STATS183 Project 7

Takao Oba

2023-05-05

Please answer the following questions:

a. Assume the multigroup model holds with short sales allowed. Find the composition of the optimal portfolio and its expected return and standard deviation and place it on the plot you constructed in previous projects with all the other portfolios and stocks. Note: Please see the numerical example of handout #37 for more details.

```
#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)]
```

```
# by industries
r1 <- r[,2:7]
r2 <- r[,8:13]
r3 <- r[,14:19]
r4 <- r[,20:25]
r5 <- r[,26:31]

# make sure all of the industries has the same dimension
all(dim(r1) == dim(r2), dim(r1) == dim(r3), dim(r1) == dim(r4),dim(r1) == dim(r5))
```

```
## [1] TRUE
```

```
rtemp <- r[,-1]

corrr_simplified <- diag(5)
corrr <- diag(30)
for(i in 1:5){
  for(j in 1:5){
    if(i == j){
      corrr[(1 + 6*(i-1)):(1+5 + 6*(i-1)), (1 + 6*(i-1)):(1+5 + 6*(i-1))] <- (sum(cor(rtemp[, (1 + 6*(i
      corrr_simplified[i,j] <- (sum(cor(rtemp[, (1 + 6*(i-1)):(1+5 + 6*(i-1))])) - 6)/(6*(6-1))
    }else{
      corrr[(1 + 6*(i-1)):(1+5 + 6*(i-1)), (1 + 6*(j-1)):(1+5 + 6*(j-1))] <- (mean(cor(rtemp[, (1 + 6*(i
      corrr_simplified[i,j] <- (mean(cor(rtemp[, (1 + 6*(i-1)):(1+5 + 6*(j-1))])))
```

```
    }
  }
}

diag(corrr) <- 1
```

Finding the variance

```
r1_var <- var(r1)
r2_var <- var(r2)
r3_var <- var(r3)
r4_var <- var(r4)
r5_var <- var(r5)
r_var <- var(rtemp)
r_mean <- colMeans(rtemp)
# setting R_f
r_f <- 0.005


C_1 <- 0
for (i in 1:6){
  C_1 <- C_1 + (mean(rtemp[,i]) - r_f)/(sqrt(r1_var[i,i])*(1-corrr_simplified[1,1]))
}


C_2 <- 0
for (i in 7:12){
  C_2 <- C_2 + (mean(rtemp[,i]) - r_f)/(sqrt(r2_var[i-6,i-6])*(1-corrr_simplified[2,2]))
}

C_3 <- 0
for (i in 13:18){
  C_3 <- C_3 + (mean(rtemp[,i]) - r_f)/(sqrt(r3_var[i-12,i-12])*(1-corrr_simplified[3,3]))
}

C_4 <- 0
for (i in 19:24){
  C_4 <- C_4 + (mean(rtemp[,i]) - r_f)/(sqrt(r4_var[i-18,i-18])*(1-corrr_simplified[4,4]))
}

C_5 <- 0
for (i in 25:30){
  C_5 <- C_5 + (mean(rtemp[,i]) - r_f)/(sqrt(r5_var[i-24,i-24])*(1-corrr_simplified[5,5]))
}


C <- matrix(c(C_1, C_2, C_3, C_4, C_5), nrow = 5)
```

Finding A

```
A <- matrix(0, nrow = 5, ncol = 5)

for (i in 1:5){
```

```
  for (j in 1:5){
    A[i,j] <- (2*corrr_simplified[i,j])/(1/corrr_simplified[i,i])
    if(i==j){
      A[i,j] <- A[i,j] + 1
    }
  }
}
```

Finding Phi

```
# Phi = A^-1 C

phi <- solve(A)%*%C
```

Finding z

```
z <- matrix(0, nrow = 5, ncol =6)

for(i in 1:5){
  for(j in 1:6){
    denom <- 1/(sqrt(r_var[(i-1)*6 + j,(i-1)*6 + j])*(1-corrr_simplified[i,i]))
    first_term <- (r_mean[(i-1)*6 + j] - r_f)/(sqrt(r_var[(i-1)*6 + j,(i-1)*5 + j]))

    temp <- 0
    for(k in 1:5){
      temp <- temp + corrr_simplified[i,k]*phi[k]
    }

    z[i,j] <- denom*(first_term - temp)
  }
}

z
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] -25.50189 -20.06474 -21.93179 -23.22075 -13.81878 -11.22768
## [2,] -44.98323 -45.95660 -53.00325 -39.58453 -51.72009 -65.40930
## [3,] -54.18068 -31.85770 -48.68078 -37.42379 -28.09887 -30.99602
## [4,] -44.55707 -24.20434 -38.20912 -32.30489 -30.37962 -37.65748
## [5,] -26.22310 -32.36563 -35.61925 -37.64460 -15.68018 -39.74742
```

Finding the composition

```
x <- z/sum(z)
x
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,] 0.02446804 0.01925131 0.02104267 0.02227937 0.01325857 0.01077251
## [2,] 0.04315960 0.04409351 0.05085449 0.03797976 0.04962335 0.06275759
## [3,] 0.05198418 0.03056618 0.04670725 0.03590662 0.02695974 0.02973944
## [4,] 0.04275071 0.02322309 0.03666011 0.03099525 0.02914802 0.03613084
## [5,] 0.02516001 0.03105352 0.03417524 0.03611848 0.01504451 0.03813605
```

```r
x_composition <- c(t(x))

# Expected Return
expected_return <- t(x_composition) %*% r_mean


ones <- rep(1, 30)
# var_x <- ((expected_return - r_f)^2)/(as.numeric((r_mean - r_f%*%ones)%*%solve(r_var)%*%t(r_mean - r_
var_x <- t(x_composition) %*% r_var %*% x_composition
sd_x <- sqrt(var_x)
```

Plotting

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

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

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

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

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

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

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

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

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


#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)
```

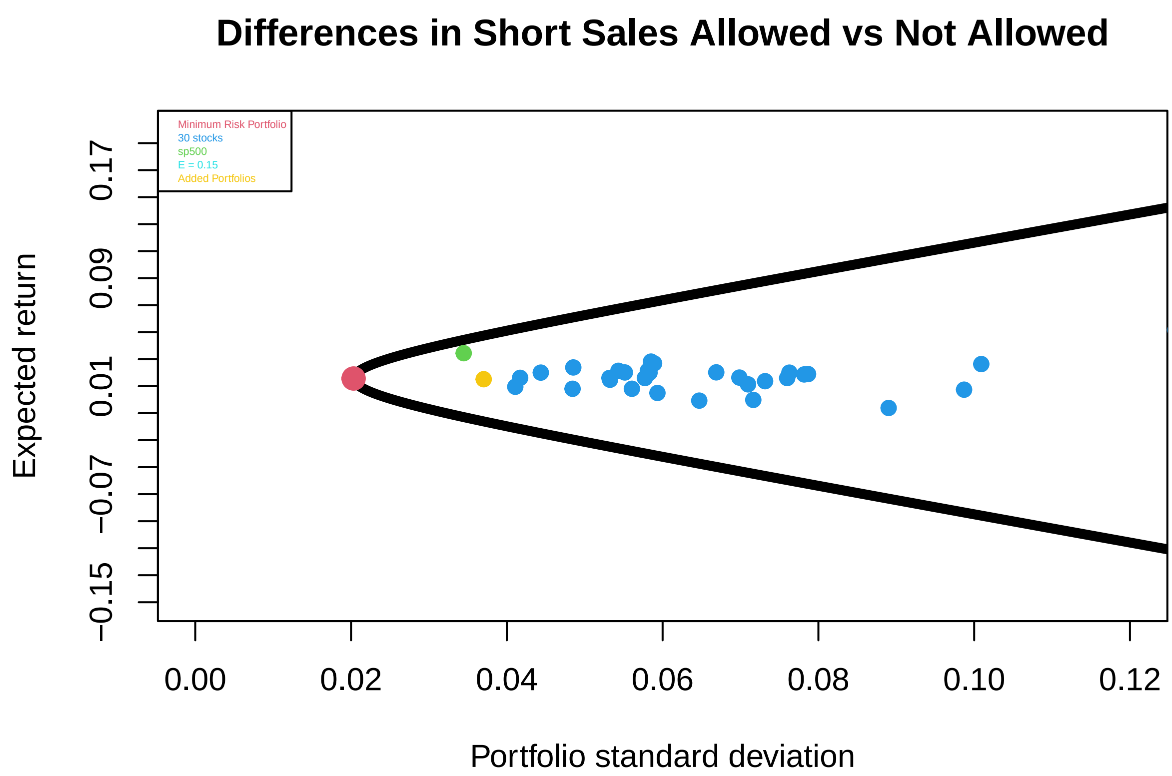
```
## Warning in C * sdeff^2: Recycling array of length 1 in array-vector arithmetic is deprecated.
##   Use c() or as.vector() instead.

## 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 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
    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 ,0.12), ylim=c(-0.15,0.2),
     xlab="Portfolio standard deviation", ylab="Expected return",
     xaxt="no", yaxt="no", main = "Differences in Short Sales Allowed vs Not Allowed")

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")
    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)
# added points multigroup model
```

```r
points(sd_x, expected_return, pch = 19, col = 7)

legend(x = 'topleft',cex  = 0.35,
       legend = c("Minimum Risk Portfolio","30 stocks", "sp500",  "E = 0.15", "Added Portfolios"),
       text.col = c(10, 12, 3,  5, 7))
```



b. Evaluate your portfolios that you constructed in the previous projects. In your analysis you should include the following:

1. Time plots of the performance of all portfolios compared to the S&P 500 (see the graph constructed using handout #17) under "Labs".

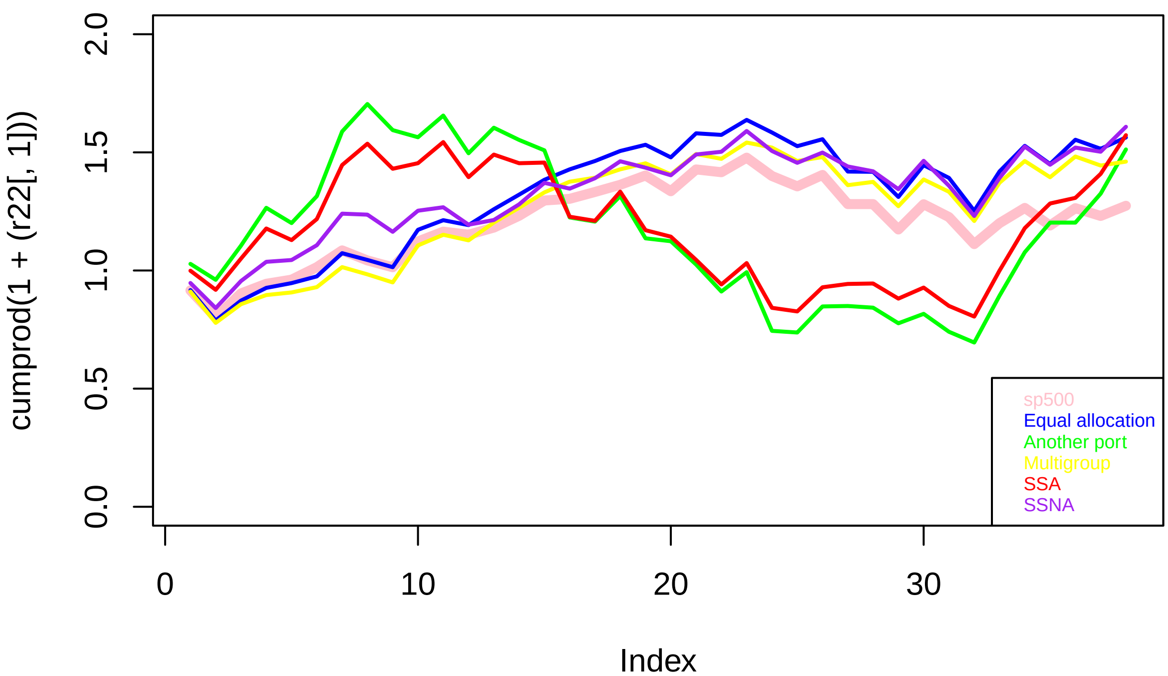
Using test data

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

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

```r
##########################################################################
# SP 500
##########################################################################
#Monthly return in period 2015-01-01 to 2018-05-01:
```

```r
r22 <- as.matrix(r_test)
#Market (S&P500) performance in period 2015-01-01 to 2018-05-01:
plot(cumprod(1+(r22[,1])), ylim=c(0,2), type="l",col="pink", lwd=5)


#########################################################################
#Assume equal allocation:
#########################################################################
x <- rep(1/30, 30)

#Compute montly returns in period 2015-01-01 to 2018-05-01:
r22 <- as.matrix(r_test)

EquRet <-  r22[,-1] %*% x
lines(cumprod(1+EquRet), col="blue", lwd=2)


#########################################################################
#Add another portfolio:
#Use Rf=0.005 to find the optimal portfolio G (tangency point):
#########################################################################
#Compute the mean returns:
R_ibar <- as.matrix(colMeans(r_train[,-1]))

#Compute the variance-covariance matrix:
var_covar <- cov(r_train[,-1])

#Compute the inverse of the variance-covariance matrix:
var_covar_inv <- solve(var_covar)

#Create the vector R:
Rf <- 0.005
R <- R_ibar-Rf

#Compute the vector Z:
z <- var_covar_inv %*% R

#Compute the vector X:
xopt <- z/sum(z)

TangencyRet <- r22[,-1] %*% xopt

lines(cumprod(1+ TangencyRet), col="green", lwd=2)


#########################################################################
# Multigroup Model
#########################################################################
multi <- r22[,-1] %*% x_composition
lines(cumprod(1+multi), col = "yellow", lwd = 2)


#########################################################################
```

```r
# Single Index Model Short Selling Allowed
##########################################################################
load("ssa.RData")
ssa <- r22[,-1] %*% xx2
lines(cumprod(1+ssa), col = "red", lwd = 2)


##########################################################################
# Single Index Model SSNA
##########################################################################
load("ssna.RData")
# MCD, LULU, UNH, SBUX, ZTS, NVDA, ADBE, MA, MSFT, V, INTU
ssna <- r22[,c(2,7,15,4,19, 30,28, 10, 27,8,29)] %*% xx3
lines(cumprod(1+ssna), col = "purple", lwd = 2)


legend(x = 'bottomright',cex  = 0.55,
       legend = c("sp500",  "Equal allocation",  "Another port", "Multigroup", "SSA", "SSNA"),
       text.col = c("pink", "blue", "green", "yellow", "red", "purple"))
```



2. Average growth of each portfolio (use geometric mean).

```
##########################################################################
#Compute average return for the equal allocation portfolio:
##########################################################################
arithMean <- mean(EquRet)
#But (1+r)^39 > (1+r1)(1+r2)...(1+r39)
# (1+ arithMean)^39

#Instead compute geometric average:
comp <- cumprod(1+EquRet)
geoMean <- comp[length(comp)]^(1/length(comp)) - 1


##########################################################################
#Compute average return for the tangency portfolio:
##########################################################################
arithMeanG <- mean(TangencyRet)
#But (1+r)^39 > (1+r1)(1+r2)...(1+r39)
# (1+ arithMeanG)^39

#Instead compute geometric average:
comp <- cumprod(1+ TangencyRet)
geoMeanG <- comp[length(comp)]^(1/length(comp)) - 1


##########################################################################
# SP500
##########################################################################
comp <- cumprod(1 + r22[,1])
geoMeanSP <- as.numeric(comp[length(comp)]^(1/length(comp)) - 1)

##########################################################################
# Multigroup
##########################################################################
comp <- cumprod(1+multi)
geoMeanMG <- comp[length(comp)]^(1/length(comp)) - 1


##########################################################################
# SSA
##########################################################################
comp <- cumprod(1+ssa)
geoMeanSSA <- comp[length(comp)]^(1/length(comp)) - 1


##########################################################################
# SSNA
##########################################################################
comp <- cumprod(1+ssna)
geoMeanSSNA <- comp[length(comp)]^(1/length(comp)) - 1


geoMean
```

```
## [1] 0.01182495
```

```
geoMeanG
```

```
## [1] 0.01094669
```

```
geoMeanSP
```

```
## [1] 0.006393005
```

```
geoMeanMG
```

```
## [1] 0.01002531
```

```
geoMeanSSA
```

```
## [1] 0.01198665
```

```
geoMeanSSNA
```

```
## [1] 0.01258297
```

3. Calculate the Sharpe ratio, differential excess return, Treynor measure, and Jensen differential perfor-mance index.

```
# sharpe ratio
sr <- (colMeans(r_test[,-1] - Rf))/diag(cov(r_test[,-1]))^0.5
sr
```

```
##          MCD          NKE         SBUX            F          CMG         LULU
##  0.095121433  0.068523991  0.066900381  0.123860730  0.168659064  0.106841065
##            V          JPM           MA          BAC           MS          WFC
##  0.024034699  0.012663921  0.035987681 -0.013797974  0.158939759 -0.022290744
##          JNJ          UNH          PFE          CVS           CI          ZTS
## -0.004922135  0.195731430  0.064196792  0.028011622  0.079354756  0.052548802
##          RTX           BA          LMT           DE          CAT           GE
##  0.024860650 -0.018950879  0.032662278  0.283292748  0.169938373  0.067753362
##         AAPL         MSFT         ADBE         INTU         NVDA          CRM
##  0.208545117  0.171276541  0.030077402  0.130252763  0.312167751  0.034260826
```

```
# differential excess return
der <- colMeans(r_test[,-1]) - Rf
der
```

```
##           MCD           NKE          SBUX             F           CMG
##  0.0058939784  0.0064610358  0.0056762705  0.0178750457  0.0186984070
##          LULU             V           JPM            MA           BAC
```

```
##  0.0128842556  0.0019195603  0.0011369156  0.0033417609 -0.0014342688
##            MS           WFC           JNJ           UNH           PFE
##  0.0165816533 -0.0025140859 -0.0002687015  0.0126893919  0.0057467430
##           CVS            CI           ZTS           RTX            BA
##  0.0019639514  0.0065359091  0.0037024614  0.0023329917 -0.0029648598
##           LMT            DE           CAT            GE          AAPL
##  0.0023843881  0.0253909910  0.0158122983  0.0088067533  0.0201208661
##          MSFT          ADBE          INTU          NVDA           CRM
##  0.0121751896  0.0032335955  0.0118238490  0.0483448277  0.0041721502
```

```r
# treynor measure
r_m <- mean(r_test[,1])
bm <- cov(r_test)[1,-1]/cov(r_test)[1,1]
tm <- (colMeans(r_test[,-1]) - Rf)/bm
tm
```

```
##           MCD           NKE          SBUX             F           CMG
##  0.0077278956  0.0058109758  0.0054973483  0.0108627914  0.0134212711
##          LULU             V           JPM            MA           BAC
##  0.0095981734  0.0019083523  0.0010705017  0.0029257373 -0.0010857564
##            MS           WFC           JNJ           UNH           PFE
##  0.0122155865 -0.0022119882 -0.0005986683  0.0181831191  0.0084262033
##           CVS            CI           ZTS           RTX            BA
##  0.0037753172  0.0117976694  0.0045820751  0.0025944005 -0.0019246002
##           LMT            DE           CAT            GE          AAPL
##  0.0044716865  0.0261779558  0.0164402949  0.0070262446  0.0158569035
##          MSFT          ADBE          INTU          NVDA           CRM
##  0.0134223213  0.0024614347  0.0092487561  0.0303570062  0.0031722011
```

```r
# jensen differential performance index
jensen <- colMeans(r_train[,-1]) - (r_f + bm * (r_m - r_f))
jensen
```

```
##           MCD           NKE          SBUX             F           CMG
##  0.0087567559  0.0075083437  0.0064921122 -0.0109289927 -0.0020220398
##          LULU             V           JPM            MA           BAC
##  0.0171081315  0.0118472658  0.0116159818  0.0152383344  0.0095044658
##            MS           WFC           JNJ           UNH           PFE
##  0.0020504737 -0.0036360518  0.0031450751  0.0129420659  0.0008899140
##           CVS            CI           ZTS           RTX            BA
## -0.0068117926  0.0069532105  0.0138119637  0.0002266804  0.0090495632
##           LMT            DE           CAT            GE          AAPL
##  0.0093918212  0.0082807333  0.0079654850 -0.0151190471  0.0109830088
##          MSFT          ADBE          INTU          NVDA           CRM
##  0.0189829320  0.0190439661  0.0123769715  0.0415706613  0.0110603404
```

4. Decompose the overall performance using Fama's decomposition (net selectivity and diversification) for the single index model when short sales are not allowed. Please show this decomposition on the plot expected return against beta.

```r
# Get the number of portfolios
n_portfolios <- ncol(r_train[,-1])

# Prepare a vector to store the beta values
beta <- numeric(n_portfolios)

# Loop over the portfolios and run the regression for each one
for (i in 1:n_portfolios) {
  model <- lm(r_train[,i+1] ~ r_train[,1])
  beta[i] <- coef(model)[2]
}

# Now you can plot
plot(beta, colMeans(r_train[,-1]), ylim = c(0, 0.025), xlim = c(0,1.5), xlab = "Beta", ylab = "Expected
abline(a = Rf, b = mean(r_m - Rf), col = "blue")  # Capital Market Line
```