BDA - Assignment 5

Anonymous

Contents

Exercise 1

a)

```
library(aaltobda)
data("bioassay")

density_ratio <- function(alpha_propose, alpha_previous, beta_propose, beta_previous, x, y, n) {
  previous = bioassaylp(alpha_previous, beta_previous, x, y, n)
  propose = bioassaylp(alpha_propose, beta_propose, x, y, n)
  covariance = matrix(c(4,12,12,100), nrow=2)
  prior_previous = dmvnorm(c(alpha_previous, beta_previous), c(0,10),covariance, log=TRUE)
  prior_propose = dmvnorm(c(alpha_propose, beta_propose), c(0,10),covariance, log=TRUE)
  r = exp((propose+prior_propose)-(previous+prior_previous))
  return (r)
}
```

b)

```
library(ggplot2)
library(reshape2)
metropolis_bioassay <- function() {
  chains = 20
  df_comb = data.frame(a = numeric(length), b = numeric(length))
  for (x in 1:chains) {
    df = data.frame(a = numeric(length), b = numeric(length))
    start_a = rnorm(1, mean=0,sd=10)
    start_b = rnorm(1, mean=10,sd=20)
    df$a[0] <- start_a
    df$b[0] <- start_b
    sd_a = 1
    sd_b = 5
    for (i in 1:length) {
```

```
      proposal_a = rnorm(1, start_a, sd_a)
      proposal_b = rnorm(1, start_b, sd_b)
      r = density_ratio(alpha_propose = proposal_a, alpha_previous = start_a, beta_propose = proposal_b
      if (!is.na(r) & r > runif(1)) {
        start_a = proposal_a
        start_b = proposal_b
      }
      df$a[i] = start_a
      df$b[i] = start_b
    }
    df_comb <- cbind(df_comb, df)

  }
  return (df_comb)

}
length = 300
data = metropolis_bioassay()
draws <- data.frame(n = 1:length)
a_data <- data[, colnames(data)=="a", drop = FALSE]
a_data <- subset(a_data, select = -a)
a_data <- cbind(a_data, draws)
a_data <- melt(a_data, id = "n", variable.name = "chain")

b_data <- data[, colnames(data)=="b", drop = FALSE]
b_data <- subset(b_data, select = -b)
b_data <- cbind(b_data, draws)
b_data <- melt(b_data, id = "n", variable.name = "chain")
```

Exercise 2

a)

In metropolis algorithm we select one starting point for the algorithm. Then we choose proposal distirbution, that gives us distribution of possible next time points. Proposal distribution does not have direct relation with the distribution of data (even though similar form can help), but it needs to be symmetric so that moving from point a to point b is as likely as moving from point b to point a. After new point is selected we move there with probability of min(r,1), where r is the ratio between probability density of the original point and the probability density of new point. This jumping is then repeated for certain amount of jumps to complete the algorithm.

b)

As proposal distributions I used:

For alpha, normal distribution with standard deviation of 1

For beta, normal distribution with standard deviation of 5

I tried a few different standard distribution standard deviations and with the used distributions, chains converged fastest and the Rhat was smallest.

c)

Initial points are selected as random variables from normal distribution. Mean 0 and standard deviation of 10 used for alpha. Mean 10 and standard deviation of 20 used for beta.

d)

Chain length used is 300

e)

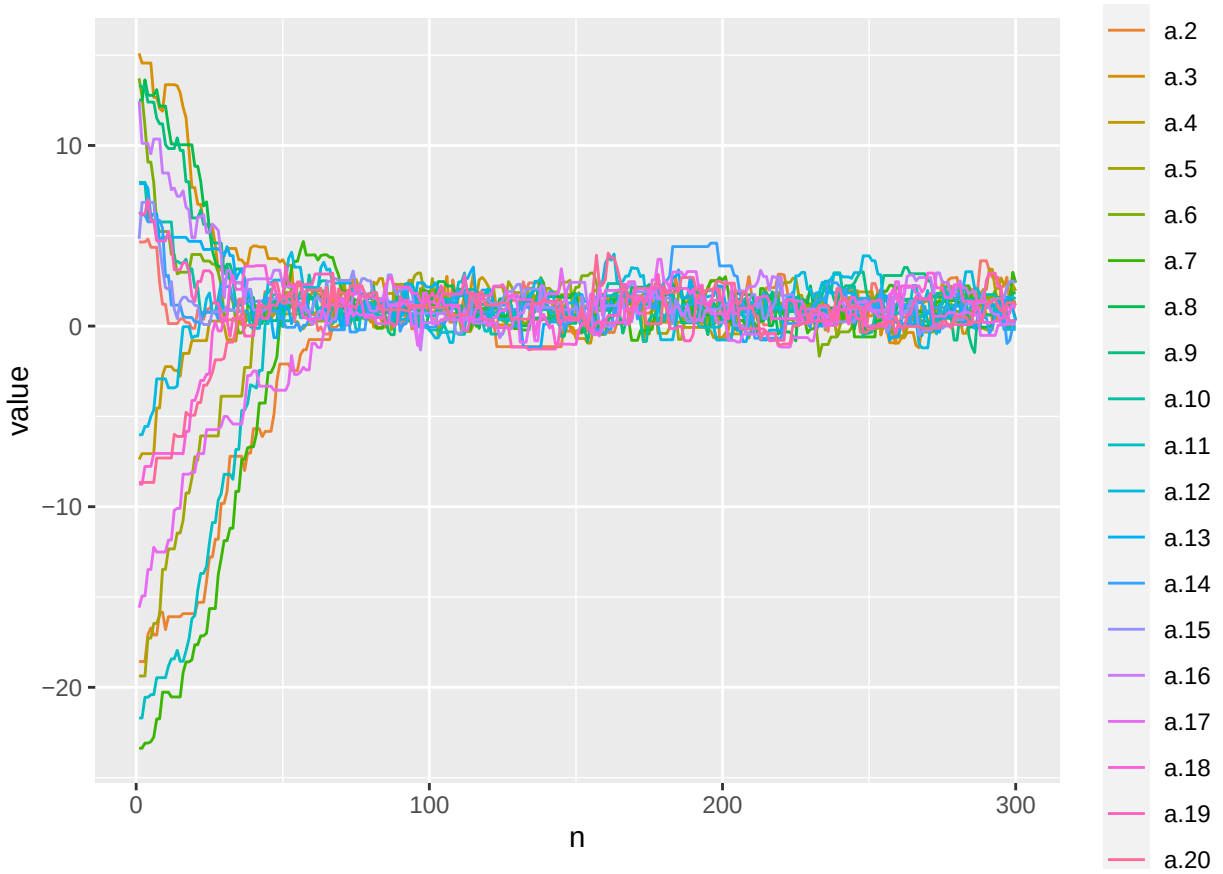
Warm up length used is 150

f)

The number of chains used is 20

g)

```
ggplot(data=a_data, aes(y=value, x = n)) + geom_line(aes(colour = chain))
```



h)

```
ggplot(data=b_data, aes(y=value, x = n)) + geom_line(aes(colour = chain))
```

Exercise 3

```
library(posterior)
a_matrix <- data.matrix(data[, colnames(data) == "a", drop=FALSE])
a_matrix <- subset(a_matrix, select = -a)
a_matrix <- tail(a_matrix,150)
print(paste("Rhat for alpha:",rhat_basic(a_matrix)))
```

```
## [1] "Rhat for alpha: 1.05209010193368"
```

```
b_matrix <- data.matrix(data[, colnames(data)=="b", drop=FALSE])
b_matrix <- subset(b_matrix, select = -b)
b_matrix <- tail(b_matrix,150)
print(paste("Rhat for beta:",rhat_basic(b_matrix)))
```

```
## [1] "Rhat for beta: 1.0956279082529"
```

Based on plots in 2g and 2h and the Rhat < 1.1 achieved above, it seems that chains have converged. From the plots we can see that after warm-up period all chains for alpha are producing values from one distribution and all chains from beta from one distribution. This is visually seen as overlapping chains.

The used Rhat function is rhat_basic from posterior package. rhat_basic is used with default settings which at the time of writing is 'split = TRUE'. rhat_basic implements equation 11.4 from BDA3 book.

a)

Rhat describes the ratio of within-chain variance and between-chain variance. The smaller the Rhat value is the closer the chains are to each other and the variance between the chains is due to within-chain variance.

b)

I obtained above Rhat values with the first try (process for selecting proposal distribution described in 2b, but this was the first try). The Rhat values around 1.1 are acceptable as Rhat < 1.1 indicates convergence with this algorithm. Partially the reason why I achieved good Rhat with the first try is that I checked the convergence of chains from graphs before calculating Rhat values thus knowing good chain length and warmup length already.

Exercise 4

```
combined = cbind(melt(a_matrix)['value'], melt(b_matrix)['value'])
colnames(combined) <- c("alpha", "beta")
ggplot(data = combined, aes(x=alpha, y=beta)) + geom_point()
```