif(9),a=runif(10),

logelr=runif(1,-0.75,-0.5),gamma=rnorm(1,0,0.1),

delta=rnorm(1,0,0.02))

}

pars.list=c("alpha","beta","gamma","delta","logelr","sig")

#

# function for Schedule P triangle data given ins group and line of business

#

ins.line.data=function(g.code){

b=subset(a,a$GRCODE==g.code)

name=b$GRNAME

grpcode=b$GRCODE

w=b$AccidentYear

d=b$DevelopmentLag

cum\_incloss=b[,6]

cum\_pdloss=b[,7]

bulk\_loss=b[,8]

dir\_premium=b[,9]

ced\_premium=b[,10]

net\_premium=b[,11]

single=b[,12]

posted\_reserve97=b[,13]

# get incremental paid losses - assume data is sorted by ay and lag

inc\_pdloss=numeric(0)

for (i in unique(w)){

s=(w==i)

pl=c(0,cum\_pdloss[s])

ndev=length(pl)-1

il=rep(0,ndev)

for (j in 1:ndev){

il[j]=pl[j+1]-pl[j]

}

inc\_pdloss=c(inc\_pdloss,il)

}

data.out=data.frame(grpcode,w,d,net\_premium,dir\_premium,ced\_premium,

cum\_pdloss,cum\_incloss,bulk\_loss,inc\_pdloss,single,posted\_reserve97)

return(data.out)

}

#

# initialization function for scodeU

#

initU=function(chain\_id){

set.seed(12345+chain\_id)

list(r\_alpha=rnorm(9,0,0.2),r\_beta=runif(9),a=runif(10),

logelr=runif(1,-0.75,-0.5),gamma=rnorm(1,0,0.1),

delta=rnorm(1,0,0.02))

}

#

# other initializations

pars.list=c("alpha","beta","gamma","delta","logelr","sig")

#

# univariate line 1

#

a=read.csv(line1.data)

cdata=ins.line.data(grpcode)

set.seed(12345)

w=cdata$w-1987

d=cdata$d

#

# sort the data in order of d, then w within d

#

o1=100\*d+w

o=order(o1)

w=w[o]

d=d[o]

premium=cdata$net\_premium[o]

cpdloss=cdata$cum\_pdloss[o]

cpdloss=pmax(cpdloss,1)

adata=data.frame(grpcode,w,d,premium,cpdloss)

rdata1=subset(adata,(adata$w+adata$d)<12)

rloss1=rdata1$cpdloss

aloss1=adata$cpdloss

Premium1=subset(rdata1,rdata1$d==1)$premium

#

# data for univariate 1

#

data.u1=list(logprem = log(rdata1$premium[1:10]),

logloss = log(rloss1),

w = rdata1$w,

d = rdata1$d)

#

# run the univariate model for line 1

#

stan\_thin=1

stan\_iter=5000

Rhat\_target=1.05

max\_Rhat=2

while ((max\_Rhat > Rhat\_target)&(stan\_thin<65)){

sflist <-

mclapply(1:4, mc.cores = 4,

function(i) stan(fit = fitU, data = data.u1,init=initU,

seed = 12345,iter=stan\_iter,thin=stan\_thin,

chains = 1, chain\_id = i, cores=4,pars=pars.list))

fitU1=sflist2stanfit(sflist)

fitU1\_summary=as.matrix(summary(fitU1)$summary)[,c(1,3,10)]

mrh=subset(fitU1\_summary,is.na(fitU1\_summary[,3])==F)

max\_Rhat=round(max(mrh[,3]),4)

mean\_lp\_\_=round(fitU1\_summary[dim(fitU1\_summary)[1],1],2)

print(paste("Maximum Rhat =",max\_Rhat," Mean lp\_\_ =",mean\_lp\_\_,

"Thin =",stan\_thin))

stan\_thin=2\*stan\_thin

stan\_iter=2\*stan\_iter

}

stan\_thin1=stan\_thin/2

max\_Rhat1=max\_Rhat

#

# extract information from stan output to process in R

#

b=extract(fitU1)

alpha1=b$alpha

beta1=b$beta

gamma1=b$gamma

delta1=b$delta

logelr1=b$logelr

sigma1=b$sig

num.mcmc=length(logelr1)

#

# get speedup rates by accident year

#

speedup1=matrix(1,num.mcmc,10)

for (i in 2:10){

speedup1[,i]=speedup1[,i-1]\*(1-gamma1-(i-2)\*delta1)

}

#

# simulate outcomes for d=10 using parallel processing

#

cl <- makeCluster(4)

registerDoParallel(cl)

at.wd10=foreach (k=1:length(gamma1),.combine=rbind) %dopar%{

set.seed(12345)

atv=rep(0,10)

for (w in 1:10){

atv[w]=rlnorm(1,log(premium[w])+logelr1[k]+alpha1[k,w],sigma1[k,10])

}

at=atv

}

stopCluster(cl)

#

# calculate loss statistics and output to data frame

#

Premium1=subset(rdata1,rdata1$d==1)$premium

ss.wd10=rep(0,10)

ms.wd10=rep(0,10)

#

ms.wd10[1]=mean(at.wd10[,1])

for (w in 2:10){

ms.wd10[w]=mean(at.wd10[,w])

ss.wd10[w]=sd(at.wd10[,w])

}

Pred.CSR=rowSums(at.wd10)

ms.td10=mean(Pred.CSR)

ss.td10=sd(Pred.CSR)

CSR.Estimate1=round(ms.td10)

CSR.SE1=round(ss.td10)

#

# repeat the above for line 2

#

a=read.csv(line2.data)

cdata=ins.line.data(grpcode)

set.seed(12345)

w=cdata$w-1987

d=cdata$d

#

# sort the data in order of d, then w within d

#

o1=100\*d+w

o=order(o1)

w=w[o]

d=d[o]

premium=cdata$net\_premium[o]

cpdloss=cdata$cum\_pdloss[o]

cpdloss=pmax(cpdloss,1)

adata=data.frame(grpcode,w,d,premium,cpdloss)

rdata2=subset(adata,(adata$w+adata$d)<12)

rloss2=rdata2$cpdloss

aloss2=adata$cpdloss

Premium2=subset(rdata2,rdata2$d==1)$premium

#

# data for univariate 2

#

data.u1=list(logprem = log(rdata2$premium[1:10]),

logloss = log(rloss2),

w = rdata2$w,

d = rdata2$d)

#

# run the univariate model for line 2

#

set.seed(12345)

stan\_thin=1

stan\_iter=5000

Rhat\_target=1.05

max\_Rhat=2

# NOTE - I changed the seed to 54321 when testing the duration of identical lines

while ((max\_Rhat > Rhat\_target)&(stan\_thin<65)){

sflist <-

mclapply(1:4, mc.cores = 4,

function(i) stan(fit = fitU, data = data.u1,init=initU,

seed = 12345,iter=stan\_iter,thin=stan\_thin,

chains = 1, chain\_id = i, cores=4,pars=pars.list))

fitU1=sflist2stanfit(sflist)

fitU1\_summary=as.matrix(summary(fitU1)$summary)[,c(1,3,10)]

mrh=subset(fitU1\_summary,is.na(fitU1\_summary[,3])==F)

max\_Rhat=round(max(mrh[,3]),4)

mean\_lp\_\_=round(fitU1\_summary[dim(fitU1\_summary)[1],1],2)

print(paste("Maximum Rhat =",max\_Rhat," Mean lp\_\_ =",mean\_lp\_\_,

"Thin =",stan\_thin))

stan\_thin=2\*stan\_thin

stan\_iter=2\*stan\_iter

}

stan\_thin2=stan\_thin/2

max\_Rhat2=max\_Rhat

#

# extract information from stan output to process in R

#

b=extract(fitU1)

alpha2=b$alpha

beta2=b$beta

gamma2=b$gamma

delta2=b$delta

logelr2=b$logelr

sigma2=b$sig

num.mcmc=length(logelr2)

#

# get speedup rates by accident year

#

speedup2=matrix(1,num.mcmc,10)

for (i in 2:10){

speedup2[,i]=speedup2[,i-1]\*(1-gamma2-(i-2)\*delta2)

}

#

# simulate outcomes for d=10 using parallel processing

#

cl <- makeCluster(4)

registerDoParallel(cl)

at.wd10=foreach (k=1:length(gamma2),.combine=rbind) %dopar%{

set.seed(12345)

atv=rep(0,10)

for (w in 1:10){

atv[w]=rlnorm(1,log(premium[w])+logelr2[k]+alpha2[k,w],sigma2[k,10])

}

at=atv

}

stopCluster(cl)

#

# calculate loss statistics and output to data frame

#

Premium2=subset(rdata2,rdata2$d==1)$premium

ss.wd10=rep(0,10)

ms.wd10=rep(0,10)

#

ms.wd10[1]=mean(at.wd10[,1])

for (w in 2:10){

ms.wd10[w]=mean(at.wd10[,w])

ss.wd10[w]=sd(at.wd10[,w])

}

Pred.CSR=rowSums(at.wd10)

ms.td10=mean(Pred.CSR)

ss.td10=sd(Pred.CSR)

CSR.Estimate2=round(ms.td10)

CSR.SE2=round(ss.td10)

#

# repeat the above for line 3

#

a=read.csv(line3.data)

cdata=ins.line.data(grpcode)

set.seed(12345)

w=cdata$w-1987

d=cdata$d

#

# sort the data in order of d, then w within d

#

o1=100\*d+w

o=order(o1)

w=w[o]

d=d[o]

premium=cdata$net\_premium[o]

cpdloss=cdata$cum\_pdloss[o]

cpdloss=pmax(cpdloss,1)

adata=data.frame(grpcode,w,d,premium,cpdloss)

rdata3=subset(adata,(adata$w+adata$d)<12)

rloss3=rdata3$cpdloss

aloss3=adata$cpdloss

Premium3=subset(rdata3,rdata3$d==1)$premium

#

# data for univariate 3

#

data.u1=list(logprem = log(rdata3$premium[1:10]),

logloss = log(rloss3),

w = rdata3$w,

d = rdata3$d)

#

# run the univariate model for line 3

#

set.seed(12345)

stan\_thin=1

stan\_iter=5000

Rhat\_target=1.05

max\_Rhat=2

# NOTE - I changed the seed to 54321 when testing the duration of identical lines

while ((max\_Rhat > Rhat\_target)&(stan\_thin<65)){

sflist <-

mclapply(1:4, mc.cores = 4,

function(i) stan(fit = fitU, data = data.u1,init=initU,

seed = 12345,iter=stan\_iter,thin=stan\_thin,

chains = 1, chain\_id = i, cores=4,pars=pars.list))

fitU1=sflist2stanfit(sflist)

fitU1\_summary=as.matrix(summary(fitU1)$summary)[,c(1,3,10)]

mrh=subset(fitU1\_summary,is.na(fitU1\_summary[,3])==F)

max\_Rhat=round(max(mrh[,3]),4)

mean\_lp\_\_=round(fitU1\_summary[dim(fitU1\_summary)[1],1],2)

print(paste("Maximum Rhat =",max\_Rhat," Mean lp\_\_ =",mean\_lp\_\_,

"Thin =",stan\_thin))

stan\_thin=2\*stan\_thin

stan\_iter=2\*stan\_iter

}

stan\_thin2=stan\_thin/2

max\_Rhat2=max\_Rhat

#

# extract information from stan output to process in R

#

b=extract(fitU1)

alpha3=b$alpha

beta3=b$beta

gamma3=b$gamma

delta3=b$delta

logelr3=b$logelr

sigma3=b$sig

num.mcmc=length(logelr3)

#

# get speedup rates by accident year

#

speedup3=matrix(1,num.mcmc,10)

for (i in 2:10){

speedup3[,i]=speedup3[,i-1]\*(1-gamma3-(i-2)\*delta3)

}

#

# simulate outcomes for d=10 using parallel processing

#

cl <- makeCluster(4)

registerDoParallel(cl)

at.wd10=foreach (k=1:length(gamma3),.combine=rbind) %dopar%{

set.seed(12345)

atv=rep(0,10)

for (w in 1:10){

atv[w]=rlnorm(1,log(premium[w])+logelr3[k]+alpha3[k,w],sigma3[k,10])

}

at=atv

}

stopCluster(cl)

#

# calculate loss statistics and output to data frame

#

Premium3=subset(rdata3,rdata3$d==1)$premium

ss.wd10=rep(0,10)

ms.wd10=rep(0,10)

#

ms.wd10[1]=mean(at.wd10[,1])

for (w in 2:10){

ms.wd10[w]=mean(at.wd10[,w])

ss.wd10[w]=sd(at.wd10[,w])

}

Pred.CSR=rowSums(at.wd10)

ms.td10=mean(Pred.CSR)

ss.td10=sd(Pred.CSR)

CSR.Estimate3=round(ms.td10)

CSR.SE3=round(ss.td10)

#

# repeat the above for line 4

#

a=read.csv(line4.data)

cdata=ins.line.data(grpcode)

set.seed(12345)

w=cdata$w-1987

d=cdata$d

#

# sort the data in order of d, then w within d

#

o1=100\*d+w

o=order(o1)

w=w[o]

d=d[o]

premium=cdata$net\_premium[o]

cpdloss=cdata$cum\_pdloss[o]

cpdloss=pmax(cpdloss,1)

adata=data.frame(grpcode,w,d,premium,cpdloss)

rdata4=subset(adata,(adata$w+adata$d)<12)

rloss4=rdata4$cpdloss

aloss4=adata$cpdloss

Premium4=subset(rdata4,rdata4$d==1)$premium

#

# data for univariate 4

#

data.u1=list(logprem = log(rdata4$premium[1:10]),

logloss = log(rloss4),

w = rdata4$w,

d = rdata4$d)

#

# run the univariate model for line 4

#

set.seed(12345)

stan\_thin=1

stan\_iter=5000

Rhat\_target=1.05

max\_Rhat=2

# NOTE - I changed the seed to 54321 when testing the duration of identical lines

while ((max\_Rhat > Rhat\_target)&(stan\_thin<65)){

sflist <-

mclapply(1:4, mc.cores = 4,

function(i) stan(fit = fitU, data = data.u1,init=initU,

seed = 12345,iter=stan\_iter,thin=stan\_thin,

chains = 1, chain\_id = i, cores=4,pars=pars.list))

fitU1=sflist2stanfit(sflist)

fitU1\_summary=as.matrix(summary(fitU1)$summary)[,c(1,3,10)]

mrh=subset(fitU1\_summary,is.na(fitU1\_summary[,3])==F)

max\_Rhat=round(max(mrh[,3]),4)

mean\_lp\_\_=round(fitU1\_summary[dim(fitU1\_summary)[1],1],2)

print(paste("Maximum Rhat =",max\_Rhat," Mean lp\_\_ =",mean\_lp\_\_,

"Thin =",stan\_thin))

stan\_thin=2\*stan\_thin

stan\_iter=2\*stan\_iter

}

stan\_thin2=stan\_thin/2

max\_Rhat2=max\_Rhat

#

# extract information from stan output to process in R

#

b=extract(fitU1)

alpha4=b$alpha

beta4=b$beta

gamma4=b$gamma

delta4=b$delta

logelr4=b$logelr

sigma4=b$sig

num.mcmc=length(logelr4)

#

# get speedup rates by accident year

#

speedup4=matrix(1,num.mcmc,10)

for (i in 2:10){

speedup4[,i]=speedup4[,i-1]\*(1-gamma4-(i-2)\*delta4)

}

#

# simulate outcomes for d=10 using parallel processing

#

cl <- makeCluster(4)

registerDoParallel(cl)

at.wd10=foreach (k=1:length(gamma4),.combine=rbind) %dopar%{

set.seed(12345)

atv=rep(0,10)

for (w in 1:10){

atv[w]=rlnorm(1,log(premium[w])+logelr4[k]+alpha4[k,w],sigma4[k,10])

}

at=atv

}

stopCluster(cl)

#

# calculate loss statistics and output to data frame

#

Premium4=subset(rdata4,rdata4$d==1)$premium

ss.wd10=rep(0,10)

ms.wd10=rep(0,10)

#

ms.wd10[1]=mean(at.wd10[,1])

for (w in 2:10){

ms.wd10[w]=mean(at.wd10[,w])

ss.wd10[w]=sd(at.wd10[,w])

}

Pred.CSR=rowSums(at.wd10)

ms.td10=mean(Pred.CSR)

ss.td10=sd(Pred.CSR)

CSR.Estimate4=round(ms.td10)

CSR.SE4=round(ss.td10)

#

# calculate the best estimate (discounted)

#

pv.paid=matrix(0,10,10)

paid=matrix(0,10,11)

cl <- makeCluster(4)

registerDoParallel(cl)

bestest=foreach (i=1:num.mcmc,.combine=rbind) %dopar%{

for (w in 1:10){

for (d in 1:10){

paid[w,d+1]=Premium1[w]\*

exp(logelr1[i]+alpha1[i,w]+beta1[i,d]\*speedup1[i,w]+sigma1[i,d]^2/2)

}

}

for (w in 2:10){

for (d in (12-w):10){

pv.paid[w,d]=(paid[w,d+1]-paid[w,d])/(1+fixed.rate)^(w+d-11.5)

}

}

be=sum(pv.paid)

}

stopCluster(cl)

best.estimate1=round(mean(bestest))

#

cl <- makeCluster(4)

registerDoParallel(cl)

bestest=foreach (i=1:num.mcmc,.combine=rbind) %dopar%{

for (w in 1:10){

for (d in 1:10){

paid[w,d+1]=Premium2[w]\*

exp(logelr2[i]+alpha2[i,w]+beta2[i,d]\*speedup2[i,w]+sigma2[i,d]^2/2)

}

}

for (w in 2:10){

for (d in (12-w):10){

pv.paid[w,d]=(paid[w,d+1]-paid[w,d])/(1+fixed.rate)^(w+d-11.5)

}

}

be=sum(pv.paid)

}

stopCluster(cl)

best.estimate2=round(mean(bestest))

#

cl <- makeCluster(4)

registerDoParallel(cl)

bestest=foreach (i=1:num.mcmc,.combine=rbind) %dopar%{

for (w in 1:10){

for (d in 1:10){

paid[w,d+1]=Premium3[w]\*

exp(logelr3[i]+alpha3[i,w]+beta3[i,d]\*speedup3[i,w]+sigma3[i,d]^2/2)

}

}

for (w in 2:10){

for (d in (12-w):10){

pv.paid[w,d]=(paid[w,d+1]-paid[w,d])/(1+fixed.rate)^(w+d-11.5)

}

}

be=sum(pv.paid)

}

stopCluster(cl)

best.estimate3=round(mean(bestest))

#

cl <- makeCluster(4)

registerDoParallel(cl)

bestest=foreach (i=1:num.mcmc,.combine=rbind) %dopar%{

for (w in 1:10){

for (d in 1:10){

paid[w,d+1]=Premium4[w]\*

exp(logelr4[i]+alpha4[i,w]+beta4[i,d]\*speedup4[i,w]+sigma4[i,d]^2/2)

}

}

for (w in 2:10){

for (d in (12-w):10){

pv.paid[w,d]=(paid[w,d+1]-paid[w,d])/(1+fixed.rate)^(w+d-11.5)

}

}

be=sum(pv.paid)

}

stopCluster(cl)

best.estimate4=round(mean(bestest))

best.estimate=best.estimate1+best.estimate2+best.estimate3+best.estimate4

#

# simulate loss for each line for the lower cumulative triangle

# in order of increasing calendar year and

# decreasing accident year within each calendar year

#

loss1.lowtri=matrix(0,num.mcmc,45)

loss2.lowtri=matrix(0,num.mcmc,45)

loss3.lowtri=matrix(0,num.mcmc,45)

loss4.lowtri=matrix(0,num.mcmc,45)

mu1.lowtri=matrix(0,num.mcmc,45)

mu2.lowtri=matrix(0,num.mcmc,45)

mu3.lowtri=matrix(0,num.mcmc,45)

mu4.lowtri=matrix(0,num.mcmc,45)

#

wmap=11-1:9

for (i in 1:8){

wmap=c(wmap,11-1:(9-i))

}

dmap=2:10

for(i in 3:10){

dmap=c(dmap,i:10)

}

cyfirst=c(1,10,18,25,31,36,40,43,45)

cylast=c(9,17,24,30,35,39,42,44,45)

#

set.seed(12345)

for (i in 1:45){

mu1.lowtri[,i]=log(Premium1[wmap[i]])+logelr1+alpha1[,wmap[i]]+

beta1[,dmap[i]]\*speedup1[,wmap[i]]

loss1.lowtri[,i]=rlnorm(num.mcmc,mu1.lowtri[,i],sigma1[,dmap[i]])

}

set.seed(12345)

for (i in 1:45){

mu2.lowtri[,i]=log(Premium2[wmap[i]])+logelr2+alpha2[,wmap[i]]+

beta2[,dmap[i]]\*speedup2[,wmap[i]]

loss2.lowtri[,i]=rlnorm(num.mcmc,mu2.lowtri[,i],sigma2[,dmap[i]])

}

set.seed(12345)

for (i in 1:45){

mu3.lowtri[,i]=log(Premium3[wmap[i]])+logelr3+alpha3[,wmap[i]]+

beta3[,dmap[i]]\*speedup3[,wmap[i]]

loss3.lowtri[,i]=rlnorm(num.mcmc,mu3.lowtri[,i],sigma3[,dmap[i]])

}

set.seed(12345)

for (i in 1:45){

mu4.lowtri[,i]=log(Premium4[wmap[i]])+logelr4+alpha4[,wmap[i]]+

beta4[,dmap[i]]\*speedup4[,wmap[i]]

loss4.lowtri[,i]=rlnorm(num.mcmc,mu4.lowtri[,i],sigma4[,dmap[i]])

}

#

# get the estimates conditioned on each scenario

#

mean.ult=matrix(0,num.mcmc,10)

for (i in 1:10){

mean.ult[,i]=exp(log(Premium1[i])+logelr1+alpha1[,i]+sigma1[,10]^2/2)

}

ultall.1=rowSums(mean.ult)

for (i in 1:10){

mean.ult[,i]=exp(log(Premium2[i])+logelr2+alpha2[,i]+sigma2[,10]^2/2)

}

ultall.2=rowSums(mean.ult)

for (i in 1:10){

mean.ult[,i]=exp(log(Premium3[i])+logelr3+alpha3[,i]+sigma3[,10]^2/2)

}

ultall.3=rowSums(mean.ult)

for (i in 1:10){

mean.ult[,i]=exp(log(Premium4[i])+logelr4+alpha4[,i]+sigma4[,10]^2/2)

}

ultall.4=rowSums(mean.ult)

#

# likelihood function of the observed new cy

#

llike=function(x,mu,sigma,cy,sz){

ll=rep(0,sz)

for (i in (cyfirst[cy]:cylast[cy])){

ll=ll+dnorm(log(x[i]),mu[,i],sigma[,dmap[i]],log=T)

}

return(ll)

}

#

# calculate the posterior post\_TVaR -

#

post\_assets=function(post1,ult1,post2,ult2,post3,ult3,post4,ult4,k){

set.seed(k)

x1=sample(ult1,10000,replace=T,post1)

set.seed(k)

x2=sample(ult2,10000,replace=T,post2)

set.seed(k)

x3=sample(ult3,10000,replace=T,post3)

set.seed(k)

x4=sample(ult4,10000,replace=T,post4)

pa=rep(0,10)

pa[1]=mean(x2+x3+x4)

pa[2]=mean(sort(x2+x3+x4)[TVaR.Range])

pa[3]=mean(x1+x3+x4)

pa[4]=mean(sort(x1+x3+x4)[TVaR.Range])

pa[5]=mean(x1+x2+x4)

pa[6]=mean(sort(x1+x2+x4)[TVaR.Range])

pa[7]=mean(x1+x2+x3)

pa[8]=mean(sort(x1+x2+x3)[TVaR.Range])

pa[9]=mean(x1+x2+x3+x4)

pa[10]=mean(sort(x1+x2+x3+x4)[TVaR.Range])

pa[11]=mean(x1)

pa[12]=mean(sort(x1)[TVaR.Range])

pa[13]=mean(x2)

pa[14]=mean(sort(x2)[TVaR.Range])

pa[15]=mean(x3)

pa[16]=mean(sort(x3)[TVaR.Range])

pa[17]=mean(x4)

pa[18]=mean(sort(x4)[TVaR.Range])

return(pa)

}

#

# get conditional estimates

#

p.mean.all=rep(0,10)

p.assets.all=rep(0,10)

p.mean.1=rep(0,10)

p.assets.1=rep(0,10)

p.mean.2=rep(0,10)

p.assets.2=rep(0,10)

p.mean.3=rep(0,10)

p.assets.3=rep(0,10)

p.mean.4=rep(0,10)

p.assets.4=rep(0,10)

p.mean.u1=rep(0,10)

p.assets.u1=rep(0,10)

p.mean.u2=rep(0,10)

p.assets.u2=rep(0,10)

p.mean.u3=rep(0,10)

p.assets.u3=rep(0,10)

p.mean.u4=rep(0,10)

p.assets.u4=rep(0,10)

#

cl <- makePSOCKcluster(4)

registerDoParallel(cl)

pred.mean.assets=foreach (i=1:num.mcmc,.combine=rbind) %dopar%{

x1=loss1.lowtri[i,]

x2=loss2.lowtri[i,]

x3=loss3.lowtri[i,]

x4=loss4.lowtri[i,]

#

loglike.1=rep(0,num.mcmc)

loglike.2=rep(0,num.mcmc)

loglike.3=rep(0,num.mcmc)

loglike.4=rep(0,num.mcmc)

posterior.init=rep(.0001,10000)

#

p0=rep(1/num.mcmc,num.mcmc)

call.pa=post\_assets(p0,ultall.1,

p0,ultall.2,

p0,ultall.3,

p0,ultall.4,i)

p.mean.1[1]=call.pa[1]

p.assets.1[1]=call.pa[2]

p.mean.2[1]=call.pa[3]

p.assets.2[1]=call.pa[4]

p.mean.3[1]=call.pa[5]

p.assets.3[1]=call.pa[6]

p.mean.4[1]=call.pa[7]

p.assets.4[1]=call.pa[8]

p.mean.all[1]=call.pa[9]

p.assets.all[1]=call.pa[10]

p.mean.u1[1]=call.pa[11]

p.assets.u1[1]=call.pa[12]

p.mean.u2[1]=call.pa[13]

p.assets.u2[1]=call.pa[14]

p.mean.u3[1]=call.pa[15]

p.assets.u3[1]=call.pa[16]

p.mean.u4[1]=call.pa[17]

p.assets.u4[1]=call.pa[18]

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,1,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,1,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,1,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,1,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[2]=call.pa[1]

p.assets.1[2]=call.pa[2]

p.mean.2[2]=call.pa[3]

p.assets.2[2]=call.pa[4]

p.mean.3[2]=call.pa[5]

p.assets.3[2]=call.pa[6]

p.mean.4[2]=call.pa[7]

p.assets.4[2]=call.pa[8]

p.mean.all[2]=call.pa[9]

p.assets.all[2]=call.pa[10]

p.mean.u1[2]=call.pa[11]

p.assets.u1[2]=call.pa[12]

p.mean.u2[2]=call.pa[13]

p.assets.u2[2]=call.pa[14]

p.mean.u3[2]=call.pa[15]

p.assets.u3[2]=call.pa[16]

p.mean.u4[2]=call.pa[17]

p.assets.u4[2]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,2,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,2,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,2,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,2,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[3]=call.pa[1]

p.assets.1[3]=call.pa[2]

p.mean.2[3]=call.pa[3]

p.assets.2[3]=call.pa[4]

p.mean.3[3]=call.pa[5]

p.assets.3[3]=call.pa[6]

p.mean.4[3]=call.pa[7]

p.assets.4[3]=call.pa[8]

p.mean.all[3]=call.pa[9]

p.assets.all[3]=call.pa[10]

p.mean.u1[3]=call.pa[11]

p.assets.u1[3]=call.pa[12]

p.mean.u2[3]=call.pa[13]

p.assets.u2[3]=call.pa[14]

p.mean.u3[3]=call.pa[15]

p.assets.u3[3]=call.pa[16]

p.mean.u4[3]=call.pa[17]

p.assets.u4[3]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,3,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,3,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,3,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,3,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[4]=call.pa[1]

p.assets.1[4]=call.pa[2]

p.mean.2[4]=call.pa[3]

p.assets.2[4]=call.pa[4]

p.mean.3[4]=call.pa[5]

p.assets.3[4]=call.pa[6]

p.mean.4[4]=call.pa[7]

p.assets.4[4]=call.pa[8]

p.mean.all[4]=call.pa[9]

p.assets.all[4]=call.pa[10]

p.mean.u1[4]=call.pa[11]

p.assets.u1[4]=call.pa[12]

p.mean.u2[4]=call.pa[13]

p.assets.u2[4]=call.pa[14]

p.mean.u3[4]=call.pa[15]

p.assets.u3[4]=call.pa[16]

p.mean.u4[4]=call.pa[17]

p.assets.u4[4]=call.pa[18]

#

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,4,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,4,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,4,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,4,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[5]=call.pa[1]

p.assets.1[5]=call.pa[2]

p.mean.2[5]=call.pa[3]

p.assets.2[5]=call.pa[4]

p.mean.3[5]=call.pa[5]

p.assets.3[5]=call.pa[6]

p.mean.4[5]=call.pa[7]

p.assets.4[5]=call.pa[8]

p.mean.all[5]=call.pa[9]

p.assets.all[5]=call.pa[10]

p.mean.u1[5]=call.pa[11]

p.assets.u1[5]=call.pa[12]

p.mean.u2[5]=call.pa[13]

p.assets.u2[5]=call.pa[14]

p.mean.u3[5]=call.pa[15]

p.assets.u3[5]=call.pa[16]

p.mean.u4[5]=call.pa[17]

p.assets.u4[5]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,5,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,5,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,5,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,5,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[6]=call.pa[1]

p.assets.1[6]=call.pa[2]

p.mean.2[6]=call.pa[3]

p.assets.2[6]=call.pa[4]

p.mean.3[6]=call.pa[5]

p.assets.3[6]=call.pa[6]

p.mean.4[6]=call.pa[7]

p.assets.4[6]=call.pa[8]

p.mean.all[6]=call.pa[9]

p.assets.all[6]=call.pa[10]

p.mean.u1[6]=call.pa[11]

p.assets.u1[6]=call.pa[12]

p.mean.u2[6]=call.pa[13]

p.assets.u2[6]=call.pa[14]

p.mean.u3[6]=call.pa[15]

p.assets.u3[6]=call.pa[16]

p.mean.u4[6]=call.pa[17]

p.assets.u4[6]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,6,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,6,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,6,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,6,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[7]=call.pa[1]

p.assets.1[7]=call.pa[2]

p.mean.2[7]=call.pa[3]

p.assets.2[7]=call.pa[4]

p.mean.3[7]=call.pa[5]

p.assets.3[7]=call.pa[6]

p.mean.4[7]=call.pa[7]

p.assets.4[7]=call.pa[8]

p.mean.all[7]=call.pa[9]

p.assets.all[7]=call.pa[10]

p.mean.u1[7]=call.pa[11]

p.assets.u1[7]=call.pa[12]

p.mean.u2[7]=call.pa[13]

p.assets.u2[7]=call.pa[14]

p.mean.u3[7]=call.pa[15]

p.assets.u3[7]=call.pa[16]

p.mean.u4[7]=call.pa[17]

p.assets.u4[7]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,7,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,7,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,7,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,7,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[8]=call.pa[1]

p.assets.1[8]=call.pa[2]

p.mean.2[8]=call.pa[3]

p.assets.2[8]=call.pa[4]

p.mean.3[8]=call.pa[5]

p.assets.3[8]=call.pa[6]

p.mean.4[8]=call.pa[7]

p.assets.4[8]=call.pa[8]

p.mean.all[8]=call.pa[9]

p.assets.all[8]=call.pa[10]

p.mean.u1[8]=call.pa[11]

p.assets.u1[8]=call.pa[12]

p.mean.u2[8]=call.pa[13]

p.assets.u2[8]=call.pa[14]

p.mean.u3[8]=call.pa[15]

p.assets.u3[8]=call.pa[16]

p.mean.u4[8]=call.pa[17]

p.assets.u4[8]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,8,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,8,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,8,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,8,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[9]=call.pa[1]

p.assets.1[9]=call.pa[2]

p.mean.2[9]=call.pa[3]

p.assets.2[9]=call.pa[4]

p.mean.3[9]=call.pa[5]

p.assets.3[9]=call.pa[6]

p.mean.4[9]=call.pa[7]

p.assets.4[9]=call.pa[8]

p.mean.all[9]=call.pa[9]

p.assets.all[9]=call.pa[10]

p.mean.u1[9]=call.pa[11]

p.assets.u1[9]=call.pa[12]

p.mean.u2[9]=call.pa[13]

p.assets.u2[9]=call.pa[14]

p.mean.u3[9]=call.pa[15]

p.assets.u3[9]=call.pa[16]

p.mean.u4[9]=call.pa[17]

p.assets.u4[9]=call.pa[18]

#

#

loglike.1=loglike.1+llike(x1,mu1.lowtri,sigma1,9,num.mcmc)

loglike=loglike.1-max(loglike.1)

postint=sum(exp(loglike))

posterior.1=exp(loglike)/postint

#

loglike.2=loglike.2+llike(x2,mu2.lowtri,sigma2,9,num.mcmc)

loglike=loglike.2-max(loglike.2)

postint=sum(exp(loglike))

posterior.2=exp(loglike)/postint

#

loglike.3=loglike.3+llike(x3,mu3.lowtri,sigma3,9,num.mcmc)

loglike=loglike.3-max(loglike.3)

postint=sum(exp(loglike))

posterior.3=exp(loglike)/postint

#

loglike.4=loglike.4+llike(x4,mu4.lowtri,sigma4,9,num.mcmc)

loglike=loglike.4-max(loglike.4)

postint=sum(exp(loglike))

posterior.4=exp(loglike)/postint

#

call.pa=post\_assets(posterior.1,ultall.1,

posterior.2,ultall.2,

posterior.3,ultall.3,

posterior.4,ultall.4,i)

p.mean.1[10]=call.pa[1]

p.assets.1[10]=call.pa[2]

p.mean.2[10]=call.pa[3]

p.assets.2[10]=call.pa[4]

p.mean.3[10]=call.pa[5]

p.assets.3[10]=call.pa[6]

p.mean.4[10]=call.pa[7]

p.assets.4[10]=call.pa[8]

p.mean.all[10]=call.pa[9]

p.assets.all[10]=call.pa[10]

p.mean.u1[10]=call.pa[11]

p.assets.u1[10]=call.pa[12]

p.mean.u2[10]=call.pa[13]

p.assets.u2[10]=call.pa[14]

p.mean.u3[10]=call.pa[15]

p.assets.u3[10]=call.pa[16]

p.mean.u4[10]=call.pa[17]

p.assets.u4[10]=call.pa[18]

#

result=c(p.mean.1,p.assets.1,p.mean.2,p.assets.2,p.mean.3,p.assets.3,

p.mean.4,p.assets.4,p.mean.all,p.assets.all,p.mean.u1,p.assets.u1,

p.mean.u2,p.assets.u2,p.mean.u3,p.assets.u3,p.mean.u4,p.assets.u4)

}

stopCluster(cl)

#

# calculate marginal capital

#

pred.E.all=as.matrix(pred.mean.assets[,81:90])

pred.A.all=as.matrix(pred.mean.assets[,91:100])

pred.C.all=pred.A.all-pred.E.all

release=pred.C.all[,1:9]\*(1+fixed.rate)-pred.C.all[,2:10]

risk.margin.all=pred.C.all[,1]

for (i in 1:9){

risk.margin.all=risk.margin.all-release[,i]/(1+risky.rate)^i

}

#

pred.E.m1=as.matrix(pred.mean.assets[,1:10])

pred.A.m1=as.matrix(pred.mean.assets[,11:20])

pred.C.m1=pred.A.m1-pred.E.m1

release=pred.C.m1[,1:9]\*(1+fixed.rate)-pred.C.m1[,2:10]

risk.margin.m1=pred.C.m1[,1]

for (i in 1:9){

risk.margin.m1=risk.margin.m1-release[,i]/(1+risky.rate)^i

}

#

pred.E.m2=as.matrix(pred.mean.assets[,21:30])

pred.A.m2=as.matrix(pred.mean.assets[,31:40])

pred.C.m2=pred.A.m2-pred.E.m2

release=pred.C.m2[,1:9]\*(1+fixed.rate)-pred.C.m2[,2:10]

risk.margin.m2=pred.C.m2[,1]

for (i in 1:9){

risk.margin.m2=risk.margin.m2-release[,i]/(1+risky.rate)^i

}

#

pred.E.m3=as.matrix(pred.mean.assets[,41:50])

pred.A.m3=as.matrix(pred.mean.assets[,51:60])

pred.C.m3=pred.A.m3-pred.E.m3

release=pred.C.m3[,1:9]\*(1+fixed.rate)-pred.C.m3[,2:10]

risk.margin.m3=pred.C.m3[,1]

for (i in 1:9){

risk.margin.m3=risk.margin.m3-release[,i]/(1+risky.rate)^i

}

#

pred.E.m4=as.matrix(pred.mean.assets[,61:70])

pred.A.m4=as.matrix(pred.mean.assets[,71:80])

pred.C.m4=pred.A.m4-pred.E.m4

release=pred.C.m4[,1:9]\*(1+fixed.rate)-pred.C.m4[,2:10]

risk.margin.m4=pred.C.m4[,1]

for (i in 1:9){

risk.margin.m4=risk.margin.m4-release[,i]/(1+risky.rate)^i

}

#

pred.E.1=as.matrix(pred.mean.assets[,101:110])

pred.A.1=as.matrix(pred.mean.assets[,111:120])

pred.C.1=pred.A.1-pred.E.1

release=pred.C.1[,1:9]\*(1+fixed.rate)-pred.C.1[,2:10]

risk.margin.1=pred.C.1[,1]

for (i in 1:9){

risk.margin.1=risk.margin.1-release[,i]/(1+risky.rate)^i

}

#

pred.E.2=as.matrix(pred.mean.assets[,121:130])

pred.A.2=as.matrix(pred.mean.assets[,131:140])

pred.C.2=pred.A.2-pred.E.2

release=pred.C.2[,1:9]\*(1+fixed.rate)-pred.C.2[,2:10]

risk.margin.2=pred.C.2[,1]

for (i in 1:9){

risk.margin.2=risk.margin.2-release[,i]/(1+risky.rate)^i

}

#

pred.E.3=as.matrix(pred.mean.assets[,141:150])

pred.A.3=as.matrix(pred.mean.assets[,151:160])

pred.C.3=pred.A.3-pred.E.3

release=pred.C.3[,1:9]\*(1+fixed.rate)-pred.C.3[,2:10]

risk.margin.3=pred.C.3[,1]

for (i in 1:9){

risk.margin.3=risk.margin.3-release[,i]/(1+risky.rate)^i

}

#

pred.E.4=as.matrix(pred.mean.assets[,161:170])

pred.A.4=as.matrix(pred.mean.assets[,171:180])

pred.C.4=pred.A.4-pred.E.4

release=pred.C.4[,1:9]\*(1+fixed.rate)-pred.C.4[,2:10]

risk.margin.4=pred.C.4[,1]

for (i in 1:9){

risk.margin.4=risk.margin.4-release[,i]/(1+risky.rate)^i

}

#

mrm1=mean(risk.margin.all)-mean(risk.margin.m1)

mrm2=mean(risk.margin.all)-mean(risk.margin.m2)

mrm3=mean(risk.margin.all)-mean(risk.margin.m3)

mrm4=mean(risk.margin.all)-mean(risk.margin.m4)

rmall=mean(risk.margin.all)

rm1=mean(risk.margin.1)

rm2=mean(risk.margin.2)

rm3=mean(risk.margin.3)

rm4=mean(risk.margin.4)

mrm.sum=mrm1+mrm2+mrm3+mrm4

mrm1.alloc=rmall\*mrm1/mrm.sum

mrm2.alloc=rmall\*mrm2/mrm.sum

mrm3.alloc=rmall\*mrm3/mrm.sum

mrm4.alloc=rmall\*mrm4/mrm.sum

#

Ult\_Estimate=c(CSR.Estimate1,CSR.Estimate2,CSR.Estimate3,CSR.Estimate4,

sum(CSR.Estimate1,CSR.Estimate2,CSR.Estimate3,CSR.Estimate4))

Best\_Estimate=round(c(best.estimate1,best.estimate2,best.estimate3,best.estimate4 ,best.estimate))

Marg\_Risk\_Margin=round(c(mrm1,mrm2,mrm3,mrm4,sum(mrm1,mrm2,mrm3,mrm4)))

Risk\_Margin\_Alloc=round(c(mrm1.alloc,mrm2.alloc,mrm3.alloc,mrm4.alloc,rmall))

Univ\_Risk\_Margin=round(c(rm1,rm2,rm3,rm4))

Univ\_Risk\_Margin=c(Univ\_Risk\_Margin,sum(Univ\_Risk\_Margin))

c1=sum(Univ\_Risk\_Margin[c(2,3,4)])-mean(risk.margin.m1)

c2=sum(Univ\_Risk\_Margin[c(1,3,4)])-mean(risk.margin.m2)

c3=sum(Univ\_Risk\_Margin[c(1,2,4)])-mean(risk.margin.m3)

c4=sum(Univ\_Risk\_Margin[c(1,2,3)])-mean(risk.margin.m4)

check=round(c(c1,c2,c3,c4,0))

outframe=data.frame(Ult\_Estimate,Best\_Estimate,Marg\_Risk\_Margin,

Risk\_Margin\_Alloc,Univ\_Risk\_Margin,check)

write.csv(outframe,outfile)

print(outframe)

#

t2=Sys.time()

print(t2-t1)