

# BDA - Assignment 5

*Anonymous*

## Contents

Exercise 1a	1
Exercise 1 b	2
Exercise 2	3
Exercise 3	4
Exercise 4	5

```
library(markmyassignment)
exercise_path = 'https://github.com/avehtari/BDA_course_Aalto/blob/master/exercises/tests/ex5.yml'
set_assignment(exercise_path)
```

```
## Assignment set:
## ex5: Bayesian Data Analysis: Assignment 5
## The assignment contain the following task:
## - density_ratio
```

## Exercise 1a

```
library(aaltobda)
data('bioassay')
library(MASS)
library(mvtnorm)

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)

density_ratio = function(alpha_propose, alpha_previous, beta_propose, beta_previous,
                         x, y, n){

  unnorm_log_prior_q = dmvnorm(x = c(alpha_previous, beta_previous),
                                mean = mu, sigma = cov, log = TRUE)
  unnorm_log_likelihood_q = bioassaylp(alpha_previous, beta_previous,
                                         x = bioassay$x, y = bioassay$y, n = bioassay$n)
  unnorm_log_posterior_q = unnorm_log_likelihood_q - abs(unnorm_log_prior_q)
```

```

unnorm_log_prior_p = dmvnorm(x = c(alpha_propose, beta_propose),
                             mean = mu, sigma = cov, log = TRUE)
unnorm_log_likelihood_p = bioassaylp(alpha_propose, beta_propose,
                                      x = bioassay$x, y = bioassay$y, n = bioassay$n)
unnorm_log_posterior_p = unnorm_log_likelihood_p - abs(unnorm_log_prior_p)

return(exp(unnorm_log_posterior_p - unnorm_log_posterior_q))

}

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

## Exercise 1 b

```

##### METROPOLIS ALGORITHM
metropolis_bioassay = function(prior_alpha, prior_beta){

  iterations = 5000
  chain = array(dim = c(iterations + 1, 2))
  chain[1, ] = c(prior_alpha, prior_beta)
  for(i in 1:iterations){
    proposal = rnorm(2, mean = chain[i,], sd= c(1, 5))
    density_ratio = density_ratio(alpha_propose = proposal[1], alpha_previous = 0.374,
                                   beta_propose = proposal[2], beta_previous = 20.04,
                                   x = bioassay$x, y = bioassay$y, n = bioassay$n)
    if (runif(1) < density_ratio){
      chain[i+1, ] = proposal
    }else{
      chain[i+1, ] = chain[i, ]
    }
  }
  warmup = seq(from = 1, to = 2500)
  return(chain)[-warmup, ]
}

# CHAIN 1,2 and 3 STARTING POINTS DRAWN RANDOMLY FROM BIVARIATE PRIOR DISTRIBUTION
prior_draw = rmvnorm(5000, mean=mu, sigma=cov)
prior_alpha = prior_draw[1]
prior_beta = prior_draw[2]
chain1 = metropolis_bioassay(prior_alpha, prior_beta)

prior_draw = rmvnorm(5000, mean=mu, sigma=cov)
prior_alpha = prior_draw[1]
prior_beta = prior_draw[2]
chain2 = metropolis_bioassay(prior_alpha, prior_beta)

```

```

prior_draw = rmvnorm(5000, mean=mu, sigma=cov)
prior_alpha = prior_draw[1]
prior_beta = prior_draw[2]
chain3 = metropolis_bioassay(prior_alpha, prior_beta)

```

## Exercise 2

- Proposal/Jumping distribution:

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

$$\beta \sim N(\beta_{t-1}, \sigma = 5)$$

- The number of chains is 3
- The starting point for the metropolis algorithm is a vector c(prior\_alpha, prior\_beta), where prior\_alpha and prior\_beta are sampled alpha and beta values from prior distribution by using 5000 samples.
- The number of draws generated from the chain is 5000
- Warm-up length for Metropolis 2500

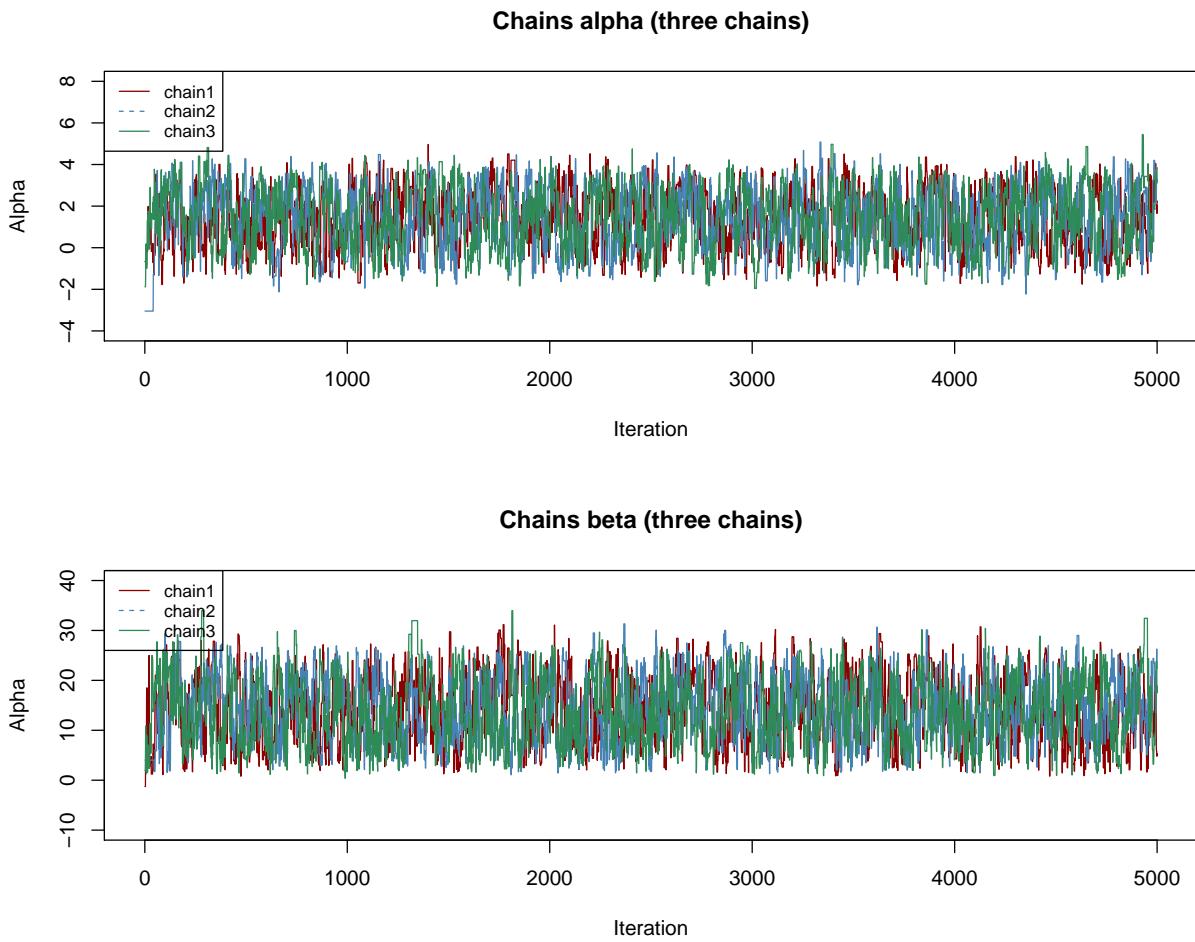
```

par(mfrow=c(2,1))

plot(chain1[,1], type = 'l', xlim=c(0,5000), ylim=c(-4,8),
      col='darkred', main='Chains alpha (three chains)', xlab='Iteration', ylab='Alpha')
lines( chain2[,1], col='steelblue')
lines(chain3[,1], col='seagreen')
legend('topleft', legend=c('chain1','chain2', 'chain3'),
       col=c('darkred','steelblue','seagreen'), lty = 1:2, cex=0.8)

plot(chain1[,2], type = 'l', xlim=c(0,5000), ylim=c(-10,40),
      col='darkred', main='Chains beta (three chains)', xlab='Iteration', ylab='Alpha')
lines( chain2[,2], col='steelblue')
lines(chain3[,2], col='seagreen')
legend('topleft', legend=c('chain1','chain2', 'chain3'),
       col=c('darkred','steelblue','seagreen'), lty = 1:2, cex=0.8)

```



To check whether the metropolis algorithm has converged or not, we run multiple chains to see if they start overlapping. Here we see that they do, indicating convergence.

### Exercise 3

For convergence analysis, I use Rhat from package rstan. This package uses split Rhat method meaning that chains have been split in half so that M is twice the number of simulated chains. Split Rhat allows us to address non-stationarity of chains.

```
suppressWarnings(suppressMessages(suppressPackageStartupMessages({library('rstan')})))
rstan_options(auto_write = TRUE)

chains = list(chain1, chain2, chain3)

for(chain in chains){
  dim(chain) = c(dim(chain),1)
  chain = aperm(chain, c(1, 3, 2))
  res = monitor(chain, probs=c(0.25, 0.5, 0.75), digits_summary=2, warmup=2500, print=TRUE)
}

## Inference for the input samples (1 chains: each with iter = 5001; warmup = 2500):
```

```

##          Q5   Q50   Q95 Mean   SD   Rhat Bulk_ESS Tail_ESS
## V1 -0.9  1.4  3.6  1.4 1.4      1     186     398
## V2  3.3 13.9 25.3 14.1 6.8      1     141     324
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
## Inference for the input samples (1 chains: each with iter = 5001; warmup = 2500):
##
##          Q5   Q50   Q95 Mean   SD   Rhat Bulk_ESS Tail_ESS
## V1 -0.9  1.5  3.6  1.4 1.4      1     212     355
## V2  4.4 14.6 25.2 14.7 6.4      1     185     257
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
## Inference for the input samples (1 chains: each with iter = 5001; warmup = 2500):
##
##          Q5   Q50   Q95 Mean   SD   Rhat Bulk_ESS Tail_ESS
## V1 -1.0  1.5  3.7  1.4 1.5  1.01     143     262
## V2  3.3 13.8 25.1 14.0 6.7  1.00     159     315
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).

```

From the above table we see that Rhat for different chains are all approximately 1 for parameters alpha and beta, which is the desired value at convergence. This means that as the n increases the statistic is approaching 1.01

## Exercise 4

```

plot(chain1, type = 'p', xlim=c(-4,8), ylim=c(-10,40),
      col='steelblue', main='Scatterplot of the draws', xlab='Alpha', ylab='Beta')
points(chain2, col='steelblue')
points(chain3, col='steelblue')

```

**Scatterplot of the draws**

