STATS183 Project 4

Takao Oba

2023-04-26

Please answer the following questions assuming the single index model holds. Use your project data in the period 01-Jan-2015 to 01-Jan-2020.

```
#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)
```

1. Compute estimates for α_i, β_i, $\sigma^2_{\epsilon_i}$, $i = 1, 2, \ldots, 30$ by regressing each stock's return on the S&P500.

```
index <- r[,1]
rest <- r[,-1]
```

```r
alpha <- c()
beta <- c()
variance_epsilon <- c()



for (i in 1:(dim(rest)[2])){
  regress_model <- lm(rest[,i] ~ index)

  alpha[i] <- regress_model$coef[1]
  beta[i] <- regress_model$coef[2]
  variance_epsilon[i] <- ((summary(regress_model))$sigma)^2

}

head(alpha)
```

```
## [1]  0.012302330  0.008728258  0.010030614 -0.009920379  0.000885200
## [6]  0.019366474
```

```r
head(beta)
```

```
## [1] 0.4415918 0.8325651 0.5401055 1.0498264 0.7424922 0.7979014
```

```r
head(variance_epsilon)
```

```
## [1] 0.001534396 0.002553086 0.002532305 0.002928348 0.009247295 0.009592057
```

2. Construct the 30 × 30 variance covariance matrix based on the single index model.

```r
vcv_matrix <- matrix(rep(0), nrow = 30, ncol = 30)

market_variance <- var(index)

for (i in 1:30) {
  for (j in 1:30) {
    if (i == j) {
      vcv_matrix[i, j] <- variance_epsilon[i] + beta[i]^2 * market_variance
    } else {
      vcv_matrix[i, j] <- beta[i] * beta[j] * market_variance
    }
  }
}

head(vcv_matrix)
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 0.0017659698 0.0004366025 0.0002832348 0.0005505357 0.0003893677
## [2,] 0.0004366025 0.0033762439 0.0005340030 0.0010379648 0.0007341031
```

```
## [3,] 0.0002832348 0.0005340030 0.0028787256 0.0006733533 0.0004762307
## [4,] 0.0005505357 0.0010379648 0.0006733533 0.0042371740 0.0009256703
## [5,] 0.0003893677 0.0007341031 0.0004762307 0.0009256703 0.0099019773
## [6,] 0.0004184246 0.0007888862 0.0005117699 0.0009947493 0.0007035388
##             [,6]         [,7]         [,8]         [,9]       [,10]
## [1,] 0.0004184246 0.0004820628 0.0006110596 0.0005306590 0.000835182
## [2,] 0.0007888862 0.0009088679 0.0011520749 0.0010004899 0.001574629
## [3,] 0.0005117699 0.0005896050 0.0007473793 0.0006490424 0.001021501
## [4,] 0.0009947493 0.0011460408 0.0014527137 0.0012615719 0.001985535
## [5,] 0.0007035388 0.0008105401 0.0010274352 0.0008922497 0.001404274
## [6,] 0.0103480984 0.0008710273 0.0011041084 0.0009588346 0.001509070
##            [,11]        [,12]        [,13]        [,14]        [,15]
## [1,] 0.0006996855 0.0005743478 0.0003628197 0.0003541182 0.0003434356
## [2,] 0.0013191678 0.0010828594 0.0006840502 0.0006676446 0.0006475039
## [3,] 0.0008557766 0.0007024776 0.0004437602 0.0004331175 0.0004200517
## [4,] 0.0016634102 0.0013654361 0.0008625560 0.0008418693 0.0008164728
## [5,] 0.0011764508 0.0009657079 0.0006100448 0.0005954140 0.0005774523
## [6,] 0.0012642444 0.0010377747 0.0006555699 0.0006398473 0.0006205452
##            [,16]        [,17]        [,18]        [,19]        [,20]
## [1,] 0.0004732078 0.0004098768 0.0004197408 0.0006477792 0.0006667979
## [2,] 0.0008921729 0.0007727705 0.0007913677 0.0012213050 0.0012571623
## [3,] 0.0005787745 0.0005013152 0.0005133797 0.0007922906 0.0008155521
## [4,] 0.0011249891 0.0009744282 0.0009978784 0.0015400097 0.0015852241
## [5,] 0.0007956512 0.0006891666 0.0007057519 0.0010891755 0.0011211535
## [6,] 0.0008550274 0.0007405963 0.0007584192 0.0011704561 0.0012048205
##            [,21]        [,22]        [,23]        [,24]        [,25]
## [1,] 0.0004884460 0.0006022310 0.0007657435 0.0006249678 0.0006780346
## [2,] 0.0009209026 0.0011354297 0.0014437117 0.0011782970 0.0012783477
## [3,] 0.0005974122 0.0007365812 0.0009365713 0.0007643903 0.0008292956
## [4,] 0.0011612160 0.0014317249 0.0018204543 0.0014857786 0.0016119379
## [5,] 0.0008212728 0.0010125908 0.0012875206 0.0010508204 0.0011400469
## [6,] 0.0008825610 0.0010881563 0.0013836030 0.0011292388 0.0012251239
##            [,26]        [,27]        [,28]       [,29]        [,30]
## [1,] 0.0006079572 0.0005738613 0.0005408096 0.001061863 0.0006339021
## [2,] 0.0011462258 0.0010819423 0.0010196275 0.002002007 0.0011951415
## [3,] 0.0007435849 0.0007018826 0.0006614574 0.001298751 0.0007753177
## [4,] 0.0014453382 0.0013642796 0.0012857035 0.002524439 0.0015070187
## [5,] 0.0010222189 0.0009648900 0.0009093168 0.001785416 0.0010658425
## [6,] 0.0010985028 0.0010368958 0.0009771754 0.001918654 0.0011453820
```

3. Answer the same question as in project 2, part (e) using the new inputs from (1) above. Draw the frontier on the same plot as in project 2. Now you will have two frontiers, one using the historical variance covariance matrix (project 2) and one using the variance covariance matrix with inputs from the single index model.

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
```

```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)
```

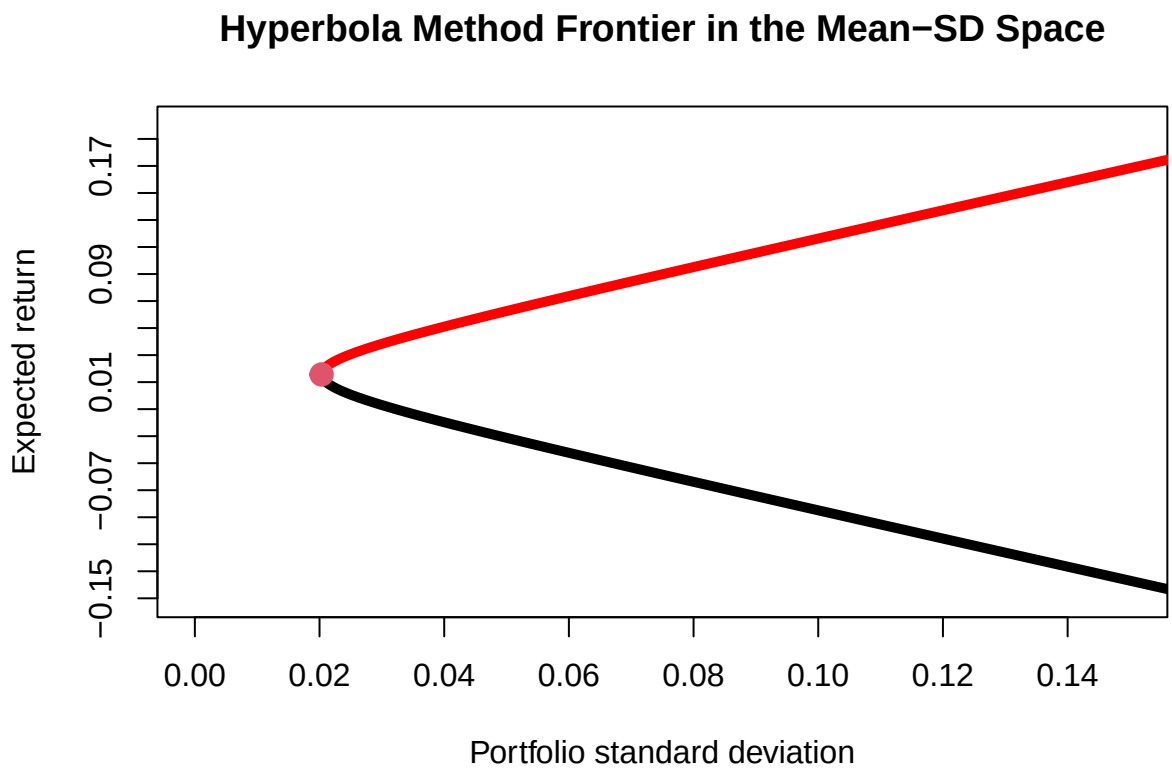
```
## Warning in C * sdeff^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## 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 i
##   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 ar
##   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 = "Hyperbola Method 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)
```



Computing A,B,CD

```r
#Compute A:
A2 <- t(ones) %*% solve(vcv_matrix) %*% means
A2
```

```
##          [,1]
## [1,] 21.57847
```

```r
#Compute B:
B2 <- t(means) %*% solve(vcv_matrix) %*% means
B2
```

```
##          [,1]
## [1,] 1.075348
```

```r
#Compute C:
C2 <- t(ones) %*% solve(vcv_matrix) %*% ones
C2
```

```
##          [,1]
## [1,] 1648.679
```

```r
#Compute D:
D2 <- B2*C2 - A2^2
D2
```

```
##          [,1]
## [1,] 1307.273
```

```r
#Single Index Model:
#Efficient frontier:
    minvar2 <- 1/C2
    minE2 <- A/C2
    sdeff2 <- seq((minvar2)^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.
```

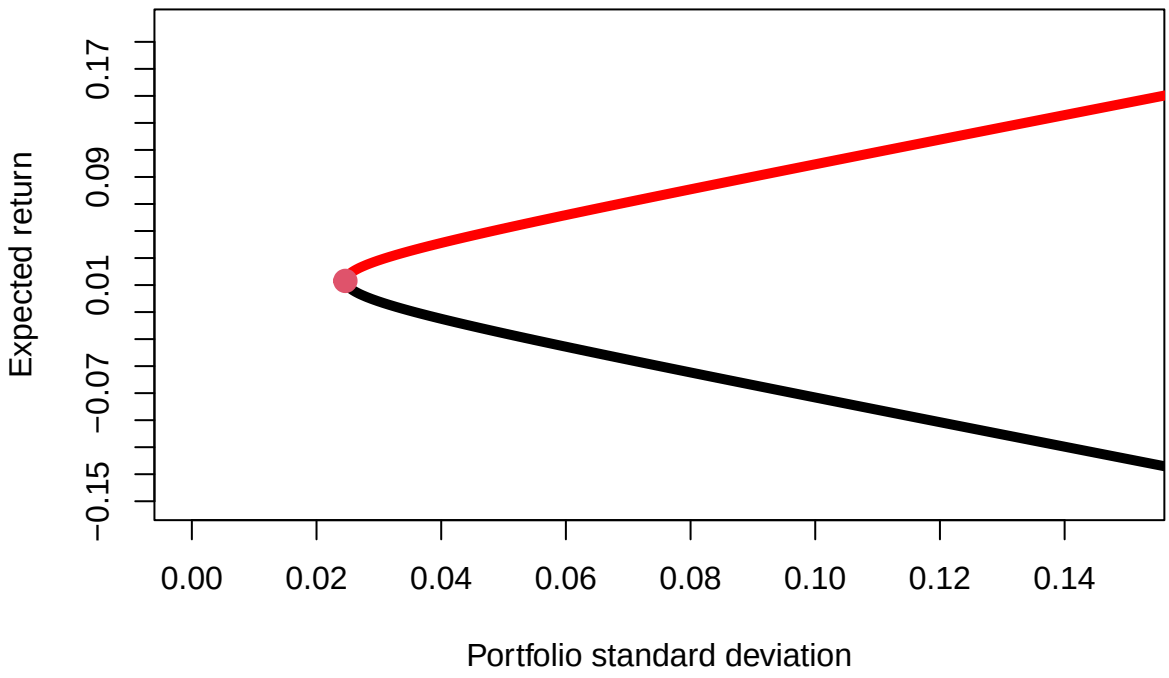
```r
#    options(warn = -1)
    y12 <- (A2 + sqrt(D2*(C2*sdeff2^2 - 1)))*(1/C2)
```

```
## Warning in C2 * sdeff2^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in D2 * (C2 * sdeff2^2 - 1): Recycling array of length 1 in array-vector arithmetic is depre
##   Use c() or as.vector() instead.

## Warning in A2 + sqrt(D2 * (C2 * sdeff2^2 - 1)): Recycling array of length 1 in array-vector arithmeti
##   Use c() or as.vector() instead.

## Warning in (A2 + sqrt(D2 * (C2 * sdeff2^2 - 1))) * (1/C2): Recycling array of length 1 in vector-arra
##   Use c() or as.vector() instead.
```

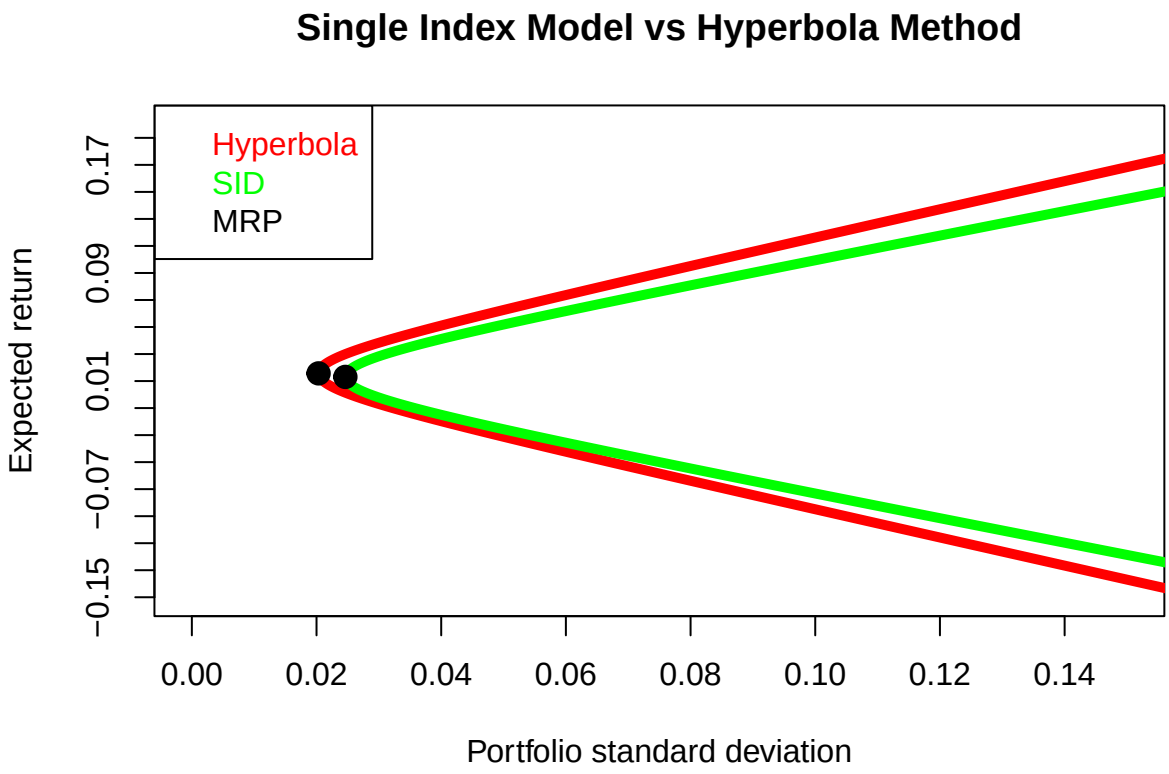
```
    y22 <- (A2 - sqrt(D2*(C2*sdeff2^2 - 1)))*(1/C2)
```

```
## Warning in C2 * sdeff2^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in D2 * (C2 * sdeff2^2 - 1): Recycling array of length 1 in array-vector arithmetic is depre
##   Use c() or as.vector() instead.

## Warning in A2 - sqrt(D2 * (C2 * sdeff2^2 - 1)): Recycling array of length 1 in array-vector arithmeti
##   Use c() or as.vector() instead.

## Warning in (A2 - sqrt(D2 * (C2 * sdeff2^2 - 1))) * (1/C2): Recycling array of length 1 in vector-arra
##   Use c() or as.vector() instead.
```

```
#    options(warn = 0)

plot(sdeff2, y12, 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 = "Single Index Model 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(sdeff2, y12, lwd=5,type = "l", col = "red")
    points(sdeff2, y22, lwd=5,type = "l")

# min risk portfolio
points(sqrt(1/C2), A2/C2, pch = 19, col = 10, lwd = 5)
```




```
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 = "Single Index Model vs Hyperbola Method")

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", col = "red")

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



points(sdeff2, y12, lwd=5,type = "l", col = "green")
points(sdeff2, y22, lwd=5,type = "l", col = "green")

# min risk portfolio
points(sqrt(1/C2), A2/C2, pch = 19, col = "black", lwd = 5)

legend(x = 'topleft',
       legend = c("Hyperbola", "SID", "MRP"),
       text.col = c("red", "green", "black"))
```


Single Index Model vs Hyperbola Method
Hyperbola
SID
MRP
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