Homework 3

Joyce Yu Cahoon

7.1

What is the gross exposure of the portfolio with weights:

$$w = (-0.2, 0.3, 0.4, -0.2, 0.1, 0.2, 0, 0.4)$$

What is the risk of this portfolio invested on "Dell", "Ford", "GE", "IBM", "J&J", "Merck", "3 mo Treasury", "S&P500" in past ten years (January 1, 2005 to January 1, 2015 using daily data). Compare it with the portfolio with equal weight.

The gross exposure can be calculated simply as:

```
w <- c(-0.2, 0.3, 0.4, -0.2, 0.1, 0.2, 0, 0.4)
sum(abs(w))
```

```
## [1] 1.8
```

The gross exposure with equal weights is:

```
w <- rep(1, length(w))/length(w)
sum(abs(w))
```

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

To calculate the risk of the original portfolio. First, we obtain the data we need. Note that we remove "DELL" as that is not provided from Yahoo!Finance:

```
# load the necessary financial data from quantmod
library(quantmod)
start <- as.Date("2005-01-01")
end <- as.Date("2015-01-01")
assets <- c(# "DVMT" # DELL is DROPPED since not provided by Quantmod
  "F",
  "GE",
  "IBM",
  "JNJ",
  "MRK",
  "SPY")
getSymbols(assets, from = start, to = end)
```

```r
# get the 3 mo tresury data
getSymbols("DGS3MO", src = "FRED")
# merge into one df
closing.prices <- merge.xts(DGS3MO,
                            F[, 4],
                            GE[, 4],
                            IBM[, 4],
                            JNJ[, 4],
                            MRK[, 4],
                            SPY[, 4])
# save as RDS
saveRDS(closing.prices, "~/workspace/st790-financial-stats/hw4/closing.rds")
```

With the daily log returns, we can then find:

```r
dat <- readRDS("~/workspace/st790-financial-stats/hw4/closing.rds")
# filter out the information for the timeframe of interest
data <- dat["2005-01-01/2015-01-01"]
data <- na.omit(data)
# get daily log returns
by_day <- data.frame()
for(i in 1:ncol(data)){
  temp <- data[, i]
  daily <- log(dailyReturn(temp)+1)
  by_day <- cbind(by_day, daily)
  colnames(by_day)[i] <- strsplit(names(temp), "[.]")[[1]][1]
}
# get the excess return
excess_return <- data.frame()
for(j in 2:ncol(by_day)){
  temp <- by_day[, j] - by_day[, 1]
  excess_return <- cbind(excess_return, temp)
  colnames(excess_return)[j-1] <- names(temp)
}
# get the covariance matrix
r <- cbind(by_day[,1], excess_return)
r[is.infinite(r)] <- NA
r <- na.omit(r)
Sigma <- cov(r)
# weight vector given
w <- c( 0.3, 0.4, -0.2, 0.1, 0.2, 0, 0.4)
w <- w/sum(w) # need to normalize since we removed DELL
# risk of portfolio
risk <- sqrt(w %*% Sigma %*% w)
# print
cat("The risk is: ", risk, "\n")
```

```
## The risk is:  0.1148775
```

```r
# equal weight
w2 <- rep(1, length(w))/length(w)
# new risk
risk2 <- sqrt(w2 %*% Sigma %*% w2)
# print
cat("The new risk is: ", risk2, "\n")
```

```
## The new risk is:  0.1633469
```

7.2

Let $X_1, \ldots, X_T$ be a sequence of stationary time series with the autocovariance function $\gamma(h) = cov(X_t, X_{t+h})$ and $\bar{X} = T^{-1} \sum_{t=1}^{T} X_t$. Show that:

$$var(\bar{X}) = T^{-2}[T\gamma(0) + 2(T-1)\gamma(1) + \cdots + 2\gamma(T-1)] \quad \text{and,}$$

$$\lim_{T \to \infty} [Tvar(\bar{X})] = \gamma(0) + 2\sum_{h=1}^{\infty} \gamma(h)$$

In other words, for a sufficiently large L:

$$var(\bar{X}) \approx T^{-1}[\gamma(0) + 2\sum_{h=1}^{L} \gamma(h)]$$

7.3

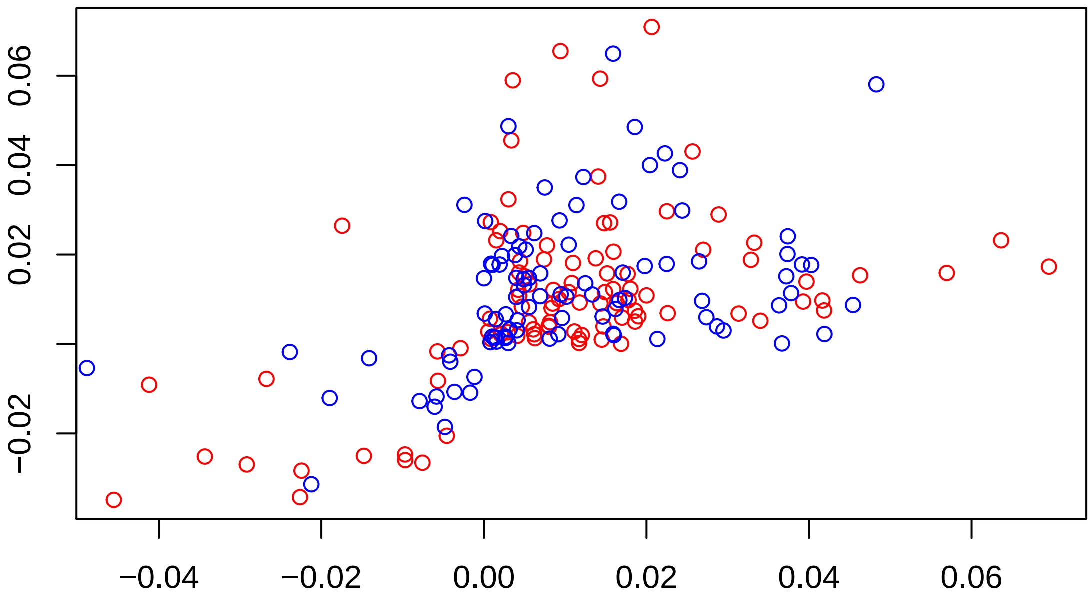
Draw 10,000 random portfolios of size $p = 100$ with gross exposure $c = 1.6$ from (7.9). Plot the weights (w_1, w_2) and (w_{99}, w_{100}).

To draw 1 random portfolio of this size, we carry out the following:

```
c <- 1.6
p <- 100
# 1. generate a positive integer k, the number of stocks with positive weights
set.seed(123)
k <- rbinom(n = 1, size = p, prob = (c+1)/(2*c))
# 2. generate independently from std exponential distribution
positive <- rexp(k)
wP <- (c+1)*positive / (2*sum(positive))
# 3. generate the negative weights
negative <- rexp(p-k)
wN <- (1-c)*negative / (2*sum(negative))
# 4. get our random permutation of w
portfolio_weights <- c(wP, wN)
```

We can thus create a function for our simulation process above to generate 10,000 portfolios:

```
generate_weights <- function(userseed, c = 1.6, p = 100){
  set.seed(userseed)
  k <- rbinom(n = 1, size = p, prob = (c+1)/(2*c))
  positive <- rexp(k)
  wP <- (c+1)*positive / (2*sum(positive))
  negative <- rexp(p-k)
  wN <- (1-c)*negative / (2*sum(negative))
  return(c(wP, wN))
}
portfolio_10000 <- sapply(c(1:10000), function(x) generate_weights(userseed = x))
# plot (w1, w2)
plot(portfolio_10000[,1], portfolio_10000[,2], xlab = "", ylab = "", col = "red")
# add in (w99, w100)
points(portfolio_10000[,99], portfolio_10000[,100], col = "blue")
```



7.5

Estimate the time-varying volatility matrix of the eight risk factors presented in Example 7.5 using the last 3-month data and the estimator (7.15) with $\lambda = 0.1$ (No estimate is needed for the initial 3 months). Present the results for the exchange rates and the S&P 500 index, similar to Figure 7.3. Repeat the exercise by using the second projection method in Section 7.2.2 and compare the results.

The eight risk factors include: (1) market risk, or the returns of the S&P 500; (2) volatility risk, or the VIX index; (3) exchange rate risk, or the index of the US dollar value; (4) maturity risk, or the yield spread between the 10 year US treasury and the 3 year treasury bills; (5) default risk, or the yield spread between AAA and BBB corporate bonds; (6) liquidity risk, or the difference between 1 month repo rates and 1 month treasury bill rates; (7) size effect, or the difference of returns between S&P500 and Russell 2000; and (8) short term rate, yields of the 3-month treasury bills.

```
library(quantmod)
start <- as.Date("2018-08-01")
end <- as.Date("2018-11-01") # getting the last 3 months data
# get the SPY
getSymbols("SPY", from = start, to = end)
# get the US dollar value, exchange rate
getSymbols("EUR=X", src="yahoo",from = start, to = end)
# get VIX
getSymbols("^VIX", src="yahoo",from = start, to = end)
# get yield spread of 3 year and 10 year treasury bills
getSymbols("T10Y3M", src = "FRED", from = start, to = end)
# get default risk based on ICE BofAML US Corporate AAA and BBB Effective Yield
getSymbols("BAMLC0A1CAAAEY", src = "FRED", from = start, to = end)
```

```r
getSymbols("BAMLC0A4CBBBEY", src = "FRED", from = start, to = end)
def <- BAMLC0A4CBBBEY - BAMLC0A1CAAAEY
# get the liquidity risk, rely on fed funds rates as surrogate for repo
getSymbols("DGS1MO", src = "FRED", from = start, to = end)
getSymbols("DFF", src = "FRED", from = start, to = end)
temp <- merge.xts(DGS1MO, DFF)["2018-08-01/2018-11-01"]
liquidity <- temp$DFF - temp$DGS1MO
# get the size effect
getSymbols("^RUA", src = "yahoo", from = start, to = end)
rua_returns <- dailyReturn(RUA)
spy_returns <- dailyReturn(SPY)
temp <- merge.xts(rua_returns, spy_returns)
size <- temp$daily.returns - temp$daily.returns.1
# get short term rate
getSymbols("DGS3MO", src = "FRED", from = start, to = end)
# combine all the returns
total <- merge.xts(spy_returns, # market risk
          `EUR=X`[,4], # exchange risk
          VIX[,4], # volatility risk
          T10Y3M, # maturity risk
          def, # default risk
          liquidity, # liquidity risk
          size, # size effect
          DGS3MO) # short term rate
names(total) <- c("market",
                  "exchange",
                  "volatility",
                  "maturity",
                  "default",
                  "liquidity",
                  "size",
                  "short")
total <- na.omit(total)
# make sure units make sense
total[,1] <- total[,1]*100 # convert to percent
total[,2] <- total[,2]*100 # convert exchange to percent
total[,7] <- total[,7]*100 # convert to basis pts
# save
saveRDS(total, "~/workspace/st790-financial-stats/hw4/risk.rds")
```

Now that we have aggregated all the necessary information, we can calculate the time varying volatility matrix. First, the correlation matrix:

```
##             market exchange volatility maturity default liquidity   size
## market       1.000   -0.044     -0.346   -0.104   0.287    -0.136  0.228
## exchange    -0.044    1.000      0.535   -0.132   0.135     0.537  0.109
```

```
## volatility -0.346    0.535       1.000    -0.086   -0.275      0.410  0.061
## maturity   -0.104   -0.132      -0.086     1.000   -0.607      0.129 -0.041
## default     0.287    0.135      -0.275    -0.607    1.000     -0.149  0.138
## liquidity  -0.136    0.537       0.410     0.129   -0.149      1.000  0.267
## size        0.228    0.109       0.061    -0.041    0.138      0.267  1.000
## short      -0.174    0.304       0.805     0.045   -0.566      0.276 -0.025
##             short
## market     -0.174
## exchange    0.304
## volatility  0.805
## maturity    0.045
## default    -0.566
## liquidity   0.276
## size       -0.025
## short       1.000
```

We can now apply thresholding to the correlation matrix:

```
temp <- cor(total)
temp2 <- temp*I(abs(temp) > .1)
round(temp2, 2)
```

```
##            market exchange volatility maturity default liquidity size
## market       1.00     0.00      -0.35    -0.10    0.29     -0.14 0.23
## exchange     0.00     1.00       0.53    -0.13    0.13      0.54 0.11
## volatility  -0.35     0.53       1.00     0.00   -0.27      0.41 0.00
## maturity    -0.10    -0.13       0.00     1.00   -0.61      0.13 0.00
## default      0.29     0.13      -0.27    -0.61    1.00     -0.15 0.14
## liquidity   -0.14     0.54       0.41     0.13   -0.15      1.00 0.27
## size         0.23     0.11       0.00     0.00    0.14      0.27 1.00
## short       -0.17     0.30       0.81     0.00   -0.57      0.28 0.00
##            short
## market     -0.17
## exchange    0.30
## volatility  0.81
## maturity    0.00
## default    -0.57
## liquidity   0.28
## size        0.00
## short       1.00
```

```
n <- nrow(total)
lambda <- .94 # for daily
Sigma <- array(0, c(2, 2, n+1))
change <- as.vector(diff(total[, 2])) # get the change for exchange rates
change[1] <- 0
```

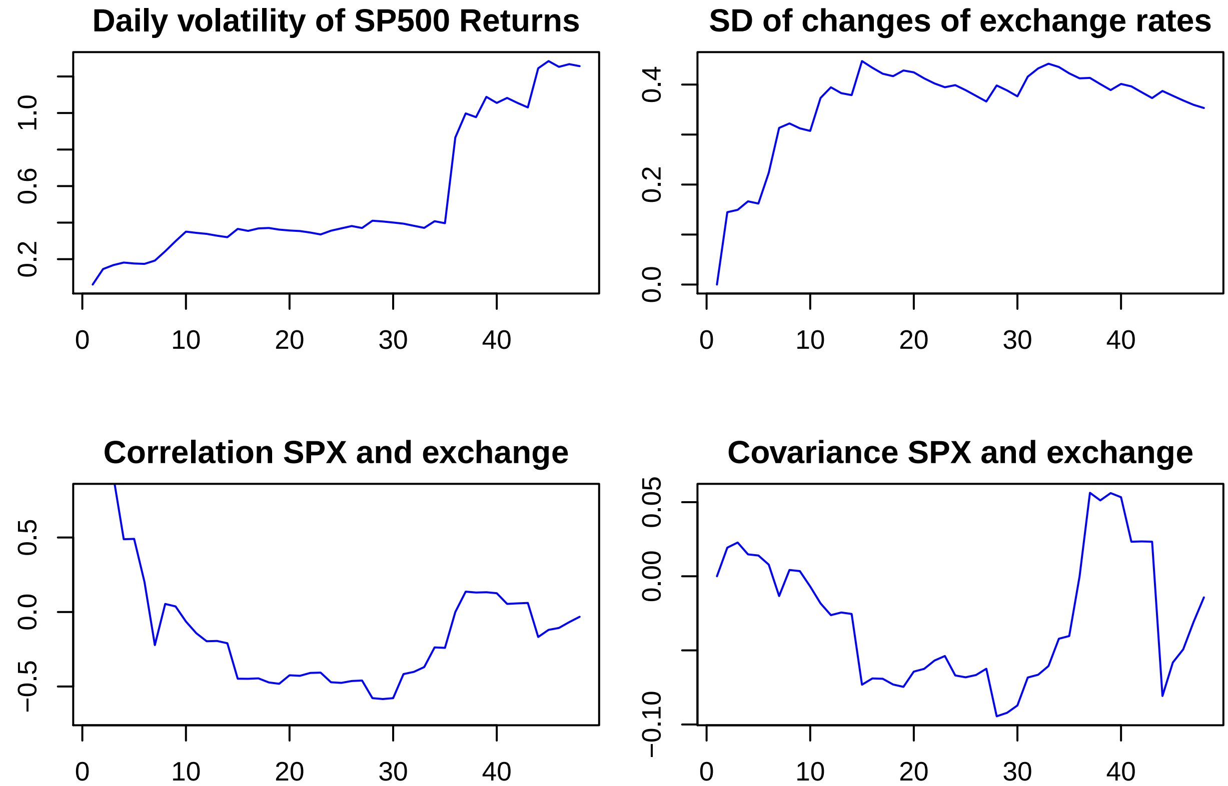
```
for(i in 1:n){
  tmp <- as.numeric(c(as.numeric(total[i, 1]), change[i]))
  Sigma[,,i+1] <- lambda*Sigma[,,i] + (1-lambda)*tmp %*% t(tmp)
}
Sigma <- Sigma[,,-1] # remove the initial values
leverage <- Sigma[1,2,]/sqrt(Sigma[1,1,])/sqrt(Sigma[2,2,])
# present the results like 7.3
par(mfrow = c(2,2), mar=c(4,2,2,1)+0.1, cex=0.8)

plot(sqrt(Sigma[1,1,]), type="l", col=4, xlab = "")
title("Daily volatility of SP500 Returns")

plot(sqrt(Sigma[2,2,]), type="l", col=4, xlab = "")
title("SD of changes of exchange rates")

plot(leverage, type="l", col=4, ylim=c(-.7,.8), xlab = "")
title("Correlation SPX and exchange")

plot(Sigma[2,1,], type="l", col=4, xlab = "")
title("Covariance SPX and exchange")
```



Using the second projection method, we want to compute:

$$\hat{\Sigma}^{+}_{\lambda} = (\hat{\Sigma}_{\lambda} + \lambda^{-}_{\min} I_p)/(1 + \lambda^{-}_{\min})$$

```r
# find the eigenvalues
min_eigen <- numeric(n)
for(i in 1:n){
  eigenval <- eigen(Sigma[,,i])$values
  min_eigen[i] <- min(eigenval)
}
```

Since the $\lambda^{-}_{\min}$ factor is always 0, our estimates do not change, and we have the same results as if we simply utilized the thresholding technique.