BDA - Assignment 4

Anonymous

Contents

```r
# To install aaltobda, see the General information in the assignment.
library(aaltobda)
library(markmyassignment)
library("ggplot2")
assignment_path <-
  paste("https://github.com/avehtari/BDA_course_Aalto/", "blob/master/assignments/tests/assignment4.yml
set_assignment(assignment_path)
```

```
## Assignment set:
## assignment4: Bayesian Data Analysis: Assignment 4
## The assignment contain the following (4) tasks:
## - log_importance_weights
## - normalized_importance_weights
## - S_eff
## - posterior_mean
```

Bioassay model

a)

Marginal distributions are of form

$$\alpha = N(\mu_\alpha, \sigma_\alpha^2) = N(0, 2^2)$$

and

$$\beta = N(\mu_\beta, \sigma_\beta^2) = N(10, 10^2)$$

The covariance between α and β is $\rho(\alpha, \beta) = 0.6$.

The mean vector is of form

$$\mu = \begin{bmatrix} \mu_\alpha \\ \mu_\beta \end{bmatrix} = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$$

The covariance matrix is of form

$$cov(\alpha, \beta) = \begin{bmatrix} \sigma_\alpha^2 & \rho\sigma_\alpha\sigma_\beta \\ \rho\sigma_\alpha\sigma_\beta & \sigma_\beta^2 \end{bmatrix} = \begin{bmatrix} 4 & 12 \\ 12 & 100 \end{bmatrix}$$

b)

MONTE CARLO STANDARD ERROR (MCSE): Monte Carlo Standard Error (MCSE) is an estimate of the inaccuracy of Monte Carlo samples, usually regarding the expectation of posterior samples, $E(\theta)$, from Monte Carlo or Markov chain Monte Carlo (MCMC) algorithms, such as with the LaplacesDemon or LaplacesDemon. (source: http://finzi.psych.upenn.edu/library/LaplacesDemon/html/MCSE.html)

Load the data and calculate the means, the required quantiles and the MCSE's. From the mean we show only the digits that are zero in MSCE of the mean and leave out the nonzero digits e.g. if MCSE is 0.0055434, then the first 2 decimals of mean are shown.

```
data("bioassay_posterior")

alpha = bioassay_posterior$alpha
beta = bioassay_posterior$beta
# number of draws
S = 4000
```

MSCE for the mean of alpha:

```
sqrt( var(alpha) / S )
```

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

So for the mean of alpha, we show only the first decimal:

```
format(round(mean(alpha), 1), nsmall = 1)
```

```
## [1] "1.0"
```

MSCE for the 5% quantile of alpha:

```
mcse_quantile(alpha, 0.05)
```

```
##         mcse
## 1 0.02600412
```

And MSCE for the 95% quantile of alpha:

```
mcse_quantile(alpha, 0.95)
```

```
##         mcse
## 1 0.04206342
```

Quantiles for alpha reported in a similar manner based on non-zero digits of the calculated MCSE's:

```
format(round(quantile(alpha, c(0.05, 0.95)), 1), nsmall = 1)
```

```
##     5%    95%
## "-0.5" " 2.6"
```

. . . and the same stuff for beta values. . .

MSCE for the mean of beta:

```
sqrt( var(beta) / S )
```

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

So for the mean of beta, we show only the first decimal:

```
format(round(mean(beta), 1), nsmall = 1)
```

```
## [1] "10.6"
```

MSCE for the 5% quantile of beta:

```
mcse_quantile(beta, 0.05)
```

```
##         mcse
## 1 0.07043125
```

And MSCE for the 95% quantile of beta:

```
mcse_quantile(beta, 0.95)
```

```
##        mcse
## 1 0.2412129
```

Quantiles for beta reported in a similar manner based on non-zero digits of the calculated MCSE's:

```
format(round(quantile(beta, c(0.05)), 1), nsmall = 1)
```

```
##    5%
## "4.0"
```

```
format(round(quantile(beta, c(0.95)), 0), nsmall = 0)
```

```
##  95%
## "19"
```

Importance sampling

c)

It is better to compute log ratios instead of ratios, in order to avoid computational underflows and overflows. This means that if the odd ratio is really small or large, it might be difficult to represent those.

```
data("bioassay")

log_importance_weights <- function(alpha, beta, x, y, n){
  bioassaylp(alpha, beta, bioassay$x, bioassay$y, bioassay$n)
}
```

d)

The effect of exponentiating and scaling so that sum is one is that we immediately get the values as probabilities.

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

 log_ratios <- bioassaylp(alpha, beta, bioassay$x, bioassay$y, bioassay$n)
 normalized <- exp(log_ratios)/(sum(exp(log_ratios)))
 return(normalized)

}
```

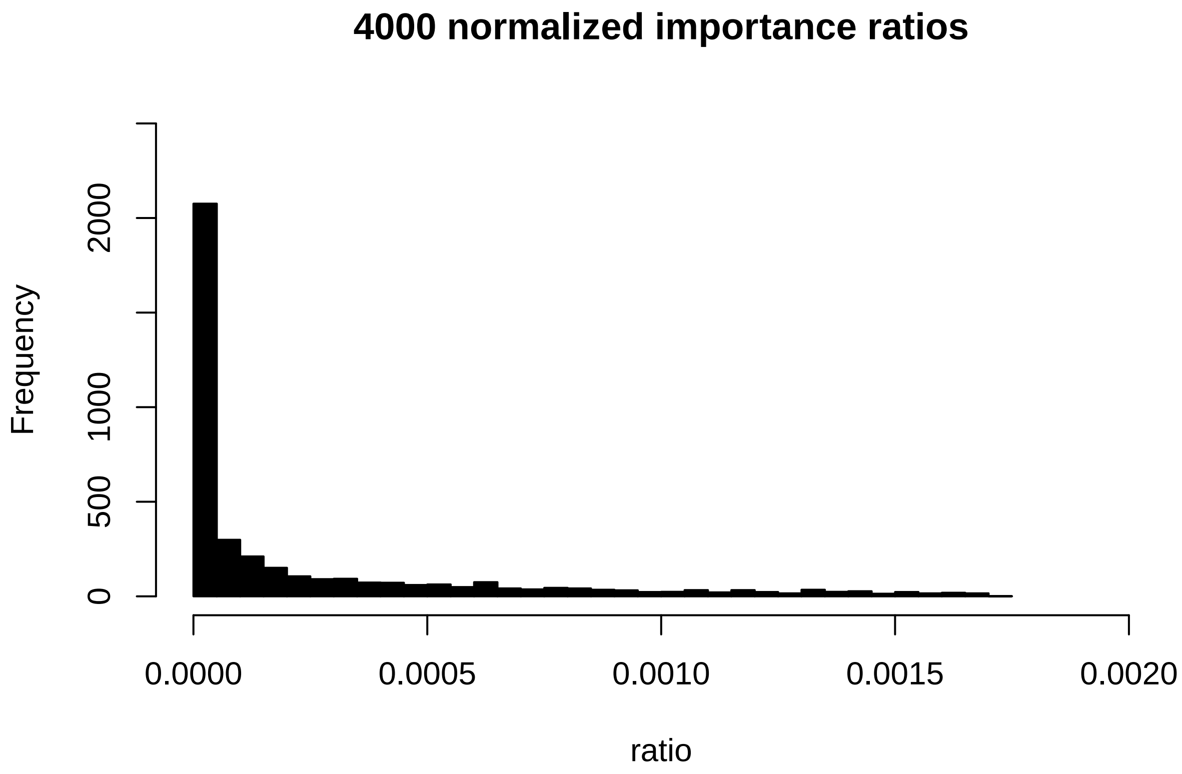
e)

```
mu = c(0, 10)
sigma_squared =  matrix(c(4, 12, 12, 100), 2, 2)
S = 4000

samples = rmvnorm(S, mu, sigma_squared)
alpha_sample = samples[,1]
beta_sample = samples[,2]

ratios = normalized_importance_weights(alpha_sample, beta_sample)

hist(ratios,
     breaks=50,
     main="4000 normalized importance ratios",
     xlab="ratio", xlim=c(0, 0.002),
     ylim = c(0,2500),
     col="black",
     freq=TRUE)
```



f)

Importance sampling effective sampling size S_{eff}:

```
S_eff <- function(alpha, beta){
 w = normalized_importance_weights(alpha, beta)
 return ( 1 / (sum(w^2)) )
}

S_eff(alpha_sample, beta_sample)
```

```
## [1] 1134.806
```

g)

BDA3, p.266: "If the variance of the weights is finite, the effective sample size can be estimated using an approximation…" "The effective sample size S_{eff} is small if there are few extremely high weights which would unduly influence the distribution. If the distribution has occasional very large weights, however, this estimate is itself noisy; it can thus be taken as no more than a rough guide."

So, by computing importance sampling effective sample size, we can analyze the accuracy and efficiency of the importance sampling estimates.

The S_{eff} calculated in part f) was high enough in order that we can say that the estimates have pretty low standard error and estimates produced are pretty stable. (BDA3, p. 594)

In the histogram the high effective sampling size can be seen in a way that there are no extremely high weights influencing the distribution.

h)

```
posterior_mean <- function(alpha, beta){
 weights = normalized_importance_weights(alpha, beta)
 c(sum(alpha*weights), sum(beta*weights))
}

means = posterior_mean(alpha_sample, beta_sample)
mean_alpha = means[1]
mean_beta = means[2]

# variance for the importance sampling
variance <- function(alpha, beta){
  w = normalized_importance_weights(alpha, beta)

  # E(x)^2
  Ealpha_squared = sum(alpha*w)^2 / sum(w)
  Ebeta_squared = sum(beta*w)^2 / sum(w)

  # E(x^2)
  E_alphasquared = sum(alpha^2*w) / sum(w)
  E_betasquared = sum(beta^2*w) / sum(w)

  # var(x) = E(x^2) - E(x)^2
  c((E_alphasquared - Ealpha_squared), (E_betasquared - Ebeta_squared))
}

# posterior mean MCSE
```

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

  v = variance(alpha, beta)
  var_alpha = v[1]
  var_beta = v[2]

  mcse_alpha = sqrt( var_alpha / S_eff(alpha, beta) )
  mcse_beta = sqrt( var_beta / S_eff(alpha, beta) )
  c(mcse_alpha, mcse_beta)
}
```

MCSE for the mean estimate of alpha:

```
posterior_mean_MCSE(alpha_sample, beta_sample)[1]
```

```
## [1] 0.02703478
```

So for the mean estimate of alpha, we show only the first decimal:

```
format(round(mean_alpha, 1), nsmall = 1)
```

```
## [1] "1.0"
```

. . . and the same procedure for beta. . .

MCSE for the mean estimate of beta:

```
posterior_mean_MCSE(alpha_sample, beta_sample)[2]
```

```
## [1] 0.1394222
```

So for the mean estimate of beta, we don't show any decimals:

```
format(round(mean_beta, 0), nsmall = 0)
```

```
## [1] "11"
```

Importance sampling is a method that can be used to compute expectations or other properties of a distribution by drawing random samples from an approximation of the target distribution. So we can still estimate parameters of intrest even if we cannot sample from the exact same distribution for which we try to estimate the parameters.

Finally, check that everything is correct:

```
mark_my_assignment()
```

```
## v |  OK F W S | Context
##
/ |   0       | log_importance_weights()
v |   5       | log_importance_weights()
##
/ |   0       | normalized_importance_weights()
v |   5       | normalized_importance_weights()
##
/ |   0       | S_eff()
v |   5       | S_eff()
##
/ |   0       | posterior_mean()
v |   5       | posterior_mean()
##
```

```
## == Results =========================================================================================
## OK:       20
## Failed:   0
## Warnings: 0
## Skipped:  0
## You're a coding rockstar!
```