Ⅰ　資産データ作成

```
N = 4           #資産数
NT = 3          #3時点
nen = 2         #1時点分の期間

#データ90/1～10/12を読み込み，timeSeriesデータ作成
data90.10.td=timeDate(data90.10[,1],in.format="%Y/%m/%d",format="%Y/%m/%d")
dataset=timeSeries(data90.10[,1:N+1],positions=data90.10.td)
dataset.r = getReturns(dataset,type="discrete")       #収益率


#事前評価データ作成
data.r = as.matrix(seriesData(dataset.r))         #timeseriesをmatirx化
data90.10.r = array(NULL,dim=c((12*nen*NT),N,8))      #初期化
#1年ずつズレたdatase(4月始まり)作成
for(i in 1:8){
   data90.10.r[,,i] = data.r[(12*(i-1)+4):(12*(i-1)+12*nen*NT+3),]
}

#93/4-99/3収益データ
data93.99 = data90.10.r[,,4]
#金利の初期値設定(99/4時点)
kinri99 = as.matrix(seriesData(dataset[113,1])


#事後評価データ(99/4-00/3)作成
postdata = as.matrix(seriesData(dataset[113:124,]))
temp = t(postdata[,2:N])/postdata[1,2:N]
postdata[,2:N] = t(temp)
postdata.list = list(aru=postdata[,1],rou=postdata[,2:N])
```

Ⅱ　事前評価

```
####シミュレーションパス生成####################################################
NPATH = 500
data = data93.99
kinri0 = kinri99

#時点間資産間を考慮したdata作成
shisan = matrix(NULL,nrow=12*nen,ncol=3*N)  #初期化
for(i in 1:N){
   for(j in 1:NT){
     shisan[,(3*(i-1)+j)] = data[(12*nen*(j-1)+1):(12*nen*j),i]
   }
}

#基本統計量算出
sbar = apply(shisan,2,mean)       #期待収益率
svar = apply(shisan,2,var)        #分散
ssd = apply(shisan,2,sd)          #SD
sQ = cor(shisan)                  #相関係数

#時点間考慮のため、matrix化
sbar = matrix(Rolbar,N,NT,byrow=T)
svar = matrix(Rolvar,N,NT,byrow=T)
ssd = matrix(Rolsd,N,NT,byrow=T)
```

```
#初期値
b0 =c(kinri0/100,1,1,1)
b0 = matrix(b0)

#パス生成
.Random.seed = seed.saved
pass = simpass.make(b0,sbar,svar,ssd,sQ)       #simpass.make関数でパス生成
senario = input.data(pass)                     #input.data関数で最適化の形に変換


####最適化##############################################################

##parameterの設定

# 確率水準設定
beta = c(0.95,0.95,0.95)
beta1 = c(0.95,0.95,0.90)
beta2 = c(0.95,0.95,0.85)
beta3 = c(0.85,0.90,0.95)

#取引コスト
tc = 0.0001

#割引率生成
ex.aru = apply(pass.list$aru,1,mean)
r0 = 1/(1+ex.aru[1])
r1 = 1/(1+ex.aru[2])
r2 = 1/(1+ex.aru[3])
df = c(r0,r0*r1,r0*r1*r2)

#重み付け設定
wf0 = c(0,0,1)
wf = c(1/3,1/3,1/3)
wf1 = c(1/16,5/16,10/16)
wf2 = c(1/21,5/21,15/21)
wf3 = c(1/111,10/111,100/111)
wf4 = c(1/102,1/102,100/102)
wf5 = c(100/111,10/111,1/111)

We = 10050       #要求期待利益の初期値
W0 = 10000       #初期費用

##効率フロンティアのため
nmu = 7          #繰り返し回数
seq = 15         #刻み幅


#効率フロンティア関数
Frontier = function(beta,NT,NPATH,senario,tc,df,wf,W0,We){
    #初期化
    optimal.mat = matrix(NULL,nrow=nmu+1,ncol=2)      #最適解の値を格納するmatrix
    colnames(optimal.mat) = c("要求期待利益","Risk")
    cvar.mat = matrix(NULL,nrow=nmu+1,ncol=4)         #CVaRの値を格納するmatrix
    colnames(cvar.mat) = c("要求期待利益","CVaR[1]","CVaR[2]","CVaR[3]")
    var.mat = matrix(NULL,nrow=nmu+1,ncol=4)          #VaRの値を格納するmatrix
    colnames(var.mat) = c("要求期待利益","VaR[1]","VaR[2]","VaR[3]")
```

```
    wealth.mat = matrix(NULL,nrow=nmu+1,ncol=4)      #各時点の富を格納するmatrix
    colnames(wealth.mat) = c("要求期待利益","W[1]","W[2]","W[3]")
    ratio.array = array(NULL,dim=c(N,NT,nmu+1))      #投資比率の値を格納するmatrix
    dimnames(ratio.array) = list(c("現金","株","債券","ＣＢ"),0:(NT-1),NULL)

    elaTime = matrix(NULL,nrow=nmu+1,ncol=2)         #計算時間を格納するmatrix
    colnames(elaTime) = c("要求期待利益","Time")

    #繰り返し
    for(i in 1:(nmu+1)){
        module(nuopt,unload=T)
        module(nuopt)
        sys.CVaRdev.tc =
        System(model=MultiLon.CVaRdev.tc,beta,NT,NPATH,senario,tc,df,wf,W0,We)
        sol.CVaRdev.tc = solve(sys.CVaRdev.tc)

        #変数を取り出し
        z = as.array(current(sys.CVaRdev.tc,z))
        v = as.array(current(sys.CVaRdev.tc,v))
        v0 = as.array(current(sys.CVaRdev.tc,v0))
        u = as.array(current(sys.CVaRdev.tc,u))

        Risk = as.array(current(sys.CVaRdev.tc,Risk))
        W.rou = as.array(current(sys.CVaRdev.tc,W.rou))
        VaR = as.array(current(sys.CVaRdev.tc,VaR))
        CVaR = as.array(current(sys.CVaRdev.tc,CVaR))
        ex.rou = as.array(current(sys.CVaRdev.tc,ex.rou))

        #時点，要求期待利益による値の表作成
        optimal.mat[i,] = c(We,Risk)
        cvar.mat[i,] = c(We,CVaR)
        var.mat[i,] = c(We,VaR)
        wealth.mat[i,] = c(We,W.rou)

        #投資比率を格納
        ratio.array[,,i] = ratio(v,v0,ex.rou,senairo.list,z,W0,W.rou)

        #計算時間を格納
        elaTime[i,] = c(We,sol.CVaRdev.tc$elapsed.time)

        We = We + seq
    }
    Result =
    list(z=z,v=v,v0=v0,u=u,optimal.mat=optimal.mat,cvar.mat=cvar.mat,var.mat=var.m
    at,wealth.mat=wealth.mat,ratio.array=ratio.array,elaTime=elaTime)#結果リスト化
}

#重み変化による結果取りだし
kekka = Frontier(beta,NT,NPATH,senario,tc,df,wf,10000,10050,7,15)
kekka11100 = Frontier(beta,NT,NPATH,senario,tc,df,wf4,10000,10050,7,15)
kekka001 = Frontier(beta,NT,NPATH,senario,tc,df,wf0,10000,10050,7,15)
kekka1510 = Frontier(beta,NT,NPATH,senario,tc,df,wf1,10000,10050,7,15)

#β変化による結果取りだし
kekka959590 = Frontier(beta1,NT,NPATH,senario,tc,df,wf,10000,10050,7,15)
kekka959585 = Frontier(beta2,NT,NPATH,senario,tc,df,wf,10000,10050,7,15)
```

```
kekka859095 = Frontier(beta3,NT,NPATH,senario,tc,df,wf,10000,10050,7,15)
```

Ⅲ　事後評価

```
##初期値設定
it = 11                #運用期間12カ月
We = 10050             #期待利益
W0 = 10000

#ﾛｰﾘﾝｸﾞ統計量
rolQ = Rolling(it,nen,NT,N,dataset.r)$rolQ
rolbar = Rolling(it,nen,NT,N,dataset.r)$rolbar
rolvar = Rolling(it,nen,NT,N,dataset.r)$rolvar
rolsd = Rolling(it,nen,NT,N,dataset.r)$rolsd

#初期化
Vec = c(W0,rep(NA,it))
a.ratio = matrix(NA,nrow=it,ncol=N)
colnames(a.ratio) = c("現金","株","債券","ＣＢ")

for(i in 1:it){
   b1 = postdata[i,]          #初期値設定
   b1[1] = b1[1]/100
   b1 = as.matrix(b0)
   Q.p = rolQ[,,i]            #基本統計量
   rbar.p = rolbar[,,i]
   rvar.p = rolvar[,,i]
   rsd.p = rolsd[,,i]

   #ﾊﾟｽ生成
   .Random.seed = seed.saved
   pass2 = simpass.make(b1,rbar.p,rvar.p,rsd.p,Q.p)
   senario2 = input.data(pass2)

   #最適化
   module(nuopt,unload=T)
   module(nuopt)
   sys.CVaRdev.tc =
   System(model=MultiLon.CVaRdev.tc,beta,NT,NPATH,senario2,tc,df,wf,W0,We)
   sol.CVaRdev.tc = solve(sys.CVaRdev.tc)

   #変数取り出し
   a.z = as.array(current(sys.CVaRdev.tc,z))
   a.v0 = as.array(current(sys.CVaRdev.tc,v0))
   a.rou0 = b1[2:N]
   a.ratio[i,] = c(a.v0,a.rou0*a.z[,1])/Vec[i]        #投資比率

   #価値算出
   Vec[i+1] = sum(postdata.list$rou[i+1,]*a.z[,1]*(1-tc)) +
   (1+postdata.list$aru[i])*a.v0

}
```

Ⅳ　設定関数

```
## 幾何ブラウン運動によるシミュレーションパス生成関数##
simpass.make = function(d0,rbar,rvar,rsd,Q){
    epsilon = rmvnorm(NPATH,cov=Q)
    epsilon.array = array(NA,dim=c(N,NT,NPATH))
    for(i in 1:NPATH){
        epsilon.array[,,i]=matrix(epsilon[i,],N,NT,byrow=T)
    }

    pass.array = array(NA,dim=c(N,NT+1,NPATH))
    for(i in 1:NT){
        for(j in 1:N){
            pass.array[j,1,] = d0[j]
            pass.array[j,i+1,] = 
    pass.array[j,i,]*exp((rbar[j,i]-0.5*rvar[j,i])+rsd[j,i]*epsilon.array[j,i,])
        }
    }

    aru = pass.array[1,,]
    rou = pass.array[2:N,,]
    pass.list = list(aru=aru,rou=rou)
    pass.list
}


##最適化用に変換する関数##
input.data = function(pass.list){
    aru0 = pass.list$aru[1,1]
    rou0 = pass.list$rou[,1,1]
    aru = pass.list$aru[-1,]
    rou = pass.list$rou[,-1,]
    list(aru0=aru0,rou0=rou0,aru=aru,rou=rou)
}


##多目標CVaR偏差最適化関数##
MultiLon.CVaRdev.t = function(beta,NT,NPATH,senario.list,tc,df,wf,W0,We)
{
    #集合と添え字の宣言
    Asset = Set()
    j = Element(set=Asset)
    Path = Set()
    i = Element(set=Path)

    Time = Set()
    t = Element(set=Time)

    #パラメーターの設定
    beta = Parameter(index=t,as.array(beta))
    T = Parameter(NT)
    I = Parameter(NPATH)
    rou0 = Parameter(index=j,as.array(senario.list$rou0))
    rou = Parameter(index=dprod(j,t,i),senario.list$rou)
    aru0 = Parameter(senario.list$aru0)
    aru = Parameter(index=dprod(t,i),senario.list$aru)
    tc = Parameter(tc)
    df=Parameter(index=t,as.array(df))
    wf=Parameter(index=t,as.array(wf))
```

```
#富の設定
W0 = Parameter(w0)
WE = Parameter(we)

#決定変数の設定
z = Variable(index=dprod(j,t))
z0 = Variable(index=dprod(j))
v = Variable(index=dprod(t,i))
v0 = Variable()
yB = Variable(index=dprod(j,t))
yS = Variable(index=dprod(j,t))


VaR = Variable(index=t)
u = Variable(index=dprod(t,i))

#t時点i経路の富
W = Expression(index=dprod(t,i))
W[1,i] ~ Sum((1-tc)*rou[j,1,i]*z0[j],j)+(1+aru0)*v0
W[t,i,t>=2] ~ Sum((1-tc)*rou[j,t,i]*z[j,t-1],j)+(1+aru[t-1,i])*v[t-1,i]

#最終時点のrouの平均値
ex.rou = Expression(index=dprod(j,t))
ex.rou[j,t] ~ Sum(rou[j,t,i],i)/I

#t時点の富の平均
W.rou = Expression(index=t)
W.rou[1] ~ Sum((1-tc)*ex.rou[j,1]*z0[j],j)+(1+aru0)*v0
W.rou[t,t>=2] ~
Sum((1-tc)*ex.rou[j,t]*z[j,t-1],j)+Sum((1+aru[t-1,i])*v[t-1,i],i)/I

#投資量保存式
z[j,1] == z[j,0] + yB[j,1] - yS[j,1]
z[j,t,t>=2] == z[j,t-1] + yB[j,t] - yS[j,t]


#制約式
W0 == Sum((1+tc)*rou0[j]*z0[j],j)+v0
Sum((1-tc)*rou[j,1,i]*yS[j,1],j)+(1+aru0)*v0 ==
Sum((1+tc)*rou[j,1,i]*yB[j,1],j)+v[1,i]
Sum((1-tc)*rou[j,t,i]*yS[j,t],j)+(1+aru[t-1,i])*v[t-1,i] ==
Sum((1+tc)*rou[j,t,i]*yB[j,t],j)+v[t,i,t>=2,t<=(NT-1)]

Sum((1-tc)*ex.rou[j,NT]*z[j,NT-1],j)+Sum((1+aru[NT-1,i])*v[NT-1,i],i)/I >= WE
u[t,i] + VaR[t] + (W[t,i]/W0 -1) - (W.rou[t]/W0 -1) >= 0
u[t,i] >= 0

z[j,t] >= 0
z0[j] >= 0
v[t,i] >= 0
v0 >= 0
yB[j,t] >= 0
yS[j,t] >= 0

#目的関数
CVaR = Expression(index=t)
CVaR[t] ~ VaR[t] + Sum(u[t,i],i)/(I*(1-beta[t]))
```

```
    Risk = Objective(type=minimize)
    Risk ~ Sum(wf[t]*df[t]*CVaR[t],t)

}


##投資比率算出関数##
ratio = function(v,v0,ex.rou,senairo.list,z,w0,W.rou){
    vbar = apply(v[-NT,],1,mean)      #現金
    vv = c(v0,vbar)

    roubar = ex.rou[,-NT]       #平均価格
    rou.pri = cbind(senario.list$rou0,roubar)
    inv.amo = rou.pri * z       #平均投資額

    amout = rbind(vv,inv.amo)   #全資産の投資額

    Wbar = c(w0,W.rou[-NT])    #0～2時点の富の平均
    inv.rat = amout/Wbar
    inv.rat
}


##ローリング統計量算出関数##
Rolling = function(it,nen,NT,N,dataset.r){
    rolQ = array(NULL,dim=c(NT*N,NT*N,it))          #統計量用のデータ初期化
    rolbar = array(NULL,dim=c(N,NT,it))
    rolvar = array(NULL,dim=c(N,NT,it))
    rolsd = array(NULL,dim=c(N,NT,it))

    for(z in 1:it){
        data = dataset.r[z:(12*nen*NT+(z-1)),]      #1ヶ月ずつずらした期間

        shisan = matrix(NULL,nrow=12*nen,ncol=3*N)  #初期化
        for(i in 1:N){
            for(j in 1:NT){
                shisan[,(3*(i-1)+j)] = data[(12*nen*(j-1)+1):(12*nen*j),i]
            }
        }                                   #時点間&資産間を考慮するため並べ替え

        Rolbar = apply(shisan,2,mean)
        Rolvar = apply(shisan,2,var)
        Rolsd = apply(shisan,2,sd)

        Rolbar = matrix(Rolbar,N,NT,byrow=T)
        Rolvar = matrix(Rolvar,N,NT,byrow=T)
        Rolsd = matrix(Rolsd,N,NT,byrow=T)

        rolQ[,,z] = cor(shisan)
        rolbar[,,z] = Rolbar
        rolvar[,,z] = Rolvar
        rolsd[,,z] = Rolsd
    }
    return(rolQ,rolbar,rolvar,rolsd)
}
```