BSDA: Assignment 4

Anonymous student

Contents

General Information to include

- **Time used for reading and self-study exercises**: ~ 5 hours.
- **Time used for the assignment**: ~11 hours
- **Good with assignment**: Implement importance sampling in a step by step way, in this way is easy to understand it and you actually know what's going on.
- **Things to improve in the assignment**: I felt like this time the assignment was a little bit harder to solve just by reading chapter 10. I had some trouble understanding why in the last question the variance identity was proposed as a hint, instead of just using var() function.

1. Bioassay model

```
library(bsda)
library(ggplot2)
library(tidyverse)
library(kableExtra)
```

a) In the prior distribution for (α, β), the marginal distributions are $\alpha \sim N(0, 2^2)$ and $\beta \sim N(10, 10^2)$, and the correlation between them is $\text{corr}(\alpha, \beta) = 0.6$. Report the mean (vector of two values) and covariance (two by two matrix) of the bivariate normal distribution.

We have a bivariate normal distribution as the joint prior distribution of the parameters α and β:

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}\mu_\alpha\\ \mu_\beta\end{pmatrix}, \begin{pmatrix}\sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta\\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2\end{pmatrix}\right)$$

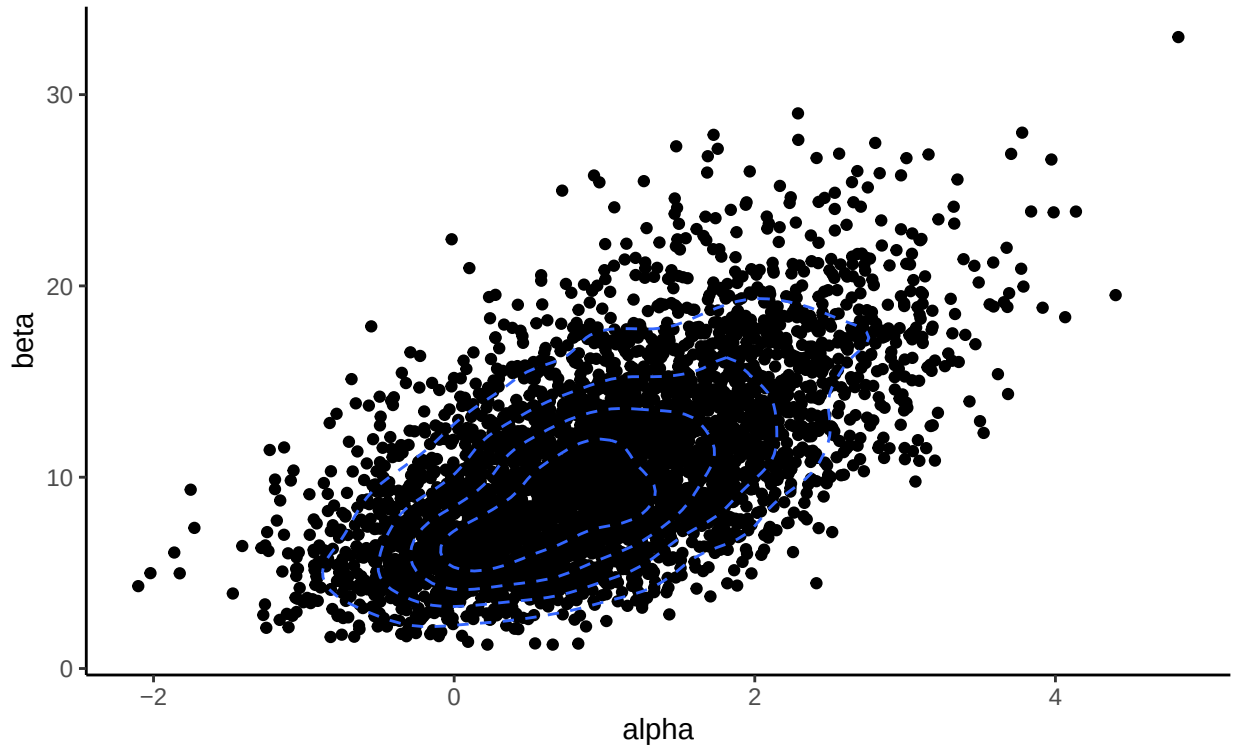
Then:

$$\begin{pmatrix}\mu_\alpha \\ \mu_\beta\end{pmatrix} = \begin{pmatrix}0 \\ 10\end{pmatrix}$$
$$\begin{pmatrix}\sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2\end{pmatrix} = \begin{pmatrix}2^2 & 0.6\cdot 2\cdot 10 \\ 0.6\cdot 2\cdot 10 & 10^2\end{pmatrix} = \begin{pmatrix}4 & 12 \\ 12 & 100\end{pmatrix}$$

$$\begin{pmatrix}\alpha \\ \beta\end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix}0 \\ 10\end{pmatrix}, \begin{pmatrix}4 & 12 \\ 12 & 100\end{pmatrix}\right)$$

b) You are given 4000 independent draws from the posterior distribution of the model. Load the draws with `data("bioassay_posterior")`. Report the mean as well as 5 % and 95 % quantiles separately for both α and β. Report also the Monte Carlo standard errors (MCSEs) for the mean and quantile estimates. Report as many digits for the mean and quantiles as the MCSEs allow. In other words, leave out digits where MCSE is nonzero. Explain in words what does Monte Carlo standard error mean and how you decided the number of digits to show.

```
data("bioassay_posterior")

bioassay_df <- tibble(bioassay_posterior)

#plot draws
ggplot(bioassay_df, aes(x = alpha, y = beta)) +
  geom_point() +
  geom_density_2d(bins = 5, linetype = 2) + theme_classic()
```



```r
#mean alpha
mean_a <- mean(bioassay_posterior$alpha)
mean_a
```

```
## [1] 0.9852263
```

```r
#quantiles alpha
quantile_5_a <- quantile(bioassay_posterior$alpha, 0.05)
quantile_95_a <- quantile(bioassay_posterior$alpha, 0.95)
c(quantile_5_a, quantile_95_a)
```

```
##         5%        95%
## -0.4675914  2.6102028
```

```r
#mean beta
mean_b <- mean(bioassay_posterior$beta)
mean_b
```

```
## [1] 10.59648
```

```r
#quantiles beta
quantile_5_b <- quantile(bioassay_posterior$beta, 0.05)
quantile_95_b <- quantile(bioassay_posterior$beta, 0.95)
c(quantile_5_b, quantile_95_b)
```

```
##        5%       95%
##  3.991403 19.340365
```

```r
#mcse mean alpha
mcse_mean_a <- sqrt(var(bioassay_posterior$alpha)/length(bioassay_posterior$alpha))
mcse_mean_a
```

```
## [1] 0.01482435
```

```r
#mcse mean beta
mcse_mean_b <- sqrt(var(bioassay_posterior$beta)/length(bioassay_posterior$beta))
mcse_mean_b
```

```
## [1] 0.07560016
```

```r
#mcse quantiles alpha
mcse_q_5_a <- mcse_quantile(bioassay_posterior$alpha, 0.05)
mcse_q_95_a <- mcse_quantile(bioassay_posterior$alpha, 0.95)
c(mcse_q_5_a$mcse, mcse_q_95_a$mcse)
```

```
## [1] 0.02600412 0.04206342
```

```r
#mcse quantiles beta
mcse_q_5_b <- mcse_quantile(bioassay_posterior$beta, 0.05)
mcse_q_95_b <- mcse_quantile(bioassay_posterior$beta, 0.95)
c(mcse_q_5_b$mcse, mcse_q_95_b$mcse)
```

```
## [1] 0.07043125 0.24121289
```

Given that we have samples $\alpha_s, \beta_s \sim p(\alpha, \beta|y)$:

the MCSEs for the α and β mean estimates are 0.0148243, and 0.0756002 respectively. Additionally, the MCSEs for the α and β quantile estimates are:

Parameter	5%	95%
alpha	0.02600412	0.04206342
beta	0.07043125	0.2412129

Then we can report the mean and quantiles estimates for α as:

$$E_{p(\alpha|y)}(\alpha) = 1.0$$
$$\text{quantiles 5\% and 95\%: } [-0.5, 2.6]$$

and the mean and quantiles estimates for β as:

$$E_{p(\beta|y)}(\beta) = 10.6$$
$$\text{quantiles 5\% and 95\%: } [4.0, 19]$$

The MCSE is an estimate of the variability in the samples obtained from a Monte Carlo simulation due to the randomness in the sampling process. We are deciding the digits to report by leaving out digits that are just random noise, for example looking at a MCSE of 0.0148243, we will only report the digits where this MCSE is 0, so we will report 1.0 instead of 0.9852263.

c) Implement a function for computing the log importance ratios (log importance weights) when the importance sampling target distribution is the posterior distribution, and the proposal distribution is the prior distribution from a). Below is a test example, the functions can also be tested with `markmyassignment`. Explain in words why it's better to compute log ratios instead of ratios.

From equation 10.3 in BDA3 importance weights are derived from the equation:

$$w(\theta^s) = \frac{q(\theta^s|y)}{g(\theta^s)}$$

where q is the notation for unnormalized densities, that is $q(\theta|y)$ equals $p(\theta|y)$ times some factor that doesn't depend on θ. In the usual use of Bayes' Theorem we work with $p(\theta)p(y|\theta)$. Substituting this expression:

$$w(\theta^s) = \frac{p(\theta^s)p(y|\theta^s)}{p(\theta^s)} = p(y|\theta^s)$$

```
data("bioassay")

log_importance_weights <- function(alpha, beta) {

  log_w <- bioassaylp(alpha, beta, bioassay$x, bioassay$y, bioassay$n)

  return(log_w)
}
```

Floating-point representation in computers can only represent a finite number of digits. This can lead to overflow issues, where the number becomes too large to be represented accurately using the available bits in a computer's memory. This can also lead to underflow issues, where the number becomes too close to zero to be precisely represented. Using logarithms instead can help, because it turns very small numbers into negative values with reasonable magnitudes, preventing them from being treated as zero in calculations. Additionally, logarithms convert multiplication and division into addition and subtraction, which makes it easier to make computations with numbers on a wide range of magnitudes.

d) Implement a function for computing normalized importance ratios from the unnormalized log ratios in c). In other words, exponentiate the log ratios and scale them such that they sum to one. Explain in words what is the effect of exponentiating and scaling so that sum is one.

Normalized weights:

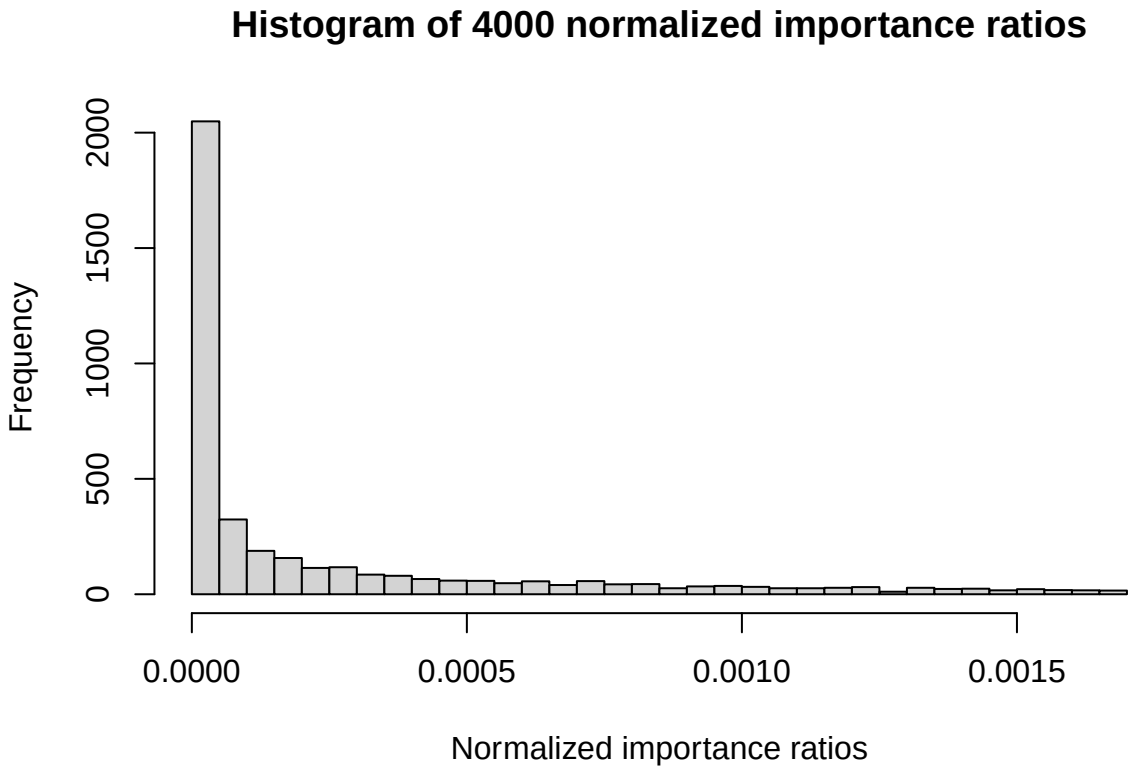
$$\tilde{w}(\theta^s) = \frac{w(\theta^s)}{\sum_{s'=1}^{S} w(\theta^{s'})}$$

```
normalized_importance_weights <- function(alpha, beta) {

  log_w <- log_importance_weights(alpha, beta)

  e_w <- exp(log_w)

  e_norm_w <- e_w/sum(e_w)

  return(e_norm_w)
}
```

Exponentiation "undoes" the logarithmic transformation and transforms the log weights back into their original scale. Scaling involves dividing each weight by the sum of all the weights to ensure that the total weight assigned to all samples is equal to one. In this way the normalized weights correspond to the volume of space represented by θ^s, these normalized weights indicate the relative probability that a specific sample represents the parameter space therefore they must sum to 1.

e) Sample 4000 draws of α and β from the prior distribution from a). Compute and plot a histogram of the 4000 normalized importance ratios. Use the functions you implemented in c) and d).

```
set.seed(987)
sample <- rmvnorm(4000, c(0,10), matrix(c(4,12,12,100),nrow = 2,
                                        ncol = 2,byrow = TRUE))
n_weights <- normalized_importance_weights(sample[,1], sample[,2])

hist(n_weights, breaks = 50,
     xlab = "Normalized importance ratios", ylab = "Frequency",
     main = "Histogram of 4000 normalized importance ratios")
```



f) Using the importance ratios, compute the importance sampling effective sample size S_{eff} and report it.

```
is.finite(var(n_weights))
```

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

```
S_eff <- function(alpha, beta) {

  s_eff <- 1/sum(normalized_importance_weights(alpha, beta)^2)

  return(s_eff)
}

S_eff(sample[,1], sample[,2])
```

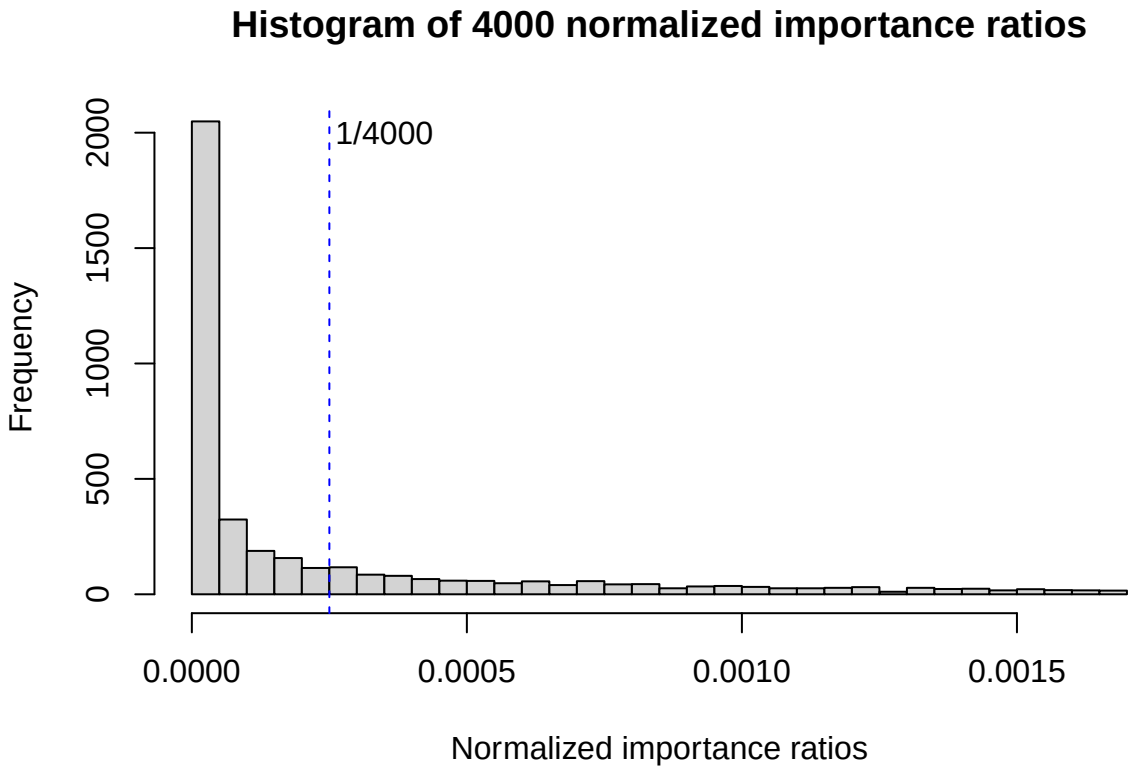
```
## [1] 1145.458
```

Given that the variance of the weights is finite, we can calculate the importance sampling effective sample size using the following approximation:

$$S_{\text{eff}} = \frac{1}{\sum_{s=1}^{S} (\tilde{w}(\theta^s))^2} = 1145.458$$

g) Explain in your own words what the importance sampling effective sample size represents. Also explain how the effective sample size is seen in the histogram of the weights that you plotted in e).

In importance sampling we generate draws and each draw is associated with a weight. These weights indicate how well each draw represents the target distribution we want to estimate. Some draws are more important, then they have higher weights, while others are less important –lower weights. The importance sampling effective sample size tells us how many equally informative, independent draws we effectively have, given the weighted sample we generated. Therefore, by looking at the equation we can see it penalizes draws with large weights (which are less common) and rewards draws with small weights (which are more common). The larger the weights, the more they contribute to reducing S_{eff}, while smaller weights increase S_{eff}.



If only one weight is dominating the effective sample approximation computes an effective sample size of 1. If all the weights are equal then the approximation would compute an effective sample size of 4000. For this case we can see that we are somewhere between there, meaning that we have not only one dominating weight neither equal weights, giving us an effective sample size of 1145.458. This is a way of estimating how good our proposal distribution is. If we would be drawing from the actual target distribution the importance weights would be $\frac{1}{4000}$, from the plot we can see that some of the draws have smaller weights and some of them have larger weights. We can see that there are some weights around $\frac{1}{4000}$, so we should expect an effective sample size smaller than 4000.

h) Implement a function for computing the posterior mean using importance sampling, and compute the mean using your 4000 draws. Explain in your own words the computation for importance sampling. Report the means for alpha and beta, and also the Monte Carlo standard errors (MCSEs) for the mean estimates. Report the number of digits for the means based on the MCSEs.

```
posterior_mean <- function(alpha, beta) {

  mean_a <- mean(alpha*normalized_importance_weights(alpha, beta)) /
    mean(normalized_importance_weights(alpha, beta))
  mean_b <- mean(beta*normalized_importance_weights(alpha, beta))/
    mean(normalized_importance_weights(alpha, beta))

  return(c(mean_a, mean_b))
}


posterior_mean(sample[,1], sample[,2])
```

```
## [1]  0.9873051 10.5835729
```

```
mean_a_2 <- mean(sample[,1]^2*normalized_importance_weights(sample[,1],
                                                            sample[,2])) /
  mean(normalized_importance_weights(sample[,1], sample[,2]))

mean_b_2 <- mean(sample[,2]^2*normalized_importance_weights(sample[,1],
                                                            sample[,2])) /
  mean(normalized_importance_weights(sample[,1], sample[,2]))

var_a <- mean_a_2 - posterior_mean(sample[,1], sample[,2])[1]^2
var_b <- mean_b_2 - posterior_mean(sample[,1], sample[,2])[2]^2

c(sqrt(var_a / S_eff(sample[,1], sample[,2])),
  sqrt(var_b / S_eff(sample[,1], sample[,2])))
```

```
## [1] 0.0272911 0.1398812
```

the MCSEs for the α and β posterior mean estimates are 0.0272911, and 0.1398812 respectively. Then we can report the mean estimates for α and β as 1.0 and 11

Importance sampling is used to estimate properties of a target probability distribution when it's difficult to directly sample from that distribution. For sampling distribution, we start with a target distribution that we are interested in, then we choose a different distribution $g(\theta)$, called the proposal distribution, which is a probability density from which we can generate S draws $\theta^1, \theta^2, \ldots, \theta^S$. For each draw generated from the proposal distribution, we calculate a weight. These weights represent how well the sample represents the target distribution. Finally, we use the draws and their weights to estimate properties of the target distribution. In importance sampling we don't require the proposal to be everywhere higher than the target distribution, since we can make a correction in both directions. We can make a correction if the target is actually lower than the proposal or if it's actually higher. When we compute the expectation we are weighting these draws by a ratio between the proposal and the target, the weights are the target divided by the proposal density. If the target distribution is higher than the proposal there is a higher weight, and if the target is lower than the proposal there is a lower weight.

$$E[f(\theta)] \approx \frac{\sum_s w_s f(\theta^{(s)})}{\sum_s w_s}, \qquad \text{where} \quad w_s = \frac{q(\theta^{(s)})}{g(\theta^{(s)})}$$