Final Project

2024-12-02

## 1(a)

is satisfied if and only if at least one of the values is less than or equal to , and at least one of the values is greater than or equal to .

The complementary event is that all observations are either below or above .

so that we can derive the probability of this event:

Since are i.i.d., we have:

Thus, we get:

As a result, the minimal sample size n that satisfies the inequality depends only on p (and not on the unknown CDF F).

## 1(b-c)

# find the minimal n to satisfy the inequality  
find\_min\_n <- function(p, value1) {  
 n <- 1  
 while (TRUE) {  
 prob <- 1 - p^n - (1 - p)^n  
 if (prob >= value1) break  
 n <- n + 1  
 }  
 return(n)  
}  
  
# values for p  
p\_values <- c(0.01, 0.025, 0.05, 0.10, 0.20, 0.40, 0.50)  
  
# let Pr=0.99  
n\_99 <- sapply(p\_values, function(p) find\_min\_n(p, 0.99))  
  
# let Pr=0.999  
n\_999 <- sapply(p\_values, function(p) find\_min\_n(p, 0.999))  
  
# Results  
results <- data.frame(p = p\_values, n\_99 = n\_99, n\_999=n\_999)  
transposed\_results <- t(results)  
print(transposed\_results,row.names = FALSE)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7]  
## p 0.01 0.025 0.05 0.1 0.2 0.4 0.5  
## n\_99 459.00 182.000 90.00 44.0 21.0 10.0 8.0  
## n\_999 688.00 273.000 135.00 66.0 31.0 14.0 11.0

## 2(a) Mean and Variance of Portfolio Return

With two random variables and as the daily log return values of stock prices, we can calculate the portfolio return as:

### Mean of the Portfolio Return

So the expected value of is:

As a result, we have:

where and .

### Variance of the Portfolio Return

The variance of the portfolio is given by:

We can expand it as:

Let:

As a result, we can express the variance of the portfolio as:

### Justification for

Because ,

## 2(b) Minimizing the Portfolio Variance

To minimize the variance , we differentiate with respect to :

Setting the derivative equal to 0:

Thus, the optimal weight that minimizes variance is:

### Case Analysis

if , , then , the variance is minimized when ;

if , then , the variance is minimized when ;

otherwise, falls between 0 and 1, so when , the variance is minimized.

## 2(c)

The sharpe ratio is .

When  and is the optimal value that maximize the sharpe ratio.

# if (!require(Deriv)) {install.packages("Deriv")}   
library(Deriv)  
f <- function(omega, a, b, c, c0, c1) {  
 (c0 + c1 \* omega) / sqrt(a \* omega^2 - 2 \* b \* omega + c)  
}  
  
# find derivatives  
f\_first <- Deriv(f, "omega", combine = TRUE)  
f\_second <- Deriv(f\_first, "omega", combine = TRUE)  
  
# extract intermediate variable expression  
extract\_variables <- function(expr) {  
 expr\_text <- deparse(expr)   
 vars <- gregexpr("\\.e[0-9]+ <- .\*", expr\_text)   
 matches <- regmatches(expr\_text, vars)   
 unlist(matches)   
}  
  
# output intermediate variables  
cat("first order derivative intermediate variables:\n")

## first order derivative intermediate variables:

cat(paste(extract\_variables(body(f\_first)), collapse = "\n"), "\n\n")

## .e1 <- 2 \* b  
## .e2 <- a \* omega  
## .e4 <- c + omega \* (.e2 - .e1)

cat("second order derivative intermediate variables:\n")

## second order derivative intermediate variables:

cat(paste(extract\_variables(body(f\_second)), collapse = "\n"), "\n\n")

## .e1 <- 2 \* b  
## .e2 <- a \* omega  
## .e4 <- 2 \* .e2 - .e1  
## .e6 <- c + omega \* (.e2 - .e1)  
## .e7 <- c0 + c1 \* omega  
## .e9 <- .e4 \* .e7/.e6

# output first and second order derivatives  
cat("f'(x) =", deparse(body(f\_first)), "\n")

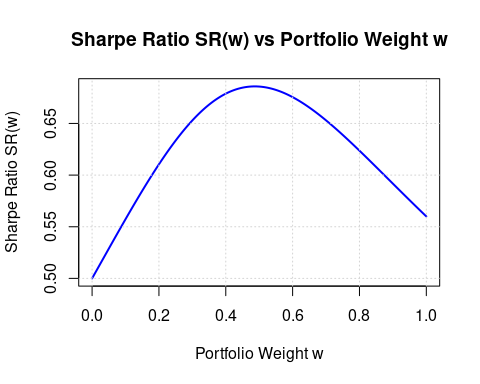
## f'(x) = { .e1 <- 2 \* b .e2 <- a \* omega .e4 <- c + omega \* (.e2 - .e1) (c1 - 0.5 \* ((2 \* .e2 - .e1) \* (c0 + c1 \* omega)/.e4))/sqrt(.e4) }

cat("f''(x) =", deparse(body(f\_second)), "\n")

## f''(x) = { .e1 <- 2 \* b .e2 <- a \* omega .e4 <- 2 \* .e2 - .e1 .e6 <- c + omega \* (.e2 - .e1) .e7 <- c0 + c1 \* omega .e9 <- .e4 \* .e7/.e6 -((0.5 \* (.e4 \* (c1 - .e9) + 2 \* (a \* .e7)) + 0.5 \* (.e4 \* (c1 - 0.5 \* .e9)))/(.e6 \* sqrt(.e6))) }

With these formulas, we can see that the second order derivatives always negative, so When , we can get the maximize the sharpe ratio directly by solving . If the the global optimal , if, then optimal is 0; else if , then optimal is 1.

# Numerical Simulation  
mu1 <- 0.056   
mu2 <- 0.04   
sigma1 <- 0.1   
sigma2 <- 0.08   
rho <- 0.2 # correlation coefficient of 1 and 2  
  
# calculate parameters  
c0 <- mu2  
c1 <- mu1 - mu2  
a <- sigma1^2 + sigma2^2 - 2 \* rho \* sigma1 \* sigma2  
b <- sigma2^2 - rho \* sigma1 \* sigma2  
c <- sigma2^2  
  
# Sharpe Ratio function  
SR <- function(w) {  
 mu\_w <- c0 + c1 \* w  
 sigma\_w <- sqrt(a \* w^2 - 2 \* b \* w + c)  
 return(mu\_w / sigma\_w)  
}  
  
# generate weight array and Sharpe Ratio array  
w\_values <- seq(0, 1, length.out = 500)  
sr\_values <- sapply(w\_values, SR)  
  
# plot Sharpe Ratio v.s. w  
plot(w\_values, sr\_values, type = "l", lwd = 2, col = "blue",  
 main = "Sharpe Ratio SR(w) vs Portfolio Weight w",  
 xlab = "Portfolio Weight w", ylab = "Sharpe Ratio SR(w)")  
abline(h = 0, col = "black", lty = 2)  
grid()



## 3

### 3.a

if (!requireNamespace("quantmod", quietly = TRUE)) install.package("quantmod")

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

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

# download data  
getSymbols(c("AAPL", "MSFT"), from = "2023-11-01", to = "2024-10-31")

## [1] "AAPL" "MSFT"

# calculate daily return  
AAPL.returns <- as.numeric(dailyReturn(AAPL)) \* 100  
MSFT.returns <- as.numeric(dailyReturn(MSFT)) \* 100  
  
# estimate moment  
mu1 <- mean(AAPL.returns)  
mu2 <- mean(MSFT.returns)  
sigma1 <- sd(AAPL.returns)  
sigma2 <- sd(MSFT.returns)  
rho <- cor(AAPL.returns, MSFT.returns)  
  
# print moment estimates  
cat(sprintf("µ1 = %.4f, µ2 = %.4f, σ1 = %.4f, σ2 = %.4f, ρ = %.4f\n", mu1, mu2, sigma1, sigma2, rho))

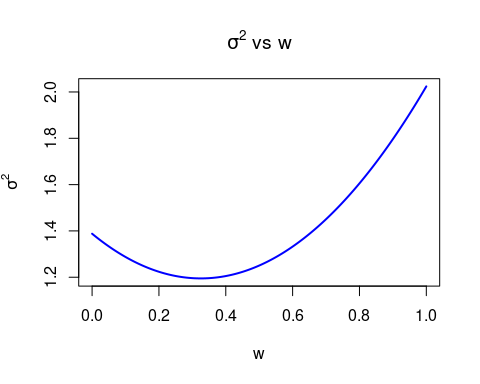
## µ1 = 0.1284, µ2 = 0.1031, σ1 = 1.4227, σ2 = 1.1781, ρ = 0.4745

# calculate parameter coefficient in 2(a)  
c0 <- mu2  
c1 <- mu1 - mu2  
a <- sigma1^2 + sigma2^2 - 2 \* rho \* sigma1 \* sigma2  
b <- sigma2^2 - rho \* sigma1 \* sigma2  
c <- sigma2^2  
  
# print parameter coefficient in 2(a)  
cat(sprintf("c0 = %.4f, c1 = %.4f, a = %.4f, b = %.4f, c = %.4f\n", c0, c1, a, b, c))

## c0 = 0.1031, c1 = 0.0252, a = 1.8215, b = 0.5928, c = 1.3880

### 3.b

sigma2 <- function(w) {  
 a \* w^2 - 2 \* b \* w + c  
}  
  
# generate weight w array and sigma array  
w\_values <- seq(0, 1, length.out = 1000)  
sigma2\_values <- sapply(w\_values, sigma2)  
  
# plot sigma v.s. w  
plot(w\_values, sigma2\_values, type = "l", col = "blue", lwd = 2,  
 main = expression(sigma^2~vs~w),  
 xlab = "w", ylab = expression(sigma^2))

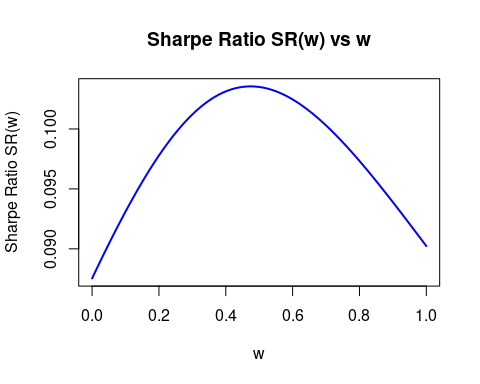


# find minimum sigma  
opt\_w <- optimize(sigma2, interval = c(0, 1))$minimum  
min\_sigma2 <- sigma2(opt\_w)  
mu\_opt <- c0 + c1 \* opt\_w  
sigma\_opt <- sqrt(min\_sigma2)  
  
# output w minimizing sigma  
cat(sprintf("Optimal w = %.2f, µ(w^) = %.4f, σ(w^) = %.4f\n", opt\_w, mu\_opt, sigma\_opt))

## Optimal w = 0.33, µ(w^) = 0.1113, σ(w^) = 1.0932

### 3.c

# calculate sharpe ratio function  
sharpe\_ratio <- function(w) {  
 mu\_w <- c0 + c1 \* w  
 sigma\_w <- sqrt(a \* w^2 - 2 \* b \* w + c)  
 mu\_w / sigma\_w  
}  
  
# calculate Sharpe Ratio  
sr\_values <- sapply(w\_values, sharpe\_ratio)  
  
# plot w v.s. sharpe ratio  
plot(w\_values, sr\_values, type = "l", col = "blue", lwd = 2,  
 main = "Sharpe Ratio SR(w) vs w",  
 xlab = "w", ylab = "Sharpe Ratio SR(w)")



# optimize w to maximize Sharpe ratio  
opt\_sr <- optimize(sharpe\_ratio, interval = c(0, 1), maximum = TRUE)  
opt\_w\_sr <- opt\_sr$maximum  
max\_sr <- opt\_sr$objective  
mu\_opt\_sr <- c0 + c1 \* opt\_w\_sr  
sigma\_opt\_sr <- sqrt(sigma2(opt\_w\_sr))  
  
# print optimal w and sharpe ratio  
cat(sprintf("Optimal w for Sharpe Ratio = %.2f, µ(w^) = %.4f, σ(w^) = %.4f, SR(w^) = %.4f\n",  
 opt\_w\_sr, mu\_opt\_sr, sigma\_opt\_sr, max\_sr))

## Optimal w for Sharpe Ratio = 0.47, µ(w^) = 0.1151, σ(w^) = 1.1115, SR(w^) = 0.1035

### 3.d

# Using Historical return data  
# calculate VaR function   
VaR <- function(returns, alpha) {  
 quantile(-returns, probs = alpha)  
}  
  
# calculating VaR  
alphas <- c(0.95, 0.975, 0.99)  
VaR\_AAPL <- sapply(alphas, VaR, returns = AAPL.returns)  
VaR\_MSFT <- sapply(alphas, VaR, returns = MSFT.returns)  
  
# print VaR  
VaR\_table <- data.frame(  
 Alpha = alphas,  
 VaR\_AAPL = round(VaR\_AAPL, 4),  
 VaR\_MSFT = round(VaR\_MSFT, 4)  
)  
print(VaR\_table)

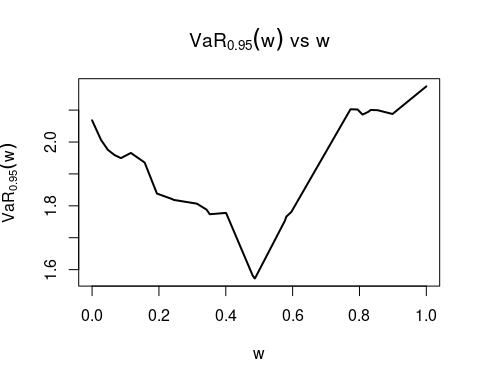
## Alpha VaR\_AAPL VaR\_MSFT  
## 95% 0.950 2.1743 2.0681  
## 97.5% 0.975 2.7633 2.4703  
## 99% 0.990 3.2464 3.2388

# Using Normal Distribution(Can be change to any other customized distribution) return  
# VaR based on Normal Distribution Return  
compute\_var\_normal <- function(mu, sigma, alpha) {  
 VaR <- -(mu - sigma \* qnorm(alpha))  
 return(VaR)  
}  
  
# Return Distribution Parameter Estimation  
mu\_AAPL <- mean(AAPL.returns, na.rm = TRUE)  
sigma\_AAPL <- sd(AAPL.returns, na.rm = TRUE)  
mu\_MSFT <- mean(MSFT.returns, na.rm = TRUE)  
sigma\_MSFT <- sd(MSFT.returns, na.rm = TRUE)  
  
# Single Stock VaR  
VaR\_AAPL\_normal <- sapply(alphas, compute\_var\_normal, mu = mu\_AAPL, sigma = sigma\_AAPL)  
VaR\_MSFT\_normal <- sapply(alphas, compute\_var\_normal, mu = mu\_MSFT, sigma = sigma\_MSFT)  
  
# Print VaRs  
VaR\_table <- data.frame(  
 Alpha = alphas,  
 VaR\_AAPL = round(VaR\_AAPL\_normal, 4),  
 VaR\_MSFT = round(VaR\_MSFT\_normal, 4)  
)  
print(VaR\_table)

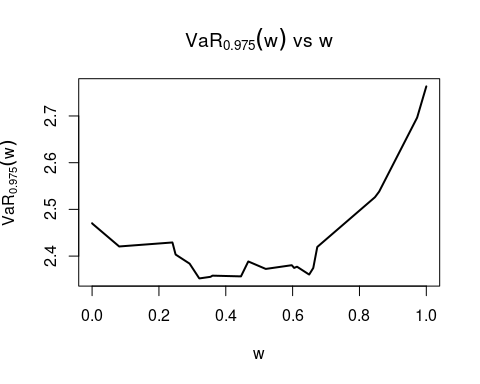
## Alpha VaR\_AAPL VaR\_MSFT  
## 1 0.950 2.2117 1.8347  
## 2 0.975 2.6600 2.2059  
## 3 0.990 3.1812 2.6376

### 3.e

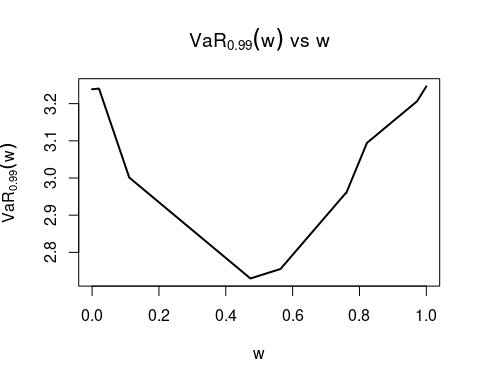
# Using Historical Return Data  
# function to calculate Portfolio VaR  
VaR\_portfolio <- function(w, alpha) {  
 portfolio\_returns <- w \* AAPL.returns + (1 - w) \* MSFT.returns  
 VaR(portfolio\_returns, alpha)  
}  
  
# calculating w and VaR(w)  
for (alpha in alphas) {  
 VaR\_values <- sapply(w\_values, VaR\_portfolio, alpha = alpha)  
 plot(w\_values, VaR\_values, type = "l", lwd = 2,  
 main = bquote(VaR[.(alpha)](w)~vs~w),  
 xlab = "w", ylab = bquote(VaR[.(alpha)](w)))  
   
 # optimize w to minimize VaR  
 opt\_VaR <- optimize(VaR\_portfolio, interval = c(0, 1), alpha = alpha)  
 opt\_w\_VaR <- opt\_VaR$minimum  
 mu\_opt\_VaR <- c0 + c1 \* opt\_w\_VaR  
 sigma\_opt\_VaR <- sqrt(sigma2(opt\_w\_VaR))  
   
 cat(sprintf("Optimal w for VaR (alpha=%.3f): w = %.2f, µ(w^) = %.4f, σ(w^) = %.4f\n",  
 alpha, opt\_w\_VaR, mu\_opt\_VaR, sigma\_opt\_VaR))  
}



## Optimal w for VaR (alpha=0.950): w = 0.49, µ(w^) = 0.1154, σ(w^) = 1.1147



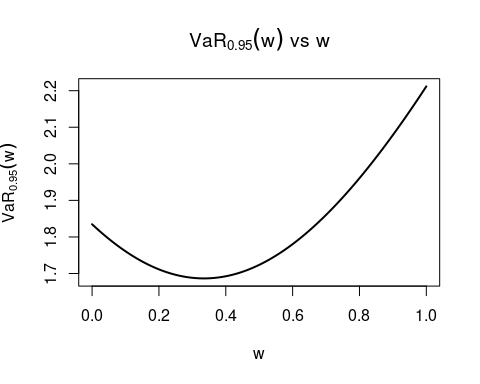
## Optimal w for VaR (alpha=0.975): w = 0.32, µ(w^) = 0.1112, σ(w^) = 1.0932



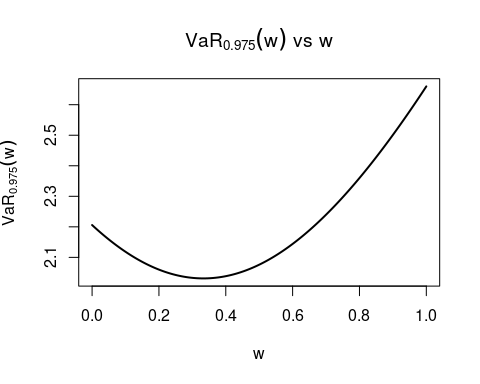
## Optimal w for VaR (alpha=0.990): w = 0.47, µ(w^) = 0.1151, σ(w^) = 1.1114

When using Normal distribution as return distribution, all optimal $$ to minimize the VaRs are the same, and they are the same as the to minimize of the portfolio.

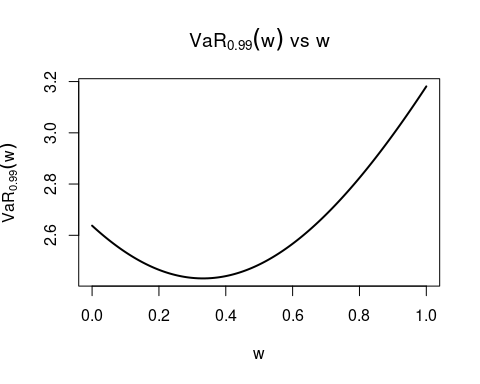
# Using Normal Distribution Return  
# Compute Portfolio VaR  
VaR\_portfolio\_normal <- function(mu1, mu2, sigma1, sigma2, rho, alpha, w) {  
 mu\_portfolio <- w \* mu1 + (1 - w) \* mu2  
 sigma\_portfolio <- sqrt((w^2) \* sigma1^2 + ((1 - w)^2) \* sigma2^2 + 2 \* w \* (1 - w) \* rho \* sigma1 \* sigma2)  
 compute\_var\_normal(mu\_portfolio, sigma\_portfolio, alpha)  
}  
  
# calculating w and VaR(w)  
for (alpha in alphas) {  
 VaR\_values <- sapply(w\_values, VaR\_portfolio\_normal, mu1 = mu\_AAPL, mu2 = mu\_MSFT, sigma1 = sigma\_AAPL, sigma2 = sigma\_MSFT, rho = rho, alpha = alpha)  
 plot(w\_values, VaR\_values, type = "l", lwd = 2,  
 main = bquote(VaR[.(alpha)](w)~vs~w),  
 xlab = "w", ylab = bquote(VaR[.(alpha)](w)))  
   
 # optimize w to minimize VaR  
 opt\_VaR <- optimize(VaR\_portfolio, interval = c(0, 1), alpha = alpha)  
 opt\_w\_VaR <- opt\_VaR$minimum  
 mu\_opt\_VaR <- c0 + c1 \* opt\_w\_VaR  
 sigma\_opt\_VaR <- sqrt(sigma2(opt\_w\_VaR))  
   
 cat(sprintf("Optimal w for VaR (alpha=%.3f): w = %.2f, µ(w^) = %.4f, σ(w^) = %.4f\n",  
 alpha, opt\_w\_VaR, mu\_opt\_VaR, sigma\_opt\_VaR))  
}



## Optimal w for VaR (alpha=0.950): w = 0.49, µ(w^) = 0.1154, σ(w^) = 1.1147



## Optimal w for VaR (alpha=0.975): w = 0.32, µ(w^) = 0.1112, σ(w^) = 1.0932



## Optimal w for VaR (alpha=0.990): w = 0.47, µ(w^) = 0.1151, σ(w^) = 1.1114