# BDA - Assignment 5

17/10/2021

We start by pre-calculating the mean $\mu$ and covariance metrics $\Sigma$

```
corr = 0.6
a_std = 2
b_std = 10

mu = c(0,10)
sigma = matrix( c(a_std^2, a_std*b_std*corr, a_std*b_std*corr, b_std^2),nrow = 2)
```

In this assignment we will use a Metropolis algorithm. The algorithm is a adaption of a random walk algorithm. At each iteration, the algorithm draws a sample from a so called jumping algorithm and either accepts or rejects the draw. The following iteration of the algorithm then uses the previously accepted draw as parameter for the jumping distribution. We then discard a warmup sample to get the final draws.

In this assignment we will use a normal distributions $\mathcal{N}(\mu, \sigma)$ as jumping distribution separately for $\alpha$ and $\beta$.

```
density_ratio = function(alpha_propose, alpha_previous, beta_propose, beta_previous, x, y, n){
  likli_propose <- bioassaylp(alpha_propose, beta_propose, x, y, n)
  likli_previous <- bioassaylp(alpha_previous, beta_previous, x, y, n)

  prior_prop <- dmvnorm(c(alpha_propose,beta_propose),mu, sigma, log = TRUE)
  prior_prev <- dmvnorm(c(alpha_previous,beta_previous),mu, sigma, log = TRUE)

  prop = prior_prop + likli_propose
  prev = prior_prev + likli_previous

  res = exp(prop - prev)
  return(res)
}
```

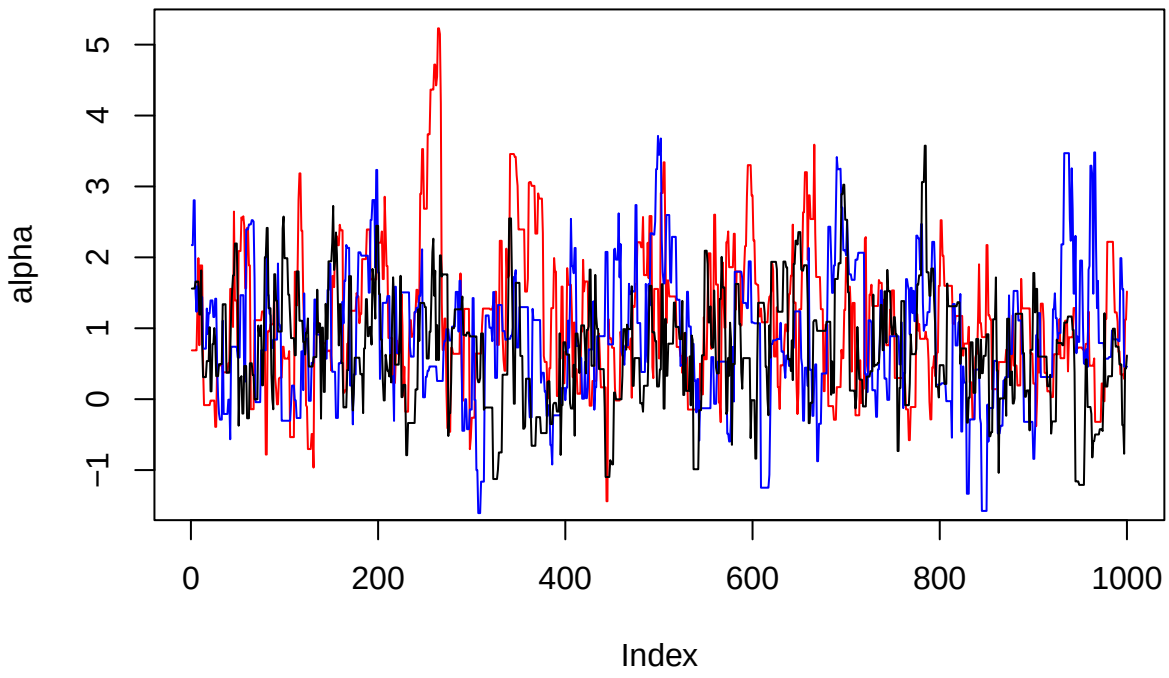
```
Metropolis_bioassay = function(n, alpha_previous, beta_previous, warmup_procent = 0.5){
  alphas = c()
  betas = c()
  for( i in 1:n){
    alpha_propose = rnorm(1, alpha_previous, 1)
    beta_propose = rnorm(1, beta_previous, 5)

     r = density_ratio(alpha_propose = alpha_propose, alpha_previous = alpha_previous, beta_propose = b
     r = min(1,r)
     if(r >= runif(1)){
       alpha_previous = alpha_propose
       beta_previous = beta_propose
     }
    alphas[i] = alpha_previous
    betas[i] = beta_previous
  }
  alpha_final = na.omit(alphas[n*warmup_procent+1:n])
```

```
  beta_final = na.omit(betas[n*warmup_procent+1:n])
  return(cbind(alpha_final,beta_final))
}
```

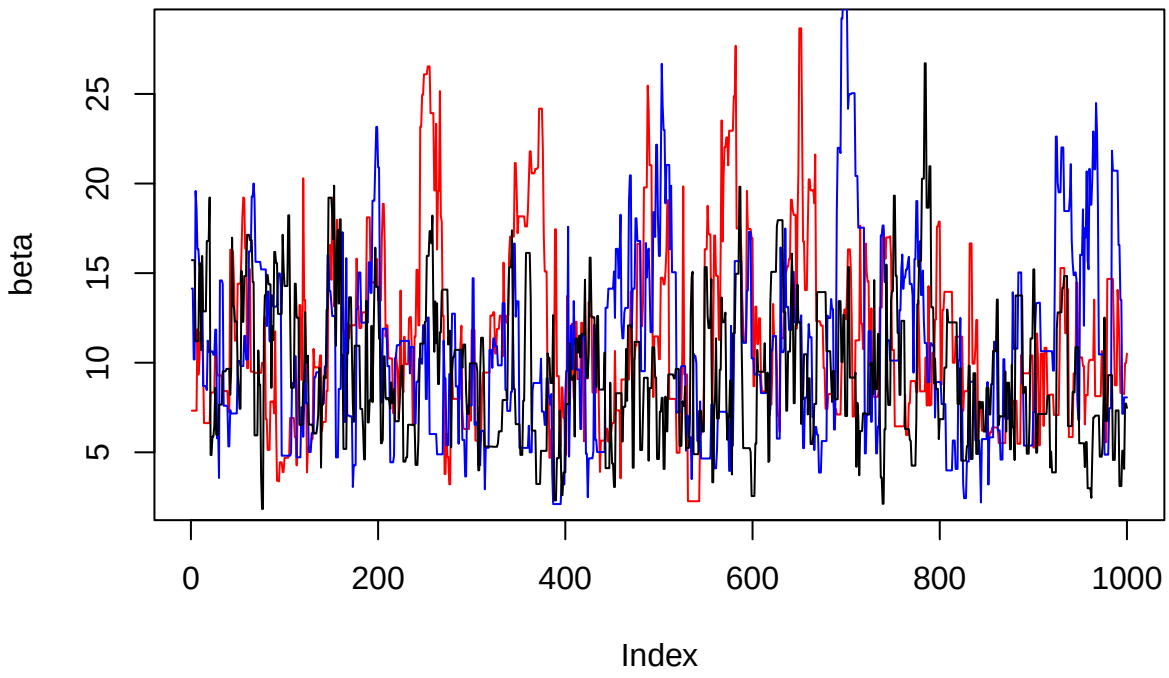
```
n=2000
warmup_procent = 0.5
alpha1 = mu[1]
alpha2 = mu[1]+sqrt(sigma[1,1])
alpha3 = mu[1]-sqrt(sigma[1,1])
beta1 = mu[2]
beta2 = mu[2] + sqrt(sigma[2,2])
beta3 = mu[2] - sqrt(sigma[2,2])
thetas1 = Metropolis_bioassay(n, alpha1, beta1, warmup_procent)
thetas2 = Metropolis_bioassay(n, alpha2, beta2, warmup_procent)
thetas3 = Metropolis_bioassay(n, alpha3, beta3, warmup_procent)
```

We will simulate 3 different chains using different initial starting points. The points are chosen as the priors means as well as one standard deviations in both positive and negative direction. This gives us the starting points $\alpha_{init} = [0, 2, -2]$ and $\beta_{init} = [10, 15, 5]$. We draw $n = 2000$ draws for each chain and use a warm-up length of half the chain, i.e. $n_{warmup} = 1000$.

```
plot(thetas1[,1],
     type='l',
     col='red',
     ylab='alpha')
lines(thetas2[,1],
     type='l',
     col='blue')
lines(thetas3[,1],
     type='l',
     col='black')
```




```
plot(thetas1[,2],
     type='l',
     col='red',
     ylab='beta')
lines(thetas2[,2],
     type='l',
     col='blue')
lines(thetas3[,2],
     type='l',
     col='black')
```




Visually we can see that for both parameters, the chains have good overlap and stay withing a the same interval. Natural noise will occur, but the chains have significant overlap.

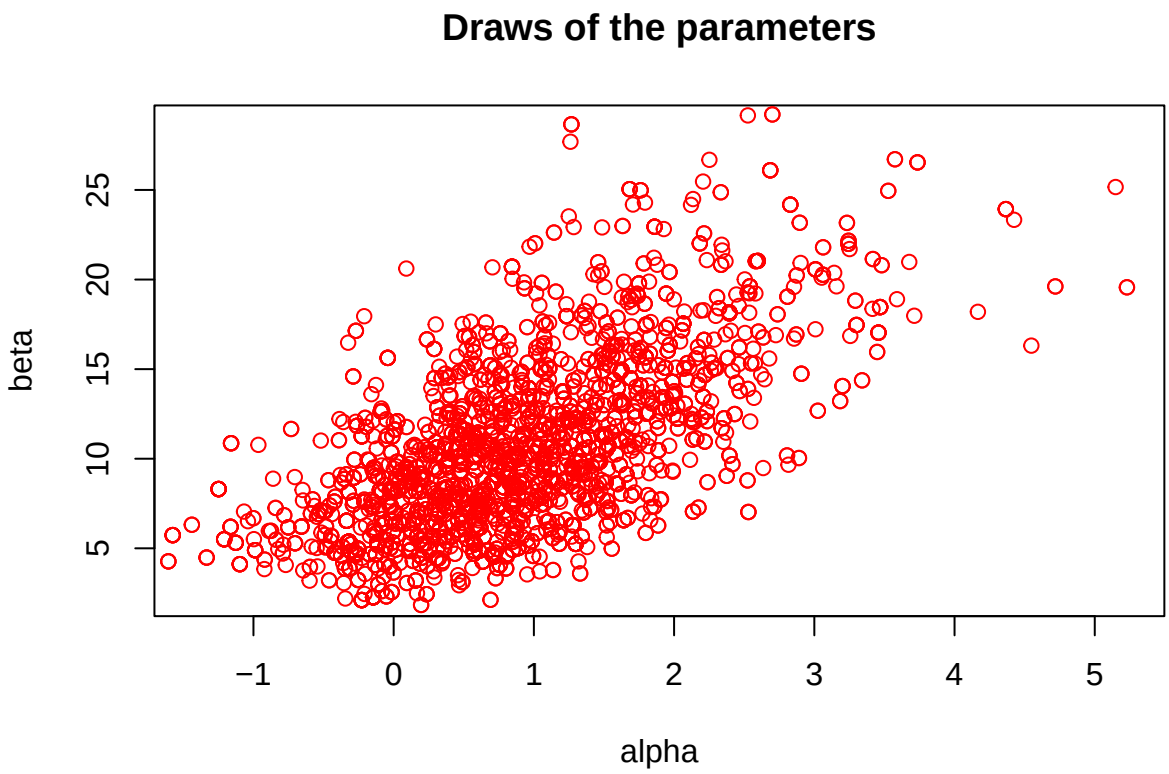
```
alphas = cbind(thetas1[,1],thetas2[,1],thetas3[,1])
alpha_Rhat = Rhat(alphas)
betas = cbind(thetas1[,2],thetas2[,2],thetas3[,2])
beta_Rhat = Rhat(betas)
```

$\widehat{R}$ is a diagnostic tool for evaluating the convergence of parameters. $\widehat{R}$ takes into account both the in-chain variance as well as the between chain variance to give a value for the convergence. If $\widehat{R} < 1.05$ then the chains have converged well, otherwise they have not.

We choose to use the implementation of $\widehat{R}$ from the Rstan package. The values are $\widehat{R}_\alpha = 1.02$ and $\widehat{R}_\beta = 1.019$. Since the value are less than 1.05, we can conclude that the chains have converged well enough.

Below we have a scatter plot of the parameters. The results look similar to a binormal distribution.

```
plot(thetas1[,1], thetas1[,2],
     col='red',
     xlab='alpha',
     ylab="beta",
     main="Draws of the parameters")
points(thetas2[,1], thetas2[,2], col='red')
points(thetas3[,1], thetas3[,2], col='red')
```


Draws of the parameters
beta
alpha
5
10
15
20
25
−1
0
1
2
3
4
5