BDA - Assignment 5

Anonymous

Contents

Import some useful libraries

```
library(aaltobda)
library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.2, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

Generalized linear model: Bioassay with Metropolis

Exercise 1

In this exercise, we implement the Metropolis algorithm for the bioassay data. We assume a prior distribution characterized by

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \sim \mathrm{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \text{ where } \boldsymbol{\mu}_0 = \begin{bmatrix}0\\ 10\end{bmatrix} \text{ and } \boldsymbol{\Sigma}_0 = \begin{bmatrix}2^2 & 10\\ 10 & 10^2\end{bmatrix}.$$

a)

We can start by implementing a function to compute the density ratio. For computational reasons, we will use the log-densities of both the proposal and previous distributions. The unnormalized posterior will be equal to the sum of the log-likelihood and the log-prior.

```
data("bioassay")

density_ratio <-
  function(alpha_propose,
           alpha_previous,
           beta_propose,
           beta_previous,
           x,
           y,
           n)
  {
    # multivariate normal prior distribution parameters
    mu_prior <- c(0, 10)
    sigma_prior <-
      matrix(data = c(4, 10, 10, 100),
             nrow = 2,
             ncol = 2)

    # Compute log-posterior of proposal distribution
    log_likelihood_proposal <-
      bioassaylp(alpha_propose, beta_propose, x, y, n)
    log_prior_proposal <- dmvnorm(c(alpha_propose, beta_propose),
                                  mu_prior, sigma_prior, log = TRUE)
    log_posterior_proposal <-
      log_likelihood_proposal + log_prior_proposal

    # Compute log-posterior of previous distribution
    log_likelihood_previous <-
      bioassaylp(alpha_previous, beta_previous, x, y, n)
    log_prior_previous <- dmvnorm(c(alpha_previous, beta_previous),
                                  mu_prior,
                                  sigma_prior,
                                  log = TRUE)
    log_posterior_previous <-
      log_likelihood_previous + log_prior_previous

    # Compute the ratio of densities
    return (exp(log_posterior_proposal - log_posterior_previous))

  }


density_ratio(
  alpha_propose = 1.89,
  alpha_previous = 0.374,
  beta_propose = 24.76,
  beta_previous = 20.04,
  x = bioassay$x,
  y = bioassay$y,
```

```
  n = bioassay$n
)
```

```
## [1] 1.187524
```

```
density_ratio(
  alpha_propose = 0.374,
  alpha_previous = 1.89,
  beta_propose = 20.04,
  beta_previous = 24.76,
  x = bioassay$x,
  y = bioassay$y,
  n = bioassay$n
)
```

```
## [1] 0.8420882
```

b)

Now, we will use the previous function to help in the implementation of the Metropolis algorithm.

```
Metropolis_bioassay <- function(sigma_alpha, sigma_beta, max_iters) {
  # First, we sample a (normal) proposal distribution as a starting point
  mu <- c(0, 10)
  sigma <- matrix(data = c(4, 10, 10, 100),
                  nrow = 2,
                  ncol = 2)
  theta_0 <- rmvnorm(n = 1, mu, sigma)

  # Set the starting point for alpha and beta
  alpha <- numeric(length = max_iters)
  beta <- numeric(length = max_iters)
  alpha[1] <- theta_0[1, 1]
  beta[1] <- theta_0[1, 2]

  # Run n times
  for (i in 2:max_iters) {
    alpha_previous <- alpha[[i - 1]]
    beta_previous <- beta[[i - 1]]

    # Sample proposal parameters from the proposal distribution
    alpha_propose <-
      rnorm(1, mean = alpha_previous, sd = sigma_alpha)
    beta_propose <- rnorm(1, mean = beta_previous, sd = sigma_beta)

    # Compute the density ratio
    r <- density_ratio(
      alpha_propose,
      alpha_previous,
      beta_propose,
      beta_previous,
      bioassay$x,
      bioassay$y,
      bioassay$n
    )
```

```
    # Set the parameters for this iteration
    if (runif(1) <= min(r, 1)) {
      # Update parameters with proposed parameters
      alpha[i] <- alpha_propose
      beta[i] <- beta_propose
    } else {
      # Update parameters with previous parameters
      alpha[i] <- alpha_previous
      beta[i] <- beta_previous
    }
  }

  # return list of alpha and beta for all iterations
  list(alpha = alpha, beta = beta)
}

# Metropolis_bioassay(sigma_alpha = 1, sigma_beta = 3, max_iters = 3000)
```

Exercise 2

a)

The main result that we hope to obtain in Bayesian statistics is the posterior distribution of our parameter(s) of interest. In simple cases, this posterior distribution can be sampled from (e.g. in the first assignments) using methods such as importance sampling. However, in more complicated cases (e.g. high dimensional distributions), the posterior distribution can not be sampled and we need to draw samples of the parameter(s) θ from approximate distributions and then iteratively improve these estimates to approximate the posterior distribution better.

b)

As a proposal distribution for α^* and β^*, we choose a normal distribution with mean the previous alpha and beta and standard deviation of scale 1 and 5, respectively. That is

$$\alpha^* \sim \mathrm{N}(\alpha_{t-1}, \sigma = 1) \text{ and } \beta^* \sim \mathrm{N}(\beta_{t-1}, \sigma = 5).$$

c)

The initial points for the Metropolis chains are chosen to be drawn from Gaussian distributions such as

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \sim \mathrm{N}(\mu_0, \mathbf{\Sigma}_0), \text{ where } \mu_0 = \begin{bmatrix}0 \\ 10\end{bmatrix} \text{ and } \mathbf{\Sigma}_0 = \begin{bmatrix}2^2 & 10 \\ 10 & 10^2\end{bmatrix}.$$

d)

The number of iterations per chain is set to a maximum of 3000. This is enough to see convergence.

e)

Since we are using a chain length of 3000 and we want to diminish the influence of the starting values, we will discard the first half of the sequence. Thus, out of the total chain length of 3000, we set a warm-up length of 1500.

f)

We generate 10 chains in order to observe the convergence of the chains. Using these many gives an idea of the overall convergence of the algorithm.

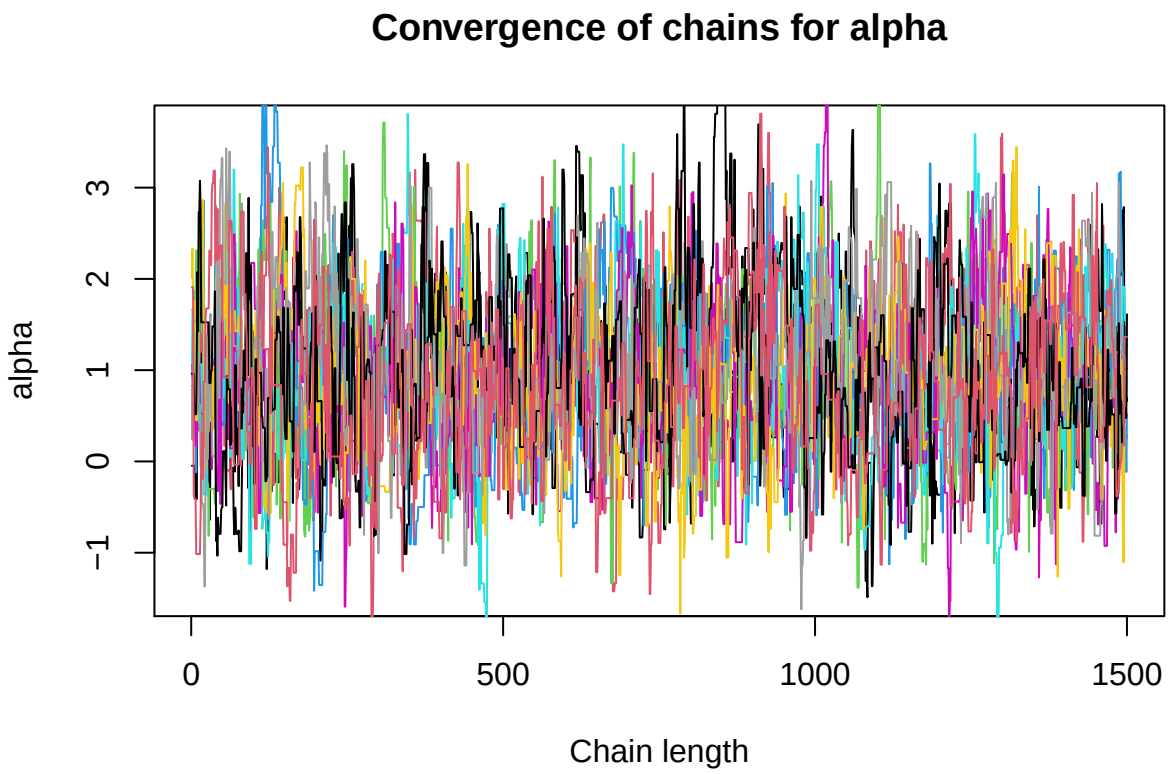
g)

Now, we will plot all 10 chains for α to see how well convergence is achieved.

```
max_iters <- 3000
N_chains <- 10
warm_up_length <- floor(max_iters / 2)

alpha_chains <-
  matrix(nrow = N_chains, ncol = max_iters - warm_up_length)
beta_chains <-
  matrix(nrow = N_chains, ncol = max_iters - warm_up_length)
for (i in 1:N_chains) {
  chains <- Metropolis_bioassay(sigma_alpha = 1,
                                sigma_beta = 5,
                                max_iters = max_iters)
  alpha_chains[i,] <- chains$alpha[(warm_up_length + 1):max_iters]
  beta_chains[i,] <- chains$beta[(warm_up_length + 1):max_iters]
}
```

```
plot(
  alpha_chains[1,],
  type = "l",
  col = 1,
  xlab = "Chain length",
  ylab = "alpha"
)
for (i in 2:N_chains) {
  lines(
    alpha_chains[i,],
    type = "l",
    col = i,
    xlab = "Chain length",
    ylab = "alpha"
  )
}
title("Convergence of chains for alpha")
```

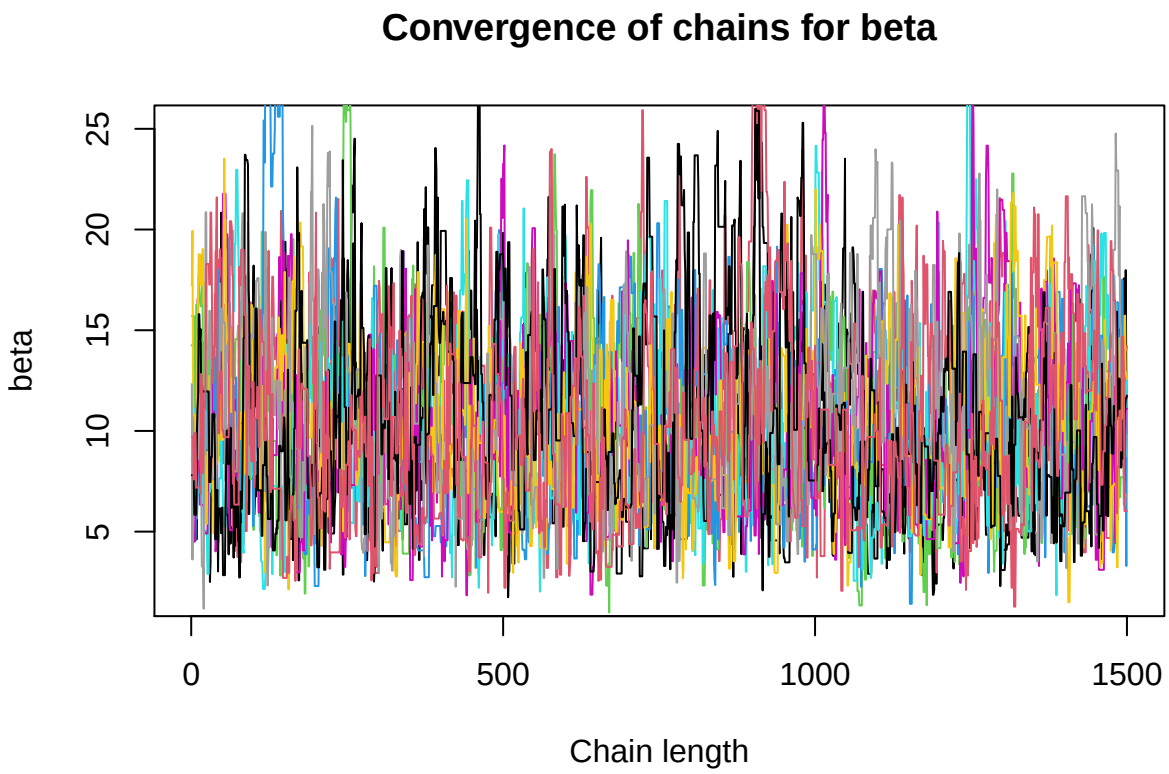


Each chain is represented with a different color. Since we are seeing many chains in the same plot, it is hard to see whether convergence has been achieved. It is sensible to compute the value of $\hat{R}$ to have a more trustworthy knoweledge of the convergence of the chains.

h)

Now, we will plot all 10 chains for β to see how well convergence is achieved.

```
plot(
  beta_chains[1,],
  type = "l",
  col = 1,
  xlab = "Chain length",
  ylab = "beta"
)
for (i in 2:N_chains) {
  lines(
    beta_chains[i,],
    type = "l",
    col = i,
    xlab = "Chain length",
    ylab = "beta"
  )
}
title("Convergence of chains for beta")
```



We can see that the range for β is larger than for α as we use a larger value for the scale parameter. As convergence is hard to see, it is better to wait for the value computed for $\hat{R}$ to see if the chains have converged.

Exercise 3

a)

Looking at the values of α and β for each chain to determine the convergence of the Markov chains can be challenging. Instead, convergence can use the metric $\hat{R}$ which is a more robust indicator of convergence. $\hat{R}$ is an indicator of the between- and within-chain variance of the estimates. If the between- or within-chain estimates are generally not agreeing, we will get a larger value. A larger value thus tells that we should continue with further simulations to improve our inference of the target distribution. If the value of $\hat{R}$ is less than 1.05, it is considered safe to use the sample.

In order to compute $\hat{R}$, we will use the function from the rstan library.

b)

Computing the $\hat{R}$ values for the chains is straightforward once we have removed the warm-up of the chain.

```
Rhat(alpha_chains)
```

```
## [1] 0.9993049
```

```
Rhat(beta_chains)
```

```
## [1] 1.002851
```

As we can see, $\hat{R}_\alpha = 1.009$ and $\hat{R}_\beta = 0.997$. Both results are below 1.05, which means that we can accept the sample as the chains have converged.

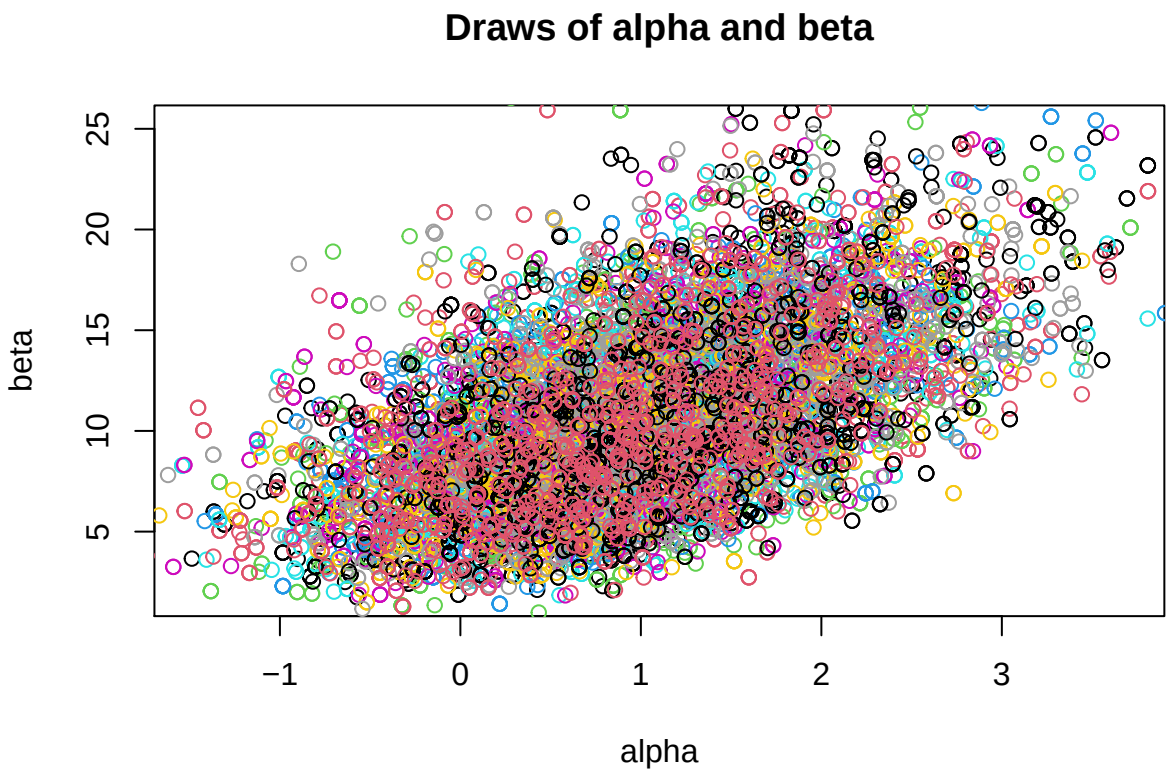
This value of $\hat{R}$ is obtained without changing the parameters or the proposal distribution that was used originally. The proposal distribution is chosen as

$$\alpha^* \sim \mathrm{N}(\alpha_{t-1}, \sigma = 1) \text{ and } \beta^* \sim \mathrm{N}(\beta_{t-1}, \sigma = 5).$$

Exercise 4

Finally, let us compare the draws for α and β to those in Figure 3.3b in BDA3 to see if the results are sensible.

```
plot(alpha_chains[1,], beta_chains[1,], col = 1, xlab = "alpha", ylab = "beta")
for (i in 2:N_chains) {
  points(alpha_chains[i,], beta_chains[i,], col = i, xlab = "alpha", ylab = "beta")
}
title("Draws of alpha and beta")
```



In the scatter plot, we can see the draws for each chain represented each with a different color. Comparing with the Figure in the book, the results are in the same scale and seem to have a similar elliptical shape.