C183 Stocks Project

Sean Nagler

4/6/2020

Part 1:

a)

```
#Read your csv file:
rm(list = ls())
a <- read.csv("stockData.csv", sep=",", header=TRUE)
```

b)

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

```
## [1] 59 31
```

c)

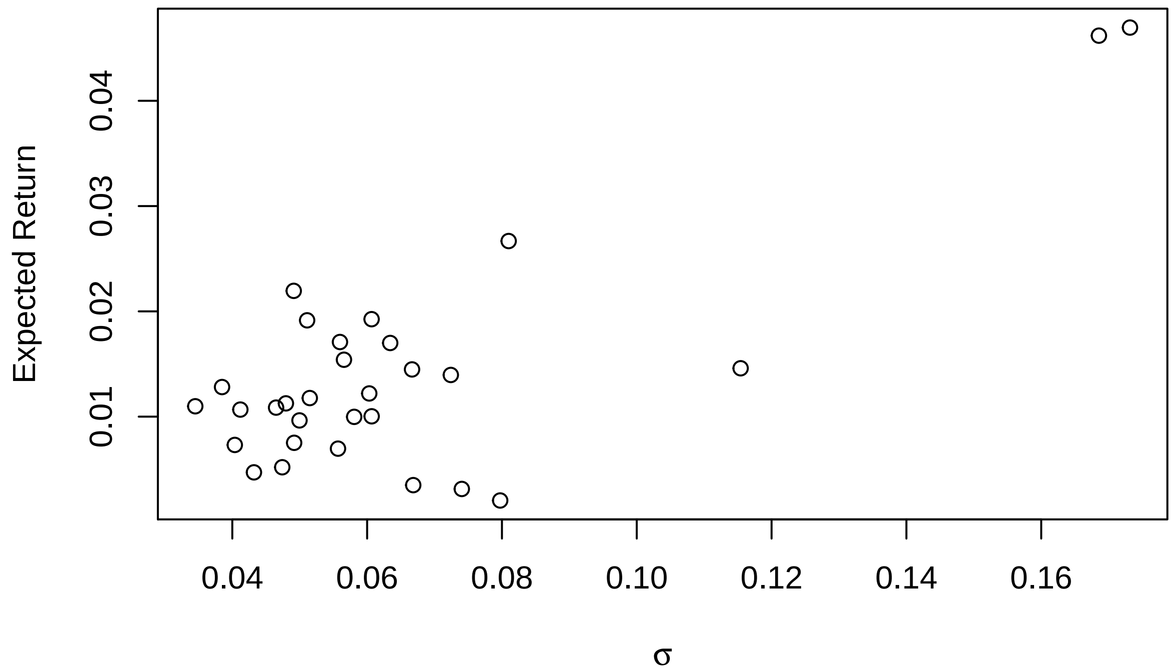
```
#Compute mean vector:
means <- colMeans(r[-1]) #Without ^GSPC
means
```

```
##        AAPL         SNE        ORCL        INTC        MSFT         XOM
## 0.013958585 0.014593472 0.007515822 0.010031562 0.016996733 0.004707575
##         CVX         SNP          BP        EQNR       GOOGL         DIS
## 0.006960666 0.003131327 0.003494190 0.002037201 0.019256606 0.019148222
##          VZ           T        NFLX        AMZN         NKE        TSLA
## 0.010857926 0.010671964 0.046953784 0.026679650 0.015405203 0.046186235
##        SBUX          TM         WMT          KO         PEP         BTI
## 0.017088705 0.011765535 0.005192586 0.007311790 0.010985854 0.012202735
##         TGT         JNJ         UNH         MRK         NVS         BMY
## 0.009979116 0.012810606 0.021946549 0.011260191 0.009642398 0.014485978
```

```
#Compute variance covariance matrix:
covmat <- cov(r[-1]) #Without ^GSPC
#Compute correlation matrix:
cormat <- cor(r[-1]) #Without ^GSPC
#Compute the vector of variances:
variances <- diag(covmat)
#Compute the vector of standard deviations:
stdev <- diag(covmat)^.5
```

d)

```
plot(stdev, means, ylab = "Expected Return", xlab = expression(sigma))
```



e)

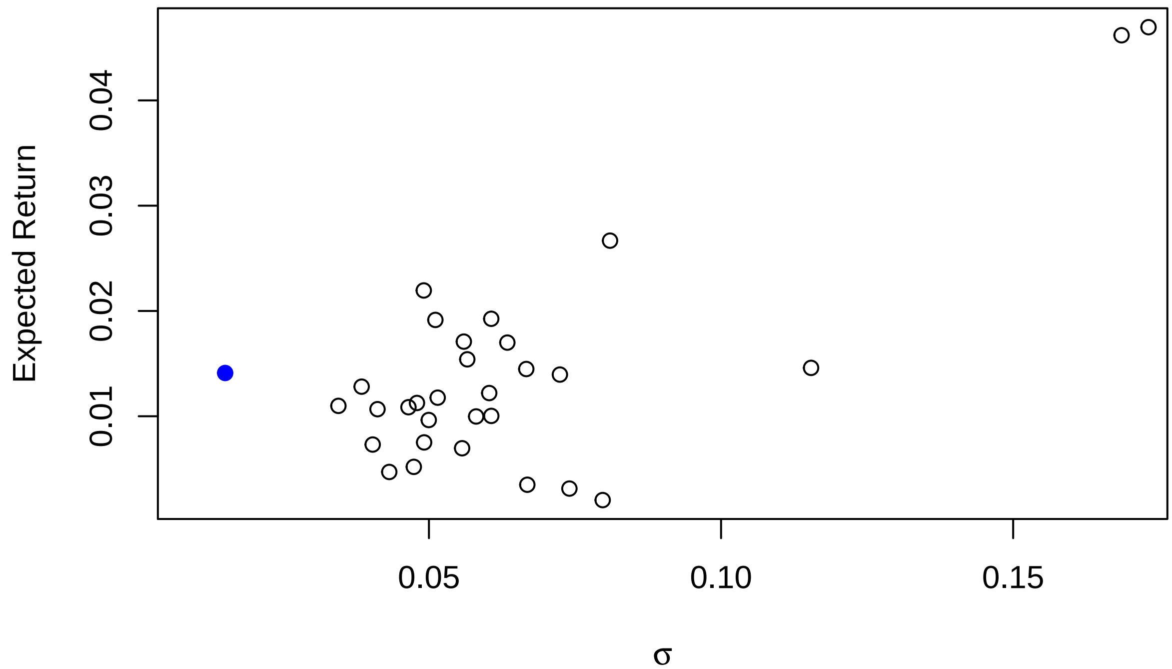
```
## mean of equal allocation portfolio
weight <- 1/30

portfolio.expected <- as.numeric((weight*means[1]) + (weight*means[2]) + (weight*means[3]) + (weight*mea

## variance of equal allocation portfolio
w.s <- (1/30)^2
cov.coef <- 2*(1/30)*(1/30)


portfolio.variance <- as.numeric((w.s*variances[1]) + (w.s*variances[2]) + (w.s*variances[3]) + (w.s*va

portfolio.sd <- sqrt(portfolio.variance)

## output expected return and expected std dev given equal allocation
portfolio.expected
```

```
## [1] 0.01410863
```

```
portfolio.sd
```

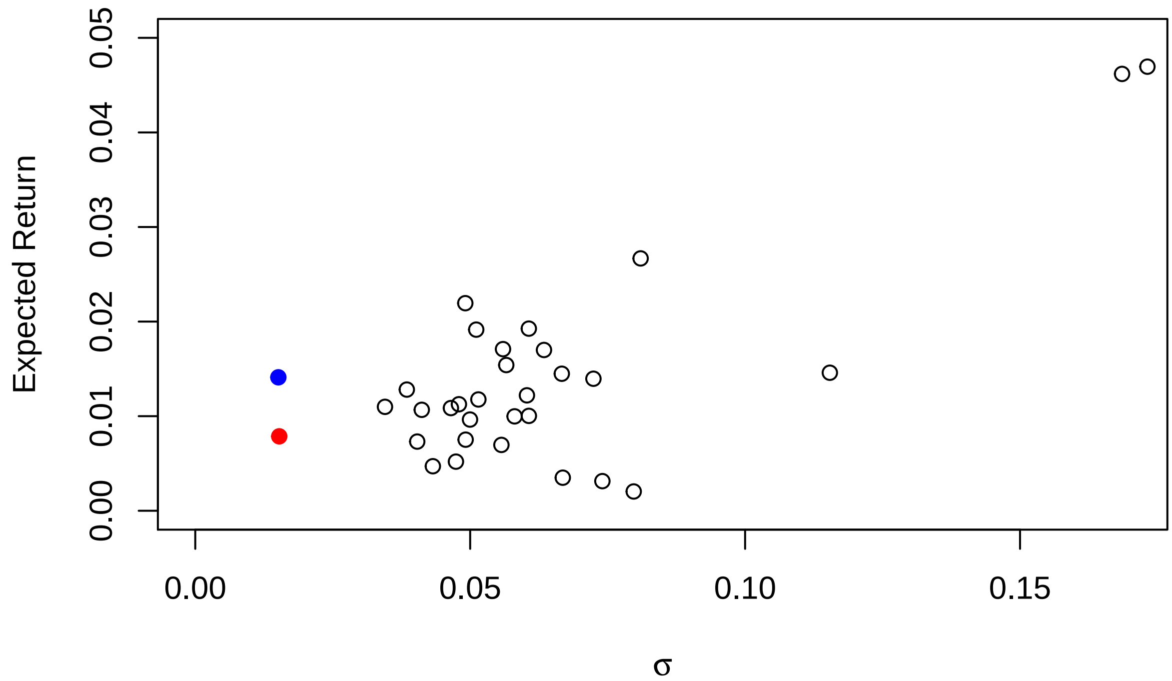
```
## [1] 0.01511409
```

```
## add this to plot from before
plot(stdev, means, ylab = "Expected Return", xlab = expression(sigma), xlim = c(0.010,0.17))
points(portfolio.sd, portfolio.expected,pch = 19, col = "blue")
```



f)

```
ones <- c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)

#Compute A:
A <- t(ones) %*% solve(covmat) %*% means
#Compute B:
B <- t(means) %*% solve(covmat) %*% means
#Compute C:
C <- t(ones) %*% solve(covmat) %*% ones
#Compute D:
D <- B*C - A^2

#plot means against sigma:
plot(stdev, means, ylab = "Expected Return", xlab = expression(sigma), xlim = c(0,0.17), ylim = c(0,.05
points(portfolio.sd, portfolio.expected,pch = 19, col = "blue")
#Add the minimum risk portfolio:
points(sqrt(1/C), A/C, pch=19, col = "red")
```



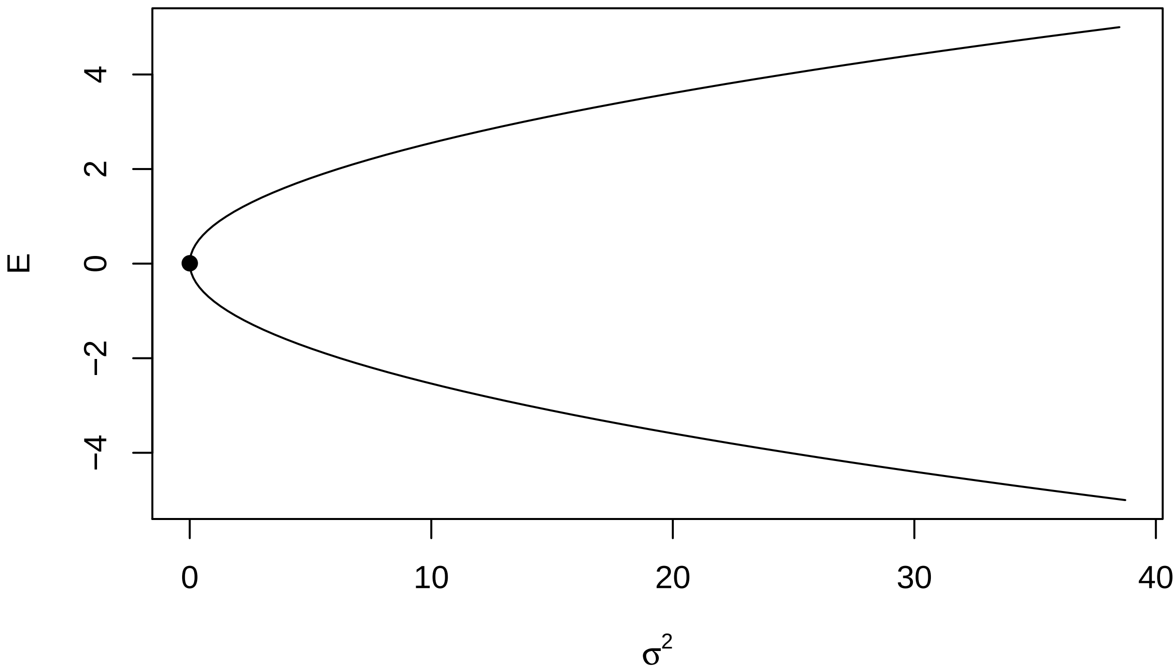
Part 2 :

a)

```
#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 depreca
##   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 dep
##   Use c() or as.vector() instead.
```

```
#Or plot E against sigma2:
plot(sigma2, E,type="l", xlab=expression(sigma^2))

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



b)

1.

```
plot(0, A/C, main = "Portfolio possibilities curve", xlab = "Risk (standard deviation)",
  ylab = "Expected Return", type = "n",
  xlim = c(-2*sqrt(1/C), 4*sqrt(1/C)),
  ylim = c(-2*A/C, 4*A/C))

#Plot center of the hyperbola:
    points(0, A/C, pch = 19)

#Plot transverse and conjugate axes:
    abline(v = 0) #Also this is the y-axis.
    abline(h = A/C)

#Plot the x-axis:
    abline(h = 0)

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

#Find the asymptotes:
    V <- seq(-1, 1, 0.001)
    A1 <- A/C + V * sqrt(D/C)
```

```
## Warning in V * sqrt(D/C): Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in A/C + V * sqrt(D/C): Recycling array of length 1 in array-vector arithmetic is deprecated
##   Use c() or as.vector() instead.
```

```r
    A2 <- A/C - V * sqrt(D/C)
```

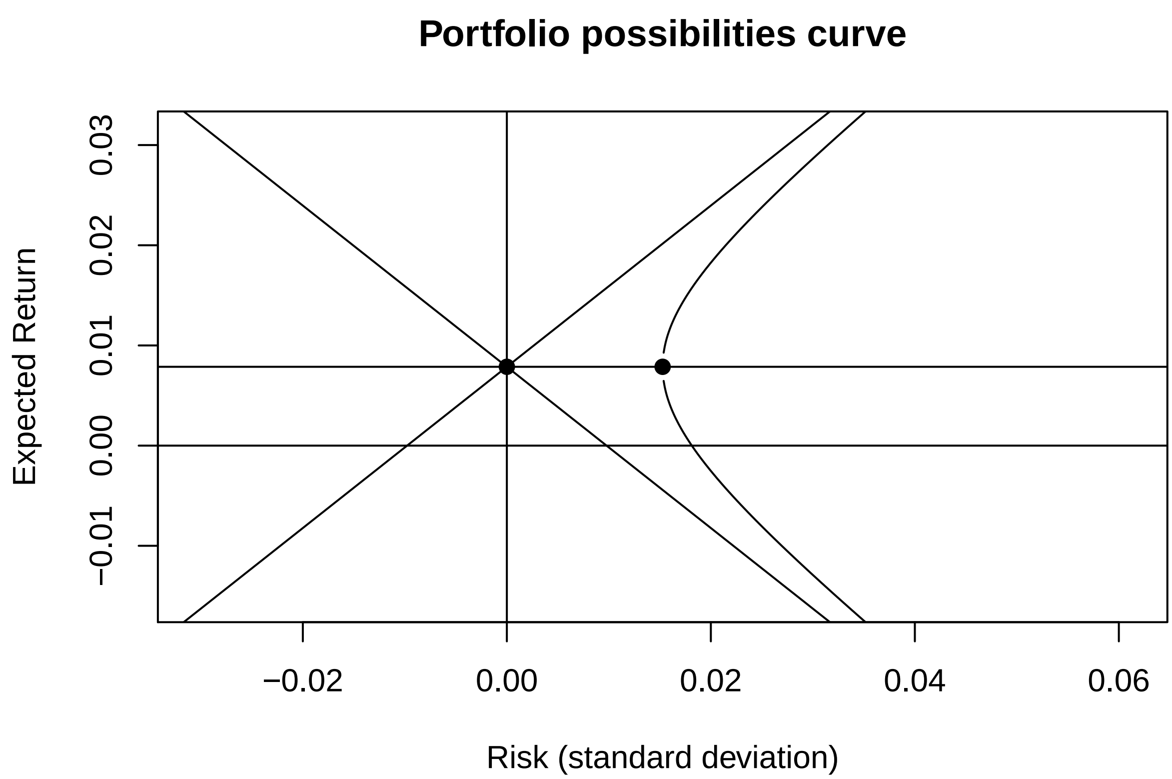
```
## Warning in V * sqrt(D/C): Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```
## Warning in A/C - V * sqrt(D/C): Recycling array of length 1 in array-vector arithmetic is deprecated
##   Use c() or as.vector() instead.
```

```r
    points(V, A1, type = "l")
    points(V, A2, type = "l")

#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)
    y2 <- (A - sqrt(D*(C*sdeff^2 - 1)))*(1/C)
    options(warn = 0)

    points(sdeff, y1, type = "l")
    points(sdeff, y2, type = "l")
```



2.

```r
#Second method:  Find two portfolios on the frontier.
#Portfolio 1:  Minimum risk portfolio.
#Composition:
x1 <- ( solve(covmat) %*% ones ) / as.numeric( t(ones) %*% solve(covmat) %*% ones )

#Mean:
m1 <- t(x1) %*% means

#Variance:
v1 <- t(x1) %*% covmat %*% x1

#Portfolio 2:  (It doesn't have to be efficient, as long as it is on the frontier).
#Need to choose a value of E.  Let's say, E=0.015.
#To find x2 we use our class notes (see week 2 - lecture 1 notes):
#x2=lambda1*Sigma^-1*means + lambda2*Sigma^-1*ones
#lambda1 = (CE-A)/D and lambda2=(B-AE)/D.

E <- 0.015
lambda1 <- (C*E-A)/D
lambda2 <- (B-A*E)/D

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

#Mean:
m2 <- t(x2) %*% means

#Variance:
v2 <- t(x2) %*% covmat %*% x2

#We also need the covariance between portfolio 1 and portfolio 2:
cov_ab <- t(x1) %*% covmat %*% x2

#Now we have two portfolios on the frontier.  We can combine them to trace out the entire frontier:
#Let a be the proportion of investor's wealth invested in portfolio 1.
#Let b be the proportion of investor's wealth invested in portfolio 2.

a <- seq(-3,3,.1)
b <- 1-a

r_ab <- a*m1 + b*m2
```

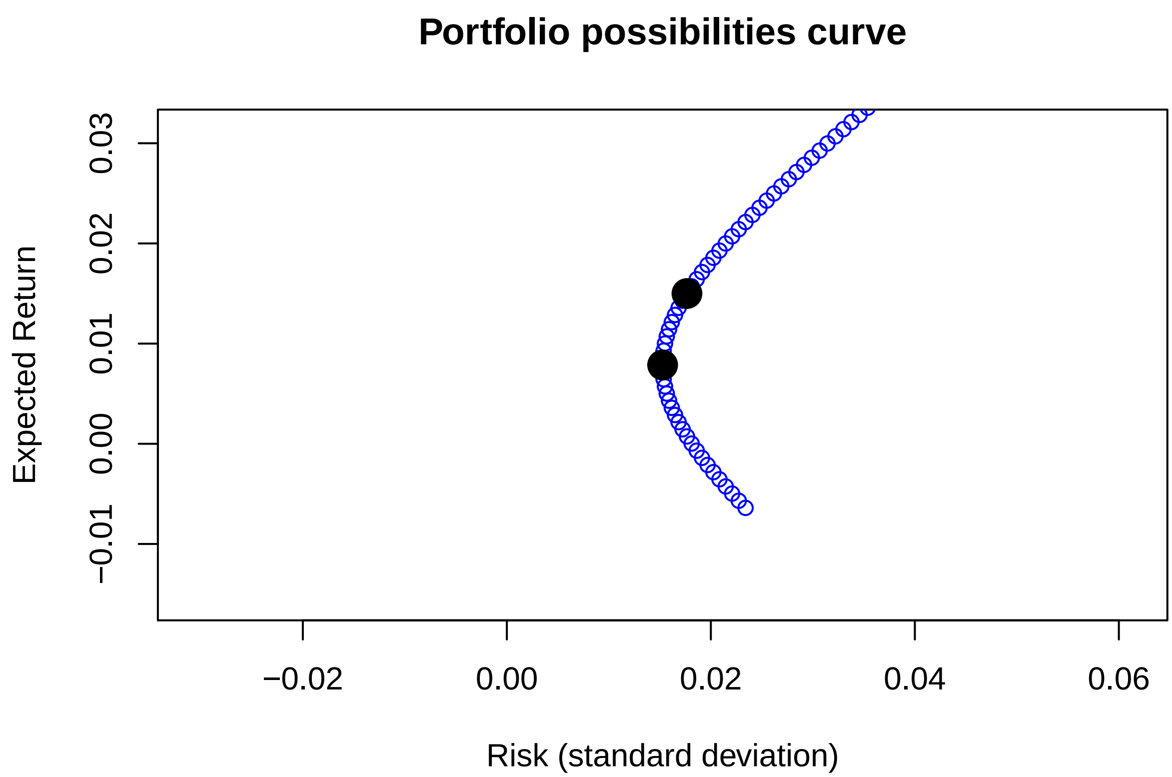
```
## Warning in a * m1: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in b * m2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```r
var_ab <- a^2*v1 + b^2*v2 + 2*a*b*cov_ab
```

```
## Warning in a^2 * v1: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in b^2 * v2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

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

```
sd_ab <- var_ab^.5


plot(0, A/C, main = "Portfolio possibilities curve", xlab = "Risk (standard deviation)",
  ylab = "Expected Return", type = "n",
  xlim = c(-2*sqrt(1/C), 4*sqrt(1/C)),
  ylim = c(-2*A/C, 4*A/C))

points(sd_ab, r_ab, col="blue")

#These are the two portfolios:
points(v1^.5, m1, pch=19, cex=2)
points(v2^.5, m2, pch=19, cex=2)
```



c)

```
x1 <- ( solve(covmat) %*% ones ) / as.numeric( t(ones) %*% solve(covmat) %*% ones )

#Mean:
m1 <- t(x1) %*% means

#Variance:
v1 <- t(x1) %*% covmat %*% x1

#Portfolio 2:  (It doesn't have to be efficient, as long as it is on the frontier).
```

```r
#Need to choose a value of E.  Let's say, E=0.015.
#To find x2 we use our class notes (see week 2 - lecture 1 notes):
#x2=lambda1*Sigma^-1*means + lambda2*Sigma^-1*ones
#lambda1 = (CE-A)/D and lambda2=(B-AE)/D.

E <- 0.015
lambda1 <- (C*E-A)/D
lambda2 <- (B-A*E)/D

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

#Mean:
m2 <- t(x2) %*% means

#Variance:
v2 <- t(x2) %*% covmat %*% x2

#We also need the covariance between portfolio 1 and portfolio 2:
cov_ab <- t(x1) %*% covmat %*% x2

#Now we have two portfolios on the frontier.  We can combine them to trace out the entire frontier:
#Let a be the proportion of investor's wealth invested in portfolio 1.
#Let b be the proportion of investor's wealth invested in portfolio 2.

a <- seq(-3,3,.1)
b <- 1-a

r_ab <- a*m1 + b*m2
```

```
## Warning in a * m1: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in b * m2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.
```

```r
var_ab <- a^2*v1 + b^2*v2 + 2*a*b*cov_ab
```

```
## Warning in a^2 * v1: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in b^2 * v2: Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

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

```r
sd_ab <- var_ab^.5


plot(0, A/C, main = "Portfolio possibilities curve", xlab = "Risk (standard deviation)",
  ylab = "Expected Return", type = "n",
  xlim = c(0,.09),
  ylim = c(-2*A/C, .09))

points(sd_ab, r_ab, col="blue")
```

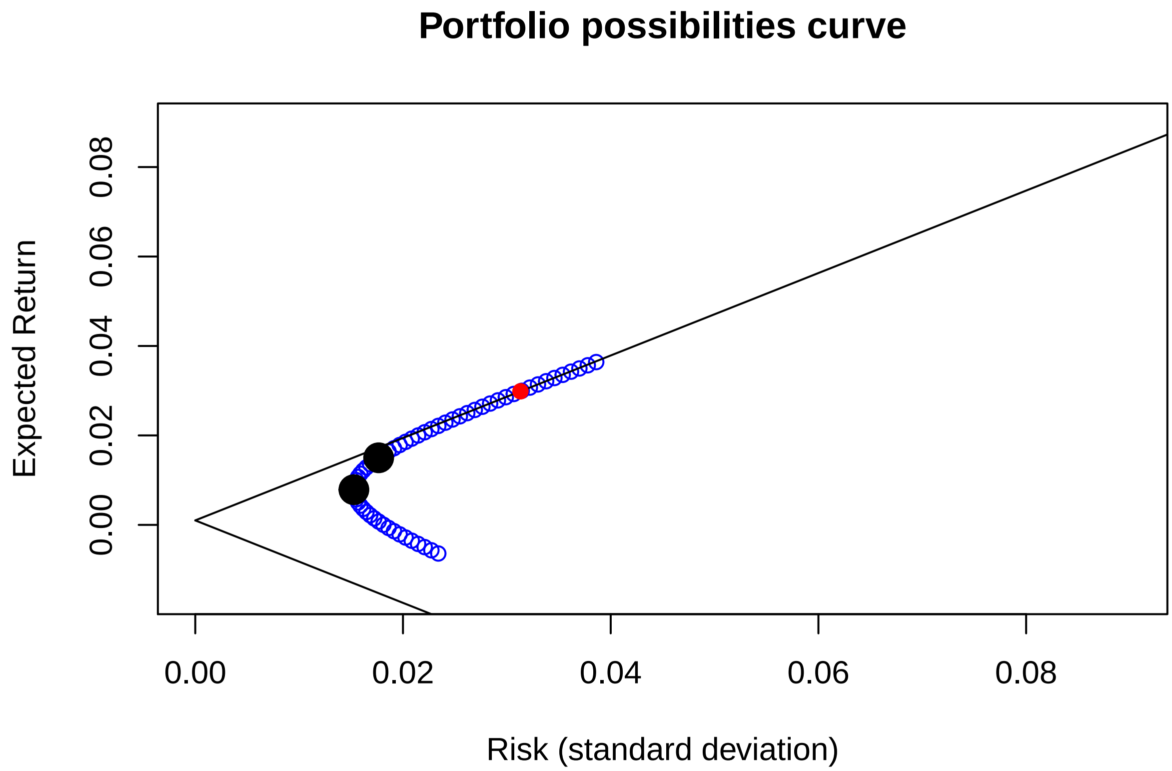
```
#These are the two portfolios:
points(v1^.5, m1, pch=19, cex=2)
points(v2^.5, m2, pch=19, cex=2)

#Choose risk-free return:
Rf <- 0.001

#Range of expected return:
sigma <- seq(0,.5, .001)

Rp1 <- Rf + sigma*sqrt(C*Rf^2-2*Rf*A+B)
```

```
## Warning in sigma * sqrt(C * Rf^2 - 2 * Rf * A + B): Recycling array of length 1 in vector-array arit
##   Use c() or as.vector() instead.
```

```
Rp2 <- Rf - sigma*sqrt(C*Rf^2-2*Rf*A+B)
```

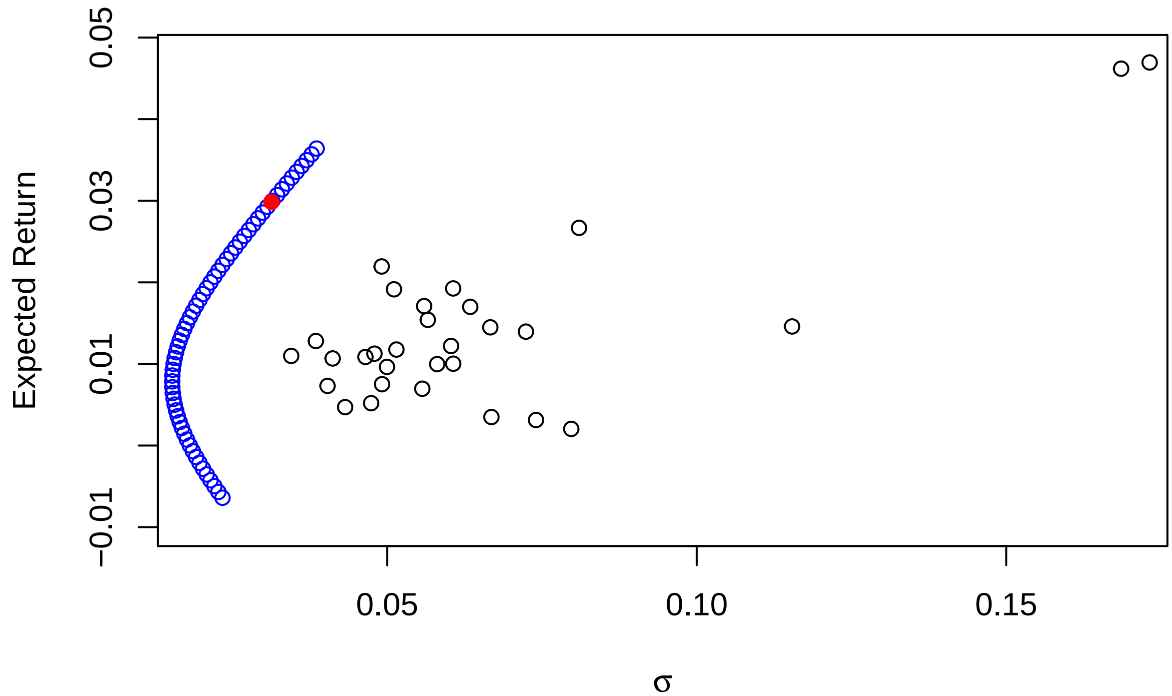
```
## Warning in sigma * sqrt(C * Rf^2 - 2 * Rf * A + B): Recycling array of length 1 in vector-array arit
##   Use c() or as.vector() instead.
```

```
points(sigma, Rp1, type="l")

points(sigma, Rp2, type="l")

#====================================
#====================================
#Point of tangency:
R <- means-Rf
z <- solve(covmat) %*% R

xx <- z/sum(z)

rr <- t(xx) %*% means
varr <- t(xx) %*% covmat %*% xx
sdev <- varr^.5

points(sdev, rr, pch=19, col = "red")
```



d)

```
#plot means against sigma:
plot(stdev, means, ylab = "Expected Return", xlab = expression(sigma), xlim = c(0.019,0.17), ylim = c(-
#Add efficient frontier
points(sd_ab, r_ab, col="blue")
#Add tangency point
points(sdev, rr, pch=19, col = "red")
```



Part 3:

```
# Read data into R
a <- read.table("http://www.stat.ucla.edu/~nchristo/statistics_c183_c283/statc183c283_5stocks.txt", head
```

a)

```
# Convert prices into returns
r1 <- (a$P1[-length(a$P1)]-a$P1[-1])/a$P1[-1]
r2 <- (a$P2[-length(a$P2)]-a$P2[-1])/a$P2[-1]
r3 <- (a$P3[-length(a$P3)]-a$P3[-1])/a$P3[-1]
r4 <- (a$P4[-length(a$P4)]-a$P4[-1])/a$P4[-1]
r5 <- (a$P5[-length(a$P5)]-a$P5[-1])/a$P5[-1]

returns <- as.data.frame(cbind(r1,r2,r3,r4,r5))
head(returns)
```

```
##              r1           r2           r3          r4           r5
## 1  0.1325966851  0.248246844  0.056577737 -0.03121342  0.097681688
## 2 -0.0103881903  0.002577924 -0.025549081  0.02520000 -0.002598077
## 3 -0.0005464481  0.042511605  0.152376033  0.06202209  0.121176813
## 4 -0.0291777188 -0.004136253 -0.028600100  0.04995540 -0.081840064
## 5  0.0595840360  0.098049693 -0.058573453 -0.02564103  0.128925121
## 6 -0.0091896408  0.039722222 -0.006103286  0.04306437 -0.034965035
```

b)

```r
#Compute the means:
means <- colMeans(returns)
means
```

```
##           r1           r2           r3           r4           r5
## 0.0027625075 0.0035831363 0.0066229478 0.0004543727 0.0045679106
```

```r
#Find the covariance matrix:
cov.matrix <- cov(returns)
cov.matrix
```

```
##             r1          r2          r3          r4          r5
## r1 0.005803160 0.001389264 0.001666854 0.000789581 0.001351044
## r2 0.001389264 0.009458804 0.003944643 0.002281200 0.002578939
## r3 0.001666854 0.003944643 0.016293581 0.002863584 0.001469964
## r4 0.000789581 0.002281200 0.002863584 0.009595202 0.003210827
## r5 0.001351044 0.002578939 0.001469964 0.003210827 0.009242440
```

c)

```r
# Extract Exxon-Mobil and Boeing stocks only, then find new means and cov-var matrix
Exx.Boe.r <- returns[,-c(2:4)]
Exx.Boe.means <- colMeans(Exx.Boe.r)
Exx.Boe.covmat <- cov(Exx.Boe.r)
Exx.Boe.covmat
```

```
##             r1          r5
## r1 0.005803160 0.001351044
## r5 0.001351044 0.009242440
```

```r
# Find vector of weights or composition for portfolio
ones <- as.matrix(rep(1,2))
numerator <- solve(Exx.Boe.covmat)%*%ones
denominator <- t(ones)%*%solve(Exx.Boe.covmat)%*%ones
X <- numerator/as.numeric(denominator)
cat("The composition of the minimum risk portfolio involving\nthe Exxon-Mobil and Boeing stocks is", X,
```

```
## The composition of the minimum risk portfolio involving
## the Exxon-Mobil and Boeing stocks is 0.6393153 0.3606847 ,respectively
```

```r
# Calculate expected return of minimum risk portfolio
cat("The expected return of the minimum risk portfolio is", t(X)%*%Exx.Boe.means,"or about 0.34%")
```

```
## The expected return of the minimum risk portfolio is 0.003413689 or about 0.34%
```

```r
# Calculate standard deviation of minimum risk portfolio
cat("The standard deviation of the minimum risk portfolio is",sqrt(t(X)%*%Exx.Boe.covmat%*%X),"or about
```

```
## The standard deviation of the minimum risk portfolio is 0.06478695 or about 6.5%
```

d)

```
ones <- rep(1,5)
#Compute A:
A <- t(ones) %*% solve(cov.matrix) %*% means
#Compute B:
B <- t(means) %*% solve(cov.matrix) %*% means
#Compute C:
C <- t(ones) %*% solve(cov.matrix) %*% ones
#Compute D:
D <- B*C - A^2

plot(0, A/C, main = "Portfolio possibilities curve", xlab = "Risk (standard deviation)",
  ylab = "Expected Return", type = "n",
  xlim = c(0,0.3),
  ylim = c(-.003,.01))

#Plot center of the hyperbola:
    points(0, A/C, pch = 19)

#Plot transverse and conjugate axes:
    abline(v = 0) #Also this is the y-axis.
    abline(h = A/C)

#Plot the x-axis:
    abline(h = 0)

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

#Find the asymptotes:
    V <- seq(-1, 1, 0.001)
    A1 <- A/C + V * sqrt(D/C)
```

```
## Warning in V * sqrt(D/C): Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in A/C + V * sqrt(D/C): Recycling array of length 1 in array-vector arithmetic is deprecated
##   Use c() or as.vector() instead.
```

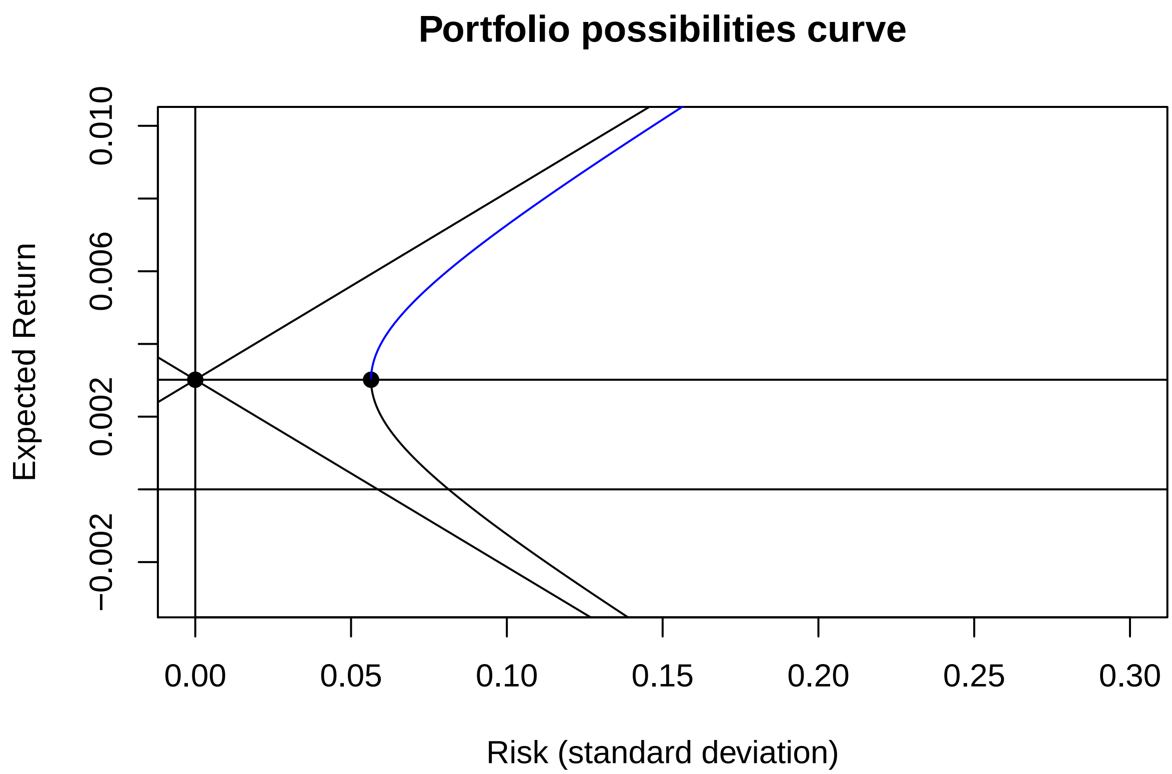
```
    A2 <- A/C - V * sqrt(D/C)
```

```
## Warning in V * sqrt(D/C): Recycling array of length 1 in vector-array arithmetic is deprecated.
##   Use c() or as.vector() instead.

## Warning in A/C - V * sqrt(D/C): Recycling array of length 1 in array-vector arithmetic is deprecated
##   Use c() or as.vector() instead.
```

```
    points(V, A1, type = "l")
    points(V, A2, type = "l")

#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.
```

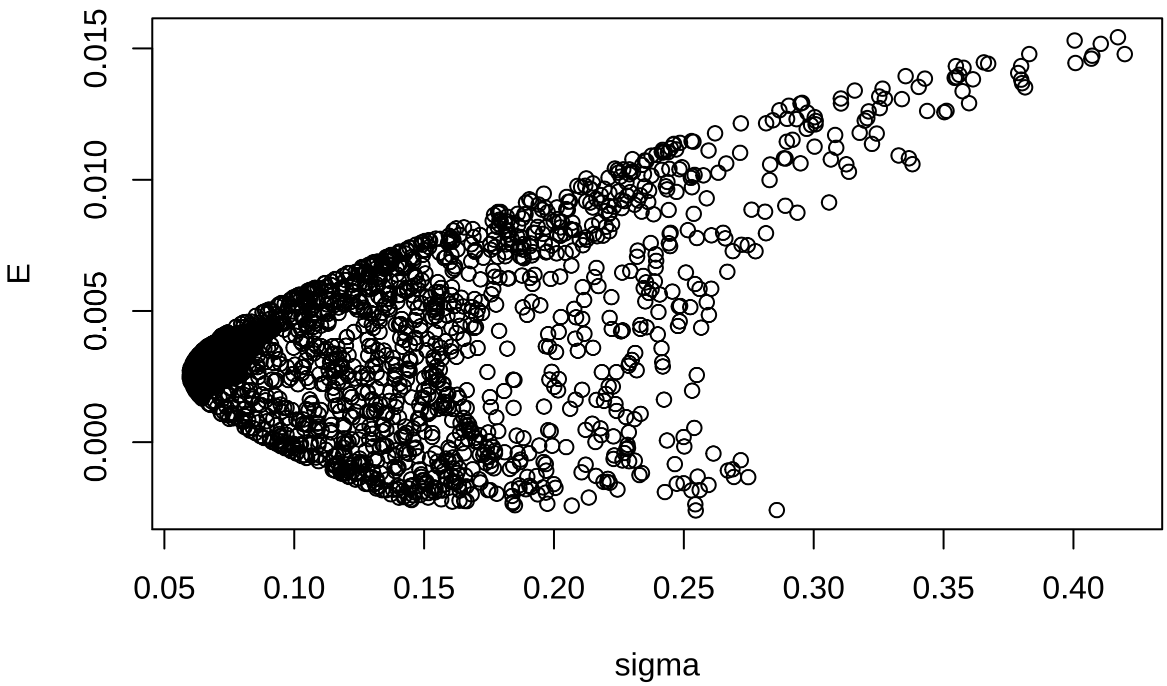
```
options(warn = -1)
y1 <- (A + sqrt(D*(C*sdeff^2 - 1)))*(1/C)
y2 <- (A - sqrt(D*(C*sdeff^2 - 1)))*(1/C)
options(warn = 0)

points(sdeff, y1, type = "l", col = "blue") # Efficient frontier shown as blue line
points(sdeff, y2, type = "l")
```



e)

```
# Read in table of many combinations which will be the weights
b <- read.table("http://www.stat.ucla.edu/~nchristo/datac183c283/statc183c283_abc.txt", header=T)
b <- as.matrix(b)
# Extract three stocks of interest from returns dataset
Exx.McD.Boe.r <- returns[,-c(2:3)]
#Calculate means of these three stock returns
Exx.McD.Boe.means <- as.matrix(colMeans(Exx.McD.Boe.r))
#Calculate var-cov matrix for these three stocks
Exx.McD.Boe.covmat <- cov(Exx.McD.Boe.r)
#Calculate expected return of many portfolio combinations
E <- b%*%Exx.McD.Boe.means
#Calculate variance of many portfolio combinations, then standard deviation
sigma2 <- diag((b%*%Exx.McD.Boe.covmat)%*%t(b))
sigma <- sqrt(sigma2)
#Plot the cloud of points
```

```r
plot(sigma,E)
```



f)

```r
# Define mean vector, var-cov matrix, and inverse var-cov matrix
R_ibar <- Exx.McD.Boe.means
var_covar <- Exx.McD.Boe.covmat
var_covar_inv <- solve(var_covar)
# Create the new R vector
Rf <- 0.001
R <- R_ibar - Rf
# Compute the Z vector
z <- var_covar_inv %*% R
# Compute the vector X, or the composition of the G portfolio
x <- z/sum(z)
cat("The weights of the three stocks in portfolio\nG are",x[1],",",x[2],", and",x[3])
```

```
## The weights of the three stocks in portfolio
## G are 0.5284782 , -0.4955882 , and 0.96711
```

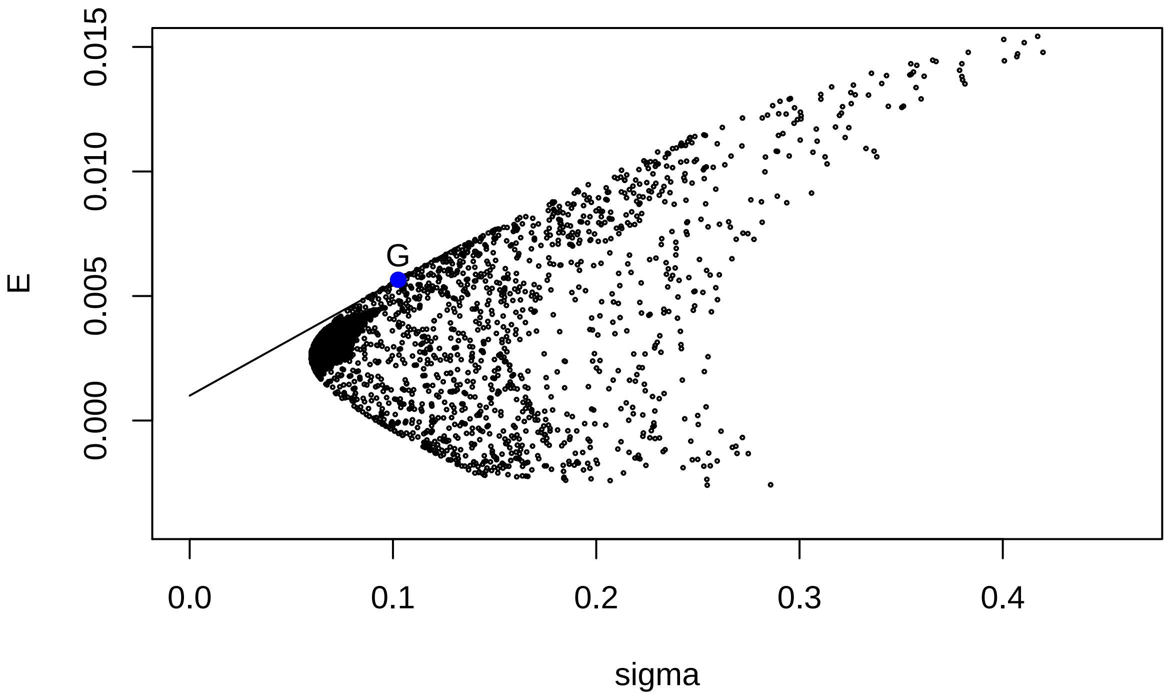
```r
# Compute the expected return of portfolio G
R_Gbar <- t(x) %*% R_ibar
cat("The expected return of portfolio G is",R_Gbar)
```

```
## The expected return of portfolio G is 0.005652415
```

```r
# Compute the variance and standard deviation of portfolio G
var_G <- t(x) %*% var_covar %*% x
sd_G <- var_G^0.5
cat("The standard deviation of the return of portfolio G is",sd_G)
```

```
## The standard deviation of the return of portfolio G is 0.1025256
```

```r
# Draw tangent line on plot from part e
# First, must calculate the slope
slope <- (R_Gbar-Rf)/(sd_G)
plot(sigma,E, xlim = c(0,0.46), ylim = c(-.004,.015), cex = 0.25)
lines(c(0,sd_G, 1.3*sd_G),c(.001,R_Gbar,0.001+slope*(1.3*sd_G)))
#Identify portfolio G:
points(sd_G, R_Gbar, cex=1, col="blue", pch=19)
text(sd_G, R_Gbar+.001, "G")
```



g)

```r
# Compute expected return
E.G.Rf <- (.6*R_Gbar) + (0.4*Rf)
cat("The expected return of a portfolio consisting of 60%\nwealth invested\n in portfolio G and the rema
```

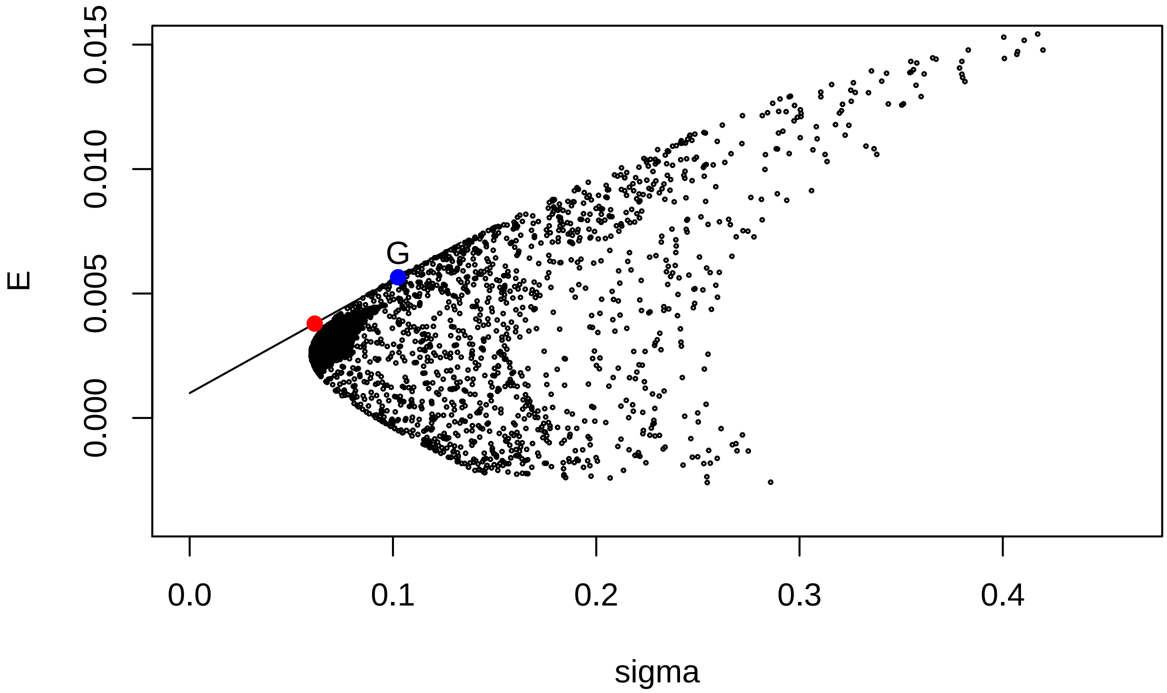
```
## The expected return of a portfolio consisting of 60%
## wealth invested
##  in portfolio G and the remaining 40%
##  invested in the risk free asset is 0.003791449
```

```r
# Compute the standard deviation
Sd.G.Rf <- ((.6)*sd_G) + ((.4)*0)
cat("The standard deviation of this portfolio is",Sd.G.Rf)
```

```
## The standard deviation of this portfolio is 0.06151535
```

```r
# Add this point to the graph from part f
plot(sigma,E, xlim = c(0,0.46), ylim = c(-.004,.015), cex = 0.25)
lines(c(0,sd_G, 1.3*sd_G),c(.001,R_Gbar,0.001+slope*(1.3*sd_G)))
#Identify portfolio G:
```

```r
points(sd_G, R_Gbar, cex=1, col="blue", pch=19)
text(sd_G, R_Gbar+.001, "G")
points(Sd.G.Rf,E.G.Rf, cex=1, col="red",pch=19)
```



h)

1.

```r
# Portfolio A was found in part f, with Rf = .001
# Define mean vector, var-cov matrix, and inverse var-cov matrix
R_ibar <- Exx.McD.Boe.means
var_covar <- Exx.McD.Boe.covmat
var_covar_inv <- solve(var_covar)
# Create the new R vector
Rf.A <- 0.001
R.A <- R_ibar - Rf.A
# Compute the Z vector
z.A <- var_covar_inv %*% R.A
# Compute the vector X, or the composition of the A portfolio
x.A <- z.A/sum(z.A)
cat("The weights of the three stocks in portfolio A are",x.A[1],",",x.A[2],", and",x.A[3])
```

```
## The weights of the three stocks in portfolio A are 0.5284782 , -0.4955882 , and 0.96711
```

```r
# Compute the expected return of portfolio A
R_Abar <- t(x.A) %*% R_ibar
cat("The expected return of portfolio A is",R_Abar)
```

```
## The expected return of portfolio A is 0.005652415
```

```
# Compute the variance and standard deviation of portfolio A
var_A <- t(x.A) %*% var_covar %*% x.A
sd_A <- var_A^0.5
cat("The standard deviation of the return of portfolio A is",sd_A)
```

```
## The standard deviation of the return of portfolio A is 0.1025256
```

```
# Must find Portfolio B
# Define mean vector, var-cov matrix, and inverse var-cov matrix
R_ibar <- Exx.McD.Boe.means
var_covar <- Exx.McD.Boe.covmat
var_covar_inv <- solve(var_covar)
# Create the new R vector
Rf.B <- 0.002
R.B <- R_ibar - Rf.B
# Compute the Z vector
z.B <- var_covar_inv %*% R.B
# Compute the vector X, or the composition of the G portfolio
x.B <- z.B/sum(z.B)
cat("The weights of Portfolio B are",x.B[1],",",x.B[2],", and",x.B[3])
```

```
## The weights of Portfolio B are 0.5312205 , -1.802663 , and 2.271443
```

```
# Compute the expected return of portfolio G
R_Bbar <- t(x.B) %*% R_ibar
cat("The expected return of portfolio B is",R_Bbar)
```

```
## The expected return of portfolio B is 0.01102417
```

```
# Compute the variance and standard deviation of portfolio G
var_B <- t(x.B) %*% var_covar %*% x.B
sd_B <- var_B^0.5
cat("The standard deviation of the return of portfolio B is",sd_B)
```

```
## The standard deviation of the return of portfolio B is 0.2365542
```

2.

```
# Compute the variances between Portfolios A and B
cov_AB <- t(x.A)%*%var_covar%*%x.B
cat("The covariance between Portfolios A and B is",cov_AB)
```

```
## The covariance between Portfolios A and B is 0.02264823
```

3.

```
# Re-plot cloud of points from before
plot(sigma,E, xlim = c(0,0.46), ylim = c(-.004,.015), cex = 0.25)
# Add points for Portolios A and B
points(sd_A, R_Abar, cex=1.5, col="blue", pch=19)
text(sd_A, R_Abar+.001, "A")
points(sd_B, R_Bbar, cex=1.5, col="red", pch=19)
text(sd_B, R_Bbar+.001, "B")
# Trace frontier using both portfolios
```

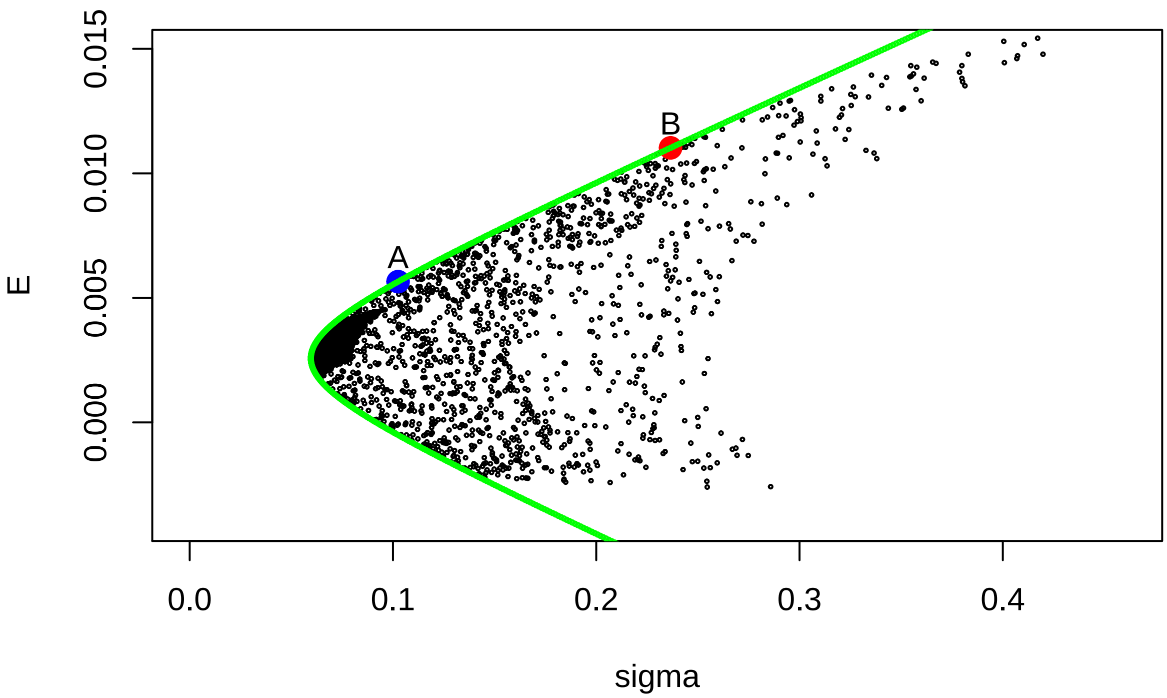
```
xa <- seq(-3, 5, 0.01)
xb <- 1-xa
#Compute the expected return and standard deviation for each combination of xa, xb:
sigma_p <- (xa^2*var_A + xb^2*var_B+ 2*xa*xb*cov_AB)^.5
```

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

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

## Warning in 2 * xa * xb * cov_AB: Recycling array of length 1 in vector-array arithmetic is deprecatec
##   Use c() or as.vector() instead.
```

```
rp_bar <- xa*R_Abar + xb*R_Bbar
```

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

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

```
#Plot:
points(sigma_p, rp_bar, cex=0.3, xaxt="no", yaxt="no", col = "green")
```



4.

```
#Compute the minimum risk portfolio in terms of the portfolios A and B:
xA_min <- (var_B - cov_AB)/(var_A+var_B-2*cov_AB)
xB_min <- 1-xA_min

#Find the composition of the minimum risk portfolio in terms of the three stocks:
x1_min <- xA_min*x.A[1] + xB_min*x.B[1]
```

```r
x2_min <- xA_min*x.A[2] + xB_min*x.B[2]
x3_min <- xA_min*x.A[3] + xB_min*x.B[3]

#Find the expected return and standard deviation of the minimum risk portfolio:

xx <- as.matrix(c(x1_min,x2_min,x3_min))
cat("The composition of the minimum risk portfolio in terms of the three stocks\nis", as.matrix(xx),"re
```

```
## The composition of the minimum risk portfolio in terms of the three stocks
## is 0.5269063 0.2536533 0.2194404 respectively
```

```r
rp_minimum <- t(xx) %*% R_ibar
sd_minimum <- (t(xx) %*% var_covar %*% xx)^.5
cat("The expected return of the minimum risk portfolio is", rp_minimum)
```

```
## The expected return of the minimum risk portfolio is 0.00257322
```

```r
cat("The standard deviation of the minimum risk portfolio", sd_minimum)
```

```
## The standard deviation of the minimum risk portfolio 0.05961942
```