STAT509-001-HW4-Xinye Xu

Q1

(c) Exercise 7 on page 181 in Ruppert/Matteson. Hint: Use eigen command in R to carry out eigen-value/vector analysis of the matrix, and find a vector w so that formula for the variance of w*X would be negative showing a contradiction if this really were a covariance matrix for

Assumed a = 0, for Matrix Y, based on eigen command, there is an negative eigenvalue. and from 7.7 for Var(w_transpose * Y) = w_tran * Y * w, which should be non-negative. Then the Y matrix should be positive dedefinite. Notice that covariance matrix is symmetric, and the symmetric matrix A is said positive semidefinite (A >= 0) if all its eigenvalues are non negative. So a should be zero. We can also have the same conclusion when w is set to be the following, the Var(w-transpose * Y) is negative, which breaks the common sense.

```r
m = matrix(c(1, 0.9, 0, 0.9, 1, 0.9, 0, 0.9, 1), nrow = 3)
eigen(m)
```

```
## eigen() decomposition
## $values
## [1]  2.2727922  1.0000000 -0.2727922
##
## $vectors
##           [,1]          [,2]       [,3]
## [1,] 0.5000000 -7.071068e-01  0.5000000
## [2,] 0.7071068 -8.722736e-16 -0.7071068
## [3,] 0.5000000  7.071068e-01  0.5000000
```

```r
w = matrix(c(0.5000000, -7.071068e-01, 0.5000000), nrow = 3)
t(w)  %*%  m %*% w
```

```
##            [,1]
## [1,] -0.2727922
```

Q2

Suppose R1.R2 are returns on an asset, and suppose that E(R1) = .02,E(R2) = .04, Var(R1) = (.03)2, Var(R2) = (.06)2, and Corr(R1, R2) = 0.5.

(a) What are E(0.6R1 + 0.4R2) = 0.028 and Var(0.6R1 + 0.4R2) = 0.001332

```r
0.6 * 0.02 + 0.4 * 0.04
```

```
## [1] 0.028
```

```r
sigma_1 = 0.03
sigma_2 = 0.06
0.6^2 * sigma_1^2 + 0.4^2 * sigma_2^2 + 2 * 0.6 * 0.4 * 0.5 * (sigma_1 * sigma_2)
```

```
## [1] 0.001332
```

(b) For what value of w is

$$Var(w \times R1 + (1w) \times R2)$$

minimized? Why would it be useful to minimize

$$Var(w \times R1 + (1w) \times R2)$$

?

$$Var(w \times R1 + (1w)\mathring{u}R2) = w^2 \times (0.03)^2 + (1w)^2 \times (0.06)2 + 2 \times w \times (1w) \times 0.5 \times 0.03 \times 0.06$$

Then, take the first derivative equal to zero:

$$0.03^2 * w - (1 - w) * 0.06^2 + (1 - 2w) * 0.5 * 0.03 * 0.06 = 0$$

So the weight for R1 is 100% and weight for R2 is 0, then variance will be minimized. The idea for w is important is that adjustment of it can help whole portfolio reduce the volatility, or standard deviation. The minimal standard deviation for the return of this portfolio is (0.03)

```
w = (sigma_2^2 - 0.5 * sigma_1 * sigma_2) / (sigma_1^2 + sigma_2^2 - 2*0.5*sigma_1*sigma_2)
w
```

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

```
sigma_1^2
```

```
## [1] 9e-04
```

(c) Assuming a portfolio of a $1 million, and a multivariate normal distribution for R1 and R2, find the value w that minimizes the expected shortfall associated with VaR at q = .005 and why might that be useful? Also, report the associated VaR with this portfolio.

Hints: Note that the random variable [wR1 + (1 − w)R2] is normally distributed for any w, and it will be easiest to use R-package – the solution can be approximate, i.e., accurate to .01.

Carried out analysis of VaR and relative shortfall as a function of w - the plots of these two vs. the VaR and shortfall are shown below. According to two plots of VaR and Shortfall against weight, there are quadratic trends against weight for both plot. Then, the optimal weight is 93% for minimal shortfall, and it is quite close to the weight 92% for minimal value of relative VaR. The relative VaR and shortfall for the weight of w = 93% is 0.05641319 and 0.06645752. The relative VaR and shortfall for the weight of w = 92% is 0.05640813 and 0.06647756. The VaR and shortfall for the weight of w = 93% is 56413.19 and 65914.67 The VaR and shortfall for the weight of w = 92% is 56408.13 and 65933.36

```
set.seed(123)
weight = seq(0,1,0.01)  # accurate to .01.
n = length(weight)
VaRv = rep(0,n)
shortfall = rep(0,n)
quantv = rep(0,n)
ER1 = 0.02
ER2 = 0.04
VarR1 = sigma_1^2
VarR2 = sigma_2^2
Cov = 0.5 * sigma_1 * sigma_2;
mu = weight * ER1+(1-weight)*ER2
sigma = sqrt(weight^2*VarR1 +(1-weight)^2*VarR2+2*weight*(1-weight)*Cov)

randnorm = rnorm(100000,0,1) # using same rnorm to simulate tail return for all i
for(i in 1:n){
  quantv[i] = qnorm(0.005,mu[i],sigma[i]) # VaR at q = .005
  VaRv[i] = -quantv[i]
  randnorm_value = mu[i] + sigma[i] * randnorm
  shortfall[i] = -mean(randnorm_value[randnorm_value < quantv[i]])
}

which.min(shortfall) # 93
```

```
## [1] 93
```

```r
print(VaRv[which.min(shortfall)])
```

```
## [1] 0.05641319
```

```r
print(shortfall[which.min(shortfall)])
```

```
## [1] 0.06591467
```

```r
print(VaRv[which.min(shortfall)]*1000000)
```

```
## [1] 56413.19
```

```r
print(shortfall[which.min(shortfall)]*1000000)
```

```
## [1] 65914.67
```

```r
which.min(VaRv) # 92
```

```
## [1] 92
```

```r
print(VaRv[which.min(VaRv)])
```

```
## [1] 0.05640813
```

```r
print(shortfall[which.min(VaRv)])
```

```
## [1] 0.06593336
```

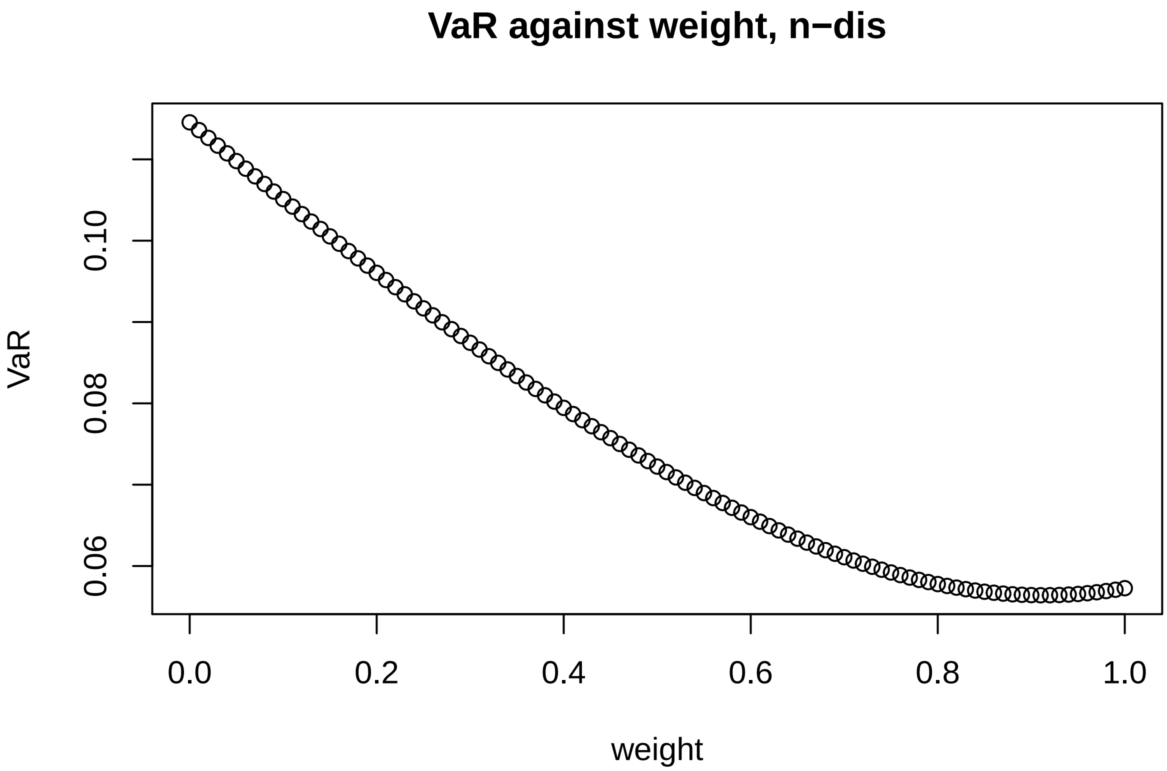
```r
print(VaRv[which.min(VaRv)]*1000000)
```

```
## [1] 56408.13
```

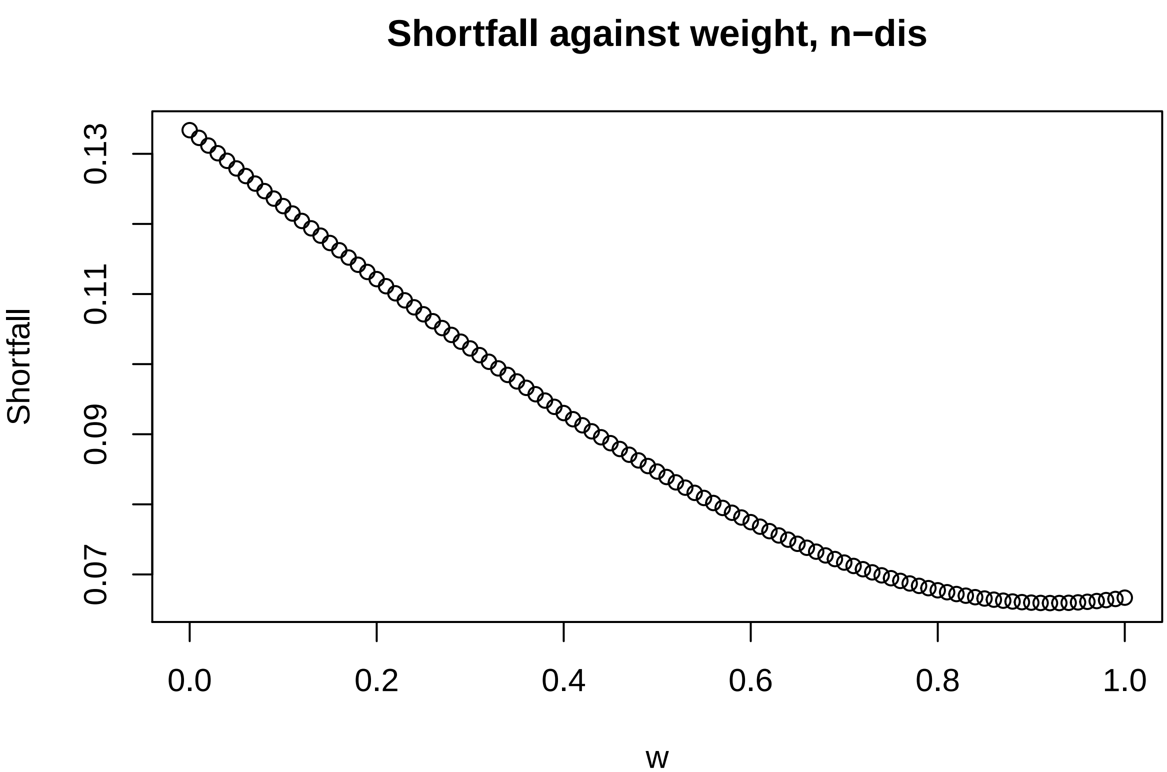
```r
print(shortfall[which.min(VaRv)]*1000000)
```

```
## [1] 65933.36
```

```r
plot(weight, VaRv,xlab = 'weight',ylab='VaR',main='VaR against weight, n-dis')
```




```
plot(weight, shortfall,xlab = 'w', ylab='Shortfall', main='Shortfall against weight, n-dis')
```



(d) Repeat the above if have multi-variate t-distribution for R1 and R2, with $= 6$.

For the t-dist, we know that

$$sigma^2 = scale^2 * (df/(df - 2))$$

, so scale = sigma * sqrt((6-2)/6). Then the scaled t-dist is

$$scale \times standardt - dist + mu$$

. According to two plots of VaR and Shortfall against weight, there are also quadratic trends against weight for both plot. Then, the optimal weight is 95% for minimal shortfall, and it is quite close to the weight 94% for minimal value of relative VaR. The relative VaR and shortfall for the weight of w = 95% is 0.07010214 and 0.09683756 The relative VaR and shortfall for the weight of w = 94% is 0.07007811 and 0.09686506 The VaR and shortfall for the weight of w = 95% is 70102.14 and 95761.96 The VaR and shortfall for the weight of w = 94% is 70078.11 and 95787.38

```r
randt = rt(1000000,6) # degree of freedom = 6
scale = sigma * sqrt((6-2)/6)
for(i in 1:n){
  quantv[i] = scale[i] * qt(0.005,6) + mu[i]  # VaR at q = .005
  VaRv[i] = -quantv[i]
  randnorm_value = mu[i] + sigma[i] * randt  # change to randt
  shortfall[i] = -mean(randnorm_value[randnorm_value < quantv[i]])
}

which.min(shortfall) # c
```

```
## [1] 95
```

```r
print(VaRv[which.min(shortfall)])
```

```
## [1] 0.07010214
```

```r
print(shortfall[which.min(shortfall)]) #
```

```
## [1] 0.09576196
```

```r
print(VaRv[which.min(shortfall)]*1000000)
```

```
## [1] 70102.14
```

```r
print(shortfall[which.min(shortfall)]*1000000)
```

```
## [1] 95761.96
```

```r
which.min(VaRv) # 92
```

```
## [1] 94
```

```r
print(VaRv[which.min(VaRv)]) #
```

```
## [1] 0.07007811
```

```r
print(shortfall[which.min(VaRv)]) #
```

```
## [1] 0.09578738
```

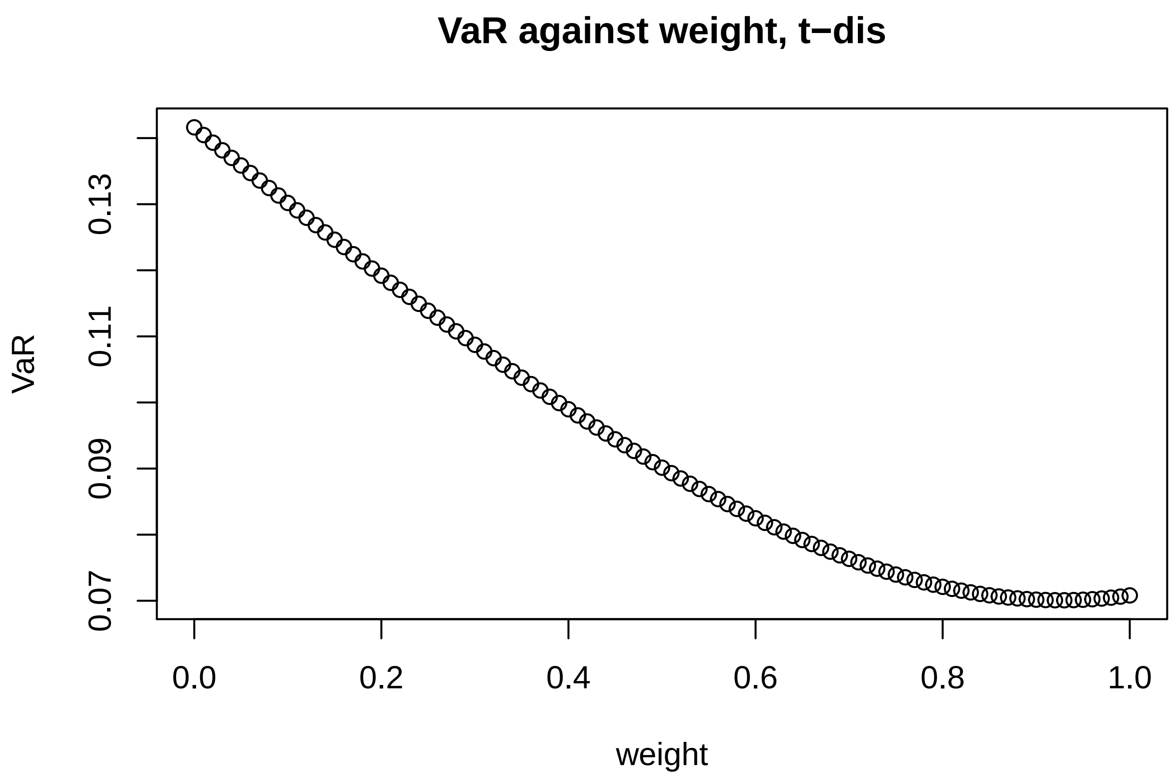
```r
print(VaRv[which.min(VaRv)]*1000000)
```

```
## [1] 70078.11
```

```r
print(shortfall[which.min(VaRv)]*1000000)
```

```
## [1] 95787.38
```

```r
plot(weight, VaRv,xlab = 'weight',ylab='VaR',main='VaR against weight, t-dis')
```




```r
plot(weight, shortfall,xlab = 'w', ylab='Shortfall', main='Shortfall against weight, t-dis')
```