Assignment 4

anonymous

1 General information

I did not use AI for solving this exercise.

2 Bioassay model

2.1 (a)

Given the prior distributions for two parameters, α and β:

$$\alpha \propto N(0, 2^2)$$
$$\beta \propto N(10, 10^2)$$

The known correlation between them is represented as ρ:

$$corr(\alpha, \beta) = 0.6$$

The mean of this bivariate normal posterior distribution is:

$$\bar{\mu} = (\mu_\alpha, \mu_\beta) = (0, 10)$$

The covariance matrix for this posterior distribution is:

$$\sum = \begin{pmatrix} \sigma_\alpha^2 & \rho\,\sigma_\alpha\,\sigma_\beta \\ \rho\,\sigma_\alpha\,\sigma_\beta & \sigma_\beta^2 \end{pmatrix} = \begin{pmatrix} 2^2 & 0.6 \times 2 \times 10 \\ 0.6 \times 2 \times 10 & 10^2 \end{pmatrix} = \begin{pmatrix} 4 & 12 \\ 12 & 100 \end{pmatrix}$$

Therefore, the posterior distribution for these parameters can be expressed as a bivariate normal distribution

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = N\left[\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \sigma_\alpha^2 & \rho\,\sigma_\alpha\,\sigma_\beta \\ \rho\,\sigma_\alpha\,\sigma_\beta & \sigma_\beta^2 \end{pmatrix}\right]$$

```
# Posterior distribution
mean_vector <- c(0,10);
covariance_matrix <- matrix(c(4, 12, 12, 100),2)
joint_posterior <- rmvnorm(4000, mean=mean_vector,
                           sigma=covariance_matrix)
head(joint_posterior)
```

```
            [,1]      [,2]
[1,] -4.01069319  5.134246
[2,] -2.25168379 10.242203
[3,]  0.69110230  1.618235
[4,] -3.12710763  0.353354
[5,] -0.01973444 20.363983
[6,]  1.14800490  1.461065
```

```r
print("\nCovariance Matrix:")
```

```
[1] "\nCovariance Matrix:"
```

```r
print(covariance_matrix)
```

```
     [,1] [,2]
[1,]    4   12
[2,]   12  100
```

```r
print("Mean Vector:")
```

```
[1] "Mean Vector:"
```

```r
print(mean_vector)
```

```
[1]  0 10
```

2.2 (b)

Loading the library and the data.

```r
# Useful functions: quantile()
# and mcse_quantile() (from aaltobda)

data("bioassay_posterior")
# The 4000 draws are now stored in the variable `bioassay_posterior`.
# The below displays the first rows of the data:
head(bioassay_posterior)
```

```
        alpha      beta
1 -0.02050577 10.032841
2  1.21738518  4.504546
3  3.04829407 16.239424
4  1.32272770  4.924268
5  1.36274817 12.880561
6  1.08593225  5.943731
```

```
# Extract alpha and beta samples from bioassay_posterior
alpha_samples <- bioassay_posterior$alpha
beta_samples <- bioassay_posterior$beta

# Calculate the number of samples (S)
S <- length(alpha_samples)

# Calculate mean and variance for alpha and beta samples
mean_alpha <- mean(alpha_samples)
mean_beta <- mean(beta_samples)
var_alpha <- var(alpha_samples)
var_beta <- var(beta_samples)

# Calculate mean Monte Carlo Standard Error (MCSE)
mean_alpha_mcse <- sqrt(var_alpha / S)
mean_beta_mcse <- sqrt(var_beta / S)

# Calculate quantiles for alpha and beta samples
alpha_5 <- quantile(alpha_samples, probs = 0.05)
alpha_95 <- quantile(alpha_samples, probs = 0.95)
beta_5 <- quantile(beta_samples, probs = 0.05)
beta_95 <- quantile(beta_samples, probs = 0.95)

# Calculate MCSE for quantiles
alpha_5_mcse <- mcse_quantile(alpha_samples, prob = 0.05)
alpha_95_mcse <- mcse_quantile(alpha_samples, prob = 0.95)
beta_5_mcse <- mcse_quantile(beta_samples, prob = 0.05)
beta_95_mcse <- mcse_quantile(beta_samples, prob = 0.95)

# Output results
cat(
  "Mean and MCSE:\n",
  paste("Mean alpha MCSE:", mean_alpha_mcse, "\n"),
  paste("Mean alpha:", mean_alpha, "\n"),
  paste("So the true value is between ", mean_alpha - 3*mean_alpha_mcse, " and ",
        mean_alpha + 3*mean_alpha_mcse, "\n"),
  paste("REPORT FOR Mean Alpha:  ", round(mean_alpha + 3*mean_alpha_mcse,1) ,
        "\n---------------\n"),
  paste("Mean beta MCSE:", mean_beta_mcse, "\n"),
  paste("Mean beta:", mean_beta, "\n"),
  paste("So the true value is between ", mean_beta - 3*mean_beta_mcse, " and ",
        mean_beta + 3*mean_beta_mcse, "\n"),
  paste("REPORT FOR Mean Beta:  ", round(mean_beta - 3*mean_beta_mcse,0) ,
        "\n---------------\n"),
  "\nAlpha Quantiles and MCSE:\n",
  paste("MCSE for 5% alpha quantile:", alpha_5_mcse, "\n"),
  paste("Quantile 5% alpha:", alpha_5, "\n"),
  paste("So the true value is between ", alpha_5 - 3*alpha_5_mcse, " and ",
        alpha_5 + 3*alpha_5_mcse, "\n"),
  paste("REPORT FOR Quantile 5% alpha:  ", round(alpha_5 + 3*alpha_5_mcse, 1) ,
        "\n---------------\n"),
  paste("MCSE for 95% alpha quantile:", alpha_95_mcse, "\n"),
  paste("Quantile 95% alpha:", alpha_95, "\n"),
```

```r
    paste("So the true value is between ", alpha_95 - 3*alpha_95_mcse, " and ",
          alpha_95 + 3*alpha_95_mcse, "\n"),
    paste("REPORT FOR Quantile 95% alpha:  ", round(alpha_95 + 3*alpha_95_mcse, 1) ,
          "\n---------------\n"),
    "\nBeta Quantiles and MCSE:\n",
    paste("MCSE for 5% beta quantile:", beta_5_mcse, "\n"),
    paste("Quantile 5% beta:", beta_5, "\n"),
    paste("So the true value is between ", beta_5 - 3*beta_5_mcse, " and ",
          beta_5 + 3*beta_5_mcse, "\n"),
    paste("REPORT FOR Quantile 5% beta:  ", round(beta_5 + 3*beta_5_mcse, 1) ,
          "\n---------------\n"),
    paste("MCSE for 95% beta quantile:", beta_95_mcse, "\n"),
    paste("Quantile 95% beta:", beta_95, "\n"),
    paste("So the true value is between ", beta_95 - 3*beta_95_mcse, " and ",
          beta_95 + 3*beta_95_mcse, "\n"),
    paste("REPORT FOR Quantile 95% beta:  ", round(beta_95 + 3*beta_95_mcse, 0) ,
          "\n---------------\n")
  )
```

```
Mean and MCSE:
 Mean alpha MCSE: 0.0148243453551196
 Mean alpha: 0.985226289184767
 So the true value is between  0.940753253119408  and  1.02969932525013
 REPORT FOR Mean Alpha:   1
---------------
 Mean beta MCSE: 0.0756001627482693
 Mean beta: 10.5964812910431
 So the true value is between  10.3696808027983  and  10.823281779288
 REPORT FOR Mean Beta:   10
---------------

Alpha Quantiles and MCSE:
 MCSE for 5% alpha quantile: 0.0260041159750117
 Quantile 5% alpha: -0.467591355167553
 So the true value is between  -0.545603703092588  and  -0.389579007242518
 REPORT FOR Quantile 5% alpha:   -0.4
---------------
 MCSE for 95% alpha quantile: 0.0420634167918579
 Quantile 95% alpha: 2.61020281115318
 So the true value is between  2.48401256077761  and  2.73639306152875
 REPORT FOR Quantile 95% alpha:   2.7
---------------

Beta Quantiles and MCSE:
 MCSE for 5% beta quantile: 0.0704312509185216
 Quantile 5% beta: 3.99140320865043
 So the true value is between  3.78010945589487  and  4.202696961406
 REPORT FOR Quantile 5% beta:   4.2
---------------
 MCSE for 95% beta quantile: 0.241212887707373
 Quantile 95% beta: 19.3403654436304
 So the true value is between  18.6167267805082  and  20.0640041067525
 REPORT FOR Quantile 95% beta:   20
```

----

3 Importance sampling

3.1 (c)

```
# Useful functions: bioassaylp (from aaltobda)
alpha_test = c(1.896, -3.6,  0.374, 0.964, -3.123, -1.581)
beta_test = c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4)

data("bioassay")

log_importance_weights <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    weights_array <- numeric(length(alpha))
    weights_array <- bioassaylp(alpha,
                                beta,
                                bioassay$x,
                                bioassay$y,
                                bioassay$n)
    return(c(weights_array))
}
```

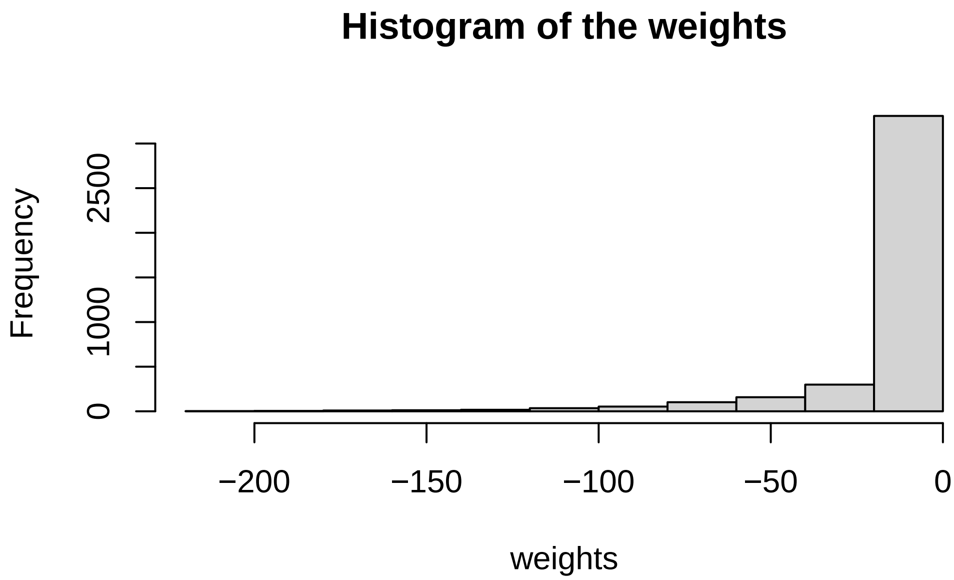
3.2 (d)

```
normalized_importance_weights <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    # Calculate weights_array_norm
    weights_array <- exp(log_importance_weights(alpha, beta))

    sum_weights <- sum(weights_array)

    weights_array_norm_final <- weights_array / sum_weights

    weights_norm <- c(weights_array_norm_final)

    return(weights_norm)
}
```

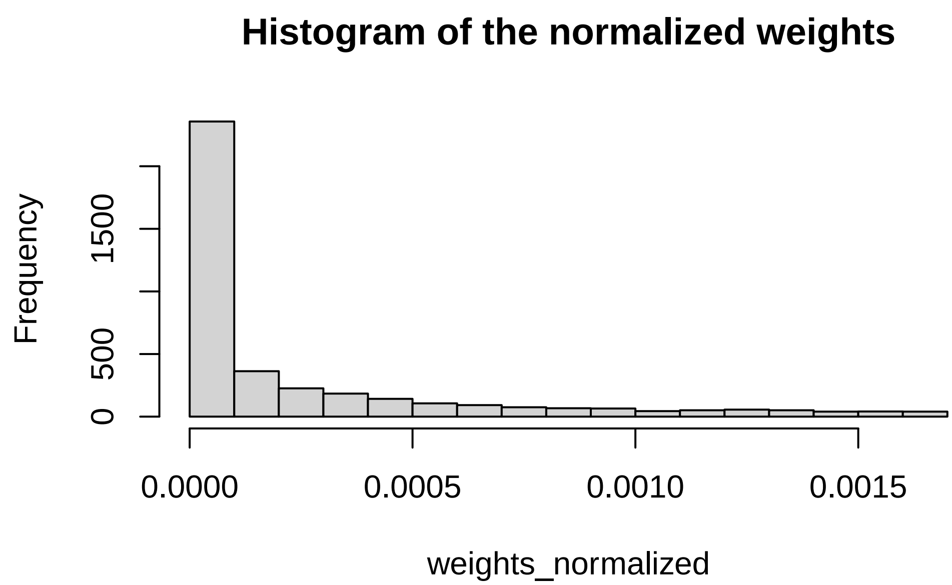
3.3 (e)

```
n <- 4000
joint_posterior <- rmvnorm(n, mean = mean_vector,
                           sigma = covariance_matrix)
alpha_samples <- (joint_posterior[,1])
beta_samples <- (joint_posterior[,2])
```

```r
weights <- log_importance_weights(alpha_samples, beta_samples)
weights_normalized <- normalized_importance_weights(alpha = alpha_samples, beta = beta_samples)
hist(weights, main = "Histogram of the weights")
```




```r
hist(weights_normalized, main = "Histogram of the normalized weights")
```




```r
head(weights_normalized)
```

```
[1] 6.226186e-08 1.336182e-03 1.237930e-04 1.190256e-03 2.495296e-24
[6] 4.534852e-04
```

3.4 (f)

```
S_eff <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    weights_normalized <- normalized_importance_weights(alpha, beta)
    seff <- 1 / sum(weights_normalized^2)
    return(seff)
}

s_eff <- S_eff(alpha = alpha_samples, beta = beta_samples)

cat("S_eff: ", s_eff, "\n")
```

```
S_eff:  1130.352
```

3.5 (g)

In the context of importance sampling, the effective sample size serves as a measure of how efficiently a set of samples represents the target distribution. A high effective sample size is desirable because it indicates that the samples are making a substantial contribution to the estimation process. Conversely, a low effective sample size suggests that many samples are redundant or don't significantly impact the final result.

Upon examining the histogram, it becomes evident that a substantial number of small weights occur quite frequently. Specifically, weights falling in the range between 0 and 0.0002 appear around 3000 times. However, our goal is to ensure that all samples carry equal importance. Consequently, the effective sample size should ideally be around $4000 - 3000 = 1000$, which aligns closely with the calculated value of S_eff. This adjustment aims to balance the importance of each sample in the estimation process.

3.6 (h)

```
posterior_mean <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    weights_normalized <- normalized_importance_weights(alpha = alpha, beta = beta)
    alpha_posterior_mean <- 0
    beta_posterior_mean <- 0
    for (i in 1:length(alpha)){
      i
      alpha_posterior_mean <- alpha_posterior_mean + alpha[i]*weights_normalized[i]
      beta_posterior_mean <- beta_posterior_mean + beta[i]*weights_normalized[i]
    }
    return(c(alpha_posterior_mean, beta_posterior_mean))
}
posterior_means <- posterior_mean(alpha=alpha_samples, beta=beta_samples)

print("Posterior Means: ")
```

```
[1] "Posterior Means: "
```

```r
print(posterior_means)
```

```
[1]  0.9525009 10.4930082
```

```r
weight = normalized_importance_weights(alpha_samples, beta_samples)

alpha_f = (alpha_samples^2) * weight
beta_f = (beta_samples^2) * weight

sq_alpha = sum(alpha_f)
sq_beta = sum(beta_f)

mean_alpha = posterior_mean(alpha_samples, beta_samples)[1]
mean_beta = posterior_mean(alpha_samples, beta_samples) [2]
var_alpha = sq_alpha - (mean_alpha^2)
var_beta =  sq_beta - (mean_beta^2)
seff = S_eff (alpha_samples, beta_samples)

mcse_alpha = sqrt(var_alpha/seff)
mcse_beta = sqrt(var_beta/seff)

cat ("MCSE alpha", mcse_alpha, "\n")
```

```
MCSE alpha 0.02736892
```

```r
cat ("MCSE beta", mcse_beta, "\n")
```

```
MCSE beta 0.1399351
```

```r
cat(
  paste("Mean alpha:", mean_alpha, "\n"),
  paste("So the true value is between ", mean_alpha - 3*mcse_alpha, " and ",
        mean_alpha + 3*mcse_alpha, "\n"),
  paste("REPORT FOR Mean Alpha:  ", round(mean_alpha + 3*mcse_alpha, 1) ,
        "\n---------------\n"),
  paste("Mean beta MCSE:", mean_beta_mcse, "\n"),
  paste("Mean beta:", mean_beta, "\n"),
  paste("So the true value is between ", mean_beta - 3*mcse_beta, " and ",
        mean_beta + 3*mcse_beta, "\n"),
  paste("REPORT FOR Mean Beta:  ", round(mean_beta + 3*mcse_beta, 0) ,
        "\n---------------\n")
)
```

```
Mean alpha: 0.95250091924671
 So the true value is between  0.870394149325556  and  1.03460768916786
 REPORT FOR Mean Alpha:   1
---------------
```

```
 Mean beta MCSE: 0.0756001627482693
 Mean beta: 10.4930081585717
 So the true value is between  10.0732028041719  and  10.9128135129716
 REPORT FOR Mean Beta:   11
---------------
```