BSDA: Assignment 5

September 30, 2023

```
knitr::opts_chunk$set(echo = TRUE)
```

Times used for reading and self-study exercise: 8
Time used for the assignment: 12
Good with the assignment:
Good that we got to work some with algorithms and at the same time try do understand the process behind it.
Things to improve in the assignment:
One area of improvement is that the structure of this assignment was a little bit odd. Once you thought you were done with question 1, you always had to go back and change the implemented function. It would be optimal if the task would be more individual, instead of combined.

Question 1

Q1a, Density ratio function

For subtask Q1a we will implement a function that computes the density ratio function. The ratio density function computes the ratio between two densities and in this case we compare the density of having previous α and β values, denoted as α^{t-1} and β^{t-1}, with density of having proposal α and β values, denoted as α^* and β^*. The ratio of densities is often being used in the jumping rule for the Metropolis algorithm where the proposal distribution is symmetric. The parameter vector $\boldsymbol{\theta} = (\alpha, \beta)$: $2x1$ consist of 2 element and the notation regarding previous and proposal parameter values is such that

$\boldsymbol{\theta}^{t-1} = \left(\alpha^{t-1}, \beta^{t-1}\right)$ and
$\boldsymbol{\theta}^* = (\alpha^*, \beta^*)$, respectively.

In this case we got a bivariate Gaussian distribution for $\boldsymbol{\theta}$ as prior distribution which got the following structure and parameter values:

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \sim N_2(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

where $\boldsymbol{\mu}_0 = \begin{bmatrix} 0 \\ 10 \end{bmatrix}$ and $\boldsymbol{\Sigma}_0 = \begin{bmatrix} 2^2 & 12 \\ 12 & 10^2 \end{bmatrix}$

Now when the prior distribution has been stated we need to compute the likelihood function for the total sample y. Below, both the likelihood and the log likelihood

being stated due to reasons related to computational overflow and underflow. The likelihood and log likelihood for the total sample is being stated below:

Likelihood function for random variable y and $\boldsymbol{\theta}$:
$L\left(\boldsymbol{\theta}|y\right) = \prod_{i=1}^{n} p(y|\boldsymbol{\theta})$

Log likelihood function for random variable y and $\boldsymbol{\theta}$
$l\left(\boldsymbol{\theta}|y\right) = \sum_{i=1}^{n} log\left(p(y|\boldsymbol{\theta})\right)$

The log-likelihood will be computed using the bioassaylp-function in R. Now to the proposal / posterior distribution. As mentioned above, due to overflow and underflow we will calculate the log posterior distribution by using the log likelihood added with the log prior distribution. This will be done for both the previous state $\boldsymbol{\theta}^{t-1}$ and proposed state $\boldsymbol{\theta}^{*}$. Hence, we get the log-posterior by computing the following:

Prior distribution, i.e $log\left(p(\boldsymbol{\theta}|y)\right)$:
$log\left(p(\boldsymbol{\theta}|y)\right) \propto l\left(\boldsymbol{\theta}|y\right) + log(p(\boldsymbol{\theta}))$

So the ratio density between $\boldsymbol{\theta}^{*}$ and $\boldsymbol{\theta}^{t-1}$ looks like:
$r = \frac{p(\theta^{*}|y)}{p(\theta^{t-1}|y)}$

and this formula for the ratio density can be rewritten to
$r = exp\left[log(p\left(\boldsymbol{\theta}^{*} \mid y\right)) - log(p\left(\boldsymbol{\theta}^{t-1} \mid y\right)\right]$

Below the implement the function that will compute the ratio density

```
library(bsda)
data("bioassay")

Sigma = matrix(c(2^2,12,12,10^2),2,2)
mu = c(0, 10)

density_ratio <- function(alpha_propose, alpha_previous,
                          beta_propose, beta_previous,
                          x = bioassay$x,
                          y = bioassay$y,
                          n = bioassay$n) {

  # Log likelihood function for Propose state
  loglikelihood1 <- bioassaylp(alpha = alpha_propose,
                               beta = beta_propose,
                               x = x, y = y, n = n)
  # Log likelihood function for Previous state
  loglikelihood0 <- bioassaylp(alpha = alpha_previous,
                               beta = beta_previous,
                               x = x, y = y, n = n)
  # log of prior distribution for theta= (alpha, beta) for propose case
  prior1 <- dmvnorm(x = c(alpha_propose, beta_propose),
                    mean = mu, sigma = Sigma, log = TRUE)
  # log of prior distribution for theta= (alpha, beta) in previous case
  prior0 <- dmvnorm(x = c(alpha_previous, beta_previous),
                    mean = mu, sigma = Sigma, log = TRUE)
  # log posterior distribution for propose case for both parameters
  posterior1 <- loglikelihood1 + prior1
  # log posterior distribution for previous case for both parameters
```

```r
  posterior0 <- loglikelihood0 + prior0
  # calculating the ratio by taking exp of log ratio
  ratio <- exp(posterior1 - posterior0)
  return(ratio)
}
density_ratio(alpha_propose = 1.89, alpha_previous = 0.374,
              beta_propose = 24.76, beta_previous = 20.04)
```

```
## [1] 1.305179
```

```r
density_ratio(alpha_propose =0.374 , alpha_previous = 1.89,
              beta_propose = 20.04 , beta_previous = 24.76)
```

```
## [1] 0.7661784
```

and we see that the ratio density corresponds to $r = 1.305179$ when having previous parameter values set to $\boldsymbol{\theta}^{t-1} = (\alpha^{t-1}, \beta^{t-1}) = (0.374, 20.04)$ and proposed parameter values set to $\boldsymbol{\theta}^{*} = (\alpha^{*}, \beta^{*}) = (1.89, 24.76)$.

Worth mentioning regarding the ratio density, we computed the ratio between two normalized posterior distributions however we would get the same result if we were to use unnormalized posterior distributions due to the normalization terms cancel out each other.

Q1b, Metropolis algorithm using the density ratio function

For subtask Q1b we will implement the Metropolis algorithm by using the ratio density function created in subtask Q1b. We have decided to use two univariate Gaussian distributions as proposal distributions, one for each parameter in $\boldsymbol{\theta} : 2x1$. We get the following two univariate normal distributions

Proposal distribution for α^{*}:
$\alpha^{*} \sim N\left(\alpha_{t-1}, \sigma_{\alpha}\right)$

Proposal distribution for β^{*}:
$\beta^{*} \sim N\left(\beta_{t-1}, \sigma_{\beta}\right)$

Based on these proposal distributions, we can see that the mean / expected value for the proposal distribution corresponds to the previous value for that parameter, i.e $E(\theta^{*}) = \theta^{t-1}$ while the standard deviation is set to a constant value independent of the previous value.

Now when we have picked a proposal distribution for $\boldsymbol{\theta}^{*} = (\alpha^{*}, \beta^{*})$ where the proposal distribution for both α and β is symmetric we need to come up with jumping rule for θ^{*} through the iterations. We have decided the following jumping rule:

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

and we can execute this by comparing the ratio for previous and proposal values with a random value from uniform distribution, $\mathcal{U}(0, 1)$. Therefore, this can be rewritten as

$$\theta_t = \begin{cases} \theta^* & u < r \\ \theta_{t-1} & \text{otherwise} \end{cases}$$

where u is a random drawn value from the uniform distribution such that $u \in \mathcal{U}(0,1)$. The implemented function also includes the acceptance rate in the output. This is important because the generic rule for acceptance rate is $10\% - 40\%$. When the acceptance rate is "too high", most likely the stochastic process will take very long to converge to the unique stationary distribution. For a very low acceptance rate, the stochastic process will be at the same place most of the time.

We will not try to find the optimal proposal distribution. However, we will test some different jumping scales and report one of these. So, we have decided to test three different combinations of jumping scales for a given starting point. We consider the arbitrary choose of $\alpha_0 = 4$ and $\beta_0 = 12$ as starting points for now. As $g = (1, 2, 3)$ is the number of tests with different jumping scale we denote it as

$\boldsymbol{\sigma}_{\theta,g} = (\sigma_{\theta,1}, \sigma_{\theta,2}, \sigma_{\theta,3})'$

where $\sigma_{\theta,g} = (\sigma_{\alpha,g}, \sigma_{\beta,g})$ is the jumping scale for each parameter for the $g : th$ test. We use

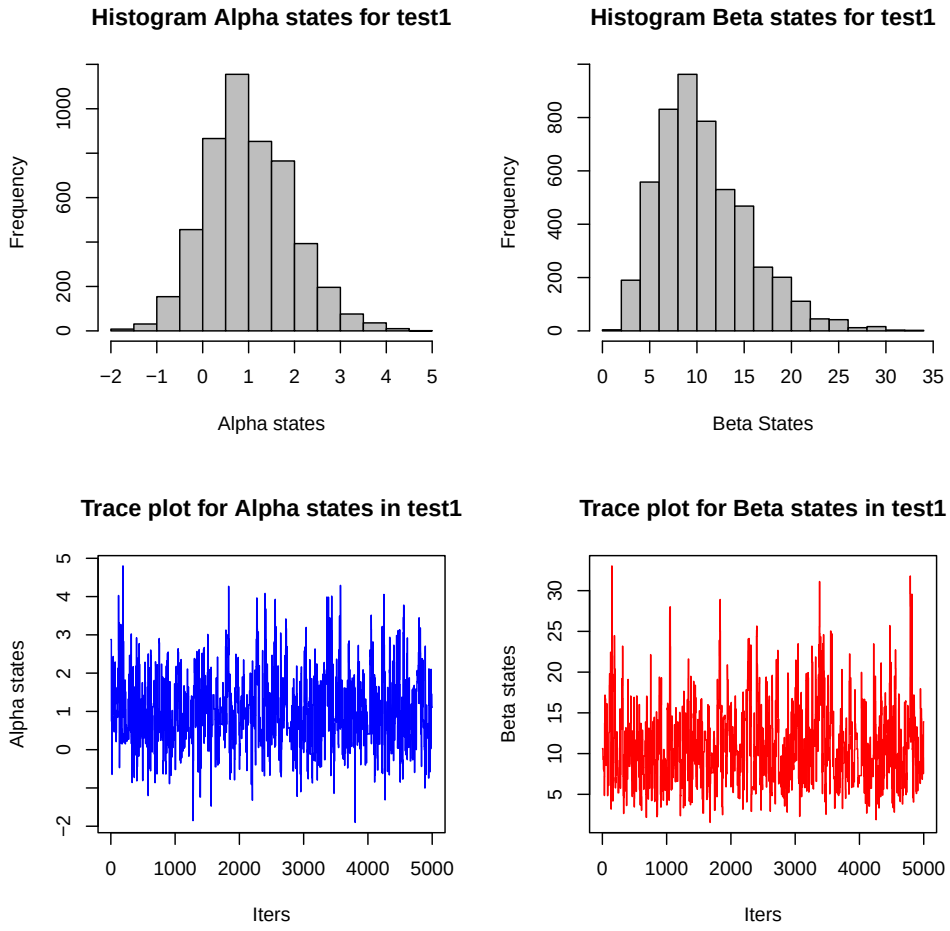
$\sigma_{\theta,1} = (\sigma_{\alpha,1} = 2, \sigma_{\beta,1} = 6)$
$\sigma_{\theta,2} = (\sigma_{\alpha,2} = 4, \sigma_{\beta,2} = 4)$
$\sigma_{\theta,3} = (\sigma_{\alpha,3} = 6, \sigma_{\beta,3} = 2)$

which represents the jumping scales for test 1, test 2 and test 3, respectively. We are going to report the distribution which got a acceptance rate within the generic rule and seems to converge in distribution when we use $n = 5000$ iters.

```
# This is the metropolis algorithm based on the ratio density
set.seed(123)
Metropolis_bioassay <- function(iters = 5000, alpha_start, beta_start,
                                alpha_sd = 2, beta_sd = 4) {

  alpha_0 <- alpha_start # starting point for alpha
  beta_0 <- beta_start # starting point for beta

  alpha_states <- c() # proposal vector for alpha
  beta_states <- c() # proposal vector for beta
  total_acceptance <- 0  # count variable to calculate accept rate

  for ( i in 1:iters) {

  alpha_1 <- rnorm(n = 1, mean = alpha_0, sd = alpha_sd)
  beta_1 <- rnorm(n = 1, mean = beta_0, sd = beta_sd)
  ratio <- density_ratio(alpha_propose = alpha_1,
                         alpha_previous = alpha_0,
                         beta_propose = beta_1,
                         beta_previous = beta_0)

    if ( runif(1) < ratio ) {
```

```
      alpha_0 <- alpha_1
      beta_0 <- beta_1

      alpha_states[i] <- alpha_1
      beta_states[i] <- beta_1

      total_acceptance <- total_acceptance + 1

    } else {
      alpha_states[i] <- alpha_0
      beta_states[i] <- beta_0
    }
  }
  acceptance_rate <- total_acceptance / iters
  output <- list(alpha_states, beta_states, acceptance_rate)
  names(output) <- c("Alpha_states", "Beta_states", "Acceptance rate")

  return(output)
}

Chain.test1 <- Metropolis_bioassay(iters =5000,
                                   alpha_start = 4, beta_start = 12,
                                   alpha_sd = 2,beta_sd = 6)

Chain.test2 <- Metropolis_bioassay(iters =5000,
                                   alpha_start = 4, beta_start = 12,
                                   alpha_sd = 4,beta_sd = 4)

Chain.test3 <- Metropolis_bioassay(iters =5000,
                                   alpha_start = 4, beta_start = 12,
                                   alpha_sd = 6,beta_sd = 2)

Chain.test1$`Acceptance rate` # 0.275

## [1] 0.2698

Chain.test2$`Acceptance rate` #  0.1694

## [1] 0.1772

Chain.test3$`Acceptance rate` # 0.1296

## [1] 0.1444

par(mfrow=c(2,2))
# Histogram and trace plot for Test 1
hist(Chain.test1$Alpha_states, col = "grey",
     main = "Histogram Alpha states for test1",
     xlab = "Alpha states")

hist(Chain.test1$Beta_states, col = "grey",
     main = "Histogram Beta states for test1",
     xlab = "Beta States")
```

```r
plot(Chain.test1$Alpha_states, type = "l", col= "blue",
     main = "Trace plot for Alpha states in test1",
     ylab = "Alpha states",
     xlab = "Iters")

plot(Chain.test1$Beta_states, type = "l", col = "red",
     main = "Trace plot for Beta states in test1",
     ylab = "Beta states",
     xlab = "Iters")
```



We have plotted the histogram and trace plot for each parameter for each test and we have decided to report Test 1 with $(4, 12)$, $\sigma_{\theta,1} = (2, 6)$ and n $= 5000$ as the starting points, jumping scale and number of iterations, respectively. Trace plot can be used as a tool to visualize convergence of a stochastic process and based on the trace plot for the different tests, test1 seemed to be close to convergence for both parameters at $n = 5000$ while test 2 and 3 was not. Also, we decided to report Test1 due to its acceptance rate which corresponds to 0.275 which is satisfied by the generic rule.

Question 2

Q2a, Describe the basic idea behind the Metropolis algorithm

The metropolis algorithm proceeds to randomly move around the sample space for a symmetric proposal density by either rejecting or accepting proposed states. This rejecting rule can be interpreted as how likely a proposed state is given for the sequence to be in current state. When the proposed state is unlikely enough we are rejecting the proposed state hence staying in the same state. Otherwise, the sequence are moving to the proposed, and more likely, state which turning the sequence into markov chain.

Q2b, Decision of prop distribution based on jumping rule

In Q1b we decided to report the proposal distribution for Test 1 with $(4, 12)$, $\sigma_{\theta,1} = (2, 6)$ and n = 5000 as the starting points, jumping scale and number of iterations, respectively, which leads us to

Reported proposal distribution for α^*
$\alpha^* \sim N(\alpha_{t-1}, \sigma = 2)$

Reported proposal distribution for β^*
$\beta^* \sim N(\beta_{t-1}, \sigma = 6)$

There is a combination of reasons why we decided to report the proposal distribution for test 1 instead of test 2 and test 3. One reason is due to the acceptance rate, all the three test has a acceptance rate within $10\% - 40\%$ however test 2s and test 3s acceptance rate was both around 0.1694 while test 1s was higher at 0.275. So we had reason to think that the proposal distribution for Test 1 would converge earlier than the two other tests at n = 5000. This though was confirmed by the different trace plots. Now when the jumping scale has been set, for the upcoming subtasks we will decide the starting points, number of iters and the warm-up period.

Q2c, Decision of prop distribution based on different starting values

Earlier, in Q1b, we decided to report Test 1 based on its trace plot, histogram and acceptance rate at n = 5000. This was being done for an arbitrary starting point at $\alpha_0 = 4$ and $\beta_0 = 12$. Now we decide to create a trace plot for several different starting points when having $\sigma_\alpha = 2$ and $\sigma_\beta = 6$.

We decided to use four different set of starting values for Test1, i.e $\sigma_\alpha = 2$ and $\sigma_\beta = 6$, and plotting respective markov chain in a trace plot. This to be able to visualize their convergence when having different starting points.

```
mean(Chain.test1$Alpha_states)
```

```
## [1] 1.007282
```

```
mean(Chain.test1$Beta_states)
```

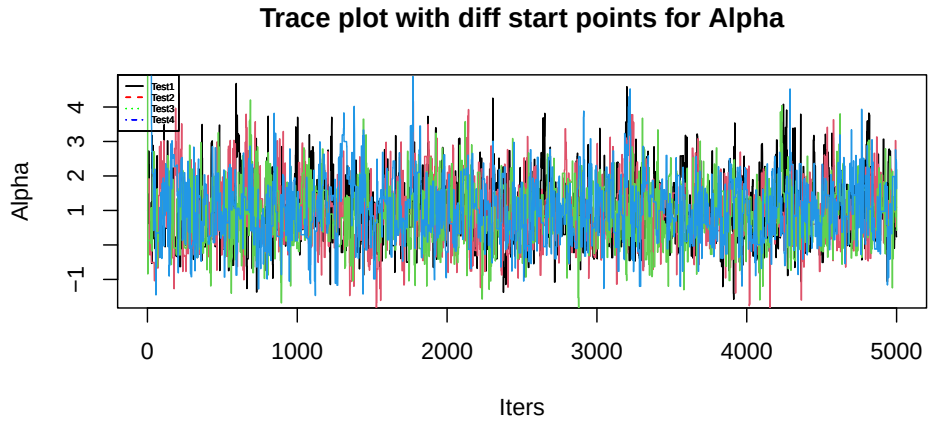
```
## [1] 10.76705
```

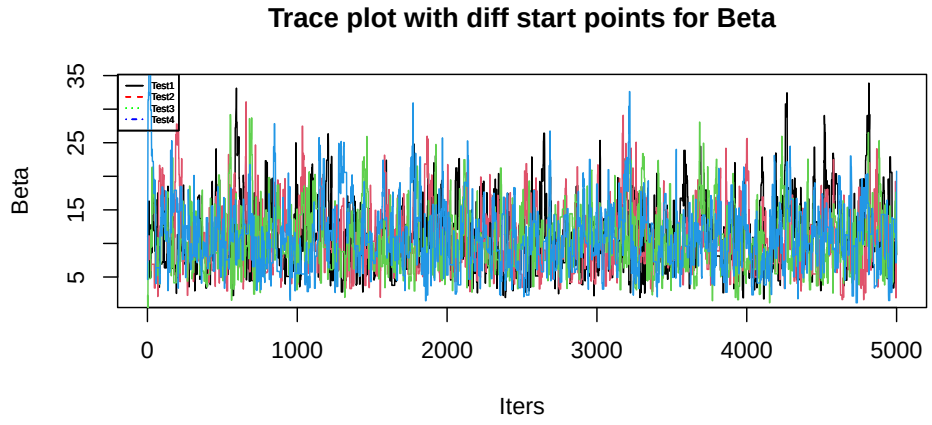
For this subtask we will have some starting points far and close to the parameters sample mean which corresponds to $\bar{\alpha} \approx 1$ and $\bar{\beta} \approx 11$. We have decided to use $(1, 11), (1, 2), (10, 11), (10, 2)$ as starting points (α_0, β_0) for Chain1, Chain2, Chain3

and Chain4 respectively. Below are we presenting all the markov chains in a trace plot at n = 5000 iterations.

```
# Chain 1 with starting points (1,11)
MC1.start <- Metropolis_bioassay(iters =5000,
                                 alpha_start = 1, beta_start = 11,
                                 alpha_sd = 2,beta_sd = 6)
# Chain 2 with starting points (1,2)
MC2.start <- Metropolis_bioassay(iters =5000,
                                 alpha_start = 1, beta_start = 2,
                                 alpha_sd = 2,beta_sd = 6)
# Chain 3 with starting points (10,11)
MC3.start <- Metropolis_bioassay(iters =5000,
                                 alpha_start = 10, beta_start = 11,
                                 alpha_sd = 2,beta_sd = 6)
# Chain 4 with starting points (10,2)
MC4.start <- Metropolis_bioassay(iters =5000,
                                 alpha_start = 10, beta_start = 2,
                                 alpha_sd = 2,beta_sd = 6)

data.alpha <- cbind( cbind(MC1.start$Alpha_states, MC2.start$Alpha_states,
              MC3.start$Alpha_states, MC4.start$Alpha_states))

n = 4
# Trace plot for parameter alpha
par(mfrow=c(2,1))
for (i in 1:n) {
  if (i == 1) {
  plot(data.alpha[,i], type = "l",
       col = i,
       main = "Trace plot with diff start points for Alpha",
       ylab = "Alpha",
       xlab = "Iters")

  } else {
  lines(data.alpha[,i], type = "l", col = i)
    legend("topleft", legend=c("Test1", "Test2", "Test3", "Test4"),
       col=c("black", "red", "green", "blue"), lty = 1:4, cex=0.4)
  }
}

# Trace plot for parameter Beta
data.beta <- cbind( cbind(MC1.start$Beta_states, MC2.start$Beta_states,
              MC3.start$Beta_states, MC4.start$Beta_states))

for (i in 1:n) {
  if (i == 1) {
  plot(data.beta[,i], type = "l",
       col = i,
       main = "Trace plot with diff start points for Beta",
       ylab = "Beta",
       xlab = "Iters")
```

```
  } else {
  lines(data.beta[,i], type = "l", col = i)
    legend("topleft", legend=c("Test1", "Test2", "Test3", "Test4"),
       col=c("black", "red", "green", "blue"), lty = 1:4, cex=0.4)
  }
}
```





In subtask Q1b we had 4 and 12 as arbitrary chosen starting points. However, this trace plots consist of several markov chains (MC) with different starting points for each parameter. Based on the trace plot, all MCs for both parameters seems to be close to convergence after 5000 iterations except maybe for MC2 and MC3 for β. Hence, we will continue with $(1, 11)$ as starting values for α and β for n $= 5000$ iters because these are close to the mass of the proposal distribution.

Q2e, Warm-up length

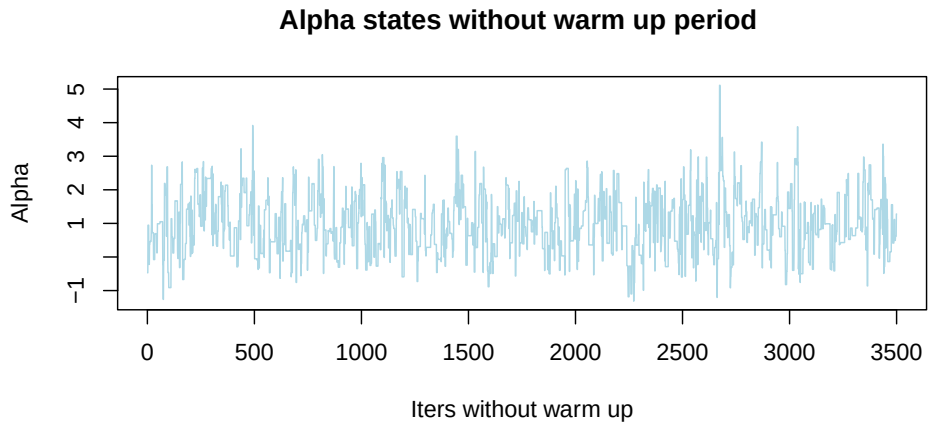
On default, the warm-up period is often set to the first half of the n iterations in the simulation where the warm-up period is the number of iterations it takes for the

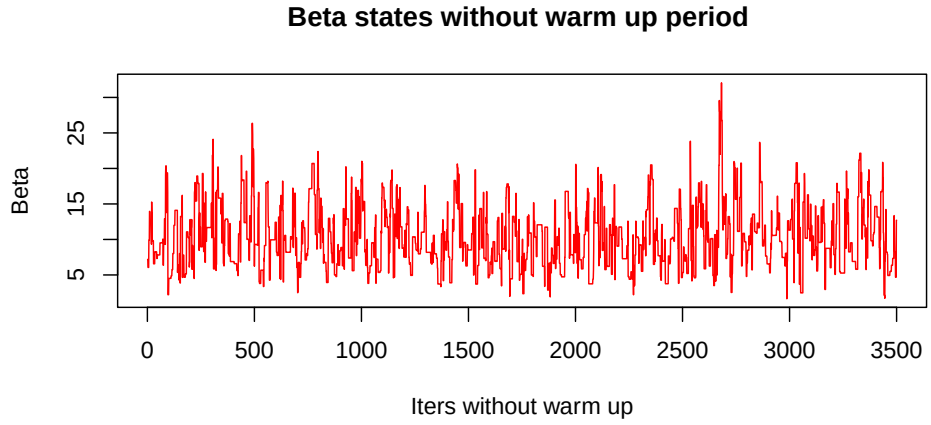
sequence / markov chain to reach the mass of the distribution.
(Reference: Bayesian Data Analysis Third edition, chapter: 11 , page: 284)

However, we decided to use starting points $\alpha_0 \approx \bar{\alpha} = 1$ and $\beta_0 \approx \bar{\beta} = 11$ from now and when the starting point of the sequence is close to the mass of the distribution / proposal distribution, theoretically, it should take fewer iterations for the sequence to reach the mass of the distribution. Based on the starting points and its trace plot we set the warm up period as the first 1500 iterations.

Hence, when excluding the warm-up period, the first 1500 iters, from the proposal distribution with $(1, 11)$ and $(\sigma_\alpha, \sigma_\beta) = (2, 6)$ as starting point and jumping scale respectively we get the following trace plots for the MC

```r
# Sequence with the warm up period
par(mfrow=c(2,1))
MC1.start <- Metropolis_bioassay(iters =5000,
                                 alpha_start = 1, beta_start = 11,
                                 alpha_sd = 2,beta_sd = 6)

# Sequence without the warm up period for alpha
plot(MC1.start$Alpha_states[-c(1:1500)],
     type = "l",
     col = "lightblue",
     main = "Alpha states without warm up period ",
     ylab = "Alpha",
     xlab = "Iters without warm up")

# Sequence without the warm up period for beta
plot(MC1.start$Beta_states[-c(1:1500)],
     type = "l",
     col = "red",
     main = "Beta states without warm up period ",
     ylab = "Beta",
     xlab = "Iters without warm up")
```





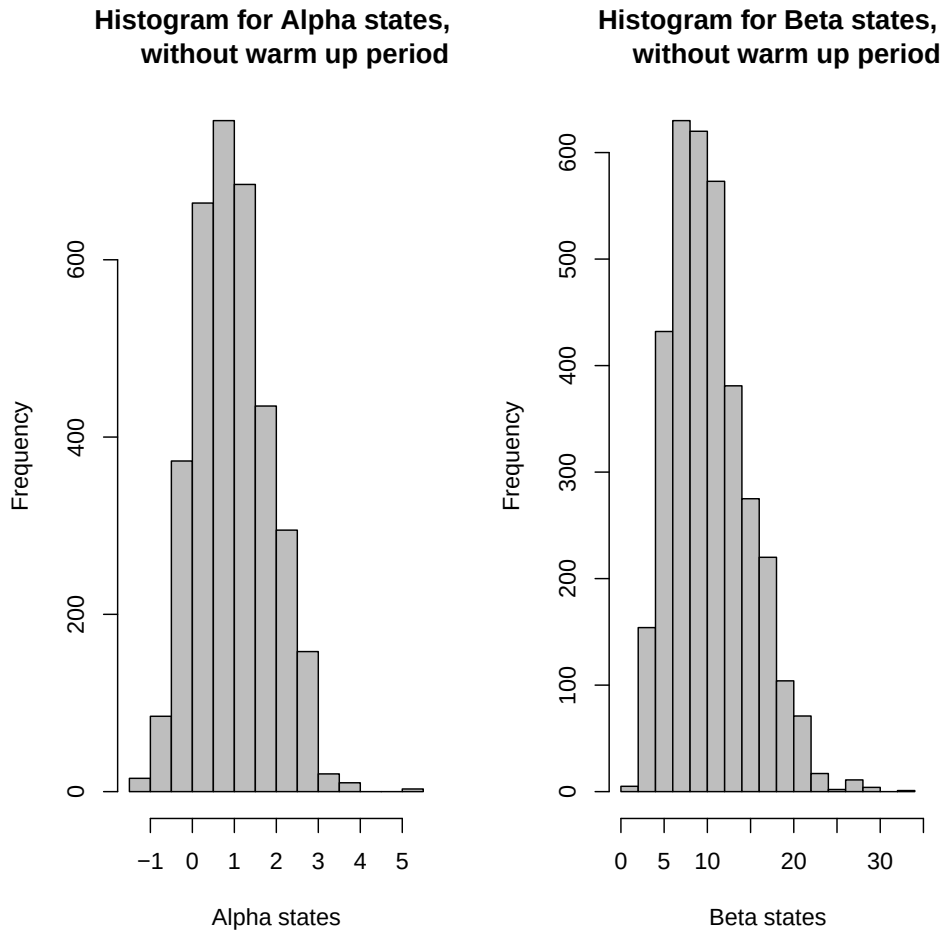
In the context of MC and convergence the proposal distribution that we decided to report after testing different starting points and jumping scales both with and without the warm-up period at n = 5000 iters was the following:

$\alpha^* \sim N(\alpha_{t-1}, \sigma_\alpha = 2)$ and
$\beta^* \sim N(\beta_{t-1}, \sigma_\beta = 6)$

where $(\sigma_\alpha, \sigma_\beta)$ is the jumping scale. Further, $(\alpha_0 = 1, \beta_0 = 11)$ is the starting points, the warm up period corresponds to the first 1500 iters in the MC. Also, we are still using n = 5000 as the total numbers of iters because it results in close convergence for both α and β. Only thing missing is the histogram for reported proposal distribution without the warm-up period. This looks like the following:

```
par(mfrow=c(1,2))
hist(MC1.start$Alpha_states[-c(1:1500)],
     main = "Histogram for Alpha states,
     without warm up period",
     xlab = "Alpha states",
     col = "grey")
```

```r
hist(MC1.start$Beta_states[-c(1:1500)],
     main = "Histogram for Beta states,
     without warm up period",
     xlab = "Beta states",
     col = "grey")
```



When looking at the histograms, it seems like the MC for both parameters are close to convergence based is close to convergence. Worth mentioning is that visual assessment is not sufficient for analyze convergence of Markov chains.

Question 3, $\hat{R}$ for convergence analysis

For this subtask we are going to use a $\hat{R}$ for convergence analysis and we have decided to use a version called split-$\hat{R}$, denoted as $\hat{R}$ in this assignment. The formula for $\hat{R}$ looks like the following:
(Reference: An improved R for assessing convergence of MCMC, page: 5)

Formula for $\hat{R}$:
$\widehat{R} = \sqrt{\frac{\widehat{\text{var}}^+(\theta|y)}{W}}$

where, $\widehat{\text{var}}^+(\theta \mid y) = \frac{N-1}{N}W + \frac{1}{N}B$

where B, W can be seen as the between chain variance and within chain variance, respectively. B and W are being computed by using the following formulas:

$$B = \frac{N}{M-1}\sum_{m=1}^{M}\left(\bar{\theta}^{(m)} - \bar{\theta}^{(.)}\right)^2, \quad \text{where} \quad \bar{\theta}^{(.m)} = \frac{1}{N}\sum_{n=1}^{N}\theta^{(nm)}, \quad \bar{\theta}^{(..)} = \frac{1}{M}\sum_{m=1}^{M}\bar{\theta}^{(.m)}$$

$$W = \frac{1}{M}\sum_{m=1}^{M}s_m^2, \quad \text{where} \quad s_m^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(\theta^{(nm)} - \bar{\theta}^{(.m)}\right)^2.$$

For a more in-depth description about $\hat{R}$, see Reference.

Q3a, Basic idea of $\hat{R}$ and how to interpret its values

The $\hat{R}$ is a estimate that is often being used when looking at the convergence of several Markov chains. This estimate uses the between-sequence information and the within-sequence information when assessing the convergence to a common proposal distribution $p(\theta|y)$. Hence, $\hat{R}$ are using the mean and the variance for W and B to test if each individual sequence has reached stationary at a common distribution.

If we have a large value of $\hat{R}$ estimate there is reason to think that an increase of iters may improve our inference about the proposal distribution.

Q3b, Compute $\hat{R}$ for the reported sequences

We will compute the $\hat{R}$ by using the Rhat-function from package rstan in R. The input for the function should be a matrix, let say $X : nxp$, where the number the rows corresponds to the number of iterations and whose columns equals the number of MCs. Hence, $X_{i,j}$ is the i:th iteration for j:th MCs.

Because the $\hat{R}$ are calculating the convergence of, preferably, several markov chains from a mixed and stationary point of view we will remove the rows that corresponds to the warm-up period for each MC in $X : nxp$. This will be done for α and β separately.

```
# installing the rstan package
library(rstan)

# The Rhat for 4 different markov chains regarding parameter alpha
X.alpha <- cbind(MC1.start$Alpha_states[-c(1:1500)],
                 MC2.start$Alpha_states[-c(1:1500)],
```

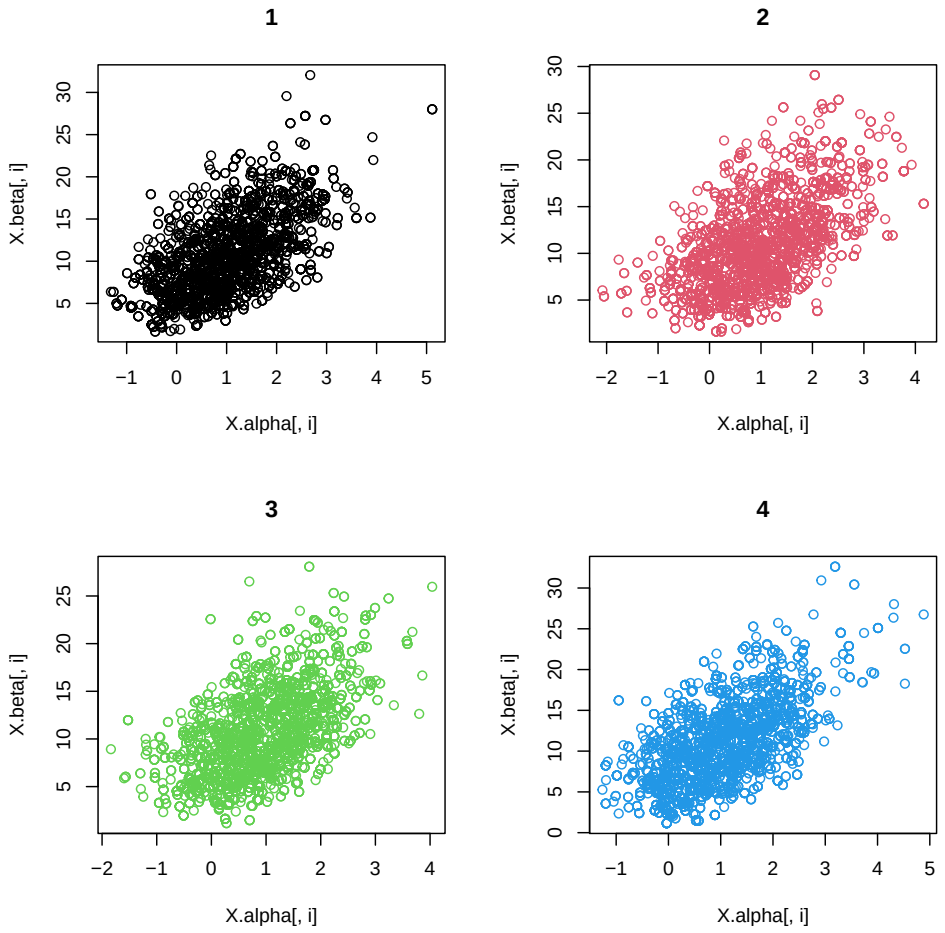
```
                    MC3.start$Alpha_states[-c(1:1500)],
                    MC4.start$Alpha_states[-c(1:1500)])
Rhat(X.alpha)

## [1] 1.004799

X.beta <-  cbind(MC1.start$Beta_states[-c(1:1500)],
                    MC2.start$Beta_states[-c(1:1500)],
                    MC3.start$Beta_states[-c(1:1500)],
                    MC4.start$Beta_states[-c(1:1500)])
Rhat(X.beta)

## [1] 1.004466
```

In our first try we got that $\hat{R}_\alpha = 1.004799$ and $\hat{R}_\beta = 1.004466$ when using n = 3500 iterations after removing the first 1500 iterations as the warm up period. Hence, we got a $\hat{R}$ close to 1 for both α and β which can be seen as a indicator for the chains to be converged. This is supported by the trace plots and the histogram in subtask Q2e.

Worth mentioning is that $\hat{R} > 1.01$ is a rule of thumb so there is still a possible that the chains is close to convergence, but not yet converged. This can be due to the starting points not being overdispersed.

Q4, Scatter plot for α and β

```
par(mfrow=c(2,2))
n = 4
for (i in 1:n) {
  plot(x = X.alpha[,i], y = X.beta[,i],
       main = i,
       col = i)
}
```



Above, each of the four MCs which consist of 3500 iters are displayed on a scatter plot. In a scatter plot we have lost the dependence and time-series aspect of the sequence, therefore we can not longer tell if there is a Markov chain or independent random draws from a joint distribution. Based on the figures it seems like all the four MC has converged to a joint distribution when using 3500 iters.