Assignment 4

anonymous

2023-09-25

General information

AI was used in part f and g to search some related information and code.

Bioassay model

(a)

The mean vector is given by:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_\alpha \\ \mu_\beta \end{bmatrix}$$

We can put our data in this vector where μ_α=0,μ_β=10

Given the result:

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

The covariance matrix is:

$$\Sigma = \begin{bmatrix} \sigma_\alpha^2 & \text{cov}(\alpha, \beta) \\ \text{cov}(\alpha, \beta) & \sigma_\beta^2 \end{bmatrix}$$

Where:

$$\text{cov}(\alpha, \beta) = \text{corr}(\alpha, \beta) \times \sigma_\alpha \times \sigma_\beta$$

Because$\sigma_\alpha = 2$ $\sigma_\beta = 10$ $\text{corr}(\alpha, \beta) = 0.6$

So $\text{cov}(\alpha, \beta) = 0.6 \times 2 \times 10 = 12$ The covariance matrix is:

$$\Sigma = \begin{bmatrix} 4 & 12 \\ 12 & 100 \end{bmatrix}$$

(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
```

```r
# Mean
mean_alpha <- mean(bioassay_posterior$alpha)
mean_beta <- mean(bioassay_posterior$beta)
cat("The mean_alpha=",mean_alpha,"\n")
```

```
## The mean_alpha= 0.9852263
```

```r
cat("The mean_beta=",mean_beta,"\n")
```

```
## The mean_beta= 10.59648
```

```r
# 5% and 95% Quantiles using the quantile function
quantile_alpha <- quantile(bioassay_posterior$alpha, probs = c(0.05, 0.
95))
quantile_beta <- quantile(bioassay_posterior$beta, probs = c(0.05, 0.9
5))
cat("The 5% and 95% quantiles for alpha:",quantile_alpha,"\n")
```

```
## The 5% and 95% quantiles for alpha: -0.4675914 2.610203
```

```r
cat("The 5% and 95% quantiles for beta:",quantile_beta,"\n")
```

```
## The 5% and 95% quantiles for beta: 3.991403 19.34037
```

```r
S <- length(bioassay_posterior$alpha)  # Number of draws
mcse_mean_alpha <- sqrt(var(bioassay_posterior$alpha) / S)
mcse_mean_beta <- sqrt(var(bioassay_posterior$beta) / S)
cat("The mcse_mean_alpha=",mcse_mean_alpha,"\n")
```

```
## The mcse_mean_alpha= 0.01482435
```

```r
cat("The mcse_mean_beta=",mcse_mean_beta,"\n")
```

```
## The mcse_mean_beta= 0.07560016
```

```r
ES_mean_alpha_lower<-mean_alpha-3* mcse_mean_alpha
ES_mean_alpha_upper<-mean_alpha+3* mcse_mean_alpha
ES_mean_beta_lower<-mean_beta-3* mcse_mean_beta
```

```
ES_mean_beta_upper<-mean_beta+3* mcse_mean_beta
cat("The estimation alpha is between",ES_mean_alpha_lower,"and",ES_mean
_alpha_upper,", so we can consider the aplha=1.","\n")

## The estimation alpha is between 0.9407533 and 1.029699 , so we can c
onsider the aplha=1.

cat("The estimation beta is between",ES_mean_beta_lower,"and",ES_mean_b
eta_upper,", so we can consider the beta=10.6.","\n")

## The estimation beta is between 10.36968 and 10.82328 , so we can con
sider the beta=11.

mcse_quantile_alpha_lower <- mcse_quantile(bioassay_posterior$alpha, 0.
05)
mcse_quantile_alpha_upper <- mcse_quantile(bioassay_posterior$alpha, 0.
95)
mcse_quantile_alpha <- c(mcse_quantile_alpha_lower$mcse,mcse_quantile_a
lpha_upper$mcse)
cat("The MCSEs_alpha:5%:",mcse_quantile_alpha[1],"95%:",mcse_quantile_a
lpha[2],"\n")

## The MCSEs_alpha:5%: 0.02600412 95%: 0.04206342

mcse_quantile_beta_lower <- mcse_quantile(bioassay_posterior$beta, 0.05)
mcse_quantile_beta_upper <- mcse_quantile(bioassay_posterior$beta, 0.95)
mcse_quantile_beta <- c(mcse_quantile_beta_lower$mcse,mcse_quantile_bet
a_upper$mcse)
cat("The MCSEs_beta:5%:",mcse_quantile_beta[1],"95%:",mcse_quantile_bet
a[2])

## The MCSEs_beta:5%: 0.07043125 95%: 0.2412129
```

The mean_alpha= 0.9852263

The mean_beta= 10.59648

The mcse_mean_alpha= 0.01482435

The mcse_mean_beta= 0.07560016

The estimation alpha is between 0.9407533 and 1.029699 , so we can consider the aplha=1.0.

The estimation beta is between 10.36968 and 10.82328 , so we can consider the beta=10.6.

The 5% and 95% quantiles for alpha: -0.4675914-2.610203

The 5% and 95% quantiles for beta: 3.991403-19.34037

The MCSEs_alpha:5%: 0.02600412 95%: 0.04206342

The MCSEs_beta:5%: 0.07043125 95%: 0.2412129

We can consider the 5% and 95% quantiles for alpha are: -0.5-2.6

We can consider the 5% and 95% quantiles for beta are: 4-19

Importance sampling

(c)

Computing the log ratios instead of ratios is often numerically more stable, especially when dealing with small probabilities. In many cases, likelihoods or probabilities can be extremely small, and their direct multiplication might cause values rounding to zero, leading to loss of information. Taking logs and working in the log space can mitigate these numerical issues. After computing in the log space, we can exponentiate the result if we need the actual probabilities or ratios.

```
# 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.
    #c(-8.95, -23.47, -6.02, -8.13, -16.61, -14.57)
    x <- bioassay$x
    y <- bioassay$y
    n <- bioassay$n
    # Compute the logarithm of the likelihood using bioassaylp
    log_likelihood <- bioassaylp(alpha, beta, x=x, y=y, n=n)

    return(log_likelihood)
}
print(round(log_importance_weights(alpha_test,beta_test),2))
```

```
## [1]  -8.95 -23.47  -6.02  -8.13 -16.61 -14.57
```

(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.
    #c(0.045, 0.000, 0.852, 0.103, 0.000, 0.000)
    # First, compute the unnormalized log importance weights
    log_weights = log_importance_weights(alpha, beta)
    weights = exp(log_weights)
    # Normalize the weights to sum to one
    normalized_weights = weights / sum(weights)
    # Round to three decimal places
```

```
    normalized_weights_rounded = round(normalized_weights, 5)

    return(normalized_weights_rounded)
}
print(round(normalized_importance_weights(alpha_test,beta_test),3))
```

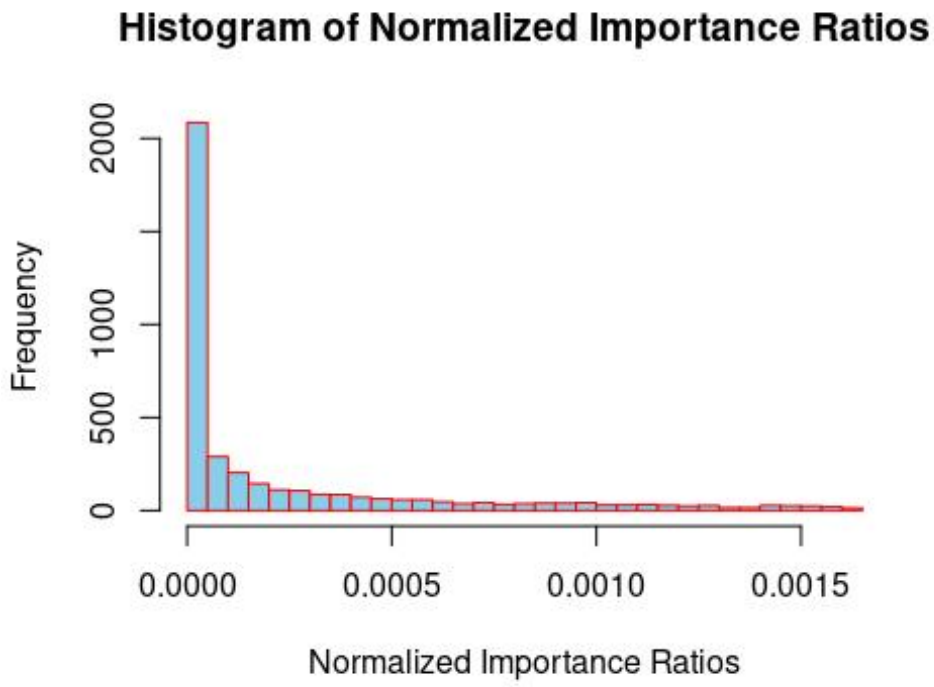
```
## [1] 0.045 0.000 0.852 0.103 0.000 0.000
```

```
#normalized_importance_weights(bioassay_posterior$alpha,bioassay_poster
ior$beta)
```

(e)

$$\text{mean\_vector} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

```
# Set up parameters
n_draws <- 4000

# Mean vector
mean_vector <- c(0, 10)

# Covariance matrix
var_alpha <- 2^2
var_beta <- 10^2
cov_alpha_beta <- 0.6 * sqrt(var_alpha * var_beta)
cov_matrix <- matrix(c(var_alpha, cov_alpha_beta, cov_alpha_beta, var_b
eta), 2, 2)

# Sample from the prior distributions using rmvnorm
samples <- rmvnorm(n_draws, mean_vector, cov_matrix)
alpha_samples <- samples[, 1]
beta_samples <- samples[, 2]

# Compute normalized importance ratios for the samples
normalized_weights <- normalized_importance_weights(alpha_samples, beta
_samples)

# Plot a histogram of the normalized importance ratios
hist(normalized_weights, main="Histogram of Normalized Importance Ratio
s",
     xlab="Normalized Importance Ratios", col="skyblue", border="red",
breaks=50)
```



(f)

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

```
S_eff <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    #1.354
    # Compute the logarithm of the importance weights
    log_weights <- log_importance_weights(alpha, beta)

    # Exponentiate the log weights to get the unnormalized importance w
eights
    weights = exp(log_weights)

    # Normalize the weights so they sum to one
    normalized_weights = weights / sum(weights)

    # Compute the effective sample size using the formula
    return(1 / sum(normalized_weights^2))
}
S_eff(alpha_samples,beta_samples)
```

```
## [1] 1157.702
```

```r
cat("We need about",round(S_eff(alpha_samples,beta_samples),0),"samples
 to be effective.","\n")

## We need about 1158 samples to be effective.
```

(g)

In my opinion, if all samples have the same weight, the effective sample size will be close to the actual sample size. However, if some samples have significantly greater weights than others, the effective sample size will decrease, as this means that only a small number of samples play a dominant role in the calculation, while the majority of samples provide little information. If the proposal distribution is a good match for the target distribution, the variance of the importance weights will be small. This means that the estimation will be more stable and less sensitive to the randomness of the samples drawn. Consequently, S_eff will be higher. From the histogram plotted in e, it can be seen that most of the weights are concentrated at the lower end of the range. This indicates that the importance ratio of most samples is low, which means that their contribution to the overall estimation is relatively small.

(h)

```r
posterior_mean <- function(alpha, beta) {
    # Do computation here, and return as below.
    # This is the correct return value for the test data provided above.
    #c(0.503, 8.275)
    log_weights <- log_importance_weights(alpha, beta)

    # Convert to actual weights
    weights = exp(log_weights)

    # Normalize the weights
    normalized_weights = weights / sum(weights)
    # Compute the means
    alpha_mean = sum(normalized_weights * alpha)
    beta_mean = sum(normalized_weights * beta)
    E_square_alpha_est = sum(normalized_weights * alpha * alpha)
    E_square_beta_est = sum(normalized_weights * beta * beta)
    V_alpha_E = E_square_alpha_est - alpha_mean^2
    V_beta_E = E_square_beta_est - beta_mean^2
    MCSE_alpha <- sqrt(V_alpha_E / 4000)
    MCSE_beta <- sqrt(V_beta_E /4000)
    cat("The MCSE for alpha:",MCSE_alpha,"\n")
    cat("The MCSE for beta:",MCSE_beta,"\n")
    return(c(alpha_mean, beta_mean))
}

result<-posterior_mean(alpha_samples,beta_samples)
```

```
## The MCSE for alpha: 0.01412631
## The MCSE for beta: 0.07336901
```

```
cat("alpha_mean:",round(result[1],3),"\n","beta_mean:",round(result[2],
3),"\n")
```

```
## alpha_mean: 0.96
##  beta_mean: 10.522
```