assignment05

Jeff Nguyen

18/09/2020

**University of Southern California**  
**Marshall School of Business**  
**FBE 506 Quantitative Method in Finance**

Assignment 05  
Student Name: Ngoc Son (Jeff) Nguyen  
Student ID: 6524885309

# **Question 1**

library(quantmod)

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: TTR

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Version 0.4-0 included new data defaults. See ?getSymbols.

# Set start date and end date of data  
start\_date <- "2014-01-02"  
end\_date <- "2020-09-18"  
  
# Get data for JPM, FB and the 10 year T-bill (TNX)  
getSymbols("JPM", src = "yahoo", from = start\_date, to = end\_date) # JPM

## 'getSymbols' currently uses auto.assign=TRUE by default, but will  
## use auto.assign=FALSE in 0.5-0. You will still be able to use  
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")  
## and getOption("getSymbols.auto.assign") will still be checked for  
## alternate defaults.  
##   
## This message is shown once per session and may be disabled by setting   
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## [1] "JPM"

getSymbols("FB", src = "yahoo", , from = start\_date, to = end\_date) # FB

## [1] "FB"

getSymbols("^TNX", src = "yahoo", from = start\_date, to = end\_date) # TNX (10-year T-bill)

## Warning: ^TNX contains missing values. Some functions will not work if objects  
## contain missing values in the middle of the series. Consider using na.omit(),  
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "^TNX"

# Get adjusted returns data for 01/2014  
rJPM <- diff(log(to.monthly(JPM)$JPM.Adjusted))  
rFB <- diff(log(to.monthly(FB)$FB.Adjusted))  
rTNX <- (to.monthly(TNX)$TNX.Adjusted) / 1200 # Using monthly rate

## Warning in to.period(x, "months", indexAt = indexAt, name = name, ...): missing  
## values removed from data

# Calculate statistics  
mean\_rJPM <- mean(rJPM, na.rm = TRUE)  
mean\_rFB <- mean(rFB, na.rm = TRUE)  
mean\_rTNX <- mean(rTNX, na.rm = TRUE)  
  
sd\_rJPM <- sd(rJPM, na.rm = TRUE)  
sd\_rFB <- sd(rFB, na.rm = TRUE)

## a. Coefficient of variation

For JPM:

# Coefficient of variation of Adjusted Returns of JPM  
cv\_rJPM <- sd\_rJPM / mean\_rJPM

For FB:

# Coefficient of variation of Adjusted Returns of FB  
cv\_rFB <- sd\_rFB / mean\_rFB

## b. Sharpe Ratio

In this case, each portfolio carry 100% of JPM of FB so weighting is 100% of each stock for each portfolio.

For JPM:

# Sharpe Ratio of a 100% JPM portfolio  
sharpe\_rJPM <- (mean\_rJPM - mean\_rTNX) / sd\_rJPM

For FB:

# Sharpe Ratio of a 100% JPM portfolio  
sharpe\_rFB <- (mean\_rFB - mean\_rTNX) / sd\_rFB

## b. Sortino Ratio

For the purpose of calculating the Sortino ratio, we’ll the mean of the Jan 2014 daily risk-free rate as the Minimum Acceptable Returns (MAR).

For JPM:

# Calculating downside deviation using lower partial moment of order 2  
# Assuming the 10 year T-bill returns as minimum acceptable returns MAR  
mar <- mean\_rTNX  
  
# Deviation from MAR, this is a data frame, remove NA values  
dev\_rJPM\_mar <- na.omit(rJPM - mar)  
  
# Get the subset of negative values  
devNegative\_rJPM\_mar <- subset(dev\_rJPM\_mar, dev\_rJPM\_mar < 0)  
  
# Calculate the Lower Partial Moment  
downsideDev\_JPM <- var(devNegative\_rJPM\_mar)  
  
# Downside deviation  
sd\_downsideDev\_JPM <- sqrt(downsideDev\_JPM)  
  
# Sortino Ratio  
sortino\_rJPM <- (mean\_rJPM - mean\_rTNX) / sd\_downsideDev\_JPM

For FB:

# Calculating downside deviation using lower partial moment of order 2  
# Assuming the 10 year T-bill returns as minimum acceptable returns MAR  
mar <- mean\_rTNX  
  
# Deviation from MAR, this is a data frame, also remove NA values  
dev\_rFB\_mar <- na.omit(rFB - mar)  
  
# Get the subset of negative values  
devNegative\_rFB\_mar <- subset(dev\_rFB\_mar, dev\_rFB\_mar < 0)  
  
# Calculate the Lower Partial Moment  
downsideDev\_FB <- var(devNegative\_rFB\_mar)  
  
# Downside deviation  
sd\_downsideDev\_FB <- sqrt(downsideDev\_FB)  
  
# Sortino Ratio  
sortino\_rFB <- (mean\_rFB - mean\_rTNX) / sd\_downsideDev\_FB

In summary:

|  |  |  |  |
| --- | --- | --- | --- |
| Portfolio | Coefficient of Variation | Sharpe Ratio | Sortino Ratio |
| JPM | 6.8327559 | 0.1191742 | 0.1460813 |
| FB | 4.2416176 | 0.2120413 | 0.3836041 |

For the period of Jan 2014:  
In terms of coefficient of variation, JPM shows more deviation from mean compared to FB, i.e. potentially having higher returns volatility.  
In terms of Sharpe Ratio, FB has better risk-adjusted returns performance compared to JPM.  
In terms of Sortino Ratio, FB also generates better risk-adjusted performance with more weighting on the downside risk.

# **Question 2**

# Set start date and end date of data  
start\_date <- "2018-01-02"  
end\_date <- "2020-09-18"  
  
# Get data for JPM, FB and the 10 year T-bill (TNX)  
getSymbols("JPM", src = "yahoo", from = start\_date, to = end\_date) # JPM

## [1] "JPM"

getSymbols("FB", src = "yahoo", from = start\_date, to = end\_date) # FB

## [1] "FB"

getSymbols("^TNX", src = "yahoo", from = start\_date, to = end\_date) # TNX (10-year T-bill)

## Warning: ^TNX contains missing values. Some functions will not work if objects  
## contain missing values in the middle of the series. Consider using na.omit(),  
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "^TNX"

# Get adjusted returns data for 01/2014  
rJPM <- diff(log(to.monthly(JPM)$JPM.Adjusted))  
rFB <- diff(log(to.monthly(FB)$FB.Adjusted))  
rTNX <- (to.monthly(TNX)$TNX.Adjusted) / 1200 # Using monthly rate

## Warning in to.period(x, "months", indexAt = indexAt, name = name, ...): missing  
## values removed from data

# Calculate statistics  
mean\_rJPM <- mean(rJPM, na.rm = TRUE)  
mean\_rFB <- mean(rFB, na.rm = TRUE)  
mean\_rTNX <- mean(rTNX, na.rm = TRUE)  
  
sd\_rJPM <- sd(rJPM, na.rm = TRUE)  
sd\_rFB <- sd(rFB, na.rm = TRUE)

## a. Coefficient of variation

For JPM:

# Coefficient of variation of Adjusted Returns of JPM  
cv\_rJPM <- sd\_rJPM / mean\_rJPM

For FB:

# Coefficient of variation of Adjusted Returns of FB  
cv\_rFB <- sd\_rFB / mean\_rFB

## b. Sharpe Ratio

In this case, each portfolio carry 100% of JPM of FB so weighting is 100% of each stock for each portfolio.

For JPM:

# Sharpe Ratio of a 100% JPM portfolio  
sharpe\_rJPM <- (mean\_rJPM - mean\_rTNX) / sd\_rJPM

For FB:

# Sharpe Ratio of a 100% JPM portfolio  
sharpe\_rFB <- (mean\_rFB - mean\_rTNX) / sd\_rFB

## b. Sortino Ratio

For the purpose of calculating the Sortino ratio, we’ll the mean of the Jan 2014 daily risk-free rate as the Minimum Acceptable Returns (MAR).

For JPM:

# Calculating downside deviation using lower partial moment of order 2  
# Assuming the 10 year T-bill returns as minimum acceptable returns MAR  
mar <- mean\_rTNX  
  
# Deviation from MAR, this is a data frame, remove NA values  
dev\_rJPM\_mar <- na.omit(rJPM - mar)  
  
# Set positive deviations to be 0 as we only want downside value  
devNegative\_rJPM\_mar <- subset(dev\_rJPM\_mar, dev\_rJPM\_mar < 0)  
  
# Calculate the Lower Partial Moment  
downsideDev\_JPM <- var(devNegative\_rJPM\_mar)  
  
# Downside deviation  
sd\_downsideDev\_JPM <- sqrt(downsideDev\_JPM)  
  
# Sortino Ratio  
sortino\_rJPM <- (mean\_rJPM - mean\_rTNX) / sd\_downsideDev\_JPM

For FB:

# Calculating downside deviation using lower partial moment of order 2  
# Assuming the 10 year T-bill returns as minimum acceptable returns MAR  
mar <- mean\_rTNX  
  
# Deviation from MAR, this is a data frame, also remove NA values  
dev\_rFB\_mar <- na.omit(rFB - mar)  
  
# Set positive deviations to be 0 as we only want downside value  
devNegative\_rFB\_mar <- subset(dev\_rFB\_mar, dev\_rFB\_mar < 0)  
  
# Calculate the Lower Partial Moment  
downsideDev\_FB <- var(devNegative\_rFB\_mar)  
  
# Downside deviation  
sd\_downsideDev\_FB <- sqrt(downsideDev\_FB)  
  
# Sortino Ratio  
sortino\_rFB <- (mean\_rFB - mean\_rTNX) / sd\_downsideDev\_FB

In summary:

|  |  |  |  |
| --- | --- | --- | --- |
| Portfolio | Coefficient of Variation | Sharpe Ratio | Sortino Ratio |
| JPM | -28.8009848 | -0.0566695 | -0.0669794 |
| FB | 10.1545518 | 0.0813225 | 0.206199 |

For the period of Jan 02, 2018 to Sep 18, 2020:  
In terms of coefficient of variation, both JPM and FB are highly volatile with JPM showing a significant negative drift from the mean returns while FB shows a positive drift from mean returns. +1 for FB  
In terms of Sharpe Ratio, FB has better risk-adjusted returns performance compared to JPM, which has negative Sharpe. +1 for FB In terms of Sortino Ratio, FB also generates better risk-adjusted performance with more weighting on the downside risk. JPM also has negative Sortino. +1 for FB

Overall, based on the above statistics, FB is the better investment with better risk-adjusted returns

# **Question 3**

Given statistics

# Mean returns of portfolio  
mean\_p <- 5/100  
  
# Standard deviation of returns of portfolio  
sd\_p <- 2.2/100  
  
# VaR threshold  
VaR\_threshold <- 5

## a. 5% 1 year VaR

Calculating statistics

# Set scaling factor based on the period of evaluation  
# In this example: {1: '1 year', 1/2: '6 months', 1/12: '1 month', 1/250: '1 day'}  
scalingFactor <- 1  
  
# Scaling returns  
meanScaled\_p <- mean\_p \* scalingFactor  
  
# Scaling standard deviation  
sdScaled\_p <- sd\_p \* sqrt(scalingFactor)  
  
# Value at Risk  
z\_stat <- qnorm(VaR\_threshold/100, 0, 1, lower.tail = TRUE)  
VaR\_1year\_p <- meanScaled\_p + sdScaled\_p \* z\_stat

Value at Risk following the standard normal distribution where the is the 5th percentile of the standard normal distribution.

Looking up the for the 5th percentile of the standard normal distribution, we have . Thus, the Value at Risk of the portfolio is:

## b. 5% 6 months VaR

Calculating statistics

# Set scaling factor based on the period of evaluation  
# In this example: {1: '1 year', 1/2: '6 months', 1/12: '1 month', 1/250: '1 day'}  
scalingFactor <- 1/2  
  
# Scaling returns  
meanScaled\_p <- mean\_p \* scalingFactor  
  
# Scaling standard deviation  
sdScaled\_p <- sd\_p \* sqrt(scalingFactor)  
  
# Value at Risk  
z\_stat <- qnorm(VaR\_threshold/100, 0, 1, lower.tail = TRUE)  
VaR\_6months\_p <- meanScaled\_p + sdScaled\_p \* z\_stat

Value at Risk following the standard normal distribution where the is the 5th percentile of the standard normal distribution.

Looking up the for the 5th percentile of the standard normal distribution, we have . Thus, the Value at Risk of the portfolio is:

## c. 5% 1 month VaR

Calculating statistics

# Set scaling factor based on the period of evaluation  
# In this example: {1: '1 year', 1/2: '6 months', 1/12: '1 month', 1/250: '1 day'}  
scalingFactor <- 1/12  
  
# Scaling returns  
meanScaled\_p <- mean\_p \* scalingFactor  
  
# Scaling standard deviation  
sdScaled\_p <- sd\_p \* sqrt(scalingFactor)  
  
# Value at Risk  
z\_stat <- qnorm(VaR\_threshold/100, 0, 1, lower.tail = TRUE)  
VaR\_1month\_p <- meanScaled\_p + sdScaled\_p \* z\_stat

Value at Risk following the standard normal distribution where the is the 5th percentile of the standard normal distribution.

Looking up the for the 5th percentile of the standard normal distribution, we have . Thus, the Value at Risk of the portfolio is:

## d. 5% 1 day VaR

Calculating statistics

# Set scaling factor based on the period of evaluation  
# In this example: {1: '1 year', 1/2: '6 months', 1/12: '1 month', 1/250: '1 day'}  
scalingFactor <- 1/250  
  
# Scaling returns  
meanScaled\_p <- mean\_p \* scalingFactor  
  
# Scaling standard deviation  
sdScaled\_p <- sd\_p \* sqrt(scalingFactor)  
  
# Value at Risk  
z\_stat <- qnorm(VaR\_threshold/100, 0, 1, lower.tail = TRUE)  
VaR\_1day\_p <- meanScaled\_p + sdScaled\_p \* z\_stat

Value at Risk following the standard normal distribution where the is the 5th percentile of the standard normal distribution.

Looking up the for the 5th percentile of the standard normal distribution, we have . Thus, the Value at Risk of the portfolio is:

In summary:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Period | 1 year | 6 months | 1 month | 1 day |
| 5% VaR | 0.0138132 | -5.879173810^{-4} | -0.0062796 | -0.0020887 |

# **Question 4**

# Set start date and end date of data  
start\_date <- "2018-01-02"  
end\_date <- "2020-09-18"  
  
# Get data for JPM, FB and the 10 year T-bill (TNX)  
getSymbols("JPM", src = "yahoo", from = start\_date, to = end\_date) # JPM

## [1] "JPM"

getSymbols("FB", src = "yahoo", from = start\_date, to = end\_date) # FB

## [1] "FB"

getSymbols("AAPL", src = "yahoo", from = start\_date, to = end\_date) # AAPL

## [1] "AAPL"

getSymbols("^TNX", src = "yahoo", from = start\_date, to = end\_date) # TNX (10-year T-bill)

## Warning: ^TNX contains missing values. Some functions will not work if objects  
## contain missing values in the middle of the series. Consider using na.omit(),  
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "^TNX"

# Get adjusted returns data for 01/2014  
rJPM <- na.omit(diff(log(to.monthly(JPM)$JPM.Adjusted)))  
rFB <- na.omit(diff(log(to.monthly(FB)$FB.Adjusted)))  
rAAPL <- na.omit(diff(log(to.monthly(AAPL)$AAPL.Adjusted)))  
rTNX <- (to.monthly(TNX)$TNX.Adjusted) / 1200 # Using monthly rate

## Warning in to.period(x, "months", indexAt = indexAt, name = name, ...): missing  
## values removed from data

# Calculate statistics  
mean\_rJPM <- mean(rJPM, na.rm = TRUE)  
mean\_rFB <- mean(rFB, na.rm = TRUE)  
mean\_rAAPL <- mean(rAAPL, na.rm = TRUE)  
mean\_rTNX <- mean(rTNX, na.rm = TRUE)  
  
sd\_rJPM <- sd(rJPM, na.rm = TRUE)  
sd\_rFB <- sd(rFB, na.rm = TRUE)  
sd\_rAAPL <- sd(rAAPL, na.rm = TRUE)  
sd\_rTNX <- sd(rTNX, na.rm = TRUE)  
  
  
cov\_rJPM\_rJPM <- cov(rJPM, rJPM)  
cov\_rAAPL\_rAAPL <- cov(rAAPL, rAAPL)  
cov\_rFB\_rFB <- cov(rFB, rFB)  
cov\_rJPM\_rFB <- cov(rJPM, rFB)  
cov\_rJPM\_rAAPL <- cov(rJPM, rAAPL)  
cov\_rFB\_rAAPL <- cov(rFB, rAAPL)  
  
cor\_rJPM\_rJPM <- cor(rJPM, rJPM)  
cor\_rAAPL\_rAAPL <- cor(rAAPL, rAAPL)  
cor\_rFB\_rFB <- cor(rFB, rFB)  
cor\_rJPM\_rFB <- cor(rJPM, rFB)  
cor\_rJPM\_rAAPL <- cor(rJPM, rAAPL)  
cor\_rFB\_rAAPL <- cor(rFB, rAAPL)

Parameters for the Markowitz portfolio model are:  
1. Expected Returns  
2. Standard deviation from mean returns  
3. Covariance of one asset returns to another.

The Covariance and Correlation formula is:

Using R, we have the parameters for the Markowitz Portfolio Model:

|  |  |  |
| --- | --- | --- |
| Stock | Mean | Standard Deviation |
| AAPL | 0.031437 | 0.1022224 |
| JPM | -0.00267 | 0.0768994 |
| FB | 0.0096887 | 0.098384 |

And the correlation matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Correlation | AAPL | JPM | FB |
| AAPL | 1 | 0.5193834 | 0.6743296 |
| JPM | 0.5193834 | 1 | 0.5553709 |
| FB | 0.6743296 | 0.5553709 | 1 |

And the covariance matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Covariance | AAPL | JPM | FB |
| AAPL | 0.0104494 | 0.0040828 | 0.0067818 |
| JPM | 0.0040828 | 0.0059135 | 0.0042018 |
| FB | 0.0067818 | 0.0042018 | 0.0096794 |

# **Question 5**

Given parameters

# Mean Returns  
mean\_rStock1 <- 10/100  
mean\_rStock2 <- 12/100  
mean\_rf <- 5/100  
  
# Variance of returns  
var\_rStock1 <- 0.025  
var\_rStock2 <- 0.030  
  
# Standard deviation of returns  
sd\_rStock1 <- sqrt(var\_rStock1)  
sd\_rStock2 <- sqrt(var\_rStock2)  
  
# Covariance of Returns  
cov\_rStock1\_rStock2 <- 0.015  
  
# Correlation of Returns  
cor\_rStock2\_rStock2 <- cov\_rStock1\_rStock2 / (sd\_rStock1 \* sd\_rStock2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Asset | Mean Returns | Variance | Pair | \_{1,2} |
| Stock 1 | 0.1 | 0.025 | (1,2) | 0.015 |
| Stock 2 | 0.12 | 0.03 |  |  |
| Risk Free | 0.05 |  |  |  |

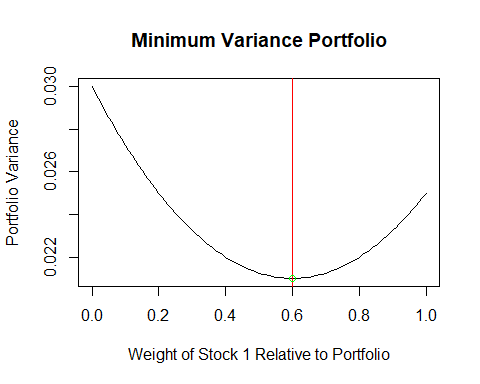
Since the investor’s objective is to minimize risk, we solve for the global minimum variance combination of the two assets. Let denotes the weight of the investment in asset i , and assume all money is invested in i, meaning .

The global minimum variance portfolio is the solution of the constrained minimization problem:

Find the critical point:

Graphing the function:

# The Variance function of the portfolio  
fVar <- function(m1) var\_rStock1 \* m1^2 + var\_rStock2 \* (1 - m1)^2 + 2 \* m1 \* (1 - m1) \* cov\_rStock1\_rStock2  
  
# Set the weighting  
m1 <- .6  
m2 <- .4  
  
# Plot of the variance function with respect to w1.  
w1 <- seq(0, 1, .05)  
plot(fVar, w1, xlab = "Weight of Stock 1 Relative to Portfolio", ylab = "Portfolio Variance", main = "Minimum Variance Portfolio"); abline(v = m1, col = "red")  
points(m1, fVar(m1), col='green')



We see indeed the minimum is at

Thus, the optimum weights of each stocks in the portfolio to satisfy the objective of minimum risk is:

|  |  |
| --- | --- |
| Asset | Weight |
| Stock 1 | 60% |
| Stock 2 | 40% |