Construct_Efficient_Portfolio

huaiyuan

```
#######################################################
if (!require(Rsolnp)) { #載入 Rsolnp 函數庫
  install.packages("Rsolnp") #安裝 Rsolnp (會自動下載，故需連接網路)
  require(Rsolnp)# 載入 Rsolnp 函數庫
}
```

```
## 載入需要的套件：Rsolnp
```

```
## Warning: 套件 'Rsolnp' 是用 R 版本 4.1.2 來建造的
```

```
#######################################################

#拉氏乘數法
# W: 投資組合權重
# MU.i: 各資產期望報酬率
# SIGMA.i: 資產報酬率共變數矩陣
#######################################################
#The return of portfolio
mu.p <- function(W, MU.i, SIGMA.i){
  return (sum(W*MU.i)) #use the sum function to return real value
}

#The variance of portfolio
sigma.2.p <- function(W, MU.i, SIGMA.i){
  return (sum((W%*%SIGMA.i)%*%W)) #use the sum function to return real
value
}

#The sum of weights
eqf1 <- function(W, MU.i, SIGMA.i){
  return (sum(W))
}
#The sum of weights and the expected return
eqf2 <- function(W, MU.i, SIGMA.i){
  SumWeight=sum(W)
  ExpectedReturn=mu.p(W, MU.i, SIGMA.i)
  return (c(SumWeight, ExpectedReturn))
}
#######################################################
#######################################################
#在要求的期望報酬之下，極小化投資組合變異數
#######################################################
```

```r
#作業找三檔個股 分別是群創 長榮 欣興
#rf 使用央行定存利率
setwd("C:\\Users\\user\\Desktop")
data<-read.csv("stockdata1.csv")

RF<-data[,4]
STOCK<-data[,1:3]
MKT<-data[,5]

W0 <-c(1/3, 1/3, 1/3)
MU.i0<-colMeans((STOCK-RF/12))*12
SIGMA.i0 <- cov((STOCK-RF/12))*12

mu.p(W0, MU.i0, SIGMA.i0)

## [1] 0.5776172

sigma.2.p(W0, MU.i0, SIGMA.i0)

## [1] 0.192487

lb<-rep(0,length(MU.i0))
ub<-NULL

if(1){
  #MKT<- rowMeans(STOCK)
  PR<- cbind(STOCK,MKT)-(RF/12)
  Ys <- colnames(STOCK)
  Rs<-list()
  for(i in (1:length(Ys))){
    Rs[[i]] <- lm(paste(Ys[i],"~MKT"),PR)
    if(i>1){
      Yhat<- cbind(Yhat,Rs[[i]]$fitted.values)
    }else{
      Yhat<- Rs[[i]]$fitted.values
    }

  }
  MU.i0 <- colMeans(Yhat)*12
  SIGMA.i0 <- cov(Yhat)*12
  for(i in(1:length(Ys))){
    SIGMA.i0[i,i]<-SIGMA.i0[i,i]+summary(Rs[[i]])$sigma^2*12
  }
}

rf<-mean(RF)
#The required expected return
TargetReturn = 0.1
minimum.variance.portfolio<-solnp(pars=W0,
                                  fun=sigma.2.p,
```

```
                                  MU.i = MU.i0, SIGMA.i = SIGMA.i0,
                                  eqfun=eqf2, eqB=c(1, TargetReturn),
                                  control=list(trace=FALSE)
)


minimum.variance.portfolio$pars

## [1]  1.24596545 -0.09996251 -0.14600294

mu.p(minimum.variance.portfolio$pars, MU.i0, SIGMA.i0)

## [1] 0.1

sigma.2.p(minimum.variance.portfolio$pars, MU.i0, SIGMA.i0)

## [1] 0.3698201

sum(minimum.variance.portfolio$pars)

## [1] 1

#######################################################
#The efficient frontier.
#######################################################
TargetReturns <- seq(0, 1, 0.01)
Min.STD <- rep(0, length(TargetReturns))
i <- 0 ;
for (TargetReturn in TargetReturns){
  i <- i+1;
  W <- solnp(pars=W0,
             fun=sigma.2.p,
             MU.i = MU.i0, SIGMA.i = SIGMA.i0,
             eqfun=eqf2, eqB=c(1, TargetReturn),
             control=list(trace=FALSE)
  )
  Min.STD[i] <- sigma.2.p(W$pars, MU.i0, SIGMA.i0)^0.5
}
plot( Min.STD, TargetReturns, type="l", lwd=2) #Plot the efficient fron
tier.

#######################################################
#Maximize the sharpe ratio.
#The Sharpe Ratio for maximization
sharpe.ratio <- function(W, MU.i, SIGMA.i, RF){
  sp <- (mu.p(W, MU.i, SIGMA.i)-RF)/sigma.2.p(W, MU.i, SIGMA.i)^.5
  return(sp)
}
#The negative Sharpe Ratio for minimization
neg.sharpe.ratio <- function(W, MU.i, SIGMA.i, RF){
  nsp <- -sharpe.ratio(W, MU.i, SIGMA.i, RF)
```

```
  return(nsp)
}

#The sum of weights for the maximixatio of sharpe ratio
sharpe.ratio.eqf1 <- function(W, MU.i, SIGMA.i, RF){
  return (eqf1(W, MU.i, SIGMA.i))
}

neg.sharpe.ratio(W0, MU.i = MU.i0, SIGMA.i = SIGMA.i0, RF=rf)

## [1] -1.598753

sharpe.ratio(W0, MU.i = MU.i0, SIGMA.i = SIGMA.i0, RF=rf)

## [1] 1.598753

sp.W <- solnp(pars=W0,
              fun=neg.sharpe.ratio,
              MU.i = MU.i0, SIGMA.i = SIGMA.i0, RF=rf,
              eqfun=sharpe.ratio.eqf1, eqB=1,
              control=list(trace=FALSE)
)
sp.W

## $pars
## [1] 0.1814743 0.2487040 0.5698217
##
## $convergence
## [1] 0
##
## $values
## [1] -1.598753 -1.761323 -1.761323
##
## $lagrange
##             [,1]
## [1,] -0.02240499
##
## $hessian
##             [,1]       [,2]        [,3]
## [1,]  3.21688521 -0.8341789 -0.02958117
## [2,] -0.83417887  5.7551417 -2.42196624
## [3,] -0.02958117 -2.4219662  2.08093293
##
## $ineqx0
## NULL
##
## $nfuneval
## [1] 47
##
## $outer.iter
## [1] 2
```

```
##
## $elapsed
## Time difference of 0.006981134 secs
##
## $vscale
## [1] 1.76132295 0.00000001 0.18147412 0.24870576 0.56982012
```

```
sp<-sharpe.ratio(sp.W$pars, MU.i0, SIGMA.i0, rf) #計算最大 sharpe ratio
sp.W.M <- mu.p(sp.W$pars, MU.i0, SIGMA.i0) #最大 sharpe ratio 的 expected
 return
sp.W.D <- sigma.2.p(sp.W$pars, MU.i0, SIGMA.i0)^.5 #最大 sharpe ratio 的
STD

sp.W.M #最大 sharpe ratio 的 expected return
```

```
## [1] 0.6488107
```

```
sp.W.D #最大 sharpe ratio 的 STD
```

```
## [1] 0.3637384
```

```
ER = rf + sp*Min.STD; #計算 CAL 線函數值
CAL <- lm(ER~Min.STD) #建構 CAL 線函數模型
points(sp.W.D,sp.W.M,pch=16,col="red") #標出最大 sharpe ratio 的位置(紅)
abline(CAL,lwd=1, col="blue") #劃出 CAL 線(藍)
```