HW 3 Group 09

Sunanda Maity, Shreya Saha, Shrasti Dwivedi, Jayansh Jain, Karan Pratap Lohiya

Q1 Generate data from bivariate unimodal and bimodal distributions and draw contour plots of them. From those contour plots, detect the number of modes of those data.

```
# Install and load required packages
# install.packages("MASS")         # For mvrnorm (multivariate normal generation)
# install.packages("ggplot2")      # For plotting
# install.packages("ggfortify")    # For GMM plot and analysis
# install.packages("mclust")       # For Gaussian Mixture Models
# install.packages("ks")
# Generate data for unimodal distribution
library(MASS)   # For generating multivariate normal data
library(ks)     # For Kernel Density Estimation (KDE)
library(ggplot2)  # For visualization
# Genration of data from bivariate normal distributions
set.seed(4)
mean_unimodal <- c(0, 0)
cov_unimodal <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data_unimodal <- mvrnorm(1e4, mu = mean_unimodal, Sigma = cov_unimodal)

# Generate data for bimodal distribution
mean_bimodal1 <- c(-3, -3)
mean_bimodal2 <- c(3, 3)
cov_bimodal <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data_bimodal1 <- mvrnorm(5e3, mu = mean_bimodal1, Sigma = cov_bimodal)
data_bimodal2 <- mvrnorm(5e3, mu = mean_bimodal2, Sigma = cov_bimodal)
data_bimodal <- rbind(data_bimodal1, data_bimodal2)
```

```
# Install and load required packages
# install.packages("MASS")         # For mvrnorm (multivariate normal generation)
# install.packages("ggplot2")      # For plotting
# install.packages("ggfortify")    # For GMM plot and analysis
# install.packages("mclust")       # For Gaussian Mixture Models
# install.packages("ks")

library(MASS)   # For generating multivariate normal data
```

```r
library(ks)     # For Kernel Density Estimation (KDE)
library(ggplot2)  # For visualization

# Generate data for unimodal distribution
set.seed(4)
mean_unimodal <- c(0, 0)
cov_unimodal <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data_unimodal <- mvrnorm(1e4, mu = mean_unimodal, Sigma = cov_unimodal)

# Generate data for bimodal distribution
mean_bimodal1 <- c(-3, -3)
mean_bimodal2 <- c(3, 3)
cov_bimodal <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
data_bimodal1 <- mvrnorm(5e3, mu = mean_bimodal1, Sigma = cov_bimodal)
data_bimodal2 <- mvrnorm(5e3, mu = mean_bimodal2, Sigma = cov_bimodal)
data_bimodal <- rbind(data_bimodal1, data_bimodal2)

# Function to find mode for unimodal case
find_mode_kde2d <- function(data) {
  kde_result <- kde2d(data[,1], data[,2], n = 200)  # High-resolution KDE
  max_idx <- which(kde_result$z == max(kde_result$z), arr.ind = TRUE)  # Find highest density index
  mode_x <- kde_result$x[max_idx[1, 1]]
  mode_y <- kde_result$y[max_idx[1, 2]]
  return(c(mode_x, mode_y))
}

# Function to find two modes for bimodal case
find_modes_kde2d <- function(data) {
  kde_result <- kde2d(data[,1], data[,2], n = 200)  # High-resolution KDE
  sorted_indices <- order(kde_result$z, decreasing = TRUE)  # Sort density values
  top_two_indices <- sorted_indices[1:2]  # Get top two peaks
  top_two_coords <- arrayInd(top_two_indices, dim(kde_result$z))
  mode1 <- c(kde_result$x[top_two_coords[1, 1]], kde_result$y[top_two_coords[1, 2]])
  mode2 <- c(kde_result$x[top_two_coords[2, 1]], kde_result$y[top_two_coords[2, 2]])
  return(rbind(mode1, mode2))
}

# Find and print mode for unimodal data
mode_unimodal <- find_mode_kde2d(data_unimodal)
print(sprintf("Unimodal mode: (%.3f, %.3f)", mode_unimodal[1], mode_unimodal[2]))
```

```
[1] "Unimodal mode: (-0.060, 0.009)"
```

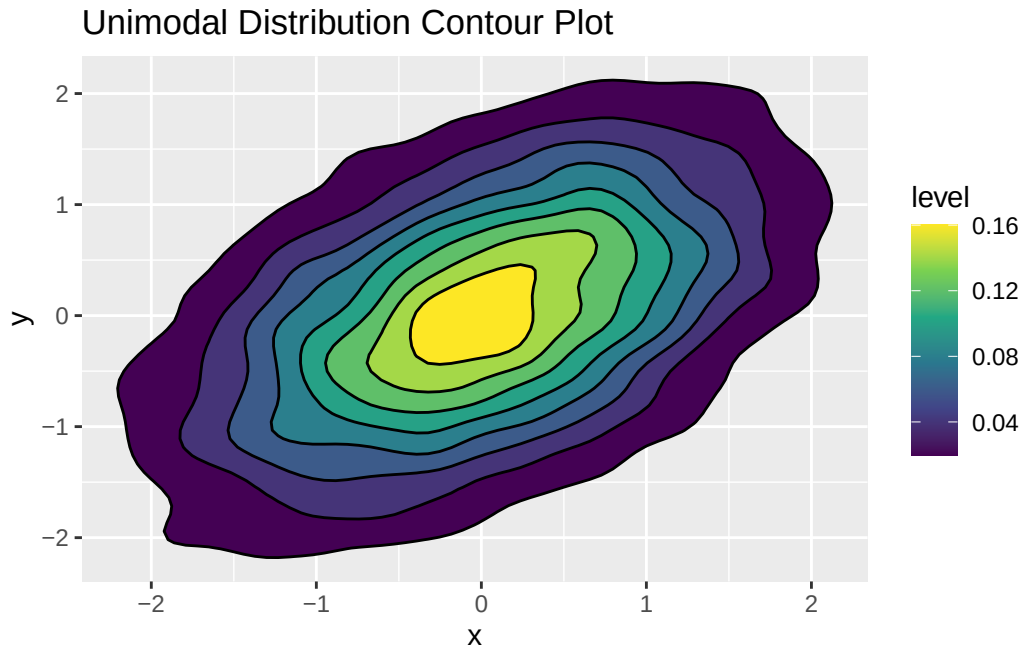
```r
# Find and print both modes for bimodal data
modes_bimodal <- find_modes_kde2d(data_bimodal)
print(sprintf("Bimodal mode 1: (%.3f, %.3f)", modes_bimodal[1, 1], modes_bimodal[1, 2]))
```

```
[1] "Bimodal mode 1: (3.036, 3.058)"
```

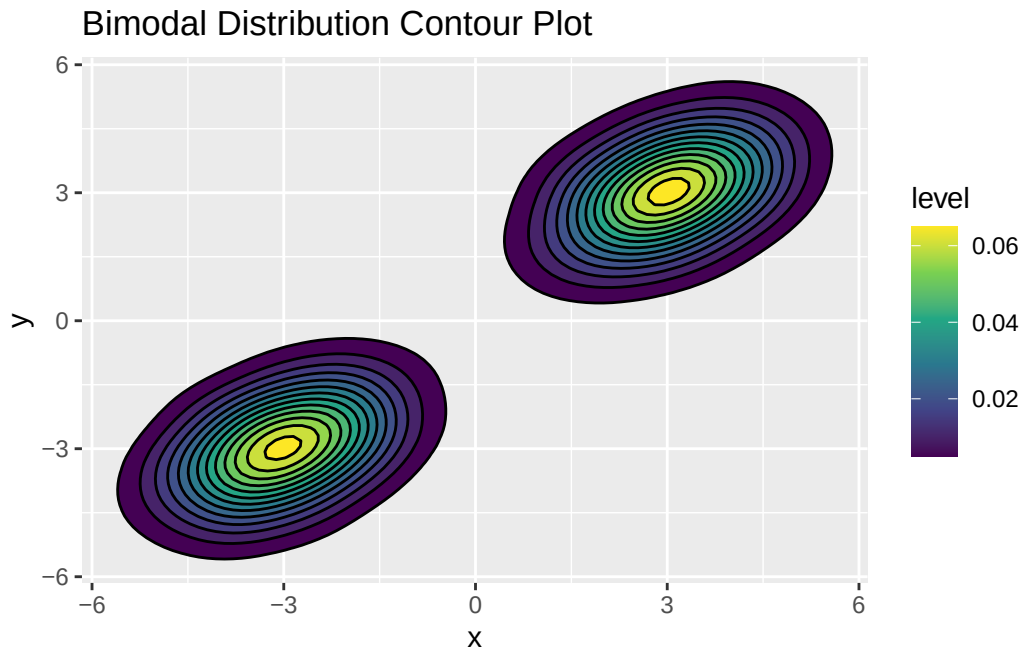
```r
print(sprintf("Bimodal mode 2: (%.3f, %.3f)", modes_bimodal[2, 1], modes_bimodal[2, 2]))
```

```
[1] "Bimodal mode 2: (3.036, 2.996)"
```

```r
# Contour plot for Unimodal distribution
ggplot(data.frame(x = data_unimodal[, 1], y = data_unimodal[, 2]), aes(x, y)) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", color = "black") +
  scale_fill_viridis_c() +
  ggtitle("Unimodal Distribution Contour Plot")
```




```r
# Contour plot for Bimodal distribution
ggplot(data.frame(x = data_bimodal[, 1], y = data_bimodal[, 2]), aes(x, y)) +
  stat_density_2d(aes(fill = ..level..), geom = "polygon", color = "black") +
  scale_fill_viridis_c() +
  ggtitle("Bimodal Distribution Contour Plot")
```



Q2 Using appropriate statistical technique, compute the following integral:

$$\int_0^1 \frac{x^8(1-x)^8(25+816x^2)}{3164(1+x^2)}dx$$

We have estimated the given integral in two methods:

1. Simple Monte Carlo
2. Simple Importance Sampling

Simple Monte Carlo

Here our concern density is $Beta(9,9)$ (first kind) and our concerned estimand under this density is

$$\theta = \int_0^1 \frac{\beta(9,9)(25+816x^2)}{3164(1+x^2)} \frac{x^8(1-x)^8}{\beta(9,9)}dx$$

To estimate θ, we have drawn sample with gradually increasing sample size from 100 to 1,00,000. So our estimator in each step is

$$\hat{\theta} = \frac{1}{N}\Sigma_{t=1}^{N} h(X_t)$$

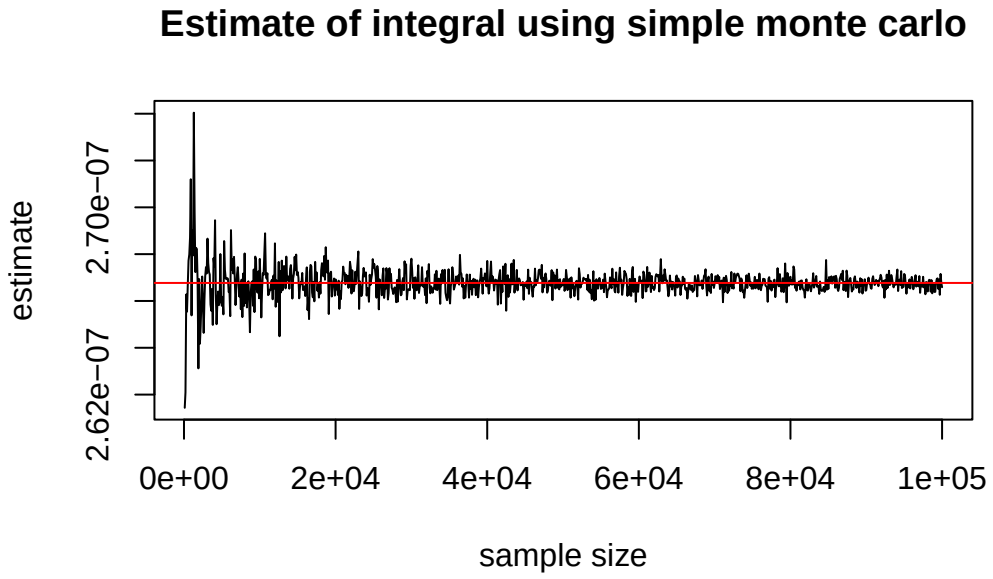
where N is the sample size in every step and

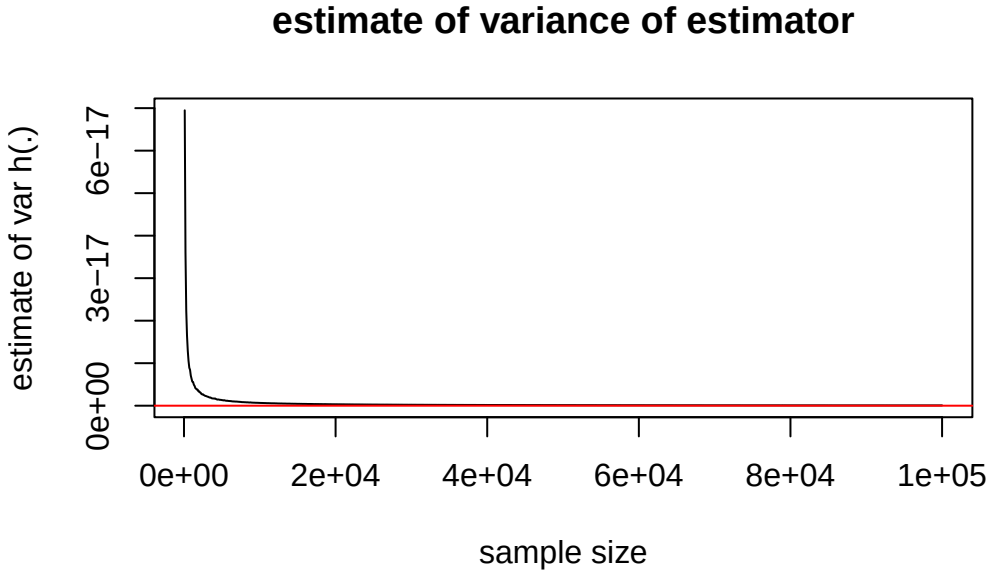
$$h(x) = \frac{\beta(9,9)(25+816x^2)}{3164(1+x^2)}$$

and the variance of the estimator in each step is

$$var(\hat{\theta}) = \frac{var_f(h(X))}{N}$$

Now after calculating the estimate of integral for different sample size, we take the mean of the estimates to get a better estimate of the integral.





From the above diagram of the estimates we can see that the values are converging to $2.667359 \cdot 10^{-}7$. And from the diagram of corresponding variance of the estimators, we can see that the values are converging to 0, which indicates the stability of the estimator for increasing sample sizes. From our study, we can get an

estimate of the above integral as $2.667359 \cdot 10^{-}7$.

Code of the carried out analysis is attached to another file.

Simple Importance Sampling

Here our goal is the same.We want to estimate

$$\theta = \int_0^1 \frac{\beta(9,9)(25+816x^2)}{3164(1+x^2)} \frac{x^8(1-x)^8}{\beta(9,9)} dx$$

Here we will choose a proposal density for drawing samples. Let G be a distribution function with density g . Here our proposal density g is $Beta(9,1)$. So the integral becomes

$$\theta = \int_0^1 \frac{\beta(9,9)(25+816x^2)}{3164(1+x^2)} \frac{x^8(1-x)^8}{\beta(9,9)} \frac{1}{g(x)} \cdot g(x) dx$$

where g is the density of $Beta(9,1)$ distribution.

To estimate θ, we have drawn sample with gradually increasing sample size from 100 to 1,00,000. So our estimator in each step is

$$\hat{\theta} = \frac{1}{N} \Sigma_{t=1}^{N} \frac{h(Z_t) f(Z_t)}{g(Z_t)}$$

where Z follows G and N is the sample size in every step and

$$h(x) = \frac{\beta(9,9)(25+816x^2)}{3164(1+x^2)}$$

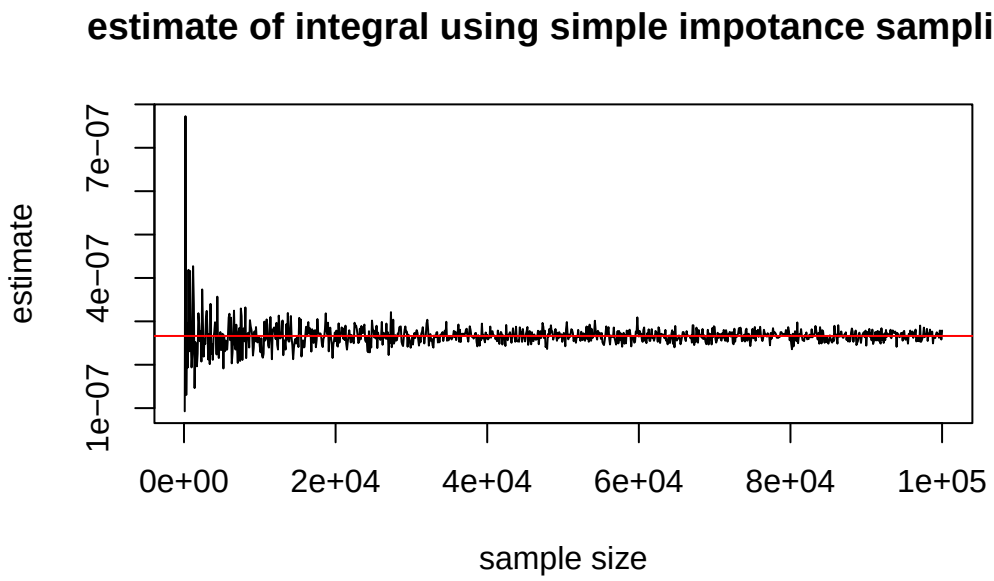
and variance of the estimator in each step is

$$var(\hat{\theta}) = var_g(\frac{h(Z)f(Z)}{g(Z)}) \frac{1}{N}$$

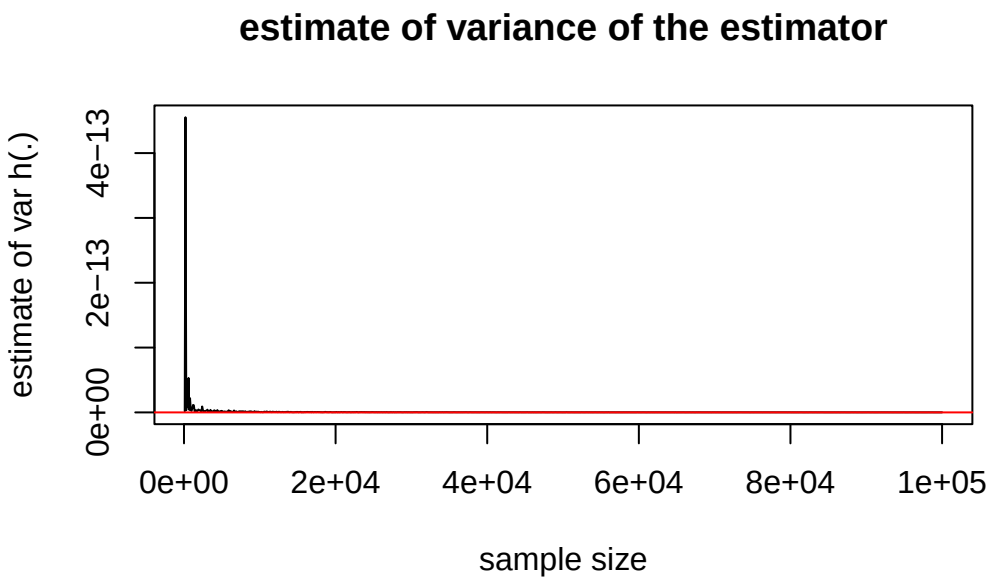
Now after calculating the estimate of integral for different sample size, we take the mean of the estimates to get a better estimate of the integral.

```
h_1 <- function(x){
  h(x)*dbeta(x,9,9)/dbeta(x,9,1)
}

N <-1e3
s.monte <- array(0, dim = c(N,3))

for (i in 1:N) {
  n <- i*1e2
  samp <- rbeta(n,9,1)
```

```
  h.samp <- h_1(samp)
  samp.mean <- mean(h.samp)
  samp.var <- var(h.samp)/n
  s.monte[i,] <- c(n, samp.mean, samp.var)
}
plot(s.monte[,1], s.monte[,2],
     xlab = "sample size", ylab = "estimate",
     main ="estimate of integral using simple impotance sampling",
     type = "l")
abline(h= mean(s.monte[,2]), col = "red")
```




```
plot(s.monte[,1], s.monte[,3],
     xlab = "sample size", ylab = "estimate of var h(.)",
     main ="estimate of variance of the estimator",
     type = "l")
abline(h= 0, col = "red")
```



From the above diagram of the estimates we can see that the values are converging to $2.667359 \cdot 10^{-}7$. And from the diagram of corresponding variance of the estimators, we can see that the values are converging to 0, which indicates the stability of the estimator for increasing sample sizes. From our study, we can get an estimate of the above integral as $2.667359 \cdot 10^{-}7$.

Code of the carried out analysis is attached to another file.