BDA - Assignment 5

Anonymous

Contents

Load packages

```r
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)
```

```r
data("bioassay_posterior")
data("bioassay")
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
bioassay
```

```
##       x n y
## 1 -0.86 5 0
## 2 -0.30 5 1
## 3 -0.05 5 3
## 4  0.73 5 5
```

Exercise 1

```r
# Density ratio function
density_ratio <- function(alpha_propose, alpha_previous, beta_propose,
                          beta_previous, x, y, n){
  p_propose <- bioassaylp(alpha_propose, beta_propose, x, y, n)
  p_previous <- bioassaylp(alpha_previous, beta_previous, x, y, n)

  # Add the prior
  mean <- c(0,10)
  sigma <- matrix(c(4, 10, 10, 100),2)
  x_vector_propose <- c(alpha_propose, beta_propose)
  log_prior_propose <- dmvnorm(x_vector_propose, mean=mean, sigma=sigma, log = TRUE)

  x_vector_previous <- c(alpha_previous, beta_previous)
  log_prior_previous <- dmvnorm(x_vector_previous, mean=mean, sigma=sigma, log = TRUE)

  log_propose <- p_propose +  log_prior_propose
  log_previous <-  p_previous + log_prior_previous

  division <- exp(log_propose - log_previous)
  return(division)
}

print(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
```

```
# Metropolis Algorithm
Metropolis_bioassay <- function(starting_point, iterations){
  markov_chain <- array(dim = c(iterations+1, 2))
  for (i in 1:iterations){
    if (i==1){
      markov_chain[1,] <- starting_point
      alpha_previous <- starting_point[1]
      beta_previous <- starting_point[2]
    }
    else{
      alpha_previous <- markov_chain[i-1,1]
      beta_previous <- markov_chain[i-1,2]
    }

    alpha_propose <- rnorm(1, alpha_previous, 3)
    beta_propose <- rnorm(1, beta_previous, 12)
    ratio <- density_ratio(alpha_propose=alpha_propose, alpha_previous=alpha_previous,
                           beta_propose=beta_propose, beta_previous=beta_previous,
                           x=bioassay$x, y=bioassay$y, n=bioassay$n)

    if (runif(1) < ratio){
      markov_chain[i+1, 1] <- alpha_propose
      markov_chain[i+1, 2] <- beta_propose
      markov_chain
    }
    else{
      markov_chain[i+1, 1] <- markov_chain[i, 1]
      markov_chain[i+1, 2] <- markov_chain[i, 2]
    }

    if (i>1){
      p1 <- c(markov_chain[i-1,1], markov_chain[i-1,2])
      p2 <- c(markov_chain[i,1], markov_chain[i,2])
    }

  }
  return(markov_chain)
}

# Number of iterations
it = 5000
```

```r
starting_point_1 <-c(floor(runif(2, min=1, max=10)))
starting_point_1
```

```
## [1] 3 9
```

```r
markov_chain_1 <- Metropolis_bioassay(starting_point=starting_point_1, iterations=it)
min_x <- min(markov_chain_1[,1])-1
max_x <- max(markov_chain_1[,1])+1
min_y <- min(markov_chain_1[,2])-1
max_y <- max(markov_chain_1[,2])+1
x_lim_1 <- c(min_x, max_x)
y_lim_1 <- c(min_y, max_y)

starting_point_2 <-c(floor(runif(2, min=1, max=10)))
starting_point_2
```

```
## [1] 4 1
```

```r
markov_chain_2 <- Metropolis_bioassay(starting_point=starting_point_2, iterations=it)
min_x <- min(markov_chain_2[,1])-1
max_x <- max(markov_chain_2[,1])+1
min_y <- min(markov_chain_2[,2])-1
max_y <- max(markov_chain_2[,2])+1
x_lim_2 <- c(min_x, max_x)
y_lim_2 <- c(min_y, max_y)


starting_point_3 <-c(floor(runif(2, min=1, max=10)))
starting_point_3
```

```
## [1] 5 3
```

```r
markov_chain_3 <- Metropolis_bioassay(starting_point=starting_point_3, iterations=it)
min_x <- min(markov_chain_3[,1])-1
max_x <- max(markov_chain_3[,1])+1
min_y <- min(markov_chain_3[,2])-1
max_y <- max(markov_chain_3[,2])+1
x_lim_3 <- c(min_x, max_x)
y_lim_3 <- c(min_y, max_y)


starting_point_4 <-c(floor(runif(2, min=1, max=10)))
starting_point_4
```

```
## [1] 5 5
```

```r
markov_chain_4 <- Metropolis_bioassay(starting_point=starting_point_4, iterations=it)
min_x <- min(markov_chain_4[,1])-1
max_x <- max(markov_chain_4[,1])+1
min_y <- min(markov_chain_4[,2])-1
max_y <- max(markov_chain_4[,2])+1
x_lim_4 <- c(min_x, max_x)
y_lim_4 <- c(min_y, max_y)
```

Exercise 2

a)

Metropolis-Hastings allows us to draw samples from a “target distribution”, whose parameters are unknown, by approximating it with other distributions that are modified in each step iteration in order to improve them and get them closer to the target one, or, in other words, to make them converge to the target distribution. It is a Markov Chain Simulation method, used when it is not possible to directly draw from the target distribution. The target distribution is the stationary distribution of the Markov Chain. Each of the samples that are drawn from the proposal distribution is called candidate sample and it can be accepted or rejected depending on a criteria based on the computation of a ratio. Given the target distribution: $p(\theta) \propto g(\theta)$, the pseudocode for the Metropolis-Hastings algorithm proceeds as following:

1. Choose an initial value for θ_0
2. For t=1. . . Iterations:

- Draw a candidate sample from the proposal distribution: $\theta^* \sim q(\theta^*|\theta_{t-1})$
- Compute the ratio of densities, r.
- Depending on the value of the ratio, the candidate θ^* is accepted as θ_t or rejected, so $\theta_t = \theta_{t-1}$:

$$\theta_i = \begin{cases} \theta^* \text{ with probability min(r,1)} \\ \theta_{i-1} \text{ otherwise} \end{cases}$$

b)

The chosen proposals are based on the example ones:

$$\alpha^* \sim N(\alpha_{t-1}, \sigma = 3)$$

$$\beta^* \sim N(\beta_{t-1}, \sigma = 12)$$

In order to choose the values of σ, the resulting plots were studied. In the figures 1 and 2, the four chains for alpha and beta are represented using the chosen proposal distributions. As it is shown, there is a clear convergence towards one point. In other cases, such as the ones in the figures 3 and 4, for which the proposal distributions have $\sigma = 1$ and $\beta = 5$, the points have a higher dispersion and the convergence is not as recognizable as in the previous case.

For higher values of σ, the acceptance rate of the Metropolis algorithm is lower, meaning that there are more rejected candidate values, so the convergence is slower and the performance, in general, is worse. This is the reason why the previous distributions were selected. To summarize, the chosen distributions meet the following requirements, stated in the book BDA3:

- It is easy to sample from the proposal distribution.
- It is easy to compute the ratio.
- Each jump goes a reasonable distance in the parameter space.
- The jumps, as explained before, are not rejected too frequently.

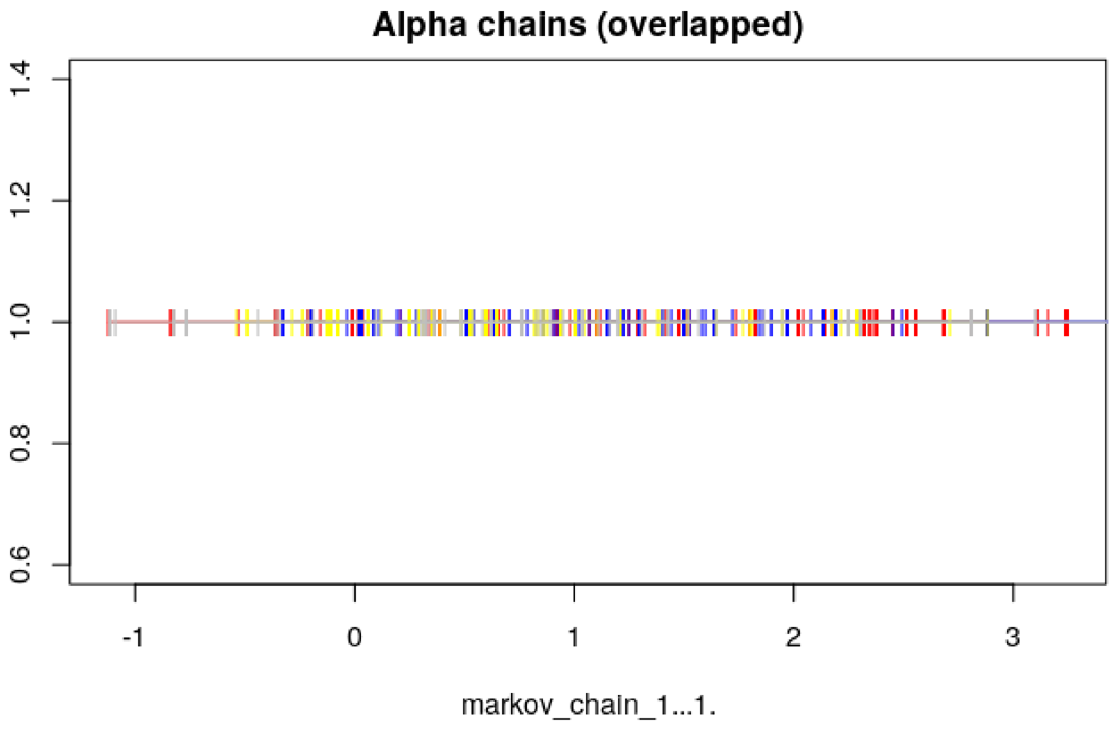

Figure 1: Alpha chains for the chosen proposal distributions.

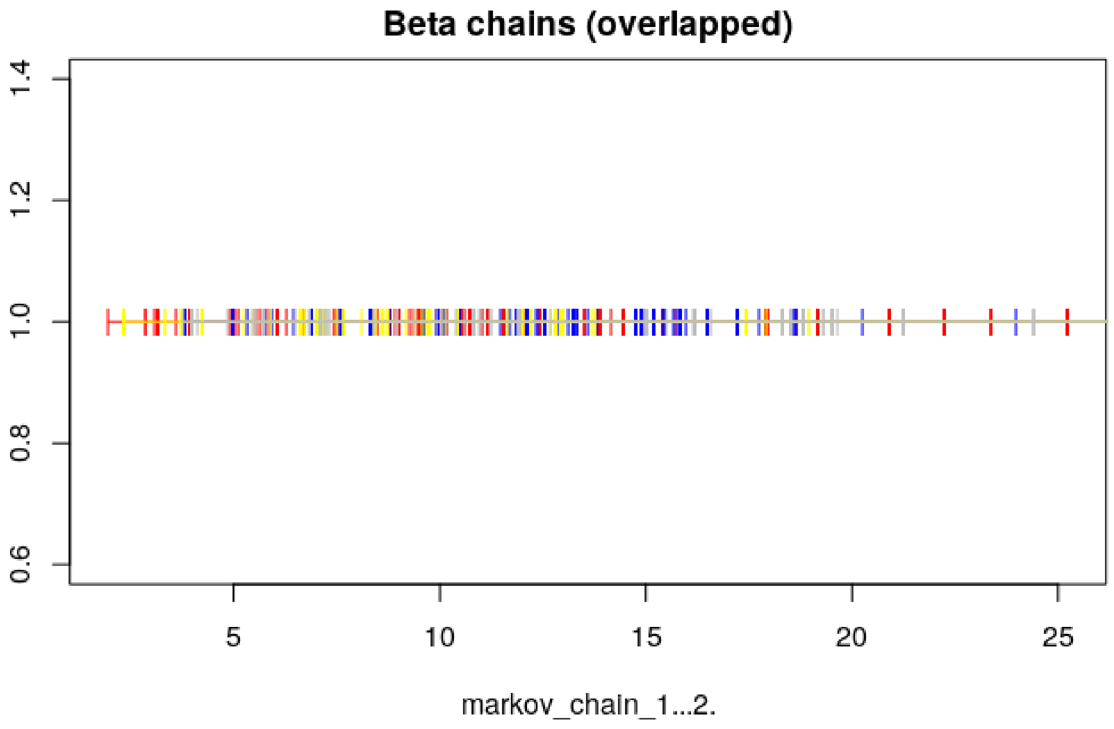

Figure 2: Beta chains for the chosen proposal distributions.

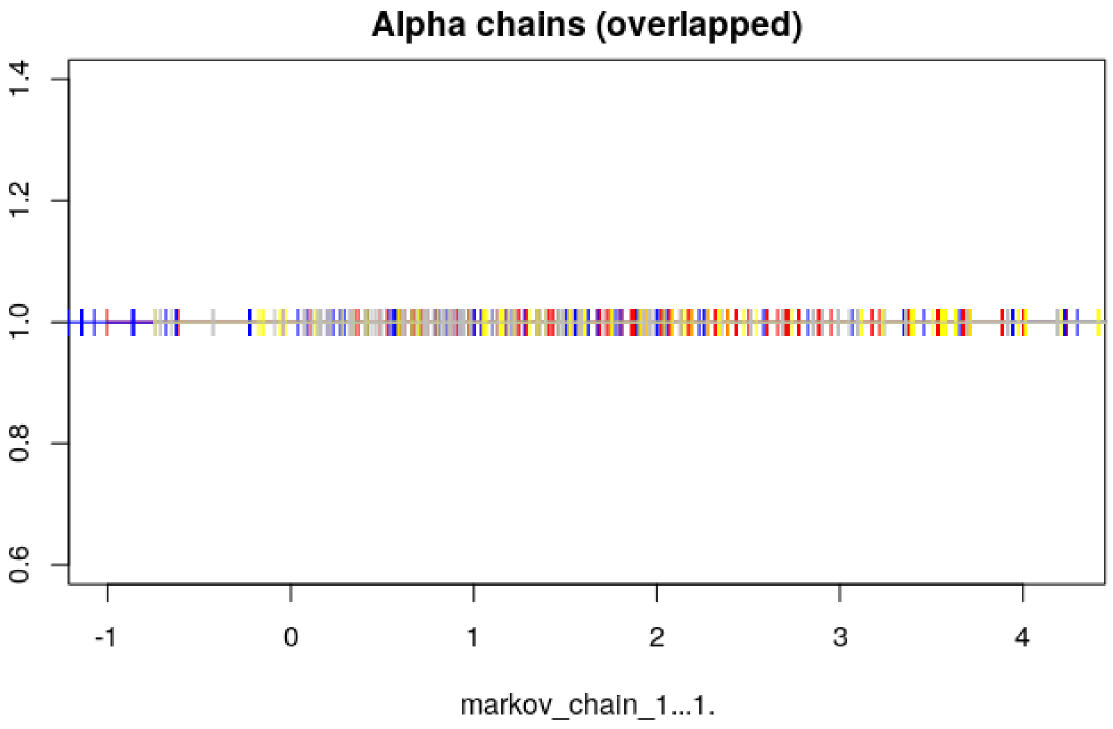

Figure 3: Alpha chains for alpha=1 and beta=5.

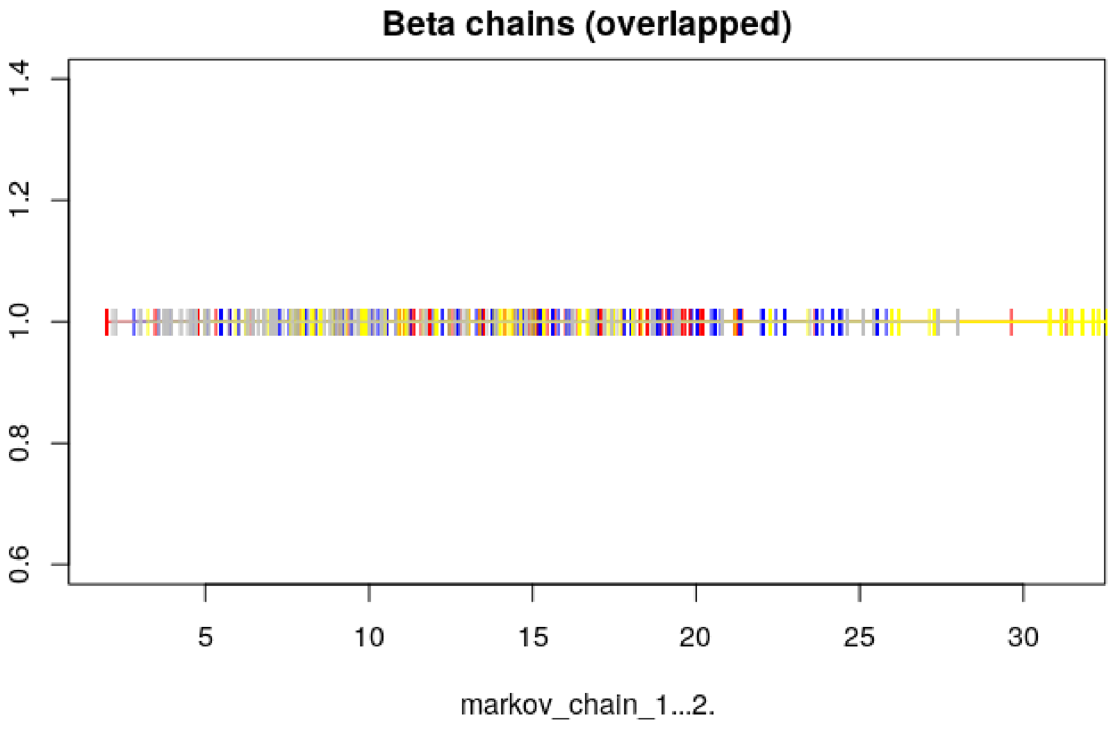

Figure 4: Beta chains for alpha=1 and beta=5.

c)

The ideal method would be choosing overdispersed starting points for each parameter. In the given case, the way they were chosen is by making use of a random number generator built-in function. The following line of code selects two integers randomly in the given range:

```
starting_point <- c(floor(runif(2, min=1, max=10)))
```

d)

The chains were built by iterating the Metropolis algorithm a total number of 5000 times, meaning that this is the length of each of the chains. The reason why this number was chosen is based on graphical evidence by carrying out a trial error process using graphs as the previously shown in figures 1 to 4 and scatter plots like the one below in this report.

e)

The warm-up length is used to "*diminish the influence of the starting values*" (BDA3). As proposed in the book, half of each sequence is discarded, meaning that the final number of points is 2500.

f)

In the given case, four Metropolis chains were computed. The ones for alpha are represented in the figure 5, while the ones for beta are represented in the figure 6.

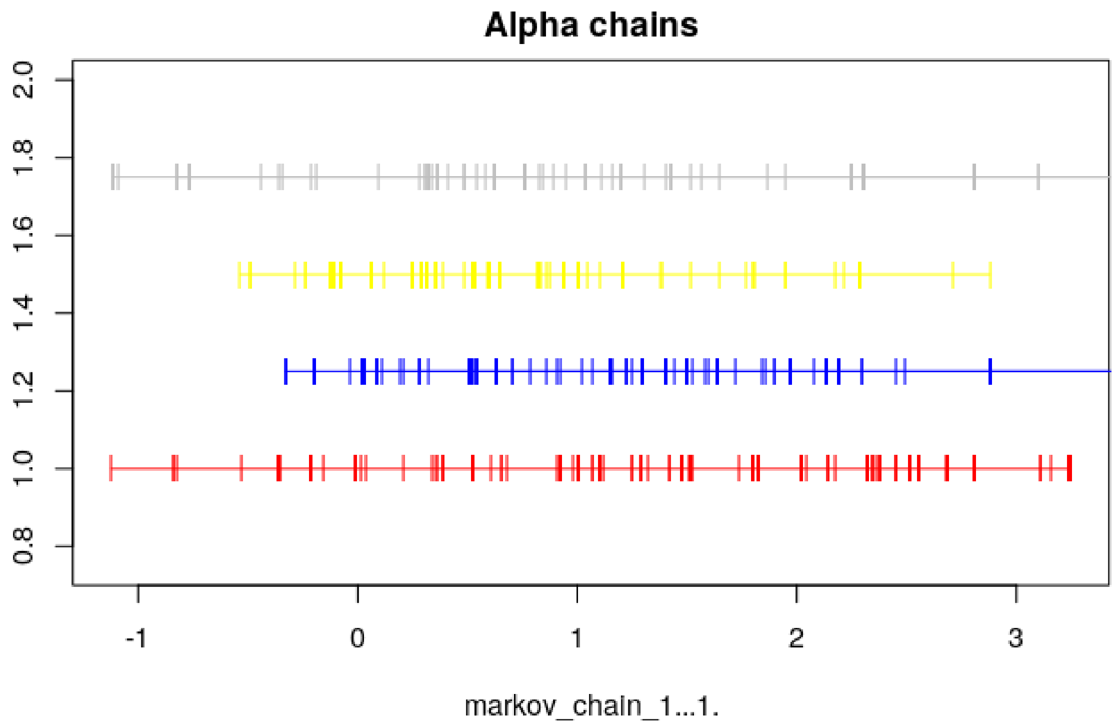

Figure 5: Alpha chains.

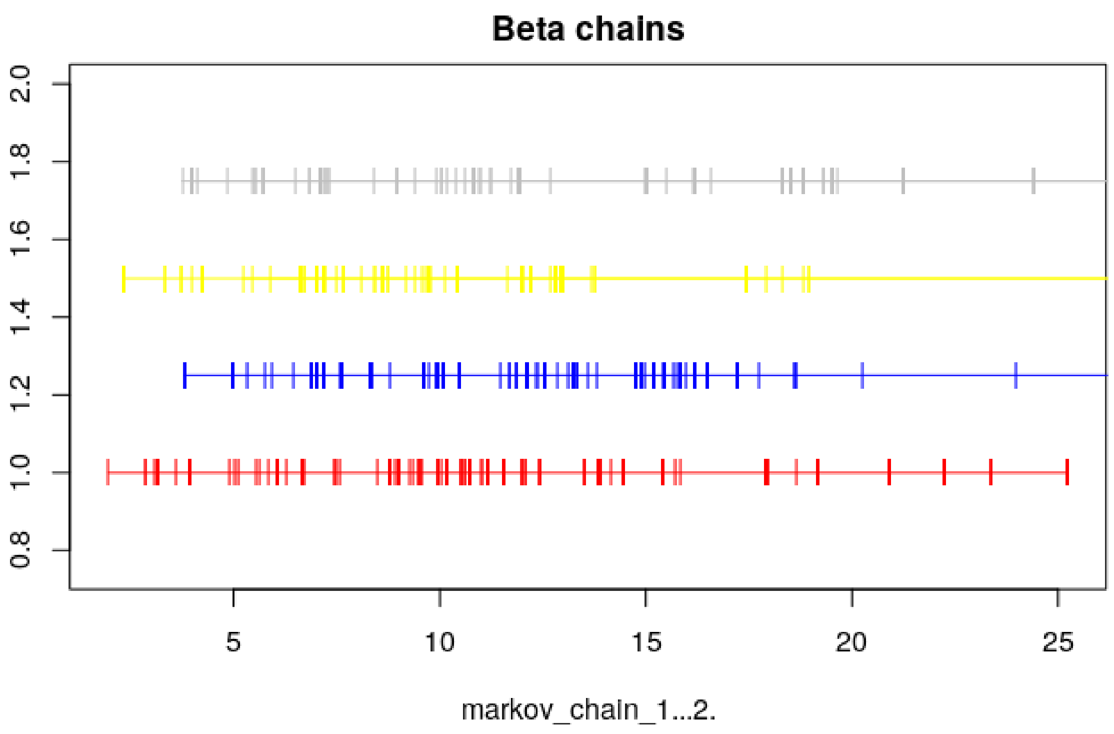

Figure 6: Beta chains.

g)

All chains for α in a single line-plot are represented in figure 7.

h)

All chains for β in a single line-plot are represented in figure 8.

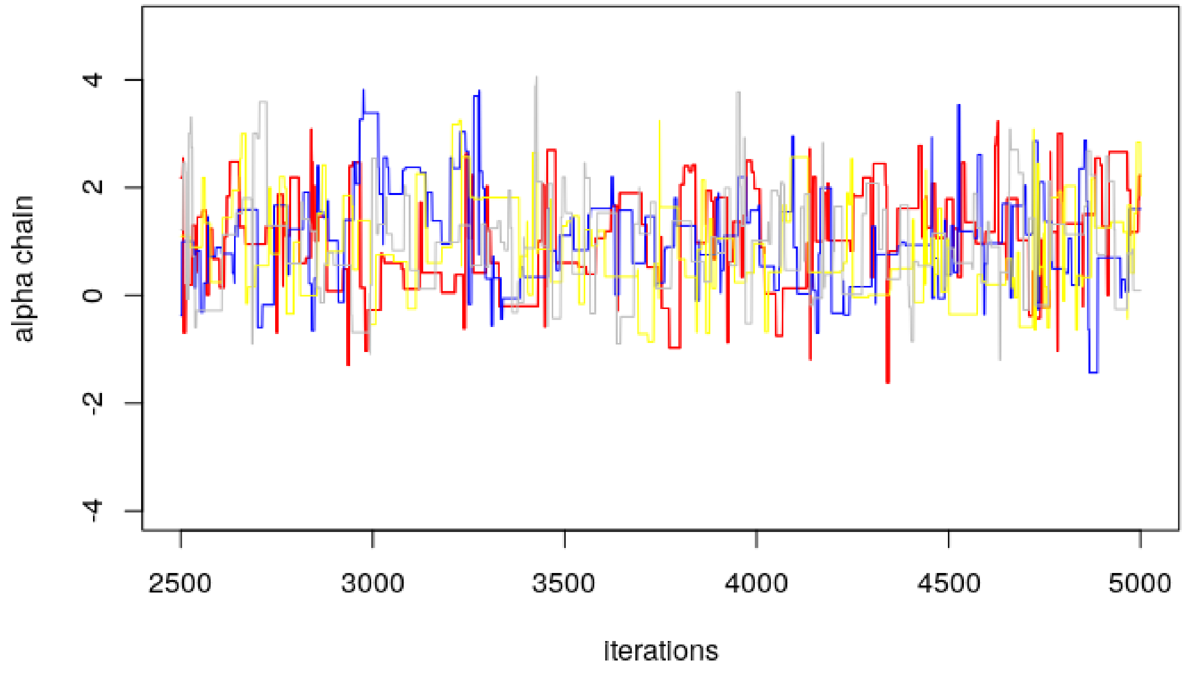

Figure 7: Alpha chains.

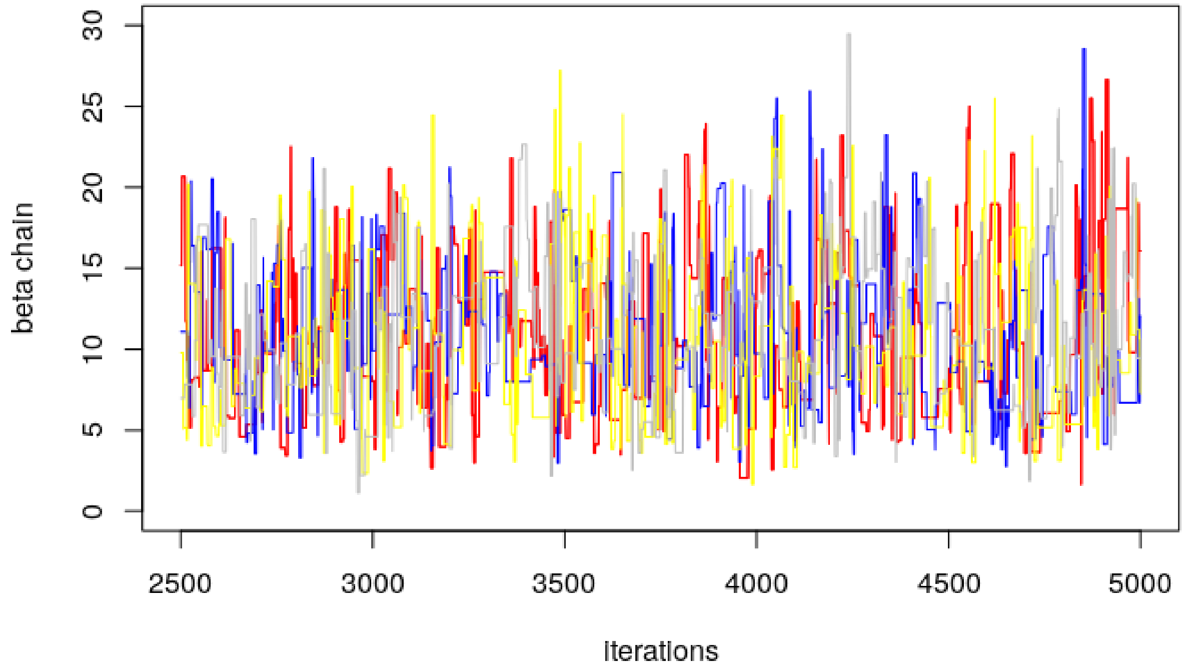

Figure 8: Beta chains.

Exercise 3

```r
sim_alpha <- array(c(x_1, x_2, x_3, x_4), dim = c(length(x_1), 4))
sim_beta <- array(c(y_1,y_2,y_3,y_4), dim=c(length(y_1),4))
R_alpha <-Rhat(sim_alpha)
R_alpha
```

```
## [1] 1.009
```

```r
R_beta <- Rhat(sim_beta)
R_beta
```

```
## [1] 1.007582
```

R is a metric that involves the variance between two parameters, B and W:

- B measures the between sequence variance, meaning how different is each sequence from the others.
- W measures the within sequence variance, meaning how different parameters within the same sequence are.

The expression for R, according to BDA3 is:

$$`R_{metric} = \sqrt[2]{(\frac{`var^{+}(\phi|y)}{W})}$$

For a greater number of samples, the previous metric tends to 1, meaning that the variability of the two parameters previously described is really small. It is an analogous concept to the clusterization metrics, in which inter-cluster and intra-cluster similarity must be compared in order to decide if a clusterization was succesful.

In the given case, with 5000 iterations, the metric (the 2500 warm-up iterations are not included for the calculation) for α is $`R_{metric,alpha} = 1.0090003 < 1.05$ and the one for $\beta$ is $`R_{metric,beta} = 1.0075815 < 1.05$. In both cases, the value is close to 1 and, according to the Stan documentation, when having a lower metric than 1.05, then, the sample can be accepted, which is the current situation.

A good measure was not obtained at first try. The number of iterations had to be increased, because initially, the first trials were performed with less than a thousand steps.

Exercise 4

```r
plot(x_2, y_2, xlim=x_lim_1, ylim=y_lim_1, xlab="alpha", ylab="beta", col="blue",
     main="Scatter plot alpha-beta")
points(x_1, y_1, col="red")
points(x_3, y_3, col="yellow")
points(x_4, y_4, col="grey")
```

Finally, the scatter plot for the draws α and β is included in the following figure.

In the figure 10, there is the scatter plot considering the draws from all the four chains.

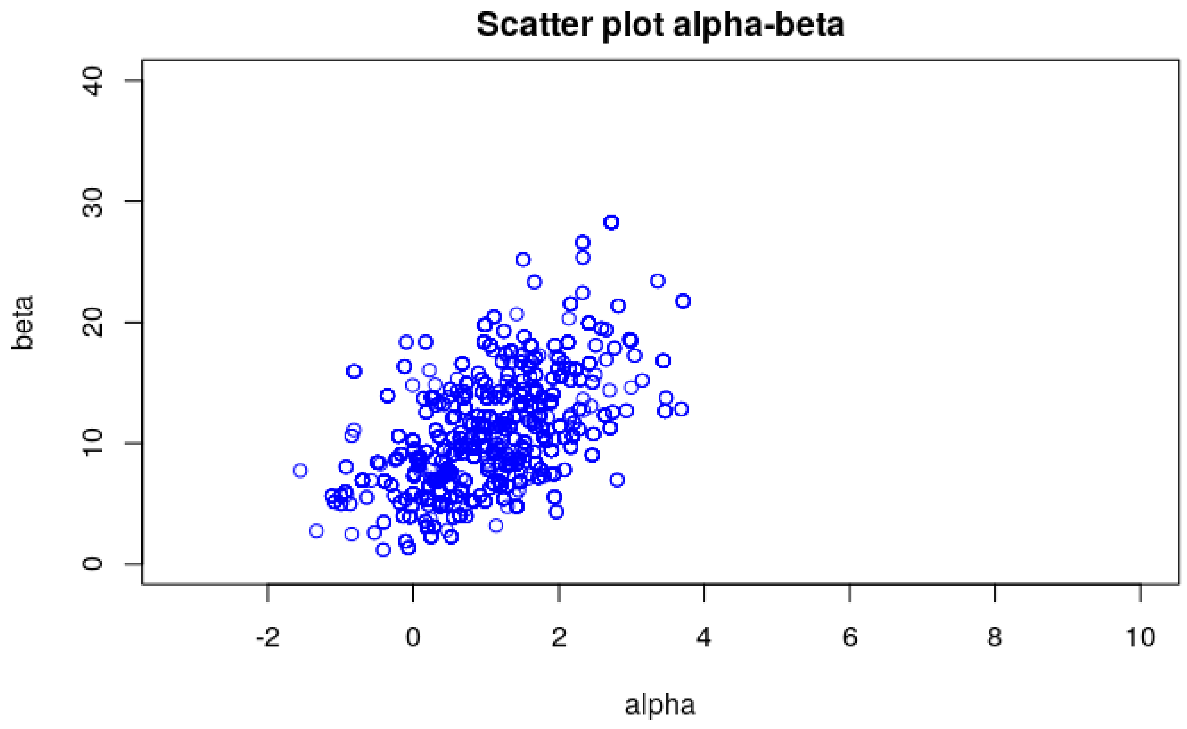

Figure 9: Scatter plot for one chain.

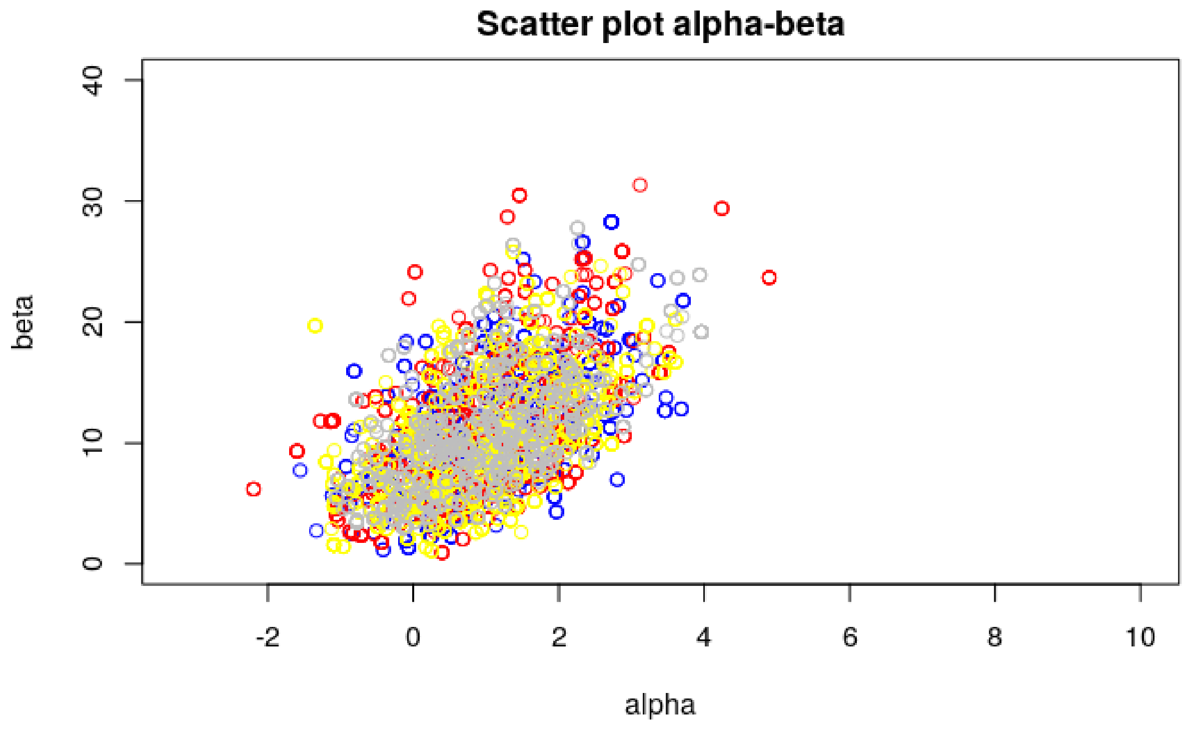

Figure 10: Scatter plot for one chain.