STATS170 Project 2

Takao Oba

2023-04-14

a. Refer to the lecture material and the paper "An Analytic Derivation of the Efficient Portfolio Frontier," (JFQA, Robert Merton, 1972).Compute A, B, C, D.

```
#Read your csv file:
a <- read.csv("/Users/takaooba/STATS 183/stockData.csv", sep=",", header=TRUE)
train <- a[1:60,]
test <- a[61:dim(a)[1],]

#Convert adjusted close prices into returns:
r <- (train[-1,3:ncol(train)]-train[-nrow(train),3:ncol(train)])/train[-nrow(train),3:ncol(train)]

#Compute mean vector:
means <- colMeans(r[-1]) #Without ^GSPC

#Compute variance covariance matrix:
covmat <- cov(r[-1]) #Without ^GSPC

#Compute the vector of variances:
variances <- diag(covmat)

#Compute the vector of standard deviations:
stdev <- diag(covmat)^.5

# mean vector of SP500
means_sp500 <- mean(r[,1])
stdev_sp500 <- sd(r[,1])

# one vector
ones <- rep(1,30)
```

Computing A,B,CD

```
#Compute A:
A <- t(ones) %*% solve(covmat) %*% means
A
```

```
##          [,1]
## [1,] 37.95192
```

```r
#Compute B:
B <- t(means) %*% solve(covmat) %*% means
B
```

```
##           [,1]
## [1,] 1.650272
```

```r
#Compute C:
C <- t(ones) %*% solve(covmat) %*% ones
C
```

```
##          [,1]
## [1,] 2419.79
```

```r
#Compute D:
D <- B*C - A^2
D
```

```
##           [,1]
## [1,] 2552.964
```

b. Compute the values of 1 and 2 (the two Lagrange multipliers).

```r
# Let E = 0.15
E = 0.15

#  1
lambda1 <- (C*E-A)/D
lambda1
```

```
##            [,1]
## [1,] 0.1273095
```

```r
#  2
lambda2 <- (B-A*E)/D
lambda2
```

```
##              [,1]
## [1,] -0.00158346
```

c. Suppose an investor has a prescribed expected return E. Find the composition of the efficient portfolio given the return E. Note: You need to choose a value of E.

```r
x2=as.numeric(lambda1)*solve(covmat) %*% means +
as.numeric(lambda2)* solve(covmat) %*% ones
x2
```

```
##             [,1]
## MCD   0.19958942
## NKE   0.46442167
## SBUX -0.07585661
## F    -0.45913467
## CMG   0.36504464
## LULU  0.46890154
## V     0.87187953
## JPM   4.60400324
## MA    0.97222997
## BAC  -0.39908051
## MS   -2.28280594
## WFC  -1.79754301
## JNJ  -0.04916994
## UNH   1.10506379
## PFE  -0.65788530
## CVS  -1.57980420
## CI    0.53088381
## ZTS   0.14930373
## RTX  -1.44773679
## BA    0.77503812
## LMT  -0.20382161
## DE    0.43786392
## CAT  -0.19411214
## GE   -0.34233377
## AAPL -0.22089695
## MSFT  0.81694784
## ADBE -1.07137986
## INTU -0.04519326
## NVDA  0.44402238
## CRM  -0.37843906
```

```
X2 <- solve(covmat) %*% (lambda1*means + lambda2*ones)
```

```
## Warning in lambda1 * means: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in lambda2 * ones: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

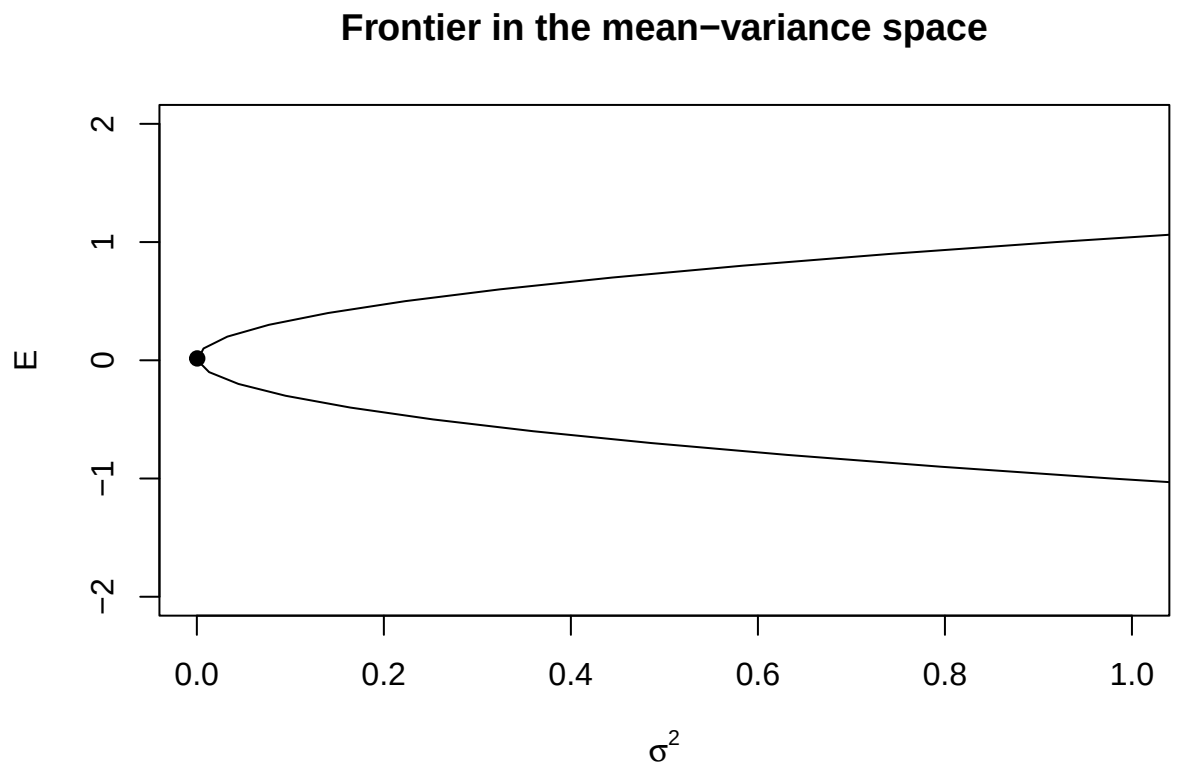
```
return2 <- t(X2) %*% means
var2 <- t(X2) %*% covmat %*% X2
sd2 <- sqrt(var2)
```

d. Use your data to plot the frontier in the mean-variance space (parabola)

```
#Give values for E:
E <- seq(-5,5,.1)

#Compute sigma2 as a function of A,B,C,D, and E:
sigma2 <- (C*E^2 - 2*A*E +B) /D
```

```
## Warning in C * E^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in 2 * A * E: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in C * E^2 - 2 * A * E + B: Recycling array of length 1 in vector-array arithmetic is deprec
##   Use c() or as.vector() instead.

## Warning in (C * E^2 - 2 * A * E + B)/D: Recycling array of length 1 in vector-array arithmetic is de
##   Use c() or as.vector() instead.
```

```
#Or plot E against sigma2:
plot(sigma2, E,type="l", xlab=expression(sigma^2), xlim = c(0, 1),   ylim = c(-2,2), main = "Frontier i

#Add the minimum risk portfolio:
points(1/C, A/C, pch=19)
```



e. Use your data to plot the frontier in the mean-standard deviation space using the hyperbola method.

```r
#Hyperbola:
#Efficient frontier:
    minvar <- 1/C
    minE <- A/C
    sdeff <- seq((minvar)^0.5, 1, by = 0.0001)
```

```
## Warning in from + (0L:n) * by: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

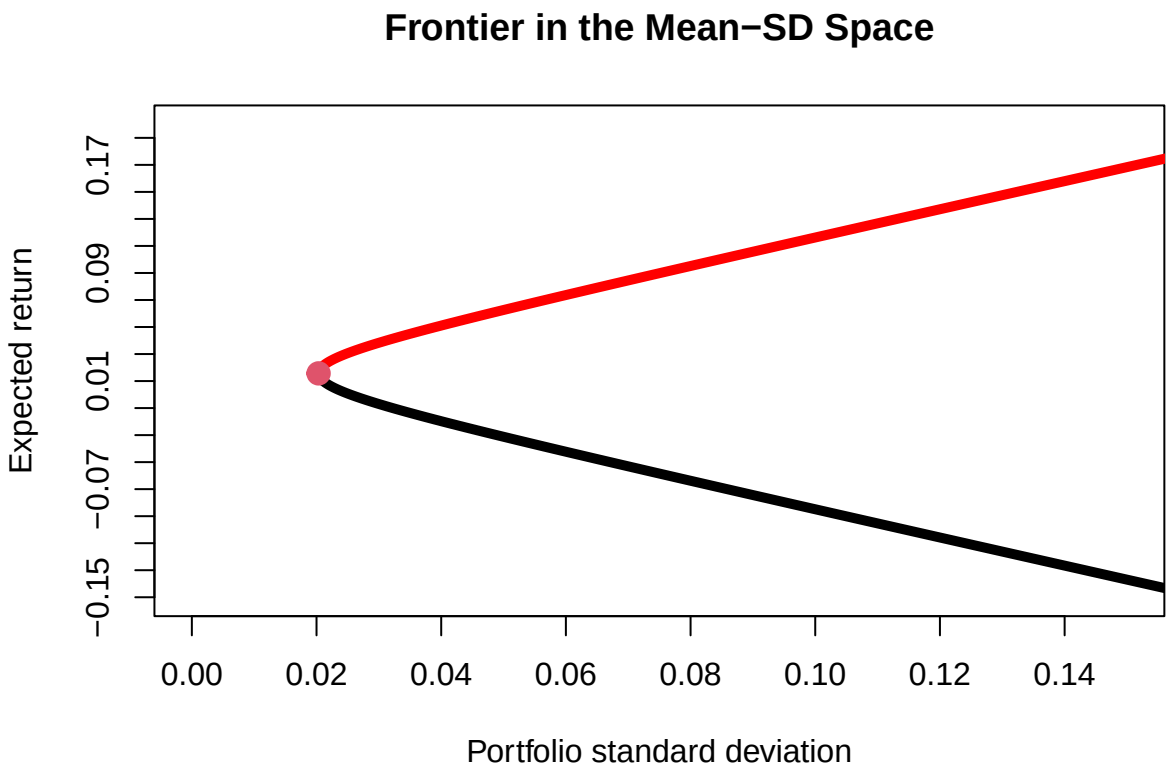
```r
#    options(warn = -1)
    y1 <- (A + sqrt(D*(C*sdeff^2 - 1)))*(1/C)
```

```
## Warning in C * sdeff^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in D * (C * sdeff^2 - 1): Recycling array of length 1 in array-vector arithmetic is deprecat
##   Use c() or as.vector() instead.

## Warning in A + sqrt(D * (C * sdeff^2 - 1)): Recycling array of length 1 in array-vector arithmetic is
##   Use c() or as.vector() instead.

## Warning in (A + sqrt(D * (C * sdeff^2 - 1))) * (1/C): Recycling array of length 1 in vector-array ari
##   Use c() or as.vector() instead.
```

```r
    y2 <- (A - sqrt(D*(C*sdeff^2 - 1)))*(1/C)
```

```
## Warning in C * sdeff^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in D * (C * sdeff^2 - 1): Recycling array of length 1 in array-vector arithmetic is deprecat
##   Use c() or as.vector() instead.

## Warning in A - sqrt(D * (C * sdeff^2 - 1)): Recycling array of length 1 in array-vector arithmetic is
##   Use c() or as.vector() instead.

## Warning in (A - sqrt(D * (C * sdeff^2 - 1))) * (1/C): Recycling array of length 1 in vector-array ari
##   Use c() or as.vector() instead.
```

```r
#    options(warn = 0)

plot(sdeff, y1, type = "n",xlim=c(0 ,0.15), ylim=c(-0.15,0.2),
     xlab="Portfolio standard deviation", ylab="Expected return",
     xaxt="no", yaxt="no", main = "Frontier in the Mean-SD Space")

axis(1, at=seq(0, 0.15, 0.02))
axis(2, at=seq(-0.15,0.2, 0.02))

    points(sdeff, y1, lwd=5,type = "l", col = "red")
    points(sdeff, y2, lwd=5,type = "l")

# min risk portfolio
points(sqrt(1/C), A/C, pch = 19, col = 10, lwd = 5)
```

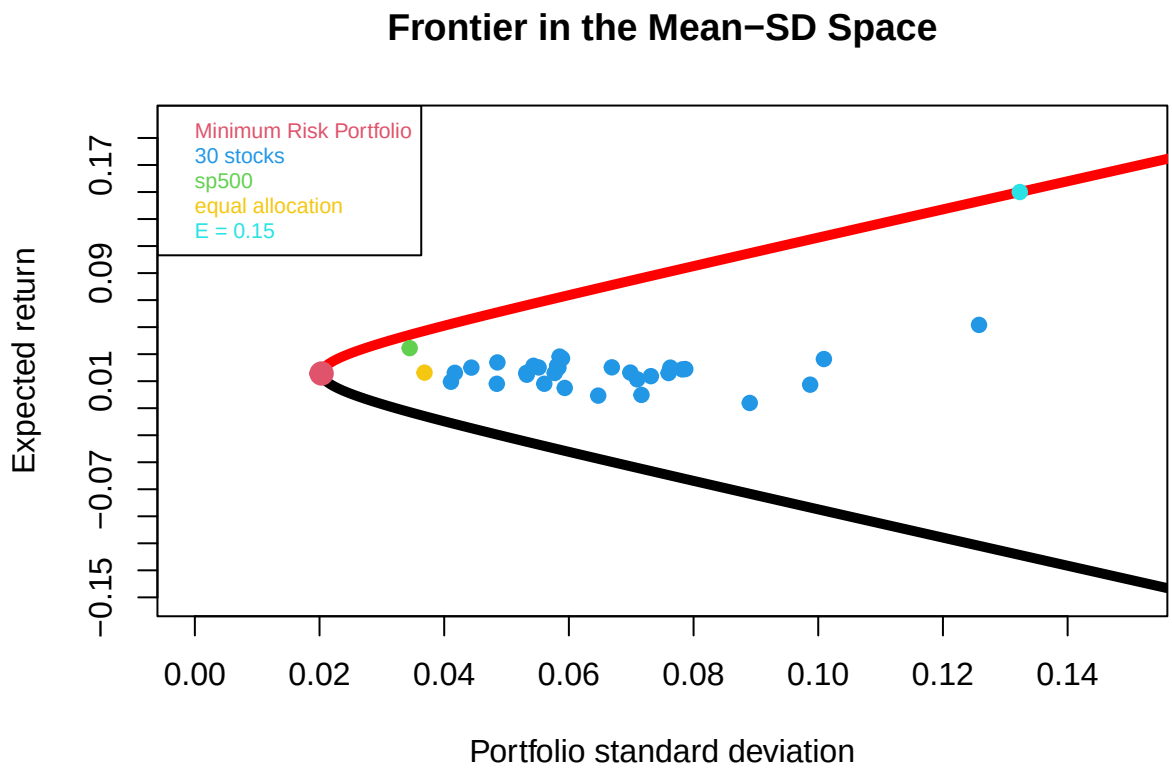


f. On the plot in (e) add the 30 stocks, the S&P500, the equal allocation portfolio, the minimum risk portfolio, and the portfolio in (c).

```
# equal allocation
eq = rep(1/30, 30)

# mean of this portfolio
r_eq = t(eq) %*% means

# sd of this portfolio
sd_eq = sqrt(t(eq) %*% covmat %*% eq)
```

```
plot(sdeff, y1, type = "n",xlim=c(0 ,0.15), ylim=c(-0.15,0.2),
     xlab="Portfolio standard deviation", ylab="Expected return",
     xaxt="no", yaxt="no", main = "Frontier in the Mean-SD Space")

axis(1, at=seq(0, 0.15, 0.02))
axis(2, at=seq(-0.15,0.2, 0.02))

    points(sdeff, y1, lwd=5,type = "l", col = "red")
    points(sdeff, y2, lwd=5,type = "l")

# min risk portfolio
```

```
points(sqrt(1/C), A/C, pch = 19, col = 10, lwd = 5)
# 30 stocks
points(stdev, means, pch = 19, col = 12)
# sp500
points(stdev_sp500, stdev_sp500, pch = 19, col = 3)
# equal allocation
points(sd_eq, r_eq, pch = 19, col = 7)
# part c
points(sd2, return2, pch = 19, col = 5)

legend(x = 'topleft',cex  = 0.65,
       legend = c("Minimum Risk Portfolio","30 stocks", "sp500", "equal allocation", "E = 0.15"),
       text.col = c(10, 12, 3, 7, 5))
```



g. Add three arbitrary portfolios on the plot of (c). You can choose any 30 weights with

$$\Sigma_{i=1}^{30} x_i = 1$$

Three arbitrary portfolios

```
# Portfolio 1, E = 0.03
E <- 0.03
lambda1_1 <- (C*E-A)/D
```

```
lambda2_1 <- (B-A*E)/D

X2_1 <- solve(covmat) %*% (lambda1_1*means + lambda2_1*ones)
```

```
## Warning in lambda1_1 * means: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in lambda2_1 * ones: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
return2_1 <- t(X2_1) %*% means
var2_1 <- t(X2_1) %*% covmat %*% X2_1
sd2_1 <- sqrt(var2_1)

# Portfolio 2, E = 0.08
E <- 0.08
lambda1_2 <- (C*E-A)/D
lambda2_2 <- (B-A*E)/D

X2_2 <- solve(covmat) %*% (lambda1_2*means + lambda2_2*ones)
```

```
## Warning in lambda1_2 * means: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in lambda2_2 * ones: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

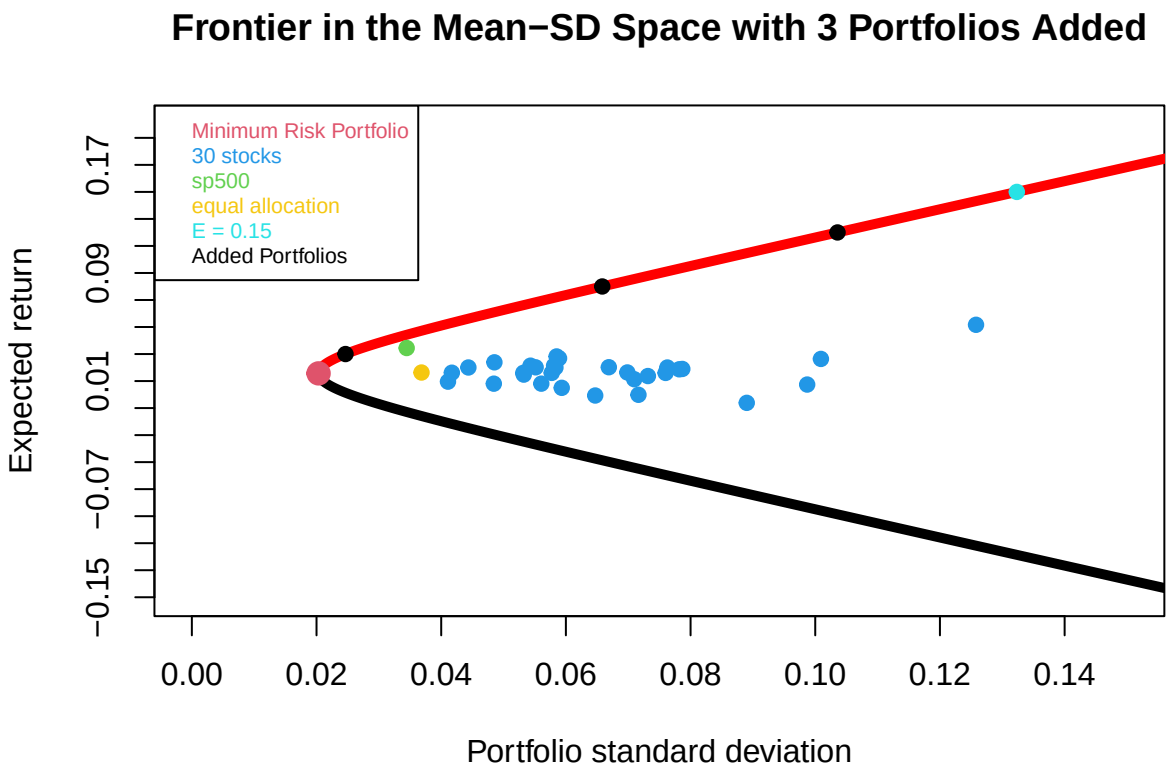
```
return2_2 <- t(X2_2) %*% means
var2_2 <- t(X2_2) %*% covmat %*% X2_2
sd2_2 <- sqrt(var2_2)

# Portfolio 3, E = 0.12
E <- 0.12
lambda1_3 <- (C*E-A)/D
lambda2_3 <- (B-A*E)/D

X2_3 <- solve(covmat) %*% (lambda1_3*means + lambda2_3*ones)
```

```
## Warning in lambda1_3 * means: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in lambda2_3 * ones: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
return2_3 <- t(X2_3) %*% means
var2_3 <- t(X2_3) %*% covmat %*% X2_3
sd2_3 <- sqrt(var2_3)
```

```
plot(sdeff, y1, type = "n",xlim=c(0 ,0.15), ylim=c(-0.15,0.2),
     xlab="Portfolio standard deviation", ylab="Expected return",
     xaxt="no", yaxt="no", main = "Frontier in the Mean-SD Space with 3 Portfolios Added")

axis(1, at=seq(0, 0.15, 0.02))
axis(2, at=seq(-0.15,0.2, 0.02))

    points(sdeff, y1, lwd=5,type = "l", col = "red")
    points(sdeff, y2, lwd=5,type = "l")

# min risk portfolio
points(sqrt(1/C), A/C, pch = 19, col = 10, lwd = 5)
# 30 stocks
points(stdev, means, pch = 19, col = 12)
# sp500
points(stdev_sp500, stdev_sp500, pch = 19, col = 3)
# equal allocation
points(sd_eq, r_eq, pch = 19, col = 7)
# part c
points(sd2, return2, pch = 19, col = 5)

points(sd2_1, return2_1, pch = 19, col = "black")
points(sd2_2, return2_2, pch = 19, col = "black")
points(sd2_3, return2_3, pch = 19, col = "black")

legend(x = 'topleft',cex  = 0.65,
       legend = c("Minimum Risk Portfolio","30 stocks", "sp500", "equal allocation", "E = 0.15", "Added
       text.col = c(10, 12, 3, 7, 5, "black"))
```


Frontier in the Mean−SD Space with 3 Portfolios Added
Minimum Risk Portfolio
30 stocks
sp500
equal allocation
E = 0.15
Added Portfolios
Expected return
0.17
0.09
0.01
−0.07
−0.15
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Portfolio standard deviation