BDA - Assignment 4

Anonymous

Contents

```r
library(markmyassignment)
exercise_path = 'https://github.com/avehtari/BDA_course_Aalto/blob/master/exercises/tests/ex4.yml'
set_assignment(exercise_path)
```

```
## Assignment set:
## ex4: Bayesian Data Analysis: Assignment 4
## The assignment contain the following (6) tasks:
## - p_prior
## - p_posterior
## - log_importance_weights
## - normalized_importance_weights
## - S_eff
## - posterior_mean
```

Exercise 1a

```r
library(MASS)
library(mvtnorm)
```

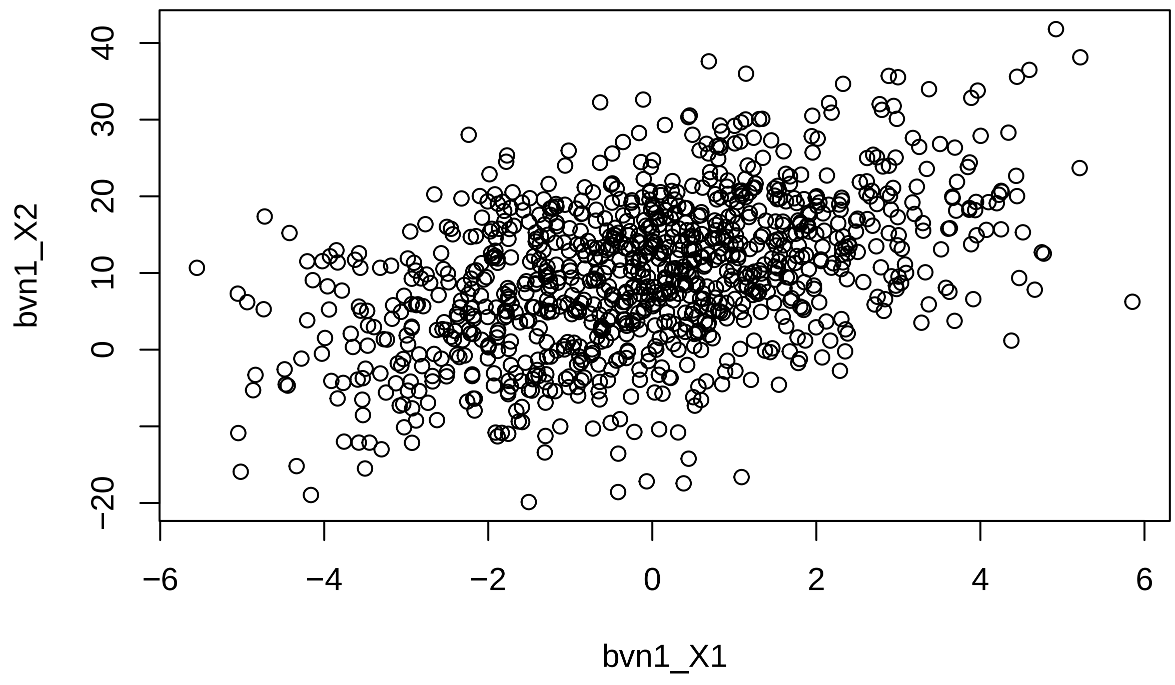
```
rho = 0.5
mu1 = 0; sd1 = 2
mu2 = 10; sd2 = 10

# Parameters for bivariate normal distribution
mu = c(mu1,mu2) # Mu
cat('Mean of the bivariate normal distribution is:',mu[1],',', mu[2], '\n')
```

```
## Mean of the bivariate normal distribution is: 0 , 10
```

```
cov = matrix(c(sd1^2,sd1*sd2*rho, sd1*sd2*rho, sd2^2),2)
cat('Covariance matrix of the bivariate normal distribution is:\n', '[',cov[1],cov[1,2],'\n',cov[2,1],co
```

```
## Covariance matrix of the bivariate normal distribution is:
##  [ 4 10
##  10 100 ]
```

```
bvnd = mvrnorm(1000, mu, cov)
colnames(bvnd) = c("bvn1_X1","bvn1_X2")
plot(bvnd)
```



Exercise 1b

```r
p_log_prior = function(alpha, beta){
  rho = 0.5
  mu1 = 0; sd1 = 2
  mu2 = 10; sd2 = 10

  # Parameters for bivariate normal distribution
  mu = c(mu1,mu2) # Mu
  cov = matrix(c(sd1^2,sd1*sd2*rho, sd1*sd2*rho, sd2^2),2) # Covariance matrix
  log_prior = dmvnorm(x = c(alpha,beta), mean = mu, sigma = cov, log = TRUE)
  cat('The logarithm of the density of the prior distribution in a) \n
      for arbitrary values of alpha and beta is:',log_prior)
}

p_log_prior(3,9)
```

```
## The logarithm of the density of the prior distribution in a)
##
##       for arbitrary values of alpha and beta is: -6.296435
```

Exercise 1c

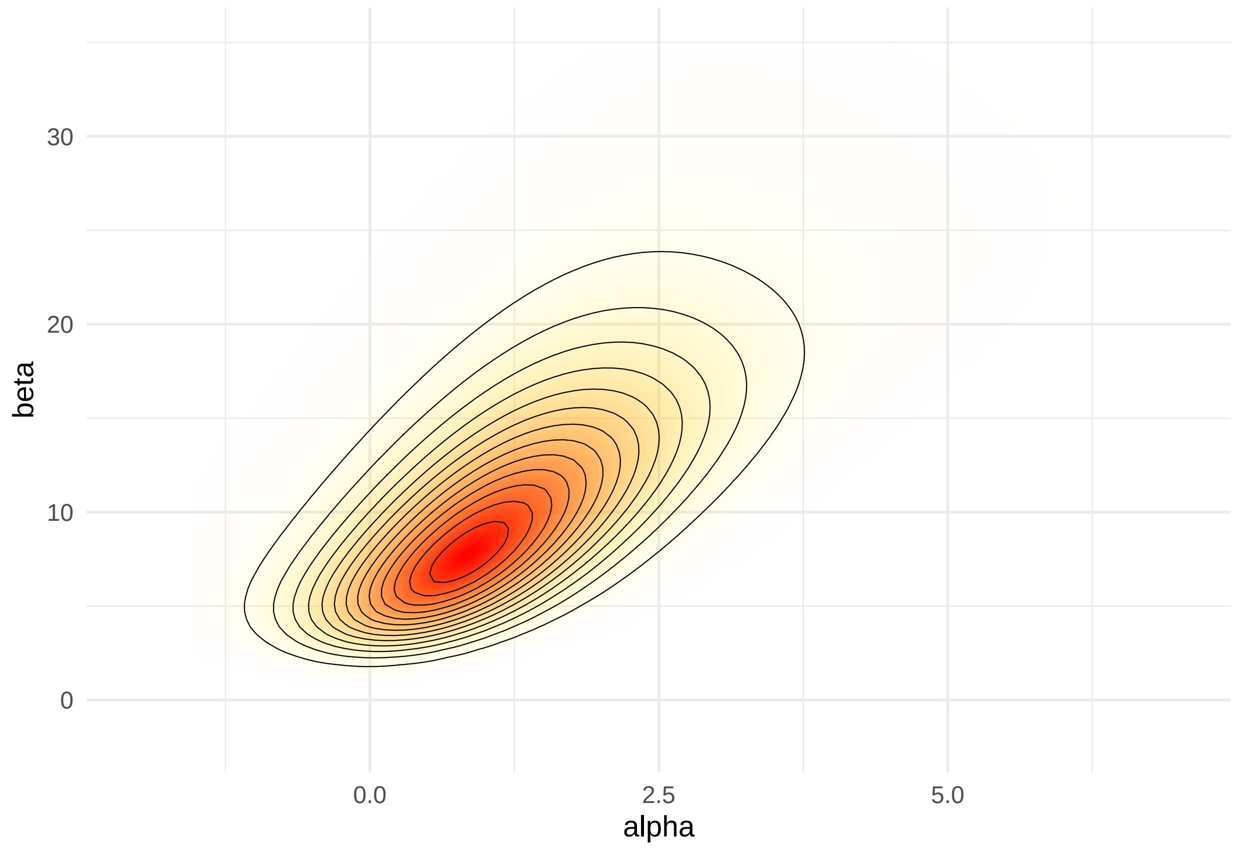
Logarithm of the product of two densities is the sum of the log-densities, i.e. log ab = log a + log b. Hence, by adopting the logarithmic prior from part 1b, we can calculate the logarithmic posterior by: logarithmic likelihood + (-6.296435).

```r
library(aaltobda)
data('bioassay')

p_log_posterior = function(alpha, beta, x = bioassay$x, y = bioassay$y, n = bioassay$n){
  log_likelihood = bioassaylp(alpha, beta, x = bioassay$x, y = bioassay$y, n = bioassay$n)
  log_posterior =  log_likelihood + -6.296435
  cat('The logarithm of the density of the posterior distribution is,',log_posterior)
}

p_log_posterior(3,9,x = bioassay$x, y = bioassay$y, n = bioassay$n)
```

```
## The logarithm of the density of the posterior distribution is, -15.78798
```

Exercise 1d

```r
library(ggplot2)
library(dplyr)
theme_set(theme_minimal())

df1 <- data.frame(
  x = c(-0.86, -0.30, -0.05, 0.73),
  n = c(5, 5, 5, 5),
  y = c(0, 1, 3, 5)
)

A = seq(-1.5, 7, length.out = 100)
B = seq(-5, 35, length.out = 100)

cA = rep(A, each = length(B))
cB = rep(B, length(A))

logl = function(df, a, b)
  df['y']*(a + b*df['x']) - df['n']*log1p(exp(a + b*df['x']))

p = apply(df1, 1, logl, cA, cB) %>%
  rowSums() %>% exp()

xl = c(-2, 7)
yl = c(-2, 35)
pos = ggplot(data = data.frame(cA ,cB, p), aes(x = cA, y = cB)) +
  geom_raster(aes(fill = p, alpha = p), interpolate = T) +
  geom_contour(aes(z = p), colour = 'black', size = 0.2) +
  coord_cartesian(xlim = xl, ylim = yl) +
  labs(x = 'alpha', y = 'beta') +
  scale_fill_gradient(low = 'yellow', high = 'red', guide = F) +
  scale_alpha(range = c(0, 1), guide = F)
pos
```



Exercise 1e

```
log_importance_weights = function(alpha, beta){

  logs = list()
  for(i in 1:length(alpha)){
    log_likelihood = bioassaylp(alpha[i], beta[i], x = bioassay$x, y = bioassay$y, n = bioassay$n)
    logs = c(logs,log_likelihood)
  }
  logs = as.vector(unlist(logs))
  cat('The log importance weights are:')
  return(round(logs,3))
}

log_importance_weights(c(1.896, -3.6,  0.374, 0.964, -3.123, -1.581), c(24.76, 20.04, 6.15, 18.65, 8.16
```

```
## The log importance weights are:

## [1]  -8.954 -23.468  -6.015  -8.130 -16.613 -14.573
```

```
normalized_importance_weights = function(alpha, beta){
  logs = list()
  for(i in 1:length(alpha)){
    log_likelihood = bioassaylp(alpha[i], beta[i], x = bioassay$x, y = bioassay$y, n = bioassay$n)
    logs = c(logs,log_likelihood)
  }

  logs = as.vector(unlist(logs))
  p = exp(logs-max(logs))
  p = round(p/sum(p),3)

  cat('The normalized importance weights are:')
  return(p)
}

normalized_importance_weights(c(1.896, -3.6,  0.374, 0.964, -3.123, -1.581),
                              c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4))
```

```
## The normalized importance weights are:

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

Exercise 1f

Below the **test** mean values for given alpha vector c(1.896, -3.6, 0.374, 0.964, -3.123, -1.581) and beta vector c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4)

```
posterior_mean = function(alpha, beta){

  logs = list()
  for(i in 1:length(alpha)){
    log_likelihood = bioassaylp(alpha[i], beta[i], x = bioassay$x, y = bioassay$y, n = bioassay$n)
    logs = c(logs,log_likelihood)
  }

  logs = as.vector(unlist(logs))
  p = exp(logs-max(logs))
  p = p/sum(p)

  S = length(alpha)
  alpha_mean = ((1/S)*sum(p * alpha)) / ((1/S)*sum(p))
  beta_mean = ((1/S)*sum(p * beta)) / ((1/S)*sum(p))
  cat('Posterior means for alpha and beta are:')
  return(c(alpha_mean, beta_mean))
}

posterior_mean(c(1.896, -3.6,  0.374, 0.964, -3.123, -1.581),
               c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4))
```

```
## Posterior means for alpha and beta are:

## [1] 0.5028453 8.2753949
```

Below the mean values for **sampled** alphas and betas. Sample size = 5000.

```
rho = 0.5
mu1 = 0; sd1 = 2
mu2 = 10; sd2 = 10
mu = c(mu1, mu2)
cov = matrix(c(sd1^2,sd1*sd2*rho, sd1*sd2*rho, sd2^2),2)
prior_draw =rmvnorm(5000, mean=mu, sigma=cov)

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

```
## Posterior means for alpha and beta are:

## [1]  0.9440291 10.5200924
```

Exercise 1g

Below the **test** effective sample size for given alpha vector c(1.896, -3.6, 0.374, 0.964, -3.123, -1.581) and beta vector c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4)

```
S_eff = function(alpha, beta){

  logs = list()
  for(i in 1:length(alpha)){
    log_likelihood = bioassaylp(alpha[i], beta[i], x = bioassay$x, y = bioassay$y, n = bioassay$n)
    logs = c(logs,log_likelihood)
  }
  logs = as.vector(unlist(logs))
  p = exp(logs-max(logs))
  p = p/sum(p)
  p_tilde = p/sum(p)
  ess = round(1/sum(p_tilde^2),3)
  cat('The effective sample size is:', ess)

}
S_eff(c(1.896, -3.6,  0.374, 0.964, -3.123, -1.581),
      c(24.76, 20.04, 6.15, 18.65, 8.16, 17.4))
```

```
## The effective sample size is: 1.354
```

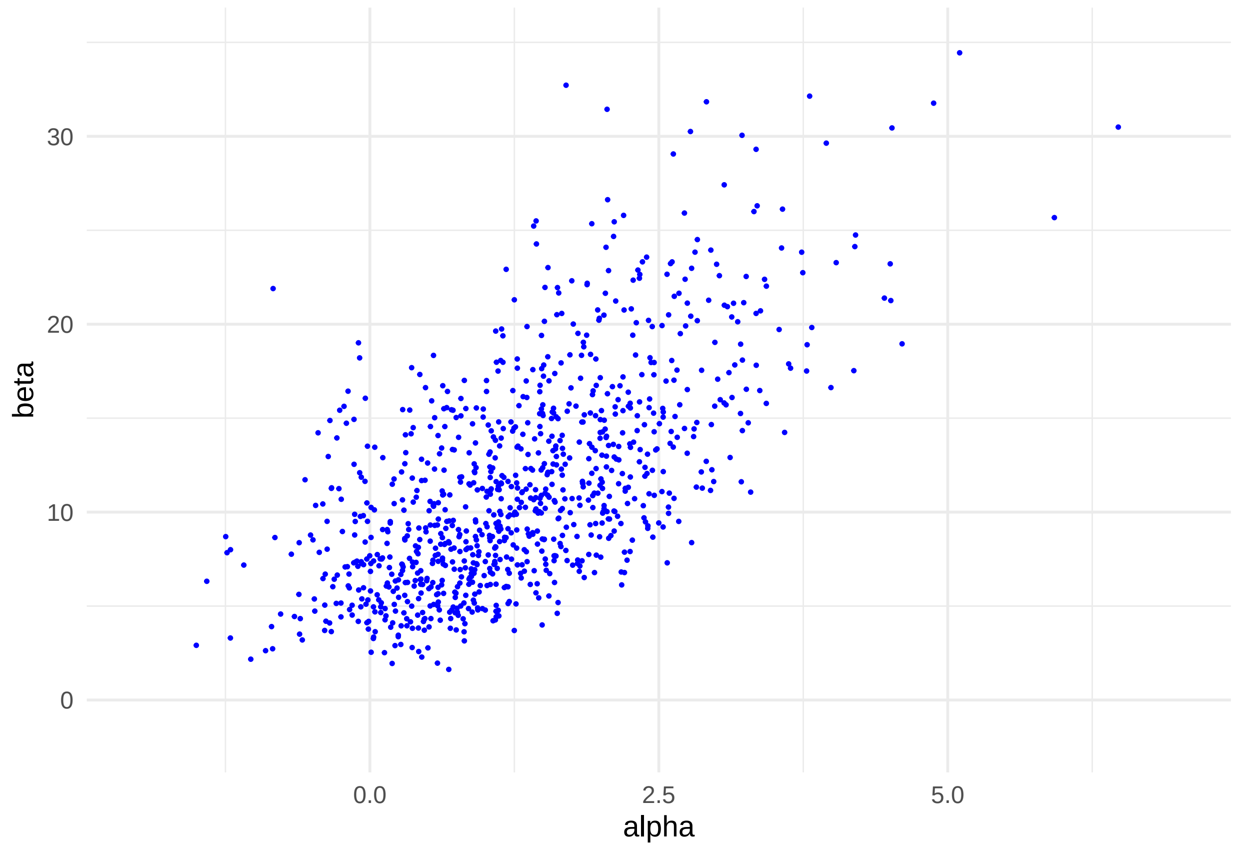
Below the effective sample size with **sampled** alphas and betas. Sample size = 5000.

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

```
## The effective sample size is: 1334.098
```

Exercise 1h

```r
library(dplyr)
library(ggplot2)
theme_set(theme_minimal())

df1 <- data.frame(
  x = c(-0.86, -0.30, -0.05, 0.73),
  n = c(5, 5, 5, 5),
  y = c(0, 1, 3, 5))

A = seq(-1.5, 7, length.out = 100)
B = seq(-5, 35, length.out = 100)
cA = rep(A, each = length(B))
cB = rep(B, length(A))

logl = function(df, a, b)
  df['y']*(a + b*df['x']) - df['n']*log1p(exp(a + b*df['x']))

p = apply(df1, 1, logl, cA, cB) %>%
  rowSums() %>% exp()

nsamp = 1000
samp_indices <- sample(length(p), size = nsamp, replace = T, prob = p/sum(p))
samp_A = cA[samp_indices[1:nsamp]]
samp_B = cB[samp_indices[1:nsamp]]

samp_A = samp_A + runif(nsamp, (A[1] - A[2])/2, (A[2] - A[1])/2)
samp_B = samp_B + runif(nsamp, (B[1] - B[2])/2, (B[2] - B[1])/2)

sam = ggplot(data = data.frame(samp_A, samp_B)) +
  geom_point(aes(samp_A, samp_B), color = 'blue', size = 0.3) +
  coord_cartesian(xlim = xl, ylim = yl) +
  labs(x = 'alpha', y = 'beta')
sam
```



Exercise 1i

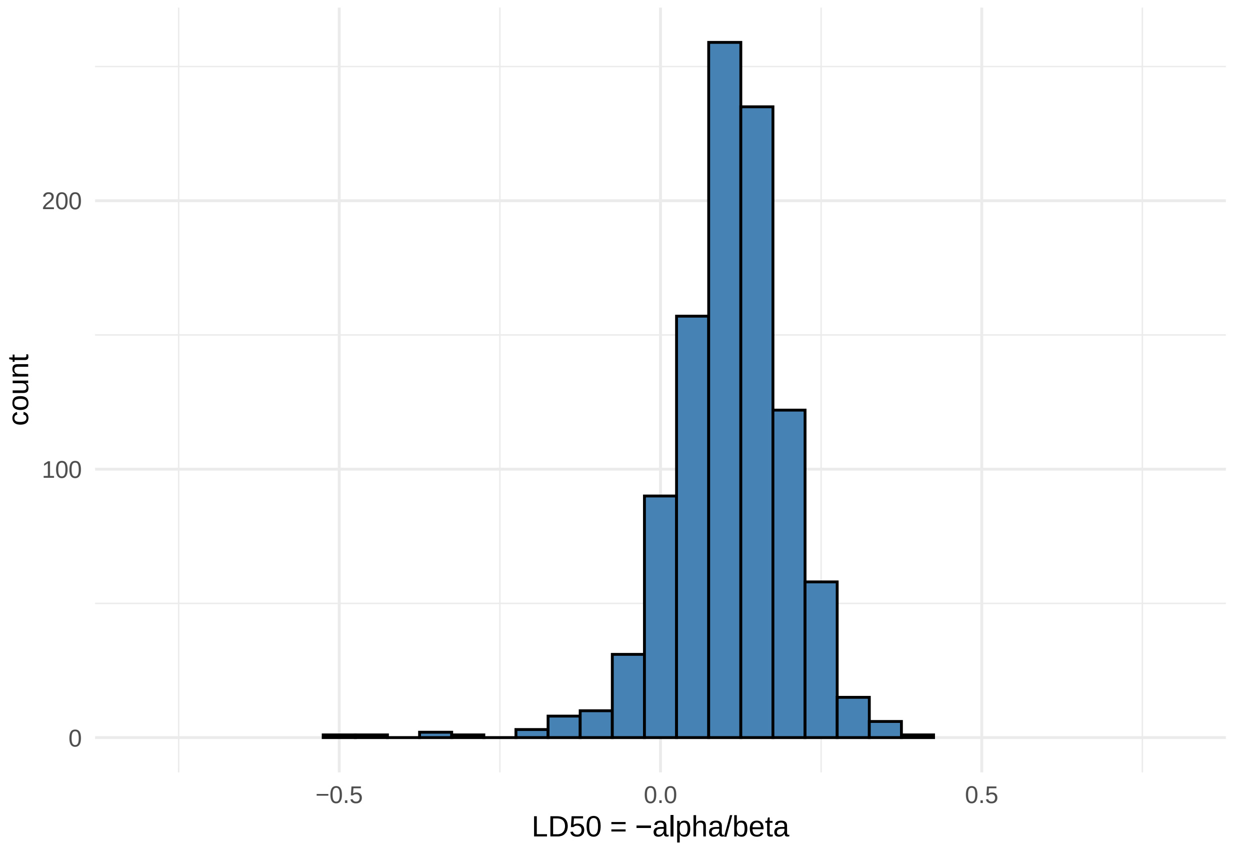
Using the posterior sample obtained via importance resampling, report an estimate for p(beta>0|x,n,y), that is, the probability that the drug is harmful

```
bpi = samp_B > 0
prob = length(bpi)/length(samp_B)
cat('The probability that the drug is harmful is:', prob)
```

```
## The probability that the drug is harmful is: 1
```

Exercise 1j

```
samp_ld50 = samp_A / samp_B
hist = ggplot() +
   geom_histogram(aes(samp_ld50), binwidth = 0.05,
                  fill = 'steelblue', color = 'black') +
   coord_cartesian(xlim = c(-0.8, 0.8)) +
   labs(x = 'LD50 = -alpha/beta')
hist
```


count
200
100
0
−0.5
0.0
0.5
LD50 = −alpha/beta