

2021

Constrained Portfolio Optimisation

Moh Tahir, M.A.

Moh Tahir, M.A. (2021) 'Constrained Portfolio Optimisation', The Plymouth Student Scientist, 14(2), pp. 429-464.

<http://hdl.handle.net/10026.1/18510>

The Plymouth Student Scientist
University of Plymouth

All content in PEARL is protected by copyright law. Author manuscripts are made available in accordance with publisher policies. Please cite only the published version using the details provided on the item record or document. In the absence of an open licence (e.g. Creative Commons), permissions for further reuse of content should be sought from the publisher or author.

R Code

The following R code is for solving the unconstrained portfolio optimisation using quadratic programming techniques.

```
library(zoo)
library(PerformanceAnalytics)
library(PortfolioAnalytics)
library(quadprog)
library(quantmod)
#Assets tickers
tickers <- c("FB", "AAPL", "AMZN", "NFLX", "TSLA","GOOG",
            "BA","PYPL","IBM","GS","AIG","MA","JMP","V"
            ,"BABA","INTC","C","AMD","WIX","ACN","BLK"
            ,"WFC","Bk","USB","SIVB","PNC","UBS","WBS",
            "TFC","COO","LH","MD","MDT","MRK","WAT","SYK"
            ,"SM","EOG","SSL","RES","PSX","APA","D","MMM"
            ,"CE","BB","BR","MSFT","EA","PTC","AON","VC"
            ,"HIG","L","CB")

portfolioprices <- NULL
for(ticker in tickers){
  portfolioprices <-
    cbind(portfolioprices,
          getSymbols.yahoo(ticker,from='2016-01-03',
                          to='2020-10-05',
                          periodicity='daily',
                          auto.assign=FALSE)[,4])
}
#portfolio prices
portfolioprices2 <-data.matrix(as.data.frame(portfolioprices))
#The return matrix.
retmatrix <- data.frame(matrix(NA, nrow = nrow(portfolioprices)-1,
                              ncol = ncol(portfolioprices)))
for(i in 1:(nrow(portfolioprices2)-1) ) {
  for (j in 1:ncol(portfolioprices2)){
    retmatrix[i,j]<-(portfolioprices2[i+1,j]-
                    portfolioprices2[i,j])/portfolioprices2[i,j]
  }
}

covariance <- cov(retmatrix) #The co-variance matrix.
print(covariance)
n <- ncol(retmatrix)
#Matrix defining the constraints.
Amat <- cbind(1, colMeans(retmatrix), diag(n))
#vector appearing in the quadratic function to be minimised.
bvec <- c(1, rep(0, n))

# create the objective matrix
Dmat <- 2*covariance

#Specify the number of equality constraints
```

```
meq <- 1
risk.Vector <- rep(0,100)
count <- 1
minret <- 0 # set min return
# setting max return to be max of average returns
maxret <- max(colMeans(retmatrix))-10^-10
number_points <- 100
return_vals <- seq(from = minret,
                   to = maxret,length.out = number_points)

# Sets of weights
weights_list <- matrix(0, number_points, n)

for ( return_val in return_vals ) {
  #dvec <- colMeans(retmatrix) * return_val
  dvec <- rep(0,n)*return_val
  b.Vector <- c(1, return_val, rep(0,n))
  out <- solve.QP( Dmat, dvec, Amat, bvec = b.Vector, meq )
  risk.Vector[count] <- out$value
  weights_list[count,] <- out$solution
  count <- count + 1
  count
}
#Plotting the efficient frontier.
plot(risk.Vector, return_vals,
     xlab="Risk",ylab="Return",col='red')
```