Assignment 1

Nguyen Xuan Binh

Homework Problem 1: Functions

In this exercise do not use the built-in functions cov, cor, cov2cor or any additional R packages.

---

a) Create an R function that takes a data matrix $\mathbf{X} \in R^{n \times p}, n > p$, as an argument and returns the unbiased estimator of the covariance matrix.

```
setwd(getwd())
```

Model reference

```
A <- matrix(rnorm(30), ncol = 3)
print(cov(A))
```

```
##             [,1]       [,2]       [,3]
## [1,]  0.54482554 -0.3398727 0.06165064
## [2,] -0.33987266  1.4559644 0.37259388
## [3,]  0.06165064  0.3725939 0.64655549
```

The function for unbiased covariance without using the R library

```
unbiased_cov <- function(X) {
  # Number of observations
  n <- nrow(X)
  # Ensure n > p to proceed
  if (n <= ncol(X)) {
    stop("The number of observations n must be greater than the number of variables p.")
  }
  # Calculate the mean for each column
  column_means <- colMeans(X)
  # Center the matrix by subtracting column means
  X_centered <- sweep(X, 2, column_means, FUN = "-")
  # Calculate the covariance matrix
  cov_matrix <- (t(X_centered) %*% X_centered) / (n - 1)
  return(cov_matrix)
}

print(unbiased_cov(A))
```

```
##             [,1]       [,2]       [,3]
## [1,]  0.54482554 -0.3398727 0.06165064
## [2,] -0.33987266  1.4559644 0.37259388
## [3,]  0.06165064  0.3725939 0.64655549
```

---

b) Create an R function that takes a full-rank covariance matrix $\mathbf{A} \in R^{p \times p}$ as an argument and returns the square root of the inverse matrix such that $\mathbf{A}^{\frac{-1}{2}}\mathbf{A}^{\frac{-1}{2}} = \mathbf{A}$

A full-rank covariance matrix is a square matrix that has linearly independent columns and rows, which means it has the maximum possible rank (number of non-zero eigenvalues) which is equal to the number of rows (or columns) of the matrix. In other words, it is a matrix that has no zero eigenvalues, which indicates that all its columns are linearly independent, thus all its rows are also linearly independent. Additionally, it is a covariance matrix, which means it is symmetric.

Model reference

```r
# First install and load expm package
# install.packages("expm")
library(expm)
```

```
## Warning: package 'expm' was built under R version 4.3.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```r
# full-rank covariance matrix
A <- matrix(c(0.99,-0.17,0.30,-0.17,0.61,0.33,0.30,0.33,1.33), nrow = 3, ncol = 3)

# Find the inverse of the matrix
inverse_A <- solve(A)

# Find the square root of the inverse of the matrix
sqrt_inverse_A <- sqrtm(inverse_A)
sqrt_inverse_A
```

```
##             [,1]       [,2]       [,3]
## [1,]  1.0768396  0.2092707 -0.1800484
## [2,]  0.2092707  1.4301472 -0.2629218
## [3,] -0.1800484 -0.2629218  0.9539935
```

The square root of a matrix can be found by using the eigenvalue decomposition of the matrix. The eigenvalue decomposition is a way of decomposing a matrix into a canonical form, where the matrix is represented as a product of a matrix of eigenvectors and a diagonal matrix of eigenvalues.

The function for the square root of the inverse of a matrix without using the R library

```r
# Define the function to calculate the square root of the inverse covariance matrix
sqrt_inv_cov_matrix <- function(A) {
  # Check if the matrix is square
  if (!is.square.matrix(A)) {
    stop("Matrix A is not a square matrix.")
  }
  # Check if the matrix is full-rank
  if (qr(A)$rank < ncol(A)) {
    stop("Matrix A is not full-rank.")
  }
  # Calculate the eigen decomposition of A
  eigen_decomp <- eigen(A)
  # Extract the eigenvalues
  eigenvalues <- eigen_decomp$values
  # Extract the eigenvectors
  eigenvectors <- eigen_decomp$vectors

  # Calculate the square root of the inverse of the eigenvalues
  sqrt_inv_eigenvalues <- 1 / sqrt(eigenvalues)

  # Construct the square root of the inverse matrix
  sqrt_inv_matrix <- eigenvectors %*% diag(sqrt_inv_eigenvalues) %*% t(eigenvectors)
  return(sqrt_inv_matrix)
}

# Check if a matrix is square
is.square.matrix <- function(M) {
  rows <- nrow(M)
  cols <- ncol(M)
  return(rows == cols)
}

# The function can be called with a covariance matrix A like so:
A_sqrt_inv <- sqrt_inv_cov_matrix(A)
```

This method will only work if the matrix is symmetric and its eigenvalues are real numbers, because the square root of real matrix is only defined for normal matrices. Thankfully, symmetry is guaranteed as it is a covariance matrix, and its eigenvalues are real numbers because it is a full-rank matrix.

---

c) Create an R function that takes a full-rank covariance matrix A as an argument and returns the corresponding correlation matrix

Model reference

```r
correlation_matrix <- cov2cor(A)
print(correlation_matrix)
```

```
##           [,1]       [,2]      [,3]
```

```
## [1,]  1.0000000 -0.2187592 0.2614435
## [2,] -0.2187592  1.0000000 0.3663728
## [3,]  0.2614435  0.3663728 1.0000000
```

The function for correlation conversion from full-rank covariance matrix without using the R library

```r
correlation_matrix <- function(A) {
  # Check if the matrix is square
  if (!is.square.matrix(A)) {
    stop("Matrix A is not a square matrix.")
  }
  # Check if the matrix is full-rank
  if (qr(A)$rank < ncol(A)) {
    stop("Matrix A is not full-rank.")
  }
  # Calculate the inverse of the square roots of the diagonal elements (variances)
  sqrt_diag_inv <- 1 / sqrt(diag(A))

  # Create a diagonal matrix with the inverse square roots
  D_inv <- diag(sqrt_diag_inv)

  # Standardize the covariance matrix to get the correlation matrix
  corr_matrix <- D_inv %*% A %*% D_inv
  return(corr_matrix)
}

# Check if a matrix is square
is.square.matrix <- function(M) {
  rows <- nrow(M)
  cols <- ncol(M)
  return(rows == cols)
}

# The function can be called with a covariance matrix A like so:
correlation_matrix_A <- correlation_matrix(A)
correlation_matrix_A
```

```
##            [,1]       [,2]      [,3]
## [1,]  1.0000000 -0.2187592 0.2614435
## [2,] -0.2187592  1.0000000 0.3663728
## [3,]  0.2614435  0.3663728 1.0000000
```

Read explanation from here: https://math.stackexchange.com/questions/186959/correlation-matrix-from-covariance-matrix