HW10

The Single-Index Model

4/9/2021

pacman::p\_load(pacman, tidyverse, tseries, psych,knitr)  
knitr::opts\_chunk$set(message = FALSE, tidy = T, fig.align = "center")

## Question 9.2

# Parameters  
beta = c(1.1, 0.9, 0.4, 1.5) # the asset betas  
sig\_eps1 = 0.2 # sd of epsilon for asset 1  
sig\_eps2 = 0.4 # sd of epsilon for asset 2  
sig\_eps3 = 0.5 # sd of epsilon for asset 3  
sig\_eps4 = 0.8 # sd of epsilon for asset 4  
sig\_m = 0.1 # sd of the market  
  
# model  
Sig1 = (sig\_m^2) \* (beta %\*% t(beta)) + diag(c(sig\_eps1, sig\_eps2, sig\_eps3,sig\_eps4)^2)  
Sig1

## [,1] [,2] [,3] [,4]  
## [1,] 0.0521 0.0099 0.0044 0.0165  
## [2,] 0.0099 0.1681 0.0036 0.0135  
## [3,] 0.0044 0.0036 0.2516 0.0060  
## [4,] 0.0165 0.0135 0.0060 0.6625

#The corresponding correlation matrix can be obtained using the function cov2cor:  
cov2cor(Sig1)

## [,1] [,2] [,3] [,4]  
## [1,] 1.00000000 0.10578704 0.03843072 0.08881210  
## [2,] 0.10578704 1.00000000 0.01750505 0.04045358  
## [3,] 0.03843072 0.01750505 1.00000000 0.01469613  
## [4,] 0.08881210 0.04045358 0.01469613 1.00000000

#Recall that, for a given asset, β\_i=ρ\_i (σ\_i/σ\_m) where σ\_i^2=Var(R\_(i,t))  
#Therefore, the vector of correlations (ρ\_1,ρ\_2,ρ\_3,ρ\_4,ρ\_5 )^T of each asset’s return with the returns on the market index is given by  
cor\_vec = beta \* (sig\_m/(diag(Sig1)^0.5))  
cor\_vec

## [1] 0.48191875 0.21951220 0.07974522 0.18428854

#Products of the form ρ\_i ρ\_j may be obtained from the off-diagonal elements of  
cor\_vec %\*% t(cor\_vec)

## [,1] [,2] [,3] [,4]  
## [1,] 0.23224568 0.10578704 0.03843072 0.08881210  
## [2,] 0.10578704 0.04818560 0.01750505 0.04045358  
## [3,] 0.03843072 0.01750505 0.00635930 0.01469613  
## [4,] 0.08881210 0.04045358 0.01469613 0.03396226

#Note that the off-diagonal elements are identical to what we computed with cov2cor(Sig1).

## Question 9.4

#Step 1. Compute monthly returns  
#a. Risk-free rate, rfree  
data1 = read.table(file.choose(),header = T,sep = ",")  
data1

## TimePeriod Yield  
## 1 2011-01 0.15  
## 2 2011-02 0.13  
## 3 2011-03 0.10  
## 4 2011-04 0.06  
## 5 2011-05 0.04  
## 6 2011-06 0.04  
## 7 2011-07 0.04  
## 8 2011-08 0.02  
## 9 2011-09 0.01  
## 10 2011-10 0.02  
## 11 2011-11 0.01  
## 12 2011-12 0.01  
## 13 2012-01 0.03  
## 14 2012-02 0.09  
## 15 2012-03 0.08  
## 16 2012-04 0.08  
## 17 2012-05 0.09  
## 18 2012-06 0.09  
## 19 2012-07 0.10  
## 20 2012-08 0.10  
## 21 2012-09 0.11  
## 22 2012-10 0.10  
## 23 2012-11 0.09  
## 24 2012-12 0.07  
## 25 2013-01 0.07  
## 26 2013-02 0.10  
## 27 2013-03 0.09  
## 28 2013-04 0.06  
## 29 2013-05 0.04  
## 30 2013-06 0.05  
## 31 2013-07 0.04  
## 32 2013-08 0.04  
## 33 2013-09 0.02  
## 34 2013-10 0.05  
## 35 2013-11 0.07  
## 36 2013-12 0.07  
## 37 2014-01 0.04  
## 38 2014-02 0.05  
## 39 2014-03 0.05  
## 40 2014-04 0.03  
## 41 2014-05 0.03  
## 42 2014-06 0.04  
## 43 2014-07 0.03  
## 44 2014-08 0.03  
## 45 2014-09 0.02  
## 46 2014-10 0.02  
## 47 2014-11 0.02  
## 48 2014-12 0.03  
## 49 2015-01 0.03  
## 50 2015-02 0.02  
## 51 2015-03 0.03  
## 52 2015-04 0.02  
## 53 2015-05 0.02  
## 54 2015-06 0.02  
## 55 2015-07 0.03  
## 56 2015-08 0.07  
## 57 2015-09 0.02  
## 58 2015-10 0.02  
## 59 2015-11 0.11  
## 60 2015-12 0.22

rff1 = data1$Yield  
table(is.na(rff1))

##   
## FALSE   
## 60

rff1

## [1] 0.15 0.13 0.10 0.06 0.04 0.04 0.04 0.02 0.01 0.02 0.01 0.01 0.03 0.09 0.08  
## [16] 0.08 0.09 0.09 0.10 0.10 0.11 0.10 0.09 0.07 0.07 0.10 0.09 0.06 0.04 0.05  
## [31] 0.04 0.04 0.02 0.05 0.07 0.07 0.04 0.05 0.05 0.03 0.03 0.04 0.03 0.03 0.02  
## [46] 0.02 0.02 0.03 0.03 0.02 0.03 0.02 0.02 0.02 0.03 0.07 0.02 0.02 0.11 0.22

# convert from percentages to proportional monthly returns  
rfree1 = (1 + rff1/100)^(1/12) - 1  
head(rfree1)

## [1] 1.249141e-04 1.082688e-04 8.329516e-05 4.998626e-05 3.332722e-05  
## [6] 3.332722e-05

#b. S&P 500  
x = get.hist.quote(instrument = "^GSPC",  
 start = "2010-12-01",  
 end = "2015-12-31",  
 quote = "AdjClose",  
 compression = "m")

## time series ends 2015-12-01

## time series ends 2015-12-01  
sp500 = as.vector(x)  
n = length(sp500)  
  
# Net returns  
sp500\_m\_ret = (sp500[-1] - sp500[-n])/sp500[-n]  
  
# Excess returns  
d = tibble(SP500 = sp500\_m\_ret - rfree1)  
d

## # A tibble: 60 x 1  
## SP500  
## <dbl>  
## 1 0.0225   
## 2 0.0318   
## 3 -0.00113  
## 4 0.0284   
## 5 -0.0135   
## 6 -0.0183   
## 7 -0.0215   
## 8 -0.0568   
## 9 -0.0718   
## 10 0.108   
## # ... with 50 more rows

# Fidelity Select Semiconductors Portfolio (symbol FSELX)  
x = get.hist.quote(instrument = "FSELX", start = "2010-12-01", end = "2015-12-31",   
 quote = "AdjClose", compression = "m") # monthly

## time series ends 2015-12-01

## time series ends 2015-12-01  
fselx\_m = as.vector(x)  
n = length(fselx\_m)  
  
fselx\_m\_ret = (fselx\_m[-1] - fselx\_m[-n])/fselx\_m[-n]  
  
stks = tibble(FSELX = fselx\_m\_ret - rfree1)  
stks

## # A tibble: 60 x 1  
## FSELX  
## <dbl>  
## 1 0.0770   
## 2 0.0251   
## 3 -0.0516   
## 4 0.0703   
## 5 -0.00169  
## 6 -0.0711   
## 7 -0.0365   
## 8 -0.0996   
## 9 -0.0418   
## 10 0.125   
## # ... with 50 more rows

# Fidelity Select Energy Portfolio (FSENX)  
x = get.hist.quote(instrument = "FSENX", start = "2010-12-01", end = "2015-12-31",   
 quote = "AdjClose", compression = "m") # monthly

## time series ends 2015-12-01

## time series ends 2015-12-01  
fsenx\_m = as.vector(x)  
n = length(fsenx\_m)  
  
fsenx\_m\_ret = (fsenx\_m[-1] - fsenx\_m[-n])/fsenx\_m[-n]  
  
stks = stks %>% mutate(FSENX = fsenx\_m\_ret - rfree1)  
stks

## # A tibble: 60 x 2  
## FSELX FSENX  
## <dbl> <dbl>  
## 1 0.0770 0.0830   
## 2 0.0251 0.0686   
## 3 -0.0516 0.0243   
## 4 0.0703 0.00319  
## 5 -0.00169 -0.0401   
## 6 -0.0711 -0.0273   
## 7 -0.0365 0.0135   
## 8 -0.0996 -0.119   
## 9 -0.0418 -0.170   
## 10 0.125 0.206   
## # ... with 50 more rows

# Fidelity Select Health Care Services Portfolio (FSHCX)  
x = get.hist.quote(instrument = "FSHCX", start = "2010-12-01", end = "2015-12-31",   
 quote = "AdjClose", compression = "m") # monthly

## time series ends 2015-12-01

## time series ends 2015-12-01  
fshcx\_m = as.vector(x)  
n = length(fshcx\_m)  
  
fshcx\_m\_ret = (fshcx\_m[-1] - fshcx\_m[-n])/fshcx\_m[-n]  
  
stks = stks %>% mutate(FSHCX = fshcx\_m\_ret - rfree1)  
stks

## # A tibble: 60 x 3  
## FSELX FSENX FSHCX  
## <dbl> <dbl> <dbl>  
## 1 0.0770 0.0830 0.0534  
## 2 0.0251 0.0686 0.0568  
## 3 -0.0516 0.0243 0.0388  
## 4 0.0703 0.00319 0.0346  
## 5 -0.00169 -0.0401 0.0144  
## 6 -0.0711 -0.0273 -0.0105  
## 7 -0.0365 0.0135 -0.0262  
## 8 -0.0996 -0.119 -0.0785  
## 9 -0.0418 -0.170 -0.0853  
## 10 0.125 0.206 0.105   
## # ... with 50 more rows

# Fidelity Real Estate Investment Portfolio (FRESX)  
x = get.hist.quote(instrument = "FRESX", start = "2010-12-01", end = "2015-12-31",   
 quote = "AdjClose", compression = "m") # monthly

## time series ends 2015-12-01

## time series ends 2015-12-01  
fresx\_m = as.vector(x)  
n = length(fresx\_m)  
  
fresx\_m\_ret = (fresx\_m[-1] - fresx\_m[-n])/fresx\_m[-n]  
  
stks = stks %>% mutate(FRESX = fresx\_m\_ret - rfree1)  
stks

## # A tibble: 60 x 4  
## FSELX FSENX FSHCX FRESX  
## <dbl> <dbl> <dbl> <dbl>  
## 1 0.0770 0.0830 0.0534 0.0395  
## 2 0.0251 0.0686 0.0568 0.0447  
## 3 -0.0516 0.0243 0.0388 -0.0145  
## 4 0.0703 0.00319 0.0346 0.0545  
## 5 -0.00169 -0.0401 0.0144 0.0142  
## 6 -0.0711 -0.0273 -0.0105 -0.0332  
## 7 -0.0365 0.0135 -0.0262 0.0177  
## 8 -0.0996 -0.119 -0.0785 -0.0616  
## 9 -0.0418 -0.170 -0.0853 -0.114   
## 10 0.125 0.206 0.105 0.155   
## # ... with 50 more rows

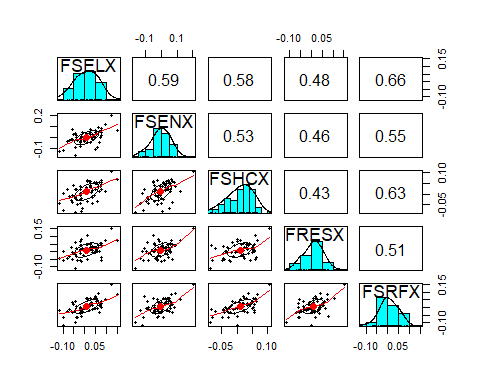
# Fidelity Select Transportation Portfolio (FSRFX)  
x = get.hist.quote(instrument = "FSRFX", start = "2010-12-01", end = "2015-12-31",   
 quote = "AdjClose", compression = "m") # monthly

## time series ends 2015-12-01

## time series ends 2015-12-01  
fsrfx\_m = as.vector(x)  
n = length(fsrfx\_m)  
  
fsrfx\_m\_ret = (fsrfx\_m[-1] - fsrfx\_m[-n])/fsrfx\_m[-n]  
  
stks = stks %>% mutate(FSRFX = fsrfx\_m\_ret - rfree1)  
stks

## # A tibble: 60 x 5  
## FSELX FSENX FSHCX FRESX FSRFX  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0.0770 0.0830 0.0534 0.0395 0.000334  
## 2 0.0251 0.0686 0.0568 0.0447 0.0174   
## 3 -0.0516 0.0243 0.0388 -0.0145 0.0214   
## 4 0.0703 0.00319 0.0346 0.0545 0.0116   
## 5 -0.00169 -0.0401 0.0144 0.0142 -0.00528   
## 6 -0.0711 -0.0273 -0.0105 -0.0332 -0.0142   
## 7 -0.0365 0.0135 -0.0262 0.0177 -0.0683   
## 8 -0.0996 -0.119 -0.0785 -0.0616 -0.0665   
## 9 -0.0418 -0.170 -0.0853 -0.114 -0.100   
## 10 0.125 0.206 0.105 0.155 0.149   
## # ... with 50 more rows

#Exploratory data analysis  
pairs.panels(stks)



#Partial correlation  
#What is the sample correlation of the returns on Fidelity Select Semiconductors Portfolio and Fidelity Select Energy Portfolio  
cor(stks$FSELX, stks$FSENX)

## [1] 0.5942167

p\_load(ppcor)  
pcor.test(stks$FSELX, stks$FSENX, d$SP500)

## estimate p.value statistic n gp Method  
## 1 0.09583273 0.4702824 0.7268667 60 1 pearson

#Calculate the partial correlation for all pairs  
pcor = pvalue = matrix(0, 5, 5)  
  
# compute  
for (i in 1:4) {  
 for (j in (i + 1):5) {  
 res = pcor.test(stks[, i], stks[, j], d$SP500)  
 pcor[i, j] = res[1, 1]  
 pvalue[i, j] = res[1, 2]  
 }  
}  
  
# add names to rows and columns  
rownames(pcor) = colnames(stks)  
colnames(pcor) = colnames(stks)  
  
rownames(pvalue) = colnames(stks)  
colnames(pvalue) = colnames(stks)  
  
# partial correlation coefficients of the vector of returns  
print(pcor)

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 0 0.09583273 0.17261108 0.03519150 0.20507468  
## FSENX 0 0.00000000 0.07123968 0.01647449 -0.01674605  
## FSHCX 0 0.00000000 0.00000000 0.03432812 0.25636011  
## FRESX 0 0.00000000 0.00000000 0.00000000 0.08360241  
## FSRFX 0 0.00000000 0.00000000 0.00000000 0.00000000

#Here’s another version of a table of partial correlations.  
print(as.table(pcor), zero.print = ".")

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX . 0.09583273 0.17261108 0.03519150 0.20507468  
## FSENX . . 0.07123968 0.01647449 -0.01674605  
## FSHCX . . . 0.03432812 0.25636011  
## FRESX . . . . 0.08360241  
## FSRFX . . . . .

print(pvalue)

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 0 0.4702824 0.1911039 0.7913104 0.11920568  
## FSENX 0 0.0000000 0.5918366 0.9014390 0.89982258  
## FSHCX 0 0.0000000 0.0000000 0.7963192 0.05000794  
## FRESX 0 0.0000000 0.0000000 0.0000000 0.52900468  
## FSRFX 0 0.0000000 0.0000000 0.0000000 0.00000000

#Here’s another version of a table of p-values.  
print(as.table(pvalue), zero.print = ".")

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX . 0.47028236 0.19110393 0.79131042 0.11920568  
## FSENX . . 0.59183663 0.90143899 0.89982258  
## FSHCX . . . 0.79631919 0.05000794  
## FRESX . . . . 0.52900468  
## FSRFX . . . . .

#The relatively large p-values suggest that there is no evidence that the single-index model is inappropriate.

## Question 9.6

# note: I convert the stks to a matrix to regress multiple variables on SP500  
stks.mm = lm(as.matrix(stks) ~ SP500, data = d)  
# or stks.mm = lm(as.matrix(stks) ~ d$SP500)  
  
# coefficients  
stks.mm$coefficients

## FSELX FSENX FSHCX FRESX FSRFX  
## (Intercept) -0.001473252 -0.01619906 0.002639488 0.002530522 -0.003147802  
## SP500 1.238971298 1.43194349 0.815446870 0.855638656 1.057797894

#Extract the intercepts  
stks.alpha = stks.mm$coefficients[1, ] # first row, all columns  
stks.alpha

## FSELX FSENX FSHCX FRESX FSRFX   
## -0.001473252 -0.016199058 0.002639488 0.002530522 -0.003147802

#Extract the beta coefficients  
stks.beta = stks.mm$coefficients[2, ]  
stks.beta

## FSELX FSENX FSHCX FRESX FSRFX   
## 1.2389713 1.4319435 0.8154469 0.8556387 1.0577979

#Estimate the standard deviation  
# First, define the f.sighat function to compute multiple sigma estimates.  
f.sighat = function(y) {  
 summary(lm(y ~ d$SP500))$sigma  
}  
  
# Now, compute the sigma estimates  
stks.s = apply(stks, 2, f.sighat)  
stks.s

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.03725984 0.04494777 0.03117458 0.03773250 0.03054574

#1. Compute the residual standard deviation (i.e., the sigma of epsilon)  
stks.Sigeps = diag(stks.s^2)  
rownames(stks.Sigeps) = colnames(stks.Sigeps) = c(labels(stks.s))  
  
print(as.table(stks.Sigeps), zero.print = ".")

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 0.0013882956 . . . .  
## FSENX . 0.0020203017 . . .  
## FSHCX . . 0.0009718547 . .  
## FRESX . . . 0.0014237412 .  
## FSRFX . . . . 0.0009330423

#2. Compute estimate ββ^T  
stks.beta %\*% t(stks.beta)

## FSELX FSENX FSHCX FRESX FSRFX  
## [1,] 1.535050 1.774137 1.0103153 1.0601117 1.3105812  
## [2,] 1.774137 2.050462 1.1676738 1.2252262 1.5147068  
## [3,] 1.010315 1.167674 0.6649536 0.6977279 0.8625780  
## [4,] 1.060112 1.225226 0.6977279 0.7321175 0.9050928  
## [5,] 1.310581 1.514707 0.8625780 0.9050928 1.1189364

#3. Now compute estimate of Σ  
stks.Sig = var(c(d$SP500)) \* (stks.beta %\*% t(stks.beta)) + diag(stks.s^2)  
  
print(stks.Sig)

## FSELX FSENX FSHCX FRESX FSRFX  
## [1,] 0.003138917 0.002023284 0.0011521966 0.0012089861 0.0014946297  
## [2,] 0.002023284 0.004358716 0.0013316534 0.0013972880 0.0017274211  
## [3,] 0.001152197 0.001331653 0.0017301896 0.0007957117 0.0009837121  
## [4,] 0.001208986 0.001397288 0.0007957117 0.0022586719 0.0010321974  
## [5,] 0.001494630 0.001727421 0.0009837121 0.0010321974 0.0022091139

cov(stks)

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 0.003115386 0.002181059 0.0013492964 0.0012576235 0.001724075  
## FSENX 0.002181059 0.004324473 0.0014297845 0.0014247550 0.001704819  
## FSHCX 0.001349296 0.001429785 0.0017137174 0.0008354072 0.001223694  
## FRESX 0.001257623 0.001424755 0.0008354072 0.0022345407 0.001126922  
## FSRFX 0.001724075 0.001704819 0.0012236936 0.0011269216 0.002193300

diag(stks.Sig)^0.5

## [1] 0.05602604 0.06602057 0.04159555 0.04752549 0.04700121

diag(cov(stks))^0.5

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.05581565 0.06576073 0.04139707 0.04727093 0.04683268

#The correlation matrix corresponding to stks.Sig…  
cov2cor(stks.Sig)

## FSELX FSENX FSHCX FRESX FSRFX  
## [1,] 1.0000000 0.5470004 0.4944129 0.4540512 0.5675899  
## [2,] 0.5470004 1.0000000 0.4849144 0.4453281 0.5566855  
## [3,] 0.4944129 0.4849144 1.0000000 0.4025152 0.5031670  
## [4,] 0.4540512 0.4453281 0.4025152 1.0000000 0.4620906  
## [5,] 0.5675899 0.5566855 0.5031670 0.4620906 1.0000000

cor(stks)

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 1.0000000 0.5942167 0.5839583 0.4766510 0.6595554  
## FSENX 0.5942167 1.0000000 0.5252117 0.4583312 0.5535575  
## FSHCX 0.5839583 0.5252117 1.0000000 0.4269082 0.6311813  
## FRESX 0.4766510 0.4583312 0.4269082 1.0000000 0.5090384  
## FSRFX 0.6595554 0.5535575 0.6311813 0.5090384 1.0000000

cov2cor(stks.Sig) - cor(stks)

## FSELX FSENX FSHCX FRESX FSRFX  
## [1,] 0.00000000 -0.047216329 -0.08954531 -0.02259984 -0.091965513  
## [2,] -0.04721633 0.000000000 -0.04029727 -0.01300315 0.003128023  
## [3,] -0.08954531 -0.040297268 0.00000000 -0.02439298 -0.128014308  
## [4,] -0.02259984 -0.013003147 -0.02439298 0.00000000 -0.046947855  
## [5,] -0.09196551 0.003128023 -0.12801431 -0.04694785 0.000000000

#Another approach to estimating the covariance matrix of the asset returns: use a shrinkage estimate.  
p\_load(ShrinkCovMat)  
  
stks.shrink = shrinkcovmat.equal(t(stks))$Sigmahat  
diag(stks.shrink)^0.5

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.05546103 0.06454044 0.04257749 0.04777208 0.04738151

#The shrinkage estimate of the correlation matrix is given by  
cov2cor(stks.shrink)

## FSELX FSENX FSHCX FRESX FSRFX  
## FSELX 1.0000000 0.5490781 0.5149037 0.4277354 0.5912157  
## FSENX 0.5490781 1.0000000 0.4688622 0.4164096 0.5023704  
## FSHCX 0.5149037 0.4688622 1.0000000 0.3701099 0.5466009  
## FRESX 0.4277354 0.4164096 0.3701099 1.0000000 0.4486393  
## FSRFX 0.5912157 0.5023704 0.5466009 0.4486393 1.0000000

#Although the three estimates of the return correlation matrix are similar, there are some differences.  
#• According to the sample covariance matrix and the shrinkage estimate of the covariance matrix, the returns of these stocks are negatively correlated.  
#• However, every stock have positive betas.  
#• Therefore, according to the single-index model, every stocks is positively correlated with the market and, hence, with each other.  
#Because the shrinkage estimate is a weighted average of the sample covariance matrix and a scaled identity matrix, it is not surprising that the shrinkage correlation estimates are closer to zero than are the sample correlations.

## Question 9.8

wgt = solve(stks.Sig, stks.alpha + stks.beta \* mean(d$SP500))  
w\_T\_si = wgt/sum(wgt)  
w\_T\_si

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.2971914 -0.6989576 0.7633261 0.5248531 0.1135871

w\_1T = solve(cov(stks), apply(stks, 2, mean))  
w\_T = w\_1T/sum(w\_1T)  
w\_T

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.3794741 -0.7570615 0.9307911 0.6180158 -0.1712195

apply(stks, 2, mean)

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.009228159 -0.003830881 0.009682777 0.009920961 0.005988754

stks.alpha + stks.beta \* mean(d$SP500)

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.009228159 -0.003830881 0.009682777 0.009920961 0.005988754

w.sh1 <- solve(stks.shrink, apply(stks, 2, mean))  
w.sh1/sum(w.sh1)

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.37586163 -0.71979096 0.80212165 0.59329445 -0.05148677

stks.beta/(stks.s^2)/sum(stks.beta/(stks.s^2))

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.2137599 0.1697683 0.2009746 0.1439482 0.2715490

## Question 9.10

#The weights for each asset:  
wbar0 = (stks.alpha/stks.s^2)/sum(stks.alpha/stks.s^2)  
wbar0

## FSELX FSENX FSHCX FRESX FSRFX   
## 0.1333205 1.0073384 -0.3412088 -0.2232960 0.4238459

#The weight given to the market index using this approach is given by  
c1 = mean(SP500)/var(SP500) - sum(stks.alpha \* stks.beta/stks.s^2)  
  
wm = c1/(c1 + sum(stks.alpha/stks.s^2))  
wm

## [1] 2.686124

1 - wm

## [1] -1.686124

#The remainder 1 - wm is invested in the portfolio with the weights given in the variable wbar0  
  
#Alternatively, the Treynor-Black portfolio may be described in terms of the weight vector for the six assets (the five stocks and the market portfolio)  
  
c((1 - wm) \* wbar0, wm)

## FSELX FSENX FSHCX FRESX FSRFX   
## -0.2247949 -1.6984974 0.5753203 0.3765047 -0.7146566 2.6861239