```r
# Portfolio Theory
## Load data
library(readr)
data=read_csv("full_close.csv")
```

```
## Rows: 1218 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (4): PFE_Close, JNJ_Close, MRK_Close, NVS_Close
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
prices=data
n=dim(prices)[1]
returns =(prices[2:n,]/prices[1:(n-1),]-1) # returns by day
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
## mean, covariance, standard deviation of PRICES
mean.p = colMeans(prices)
cov.p = cov(prices)
sd.p = sqrt(diag(cov.p))
## mean, covariance, standard deviation of RETURN
mean.r = colMeans(returns)
cov.r = cov(returns)
sd.r = sqrt(diag(cov.r))
## Skewness Coefficients, Kurtosis Coefficients and beta of PRICES
list=as.list(prices)
sk.p=unlist(lapply(list,function(data){
  (sum((data-mean(data))^3/(sd(data))^3))/length(data)
}))
kt.p=unlist(lapply(list,function(data){
  (sum((data-mean(data))^4/(sd(data))^4))/length(data)
}))
## Info Table
info=cbind(mean.r,sd.r,sk.p,kt.p)
colnames(info)=c("mean","sd","skewness","kurtosis")
sd.r[order(sd.r,decreasing = F)]
```

```
##  NVS_Close  JNJ_Close  MRK_Close  PFE_Close
## 0.01294151 0.01330685 0.01476153 0.01665709
```

```r
##Sharpe's Ratio (using sample mean &sd in month)
rf.m=1.73/100/365 #daily risk free
sharpes.ratio=(mean.r-rf.m)/sd.r
sort(sharpes.ratio,decreasing = T)
```

```
##   MRK_Close   PFE_Close   JNJ_Close   NVS_Close
## 0.038713081 0.021655465 0.016553030 0.007023985
```

```r
cat("\n The biggest sharpe ratio is MRK:",max(sharpes.ratio))
```

```
##
##  The biggest sharpe ratio is MRK: 0.03871308
```

```r
library(quadprog)
library(Ecdat)
```

```
## Loading required package: Ecfun
##
## Attaching package: 'Ecfun'
##
## The following object is masked from 'package:base':
##
##     sign
##
##
## Attaching package: 'Ecdat'
##
## The following object is masked from 'package:datasets':
##
##     Orange
```

```r
efficient.portfolio <-
function(er, cov.mat, target.return, shorts=TRUE)
{
  # compute minimum variance portfolio subject to target return
  #
  # inputs:
  # er                       N x 1 vector of expected returns
  # cov.mat                N x N covariance matrix of returns
  # target.return     scalar, target expected return
  # shorts           logical, allow shorts is TRUE
  #
  # output is portfolio object with the following elements
  # call                     original function call
  # er                       portfolio expected return
  # sd                       portfolio standard deviation
  # weights             N x 1 vector of portfolio weights
  #
  call <- match.call()
  #
  # check for valid inputs
```

```
  #
  asset.names <- names(er)
  er <- as.vector(er)                    # assign names if none exist
  N <- length(er)
  cov.mat <- as.matrix(cov.mat)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semidefinite
  #
  # compute efficient portfolio
  #
  if(shorts==TRUE){
    ones <- rep(1, N)
    top <- cbind(2*cov.mat, er, ones)
    bot <- cbind(rbind(er, ones), matrix(0,2,2))
    A <- rbind(top, bot)
    b.target <- as.matrix(c(rep(0, N), target.return, 1))
    x <- solve(A, b.target)
    w <- x[1:N]
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    Amat <- cbind(rep(1,N), er, diag(1,N))
    bvec <- c(1, target.return, rep(0,N))
    result <- solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=2)
    w <- round(result$solution, 6)
  } else {
    stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }
  #
  # compute portfolio expected returns and variance
  #
  names(w) <- asset.names
  er.port <- crossprod(er,w)
  sd.port <- sqrt(w %*% cov.mat %*% w)
  ans <- list("call" = call,
              "er" = as.vector(er.port),
              "sd" = as.vector(sd.port),
              "weights" = w)
  class(ans) <- "portfolio"
  ans
}

globalMin.portfolio <-function(er, cov.mat, shorts=TRUE){
  # Compute global minimum variance portfolio
  #
  # inputs:
  # er               N x 1 vector of expected returns
  # cov.mat      N x N return covariance matrix
  # shorts           logical, allow shorts is TRUE
  #
```

```
  # output is portfolio object with the following elements
  # call            original function call
  # er              portfolio expected return
  # sd              portfolio standard deviation
  # weights     N x 1 vector of portfolio weights
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  er <- as.vector(er)                    # assign names if none exist
  cov.mat <- as.matrix(cov.mat)
  N <- length(er)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semi-definite
  #
  # compute global minimum portfolio
  #
  if(shorts==TRUE){
    cov.mat.inv <- solve(cov.mat)
    one.vec <- rep(1,N)
    w.gmin <- rowSums(cov.mat.inv) / sum(cov.mat.inv)
    w.gmin <- as.vector(w.gmin)
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    Amat <- cbind(rep(1,N), diag(1,N))
    bvec <- c(1, rep(0,N))
    result <- solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=1)
    w.gmin <- round(result$solution, 6)
  } else {
    stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }

  names(w.gmin) <- asset.names
  er.gmin <- crossprod(w.gmin,er)
  sd.gmin <- sqrt(t(w.gmin) %*% cov.mat %*% w.gmin)
  gmin.port <- list("call" = call,
            "er" = as.vector(er.gmin),
            "sd" = as.vector(sd.gmin),
            "weights" = w.gmin)
  class(gmin.port) <- "portfolio"
  gmin.port
}

efficient.frontier <- function(er, cov.mat, nport=20, alpha.min=-0.5, alpha.max=1.5, shorts=TRUE)
{
  call <- match.call()
  #
  # check for valid inputs
```

```
#
asset.names <- names(er)
er <- as.vector(er)
N <- length(er)
cov.mat <- as.matrix(cov.mat)
if(N != nrow(cov.mat))
  stop("invalid inputs")
if(any(diag(chol(cov.mat)) <= 0))
  stop("Covariance matrix not positive definite")
#
# create portfolio names
#
port.names <- rep("port",nport)
ns <- seq(1,nport)
port.names <- paste(port.names,ns)
#
# compute global minimum variance portfolio
#
cov.mat.inv <- solve(cov.mat)
one.vec <- rep(1, N)
port.gmin <- globalMin.portfolio(er, cov.mat, shorts)
w.gmin <- port.gmin$weights
if(shorts==TRUE){
  # compute efficient frontier as convex combinations of two efficient portfolios
  # 1st efficient port: global min var portfolio
  # 2nd efficient port: min var port with ER = max of ER for all assets
  er.max <- max(er)
  port.max <- efficient.portfolio(er,cov.mat,er.max)
  w.max <- port.max$weights
  a <- seq(from=alpha.min,to=alpha.max,length=nport) # convex combinations
  we.mat <- a %o% w.gmin + (1-a) %o% w.max             # rows are efficient portfolios
  er.e <- we.mat %*% er                                        # expected returns of efficient po
  er.e <- as.vector(er.e)
} else if(shorts==FALSE){
  we.mat <- matrix(0, nrow=nport, ncol=N)
  we.mat[1,] <- w.gmin
  we.mat[nport, which.max(er)] <- 1
  er.e <- as.vector(seq(from=port.gmin$er, to=max(er), length=nport))
  for(i in 2:(nport-1))
    we.mat[i,] <- efficient.portfolio(er, cov.mat, er.e[i], shorts)$weights
} else {
  stop("shorts needs to be logical. For no-shorts, shorts=FALSE.")
}

names(er.e) <- port.names
cov.e <- we.mat %*% cov.mat %*% t(we.mat) # cov mat of efficient portfolios
sd.e <- sqrt(diag(cov.e))                          # std of efficient portfolios
sd.e <- as.vector(sd.e)
names(sd.e) <- port.names
dimnames(we.mat) <- list(port.names,asset.names)
#
# summarize results
#
```

```
  ans <- list("call" = call,
              "er" = er.e,
              "sd" = sd.e,
              "weights" = we.mat)
  class(ans) <- "Markowitz"
  ans
}

tangency.portfolio <- function(er,cov.mat,risk.free, shorts=TRUE)
{
  call <- match.call()
  #
  # check for valid inputs
  #
  asset.names <- names(er)
  if(risk.free < 0)
    stop("Risk-free rate must be positive")
  er <- as.vector(er)
  cov.mat <- as.matrix(cov.mat)
  N <- length(er)
  if(N != nrow(cov.mat))
    stop("invalid inputs")
  if(any(diag(chol(cov.mat)) <= 0))
    stop("Covariance matrix not positive definite")
  # remark: could use generalized inverse if cov.mat is positive semi-definite
  #
  # compute global minimum variance portfolio
  #
  gmin.port <- globalMin.portfolio(er, cov.mat, shorts=shorts)
  if(gmin.port$er < risk.free)
    stop("Risk-free rate greater than avg return on global minimum variance portfolio")
  #
  # compute tangency portfolio
  #
  if(shorts==TRUE){
    cov.mat.inv <- solve(cov.mat)
    w.t <- cov.mat.inv %*% (er - risk.free) # tangency portfolio
    w.t <- as.vector(w.t/sum(w.t))          # normalize weights
  } else if(shorts==FALSE){
    Dmat <- 2*cov.mat
    dvec <- rep.int(0, N)
    er.excess <- er - risk.free
    Amat <- cbind(er.excess, diag(1,N))
    bvec <- c(1, rep(0,N))
    result <- quadprog::solve.QP(Dmat=Dmat,dvec=dvec,Amat=Amat,bvec=bvec,meq=1)
    w.t <- round(result$solution/sum(result$solution), 6)
  } else {
    stop("Shorts needs to be logical. For no-shorts, shorts=FALSE.")
  }

  names(w.t) <- asset.names
  er.t <- crossprod(w.t,er)
  sd.t <- sqrt(t(w.t) %*% cov.mat %*% w.t)
```

```
  tan.port <- list("call" = call,
            "er" = as.vector(er.t),
            "sd" = as.vector(sd.t),
            "weights" = w.t)
  class(tan.port) <- "portfolio"
  return(tan.port)
}
plot.portfolio <- function(object, ...)
{
  asset.names <- names(object$weights)
  barplot(object$weights, names=asset.names,
      xlab="Assets", ylab="Weight", main="Portfolio Weights", ...)
  invisible()
}

##SHORT SELL IS NOT ALLOWED
##mvp with no short sell
mvp.noshort=globalMin.portfolio(mean.r,cov.r,shorts=FALSE)
mvp.ns.sd=mvp.noshort$sd
mvp.ns.er=mvp.noshort$er
mvp.ns.w=mvp.noshort$weights
##Tangency Portfolio
tangency.noshort=tangency.portfolio(mean.r,cov.r,risk.free = rf.m,shorts = FALSE)
tg.ns.er=tangency.noshort$er #tangency return
tg.ns.sd=tangency.noshort$sd
tg.ns.w=tangency.noshort$weights
##Efficient Portfolio Frontier
eff.front.noshort=efficient.frontier(mean.r,cov.r,nport=50, shorts = FALSE)
ef.ns.er=eff.front.noshort$er
ef.ns.sd=eff.front.noshort$sd
ef.ns.w=eff.front.noshort$weights
##Data Table
library(formattable)
noshort.table=data.frame("risk.free.day"=rbind(round(rf.m,5),0) ,
                         "mvp day"=rbind(mvp.ns.er,mvp.ns.sd),
                         "tangency.day"=rbind(tg.ns.er,tg.ns.sd),
                         "mvp year"=rbind(mvp.ns.er*365,mvp.ns.sd*sqrt(365)),
                         "Tangency year"=rbind(tg.ns.er*365,tg.ns.sd*sqrt(365)),
                         "risk.free.year"=rbind(rf.m*365,0),
                         row.names = c("return","risk"))
formattable(noshort.table)
```

risk.free.day

mvp.day

tangency.day

mvp.year

Tangency.year

risk.free.year

return

5e-05

0.0002801824

0.0006059602

0.1022666

0.2211755

0.0173

risk

0e+00

0.0112640969

0.0144013630

0.2152003

0.2751377

0.0000

```
##Value at Risk
s=100000
## t= one month
##Loss~N(-mean.mvp,sd.mvp), assuming mvp is normal
VaR.ns.mvp=(-mvp.ns.er+mvp.ns.sd*qnorm(0.95))*s
VaR.r=(-mean.r+sd.r*qnorm(0.95))*s
##Sharpe Ratio
sharpe.ns = ( ef.ns.er- rf.m) / ef.ns.sd
## compute Sharpe's ratios
(tg.ns.sharpe=max(sharpe.ns))
```

```
## [1] 0.03878502
```

```
assets.ns.sharpe=(mean.r -rf.m)/sd.r

sort(mvp.ns.w,decreasing = T)
```

```
## NVS_Close JNJ_Close MRK_Close PFE_Close
##  0.424091  0.344434  0.165458  0.066017
```

```
sort(tg.ns.w,decreasing = T)
```

```
## MRK_Close PFE_Close JNJ_Close NVS_Close
##  0.938782  0.061218  0.000000  0.000000
```

```
sort(sd.r,decreasing = F)
```

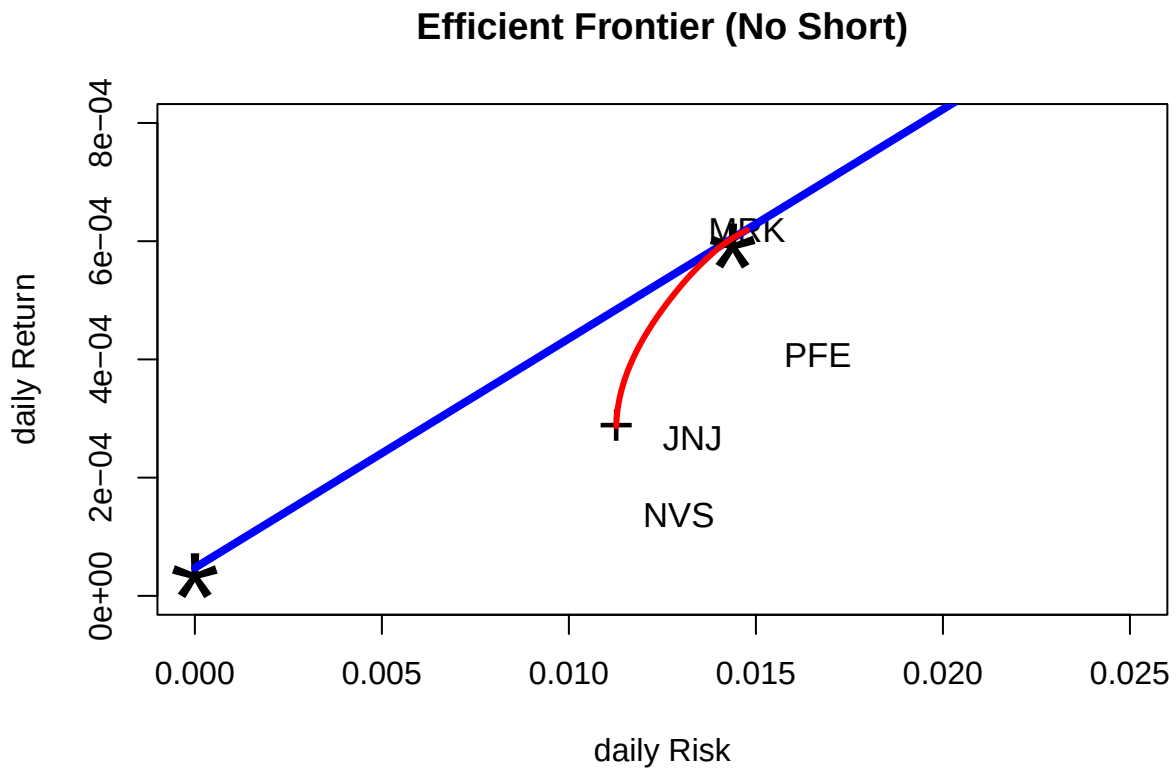
```
##  NVS_Close  JNJ_Close  MRK_Close  PFE_Close
## 0.01294151 0.01330685 0.01476153 0.01665709
```

```
##SHORT SELL IS NOT ALLOWED
##Portfolio Plot

plot(ef.ns.sd,ef.ns.er,type="l",main="Efficient Frontier (No Short)",
     xlab="daily Risk", ylab="daily Return",
     ylim = c(0,0.0008),xlim=c(0,0.025),lty=3)

text(sqrt(cov.r[1,1]),mean.r[1],'PFE',cex=1.1)
text(sqrt(cov.r[2,2]),mean.r[2],'JNJ',cex=1.1)
text(sqrt(cov.r[3,3]),mean.r[3],'MRK',cex=1.1)
text(sqrt(cov.r[4,4]),mean.r[4],'NVS',cex=1.1)

points(0, rf.m, cex = 4, pch = "*") # show risk-free asset
sharpe = ( ef.ns.er- rf.m) / ef.ns.sd # compute Sharpe's ratios
ind = (sharpe == max(sharpe)) # Find maximum Sharpe's ratio
lines(c(0, 2), rf.m + c(0, 2) * (ef.ns.er[ind] - rf.m) / ef.ns.sd[ind], lwd = 4, lty = 1, col = "blue")
points(ef.ns.sd[ind], ef.ns.er[ind], cex = 4, pch = "*") # tangency portfolio
ind2 = (ef.ns.sd == min(ef.ns.sd)) # find minimum variance portfolio
points(ef.ns.sd[ind2], ef.ns.er[ind2], cex = 2, pch = "+") # min var portfolio
ind3 = (ef.ns.er > ef.ns.er[ind2])
lines(ef.ns.sd[ind3], ef.ns.er[ind3], type = "l", lwd = 3, col = "red") # plot efficient frontier
```




```
##SHORT SELL IS ALLOWED
##ALL VARIABLES are in MONTH
##mvp
```

```
mvp.short=globalMin.portfolio(mean.r ,cov.r,shorts=TRUE)
mvp.s.sd=mvp.short$sd
mvp.s.er=mvp.short$er
mvp.s.w=mvp.short$weights
##Tangency Portfolio
tangency.short=tangency.portfolio(mean.r ,cov.r,risk.free = rf.m,shorts = TRUE)
tg.s.er=tangency.short$er #tangency return
tg.s.sd=tangency.short$sd
tg.s.w=tangency.short$weights
##SHORT SELL IS ALLOWED
##Efficient Portfolio Frontier
eff.front.short=efficient.frontier(mean.r ,cov.r,nport=50,
                                   shorts = TRUE)
ef.s.er=eff.front.short$er
ef.s.sd=eff.front.short$sd
ef.s.w=eff.front.short$weights
##Data Table
library(formattable)
short.table=data.frame("risk.free.day"=rbind(rf.m,0) ,
                       "mvp day"=rbind(mvp.s.sd,mvp.s.sd),
                       "tangency.day"=rbind(tg.s.er,tg.s.sd),
                       "mvp year"=rbind(mvp.s.er*365,mvp.s.sd*sqrt(365)),
                       "Tangency year"=rbind(tg.s.er*365,tg.s.sd*sqrt(365)),
                       "risk.free.year"=rbind(rf.m*365,0),
                       row.names = c("return","risk")
                       )
formattable(short.table)
```

risk.free.day

mvp.day

tangency.day

mvp.year

Tangency.year

risk.free.year

return

4.739726e-05

0.0112641

0.001054884

0.1022666

0.3850328

0.0173

risk

0.000000e+00

0.0112641

0.023433565

0.2152003

0.4476976

0.0000

```
##Value at Risk
s=100000
## t= one month
##Loss~N(-mean.mvp,sd.mvp), assuming mvp is normal
VaR.s.mvp=(-mvp.s.er+mvp.s.sd*qnorm(0.95))*s
VaR.s.mvp
```

```
## [1] 1824.761
```

```
VaR.s.tg=(-tg.s.er+tg.s.sd*qnorm(0.95))*s
VaR.s.tg
```

```
## [1] 3748.99
```

```
##Sharpe Ratio
sharpe.s = ( ef.s.er- rf.m) / ef.s.sd # compute Sharpe's ratios
(tg.s.sharpe=max(sharpe.s))
```

```
## [1] 0.04217991
```

```
assets.s.sharpe=(mean.r -rf.m)/sd.r

sort(mvp.s.w,decreasing = T)
```

```
## NVS_Close JNJ_Close MRK_Close PFE_Close
## 0.4240909 0.3444337 0.1654582 0.0660172
```

```
sort(tg.s.w,decreasing = T)
```

```
##  MRK_Close  PFE_Close  JNJ_Close  NVS_Close
##  1.7647639  0.3705868 -0.2433779 -0.8919728
```

```
efficient.portfolio(scale(mean.r),cov.r,0.005,shorts=FALSE)
```

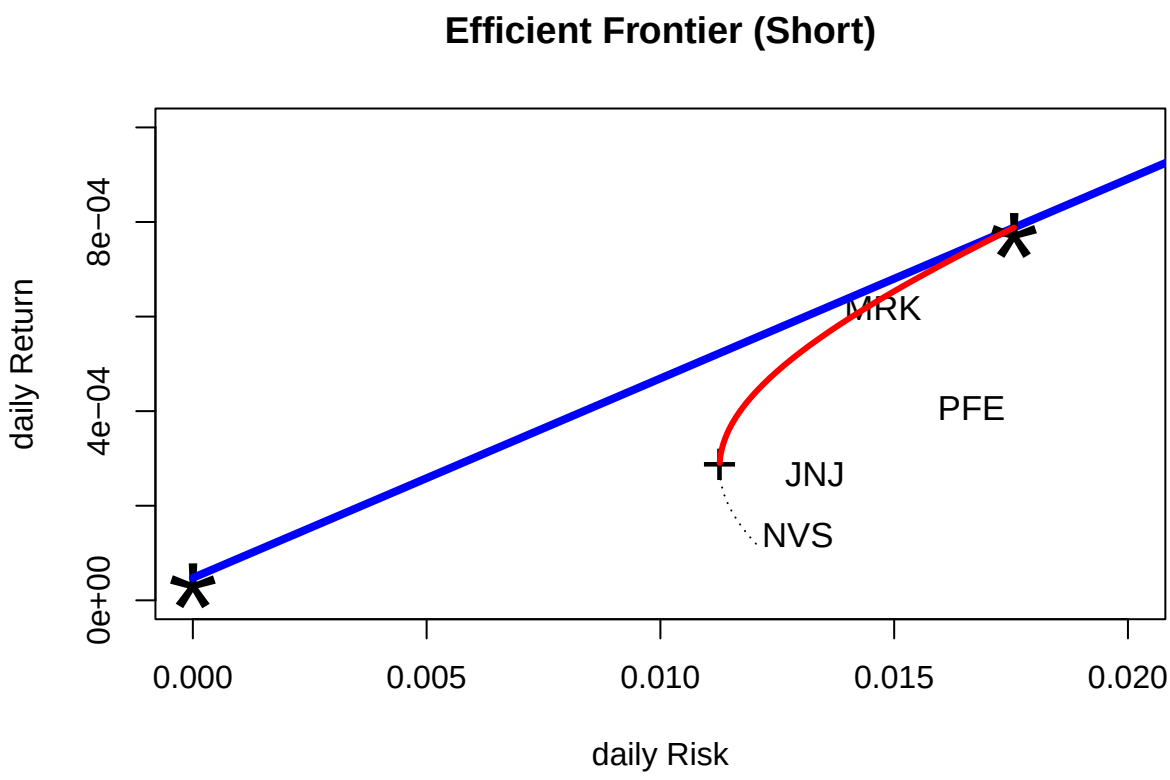
```
## $call
## efficient.portfolio(er = scale(mean.r), cov.mat = cov.r, target.return = 0.005,
##     shorts = FALSE)
##
## $er
## [1] 0.005000149
##
## $sd
## [1] 0.01145775
##
## $weights
```

```
## [1] 0.097108 0.284430 0.328715 0.289748
##
## attr(,"class")
## [1] "portfolio"
```

```
##Portfolio Plot
plot(ef.s.sd,ef.s.er,type="l",main="Efficient Frontier (Short)",
     xlab="daily Risk", ylab="daily Return",
     ylim = c(0,0.001),xlim=c(0,0.02),lty=3)

text(sqrt(cov.r[1,1]),mean.r[1],'PFE',cex=1.1)
text(sqrt(cov.r[2,2]),mean.r[2],'JNJ',cex=1.1)
text(sqrt(cov.r[3,3]),mean.r[3],'MRK',cex=1.1)
text(sqrt(cov.r[4,4]),mean.r[4],'NVS',cex=1.1)

points(0, rf.m, cex = 4, pch = "*") # show risk-free asset
sharpe = ( ef.s.er- rf.m) / ef.s.sd # compute Sharpe's ratios
ind = (sharpe == max(sharpe)) # Find maximum Sharpe's ratio
lines(c(0, 2), rf.m + c(0, 2) * (ef.s.er[ind] - rf.m) / ef.s.sd[ind], lwd = 4, lty = 1, col = "blue") #
points(ef.s.sd[ind], ef.s.er[ind], cex = 4, pch = "*") # tangency portfolio
ind2 = (ef.s.sd == min(ef.s.sd)) # find minimum variance portfolio
points(ef.s.sd[ind2], ef.s.er[ind2], cex = 2, pch = "+") # min var portfolio
ind3 = (ef.s.er > ef.s.er[ind2])
lines(ef.s.sd[ind3], ef.s.er[ind3], type = "l", xlim = c(0, 1),
      ylim = c(0, 0.3), lwd = 3, col = "red") # plot efficient frontier
```



```r
# Copula
library(copula)
library(VineCopula)
```

```
##
## Attaching package: 'VineCopula'
##
## The following object is masked from 'package:copula':
##
##     pobs
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.2     v stringr 1.4.1
## v tibble  3.2.1     v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
## Gaussian, t, archimedean, clayton, gumbel
data = read.csv("full_close.csv")
cop.norm = normalCopula(dim=4)
fit.copnorm = fitCopula(cop.norm,pobs(data),method="ml")
fit.copnorm
```

```
## Call: fitCopula(cop.norm, data = pobs(data), ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 1218 4-dimensional observations.
## Copula: normalCopula
##  rho.1
## 0.4877
## The maximized loglikelihood is 662.8
## Optimization converged
```

```r
AIC(fit.copnorm)
```

```
## [1] -1323.602
```

```r
BIC(fit.copnorm)
```

```
## [1] -1318.497
```

```r
logLik(fit.copnorm)
```

```
## 'log Lik.' 662.8008 (df=1)
```

```
cop.t = tCopula(dim=4)
fit.copt = fitCopula(cop.t,pobs(data),method = "ml")
```

```
## Warning in fitCopula.ml(copula, u = data, method = method, start = start, :
## Hessian matrix not invertible: system is computationally singular: reciprocal
## condition number = 3.54329e-08
```

```
fit.copt
```

```
## Call: fitCopula(cop.t, data = pobs(data), ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 1218 4-dimensional observations.
## Copula: tCopula
##     rho.1        df
##    0.4877 2004.9547
## The maximized loglikelihood is 662.7
## Optimization converged
```

```
AIC(fit.copt)
```

```
## [1] -1321.46
```

```
BIC(fit.copt)
```

```
## [1] -1311.251
```

```
logLik(fit.copt)
```

```
## 'log Lik.' 662.7302 (df=2)
```

```
cop.gumbel = gumbelCopula(dim=4)
fit.copgumbel = fitCopula(cop.gumbel,pobs(data),method = "ml")
fit.copgumbel
```

```
## Call: fitCopula(cop.gumbel, data = pobs(data), ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 1218 4-dimensional observations.
## Copula: gumbelCopula
## alpha
## 1.333
## The maximized loglikelihood is 398.3
## Optimization converged
```

```
AIC(fit.copgumbel)
```

```
## [1] -794.5565
```

```
BIC(fit.copgumbel)
```

```
## [1] -789.4516
```

```
logLik(fit.copgumbel)
```

```
## 'log Lik.' 398.2783 (df=1)
```

```
cop.archm = archmCopula(family = "frank",dim=4)
fit.archm = fitCopula(cop.archm, pobs(data),method = "ml")
fit.archm
```

```
## Call: fitCopula(cop.archm, data = pobs(data), ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 1218 4-dimensional observations.
## Copula: frankCopula
## alpha
## 2.665
## The maximized loglikelihood is 470.7
## Optimization converged
```

```
AIC(fit.archm)
```

```
## [1] -939.3584
```

```
BIC(fit.archm)
```

```
## [1] -934.2534
```

```
logLik(fit.archm)
```

```
## 'log Lik.' 470.6792 (df=1)
```

```
cop.clayton = claytonCopula(dim=4)
fit.copclayton = fitCopula(cop.clayton,pobs(data),method = "ml")
fit.copclayton
```

```
## Call: fitCopula(cop.clayton, data = pobs(data), ... = pairlist(method = "ml"))
## Fit based on "maximum likelihood" and 1218 4-dimensional observations.
## Copula: claytonCopula
##  alpha
## 0.6453
## The maximized loglikelihood is 592.1
## Optimization converged
```

```
AIC(fit.copclayton)
```

```
## [1] -1182.146
```

```
BIC(fit.copclayton)
```

```
## [1] -1177.041
```

```
logLik(fit.copclayton)
```

```
## 'log Lik.' 592.0729 (df=1)
```