HW3

Haojie Liu

2023-11-03

```r
library(tidyverse)
library(ggplot2)
library(reshape2)
library(boot)
```

Problem 1: Given the 4-dimensional multivariate normal distribution with mean vector

$$\mu^T = \begin{pmatrix} 2 & 1.5 & 3 & 1 \end{pmatrix}$$

and covariance matrix

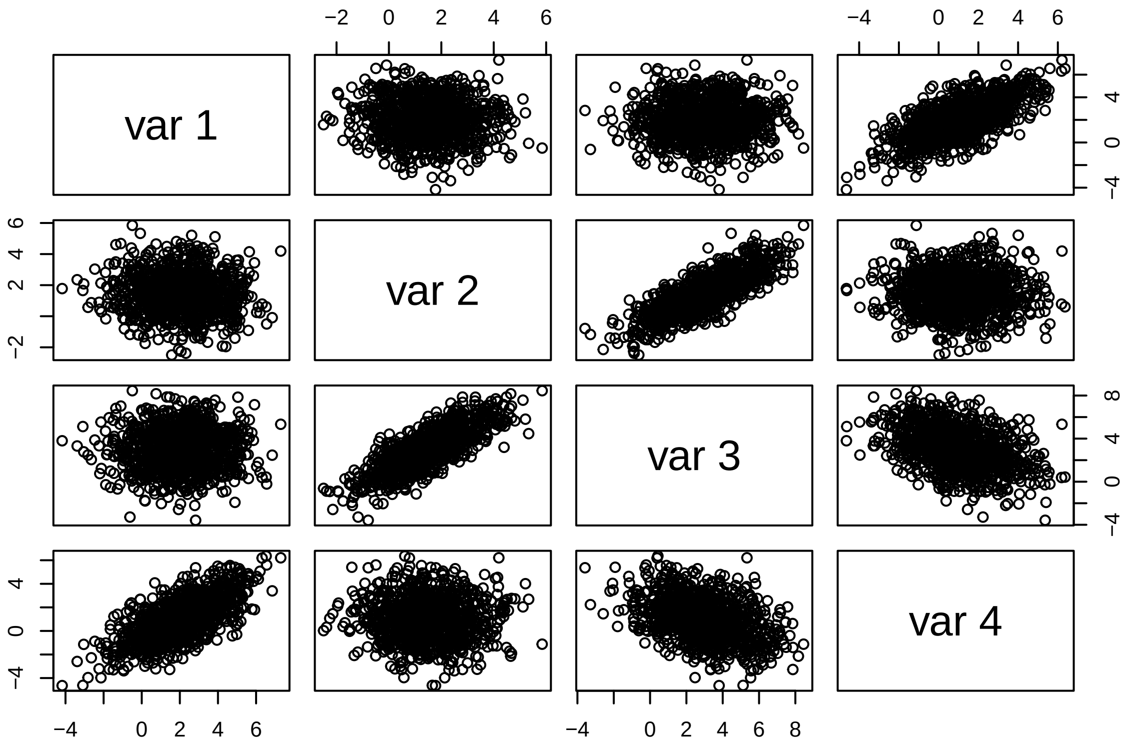
$$\Sigma = \begin{pmatrix} 2.8 & 0 & 0.2 & 2 \\ 0 & 1.7 & 2 & 0 \\ 0.2 & 2 & 3.6 & -1.2 \\ 2 & 0 & -1.2 & 3 \end{pmatrix}$$

(a) Generate 1000 random observations from this multivariate normal distribution using the Choleski factorization method.

```r
mu <- c(2,1.5,3,1)
covmat <- matrix(c(2.8,0,0.2,2,0,1.7,2,0,0.2,2,3.6,-1.2,2,0,-1.2,3), nrow = 4)
x <- matrix(rnorm(4000),ncol = 4)
sample <- matrix(numeric(4000), nrow = 1000)
for(i in 1:1000){
  sample[i,] <- mu+(x%*%chol(covmat))[i,]
}
```

(b) Draw an array of scatter plots for each pair of variables and examine if they agree with the parameters. (You may use pairs in R)

```r
pairs(sample)
```



Base on the pairs plot, I can tell that all means are around the μ vector that was given, and similar Σ that was given.

Problem 2: Write R code to standardize an d-dimensional multivariate normal sample X with the known Σ and sample size n, where d and n can be arbitrary integer numbers.

(a) Derive an algorithm for standardizing a multivariate normal sample.

To make the multivariate normal sample become standardized, we will make each sample subtract by their mean and divide by the standard deviation:

$$x_{standarized} = \frac{x - \mu}{\sigma}$$

Or in this example we have d-dimensional multivariate normal X with size n

$$X_{standarized} = (X - \mu)\Sigma^{-\frac{1}{2}}$$

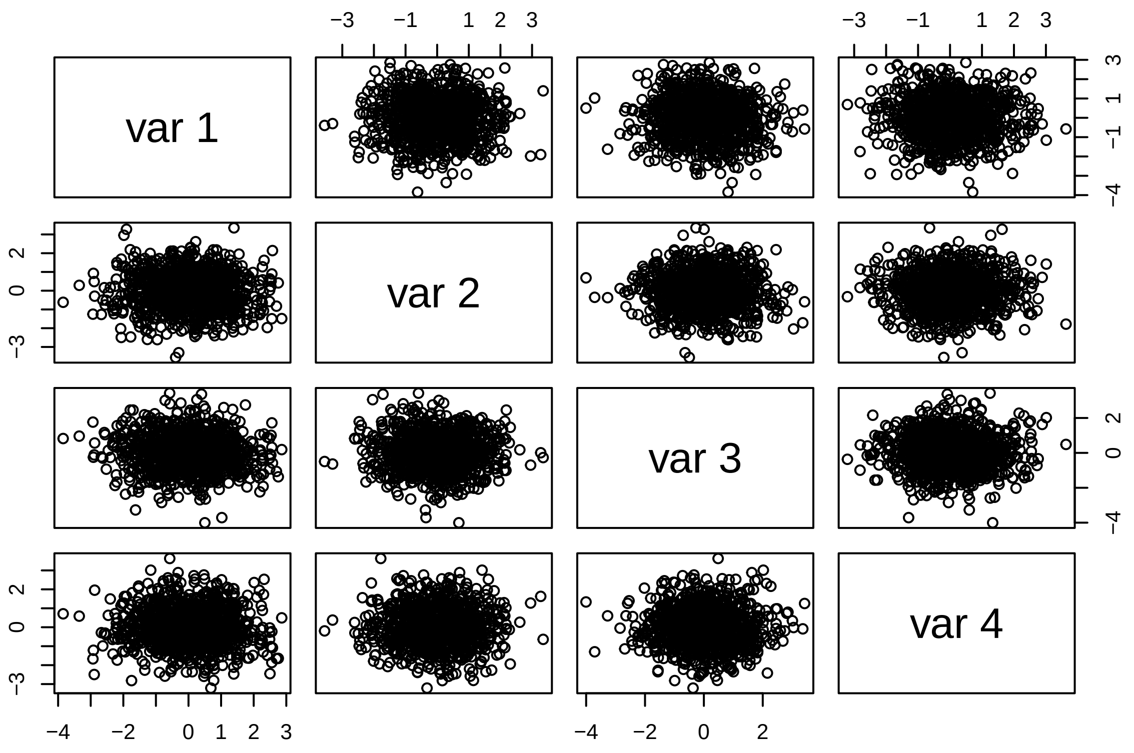
(b) Implement your algorithm in R.

```
standarize <- function(X, mu, covmat){

  df <- matrix(numeric(length(X)), nrow = nrow(X))
  for(i in 1:nrow(X)){
    df[i,] <- X[i,]-mu
  }

  eigen_value <- eigen(covmat)
  covmat_inverse <- eigen_value$vectors %*% diag(1/sqrt(eigen_value$values)) %*% t(eigen_value$vectors)
```

```r
  return(df %*% covmat_inverse)
}
```

(c) Use the generated data from Problem 1 to verify your algorithm.

```r
df <- standarize(sample, mu,covmat)
pairs(df)
```



```r
colMeans(df)
```

```
## [1] -0.0006241406 -0.0127270786  0.0286233730 -0.0006716194
```

```r
cor(df)
```

```
##             [,1]        [,2]        [,3]        [,4]
## [1,]  1.00000000 -0.01297632 -0.05891807 -0.02235814
## [2,] -0.01297632  1.00000000 -0.02500995  0.01385240
## [3,] -0.05891807 -0.02500995  1.00000000 -0.01476958
## [4,] -0.02235814  0.01385240 -0.01476958  1.00000000
```

Since all of the scatter plots looks randomly spread around mean equals 1, we can verify that the algorithm works at standarize the data from problem 1.

Problem 3: Given X is a continuous random variable from the density f(x). Let $\theta = \int g(x)f(x)dx = E[g(x)]$. Suppose we draw iid samples $X_1, ..., X_m$ from $f(x)$. Let $\hat{\theta} = \frac{1}{m}\Sigma_{i=1}^{m} g(X_i)$.

(a) Prove $E[\hat{\theta}] = \theta$

$$E[\hat{\theta}] = E[\frac{1}{m}\Sigma_{i=1}^{m} g(X_i)] = \frac{1}{m}E[\Sigma_{i=1}^{m} g(X_i)]$$
$$E[\hat{\theta}] = \frac{1}{m}\Sigma_{i=1}^{m}\theta = \frac{m\theta}{m} = \theta$$

(b) Prove $Var[\hat{\theta}] = Var[g(X)]/m$, and specify how to estimate $Var[g(x)]$

$$Var[\hat{\theta}] = Var[\frac{1}{m}\Sigma_{i=1}^{m} g(X_i)] = \frac{1}{m^2}Var[\Sigma_{i=1}^{m} g(X_i)]$$
$$Var[\hat{\theta}] = \frac{1}{m^2}(Var[g(X_1)] + ...Var[g(X_m)]) = \frac{1}{m^2}mVar[g(X)] = \frac{Var[g(X)]}{m}$$

(c) Specify how to construct 99% confidence interval of θ using central limit theorem.

$$CI = \hat{\theta} \pm (z_{\alpha/2} \times SE)$$
$$CI = \frac{1}{m}\Sigma_{i=1}^{m} g(X_i) \pm (2.576 \times \sqrt{\frac{Var[g(X)]}{m}})$$

(d) Suppose $f(x)$ is the exponential density with the rate, 1/3. Write a function (`mc2()`) to calculate a Monte Carlo estimate of $E[\sqrt{X}]$

```
mc2 <- function(m){

  lambda <- 1/3
  u <- runif(m)
  exp <- -log(1-u)/lambda
  return(c(mean(sqrt(exp)), sd(sqrt(exp))))

}
```

(e) Construct the 95% confidence interval of $E[\sqrt{X}]$. Repeat your function 1000 times, how often the confidence interval capture the true value of $E[\sqrt{X}]$.

```
means <- replicate(1000,mc2(1000))
true_mean <- integrate(function(x) sqrt(x) * 1/3 * exp(-1/3 * x), lower = 0, upper = Inf)$value
CI <- list(means[1,] + qnorm(0.975)*means[2,]/sqrt(1000), means[1,] - qnorm(0.975)*means[2,]/sqrt(1000)
result <- logical(0)
for (i in 1:1000) {
  if(true_mean>CI[[1]][i]){
    result <- c(result,FALSE)
  }else if(true_mean<CI[[2]][i]){
    result <- c(result,FALSE)
  }else{
    result <- c(result,TRUE)
  }
}
mean(result)
```

```
## [1] 0.954
```

Since all we have all TURE inside the result vector, we can tell that all of the estimated confidence interval captures the theoretical true mean.

Problem 4: Find the air-conditioning data set aircondit from the boot package. The data includes the 12 time intervals in hours between successive failures of the air-conditioning equipment. Assume that the time intervals between failures follow an exponential distribution with the hazard rate λ. Please use bootstrap to estimate the bias and standard error of $\hat{\lambda}_{MLE}$.

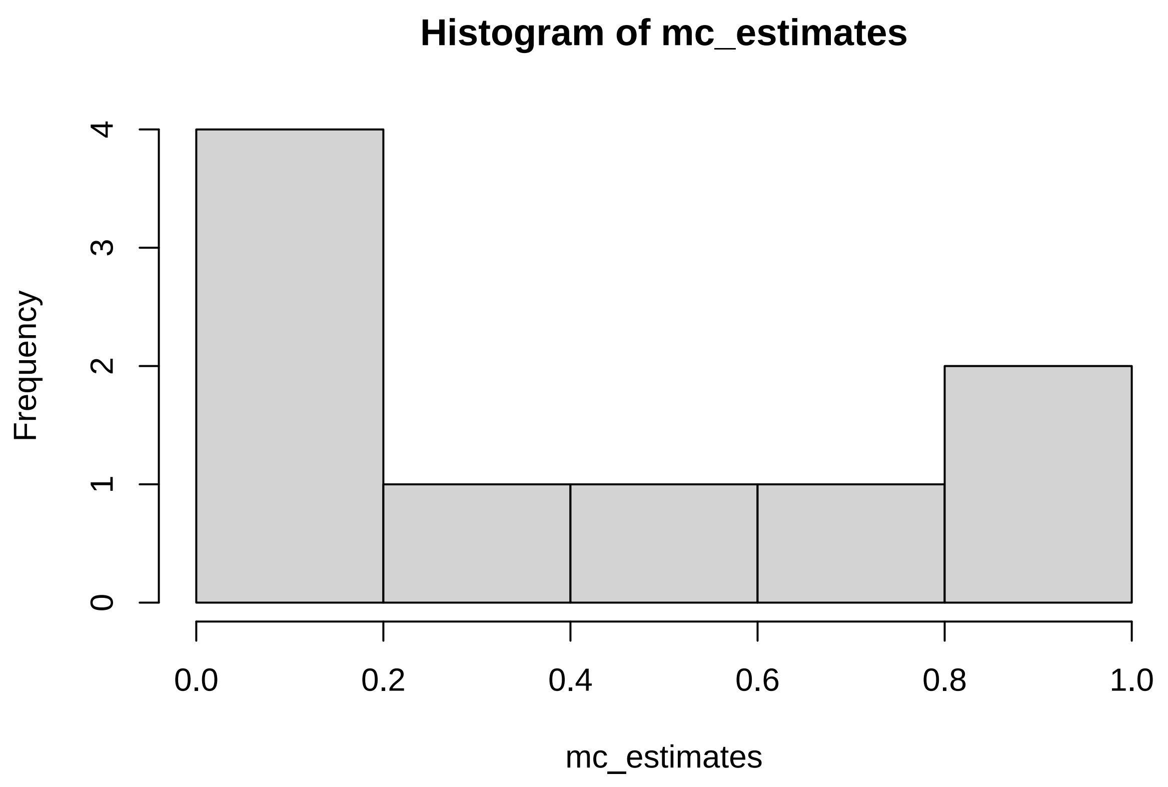
```
# By MLE, we can estimate our lambda as the following:
MLE_lambda <- function(df, index){
  resampled_data <- df[index, ]
  return(1/mean(resampled_data))
}


boot(aircondit, MLE_lambda, R = 10000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = aircondit, statistic = MLE_lambda, R = 10000)
##
##
## Bootstrap Statistics :
##       original      bias    std. error
## t1* 0.00925212 0.001276383 0.004281257
```

Problem 5: Suppose X is a random variable from Beta($\alpha = 3$, $\beta = 2$).

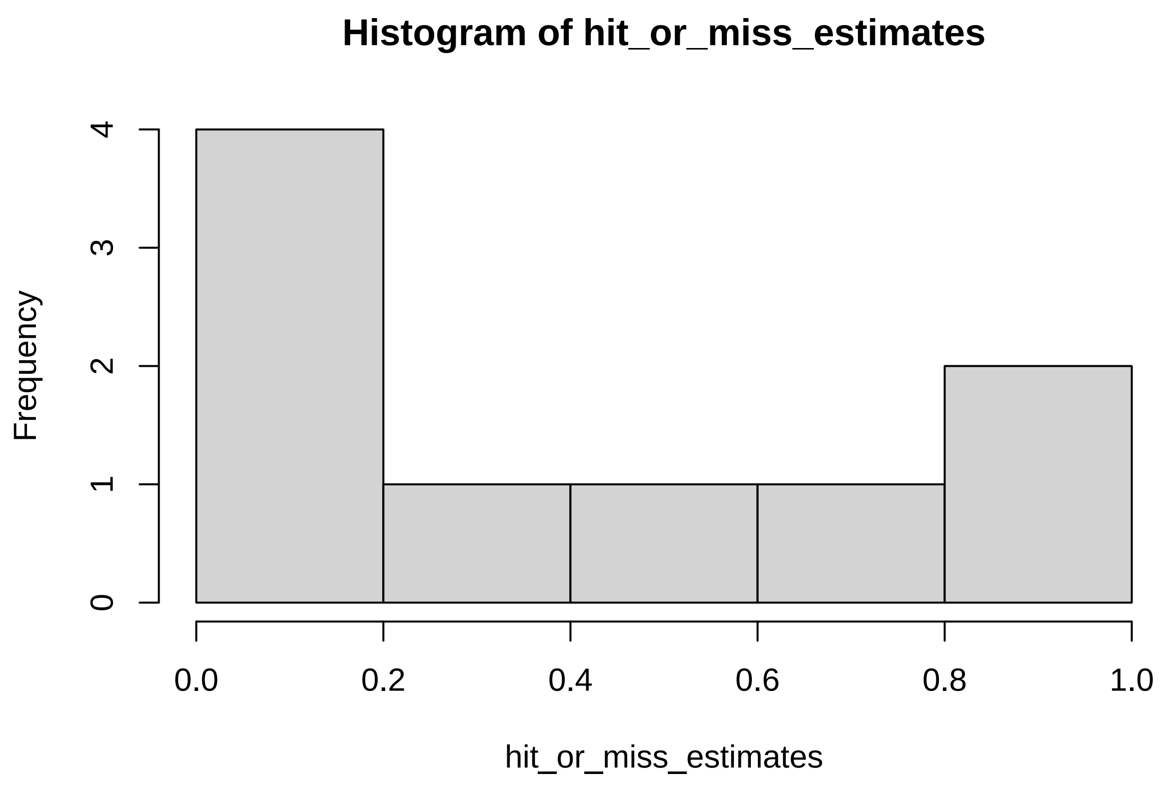
(a) Write R code to compute the Monte Carlo estimator of the CDF.

```
simulated_values <- rbeta(5000, 3, 2)

estimate_cdf <- function(x) {
  mean(simulated_values <= x)
}

mc_estimates <- sapply(seq(0.1, 0.9, by = 0.1), estimate_cdf)
hist(mc_estimates)
```



(b) Write R code using the "hit-or-miss" approach to estimate the CDF.

```
hit_or_miss <- function(x, z, alpha, beta) {
  beta_sample <- rbeta(length(z), alpha, beta)
  mean(beta_sample < x)
}
hit_or_miss_estimates<- sapply(seq(0.1, 0.9, by = 0.1), hit_or_miss, z=runif(5000), alpha=3, beta=2)

hist(hit_or_miss_estimates)
```

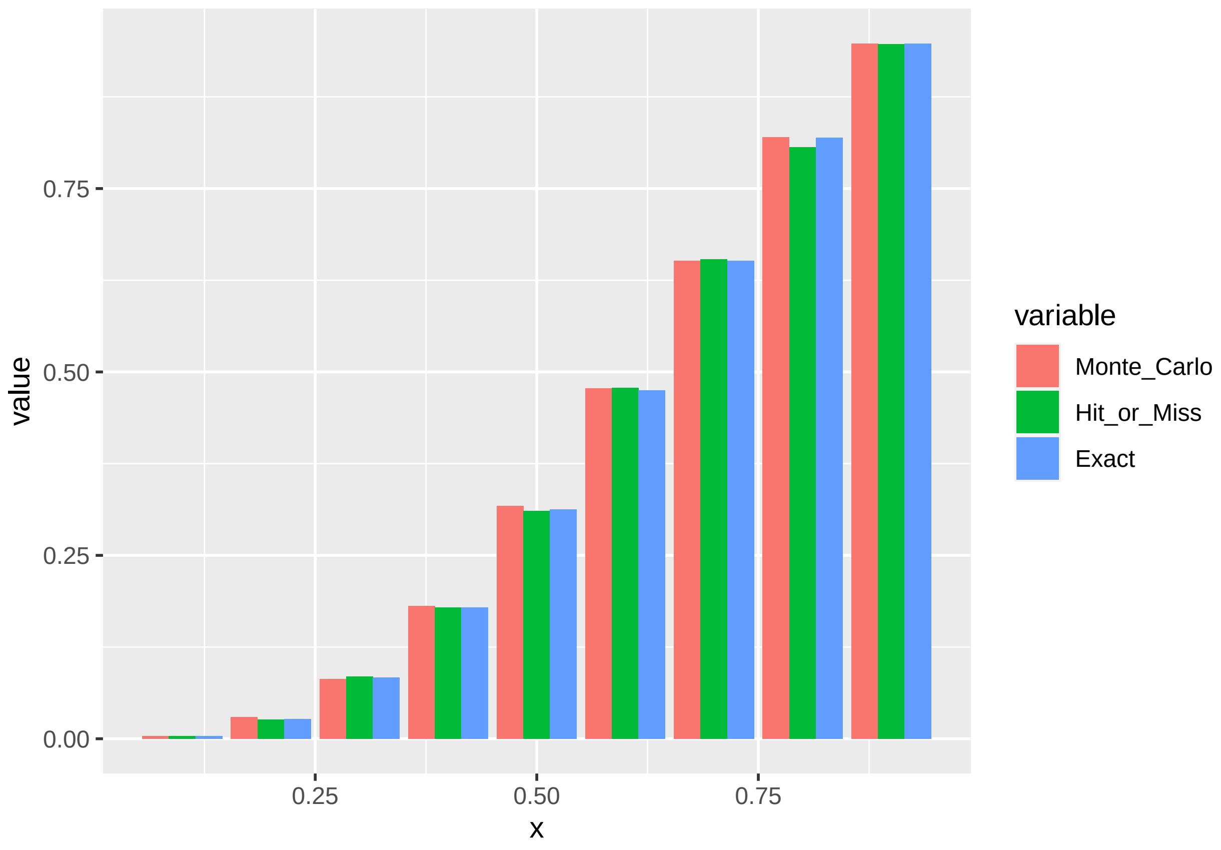


(c) Compare your estimates with the outputs of the pbeta function in R for x = 0.1, 0.2, . . . , 0.9.

```
exact_cdf <- pbeta(seq(0.1, 0.9, by = 0.1), 3, 2)

comparison <- data.frame(
  x = seq(0.1, 0.9, by = 0.1),
  Monte_Carlo = mc_estimates,
  Hit_or_Miss = hit_or_miss_estimates,
  Exact = exact_cdf
)

comparison
```

```
##     x Monte_Carlo Hit_or_Miss  Exact
## 1 0.1      0.0036      0.0040 0.0037
## 2 0.2      0.0300      0.0260 0.0272
## 3 0.3      0.0812      0.0848 0.0837
## 4 0.4      0.1808      0.1788 0.1792
## 5 0.5      0.3174      0.3108 0.3125
## 6 0.6      0.4780      0.4786 0.4752
## 7 0.7      0.6514      0.6538 0.6517
## 8 0.8      0.8198      0.8066 0.8192
## 9 0.9      0.9476      0.9468 0.9477
```

```
comparison %>%
  melt(id.vars = "x")%>%
  ggplot(aes(x = x,
             y = value,
             fill = variable))+geom_bar(stat="identity", position="dodge")
```



0.00
0.25
0.50
0.75
value
0.25
0.50
0.75
x
variable
Monte_Carlo
Hit_or_Miss
Exact