Assignment 5

anonymous

1 General information

2 Generalized linear model: Bioassay model with Metropolis algorithm

2.1 (a)

Write your answers/code here!

```r
data("bioassay")

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

  mu <- c(0, 10)
  cov <- matrix(c(4, 12, 12, 100), nrow = 2)

  log_posterior_propose <-  bioassaylp(alpha_propose,beta_propose, x, y ,n )   + dmvnorm(c(alpha_pro
  log_posterior_previous <- bioassaylp(alpha_previous,beta_previous, x, y ,n ) + dmvnorm(c(alpha_pre
  return(exp(log_posterior_propose - log_posterior_previous))

}


metropolis_bioassay <- function(alpha_initial, beta_initial, alpha_sigma, beta_sigma, no_draws, x, y
    alpha_samples <- numeric(no_draws)
    beta_samples <- numeric(no_draws)

    alpha_current <- alpha_initial
    beta_current <- beta_initial

    for (i in 1:no_draws) {
        alpha_propose <- rnorm(1, mean = alpha_current, sd = alpha_sigma)
        beta_propose <- rnorm(1, mean = beta_current, sd = beta_sigma)

        acceptance_prob <- min(1, density_ratio(alpha_propose, alpha_current, beta_propose, beta_cur

        if (runif(1) < acceptance_prob) {
            alpha_current <- alpha_propose
            beta_current <- beta_propose
        }
        alpha_samples[i] <- alpha_current
        beta_samples[i] <- beta_current
    }

    result <- data.frame(alpha = alpha_samples, beta = beta_samples)
    return(result)
}
```

```
[1] 1.305179
```

```
[1] 0.7661784
```

Metropolis algorithm:

```
1) Select initial value for $\theta = (\alpha, \beta)$
2) For $i = 1, ..., draws$
    1) Draw $\alpha, \beta$ from normal distribution
    2) Calculate Density ratio
        1) If density ratio $> 1$ => accept $\theta^*$
        2) else randomly accept or reject $\theta^*$
```

Its essential for controlling the exploration of the parameter space by randomly rejecting and accepting. It ensures that you move towards regions of higher posterior density but allows occasional moves to regions of lower density to explore the entire space.

2.2 (b)

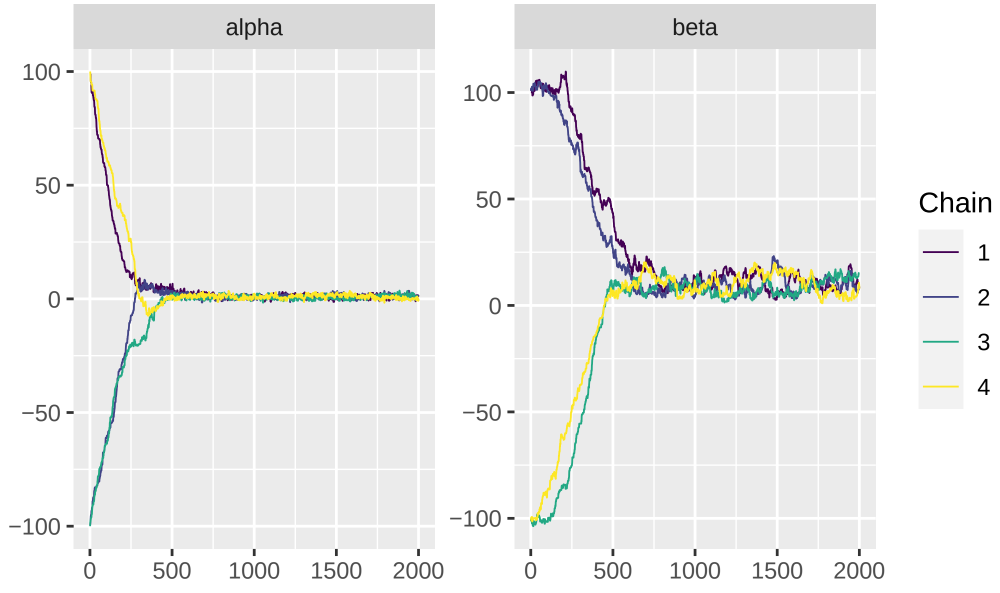
The proposal/jumping distribution for θ is a bivariate distribution that combines α and β which are drawn from independent normal distribution. We use $\sigma = 1$ for both α and β. Increasing or decreasing will change the amount of iterations it takes for the distribution to get to the "true" target distribution.

```r
samples = 2000
df1 = metropolis_bioassay(100, 100, 1, 1, samples, bioassay$x, bioassay$y, bioassay$n)
df2 = metropolis_bioassay(-100, 100, 1, 1, samples, bioassay$x, bioassay$y, bioassay$n)
df3 = metropolis_bioassay(-100, -100, 1, 1, samples, bioassay$x, bioassay$y, bioassay$n)
df4 = metropolis_bioassay(100, -100, 1, 1, samples, bioassay$x, bioassay$y, bioassay$n)


usable_df <- array(data = c(df1$alpha, df2$alpha, df3$alpha,df4$alpha, df1$beta, df2$beta, df3$beta
                    dim = c(samples, 4, 2),
                    dimnames = list(c(), c('df1', 'df2', 'df3','df4'), c('alpha', 'beta')))


color_scheme_set("viridis")
mcmc_trace(usable_df) + labs(title = "Graph 1")   +theme(plot.title = element_text(hjust = 0.5))
```

Graph 1



The proposal distribution for θ is a bivariate distribution that combines α and β which are drawn from independent normal distribution. We use $\sigma = 1$ for both α and β. Increasing or decreasing will change the amount of iterations it takes for the distribution to get to the "true" target distribution. This can be seen in the graph below.

3 2.C

The starting points, I have chosen all combinations of 100 and -100 to show that it converges in the red ellipse. This can bee observed in graph 2, I chose these values as I knew its not close to the final distribution which makes it easy to see it converging. In next analysis I would try to get starting points in the middle of the distribution.

4 2.D

The draws per chain length is 2000. Its important to choose long enough length to converge but if its to long it will take unnecessary computations and clutter the visualization.

5 2.E

The warmup length in my case is 600 because this is the value where α and β has converged, which can be seen in the graph 1. The result of filtering out the warmup length from the start can be observed in graph 3 and 4.

6 2.F

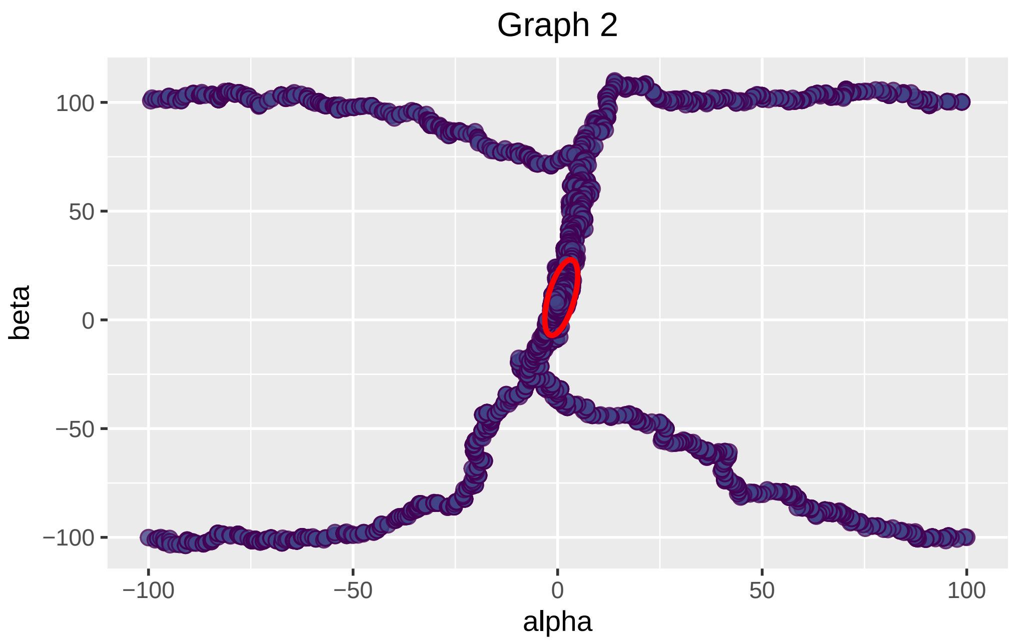
As seen in all graphs we have used 4 chains. The reason for this is to most efficently search the parameter space. This is trivial when observing graph 2.

In summary the important to understand. That there is a relationship between inital points, chain length, warmup length and how these are chosen. The worse the inital points are chosen the longer the chain length will be needed to to converge. The warmup length will be directly correlated to warmup distance between points and the size of the jump lenght which is the speed that it converges to the true distribution. A to high jumping scale will make it harder to analyse.

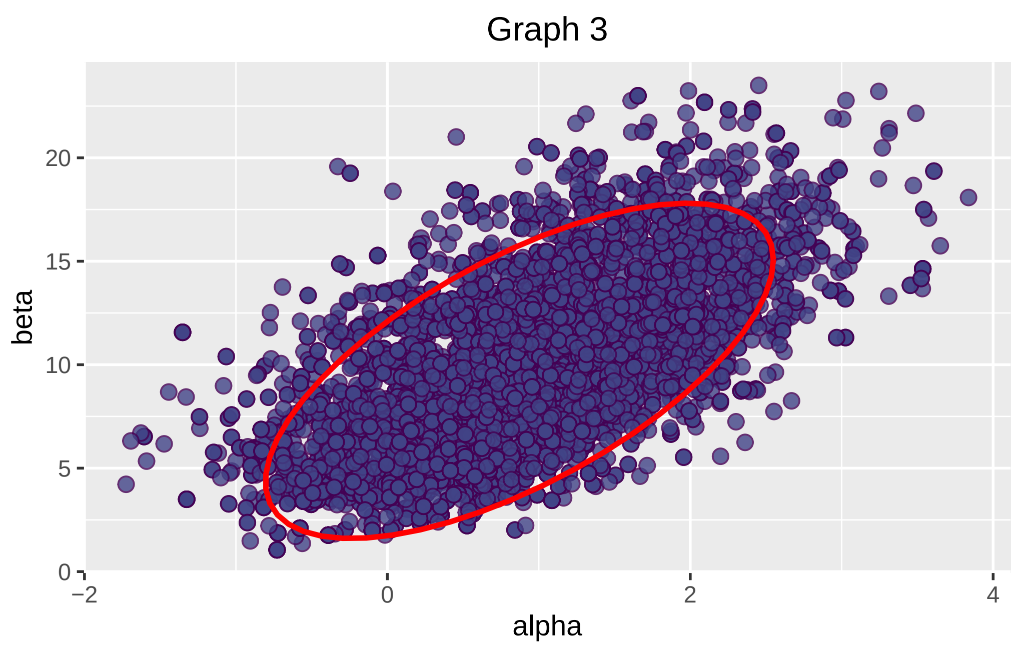
I would recommend to always atleast choose 4 chains around the distribution to make sure that the whole space is searched.

Lastly, we see alpha converges to value 1 and beta to value 10

```r
graph_2b <-mcmc_scatter(usable_df)
graph_2b +  stat_ellipse(level = 0.9, color = "red", linewidth = 1) + labs(title = "Graph 2")   +the
```



```r
graph_2b <-mcmc_scatter(usable_df[600:samples, ,])
graph_2b +  stat_ellipse(level = 0.9, color = "red", linewidth = 1) + labs(title = "Graph 3")   +the
```



R_hat, also known as the Gelman-Rubin statistic , is a diagnostic tool used in the context of Markov Chain Monte Carlo (MCMC) simulations, particularly for assessing the convergence of multiple chains. It helps determine whether multiple chains have mixed well and whether the MCMC sampler has adequately explored the target distribution. Common practice is to consider chains converged if R_hat < 1.05.

```
converged_df = usable_df[800:samples, ,]

r_hat_alpha = rhat_basic(converged_df[,,1])
r_hat_beta = rhat_basic(converged_df[,,2])

# in function rhat_baic(). R_hat is computed with convergence diagnostic for a single variable as de

print( paste("Alpha hat is: ",round(r_hat_alpha,3) ))
```

```
[1] "Alpha hat is:  1.022"
```

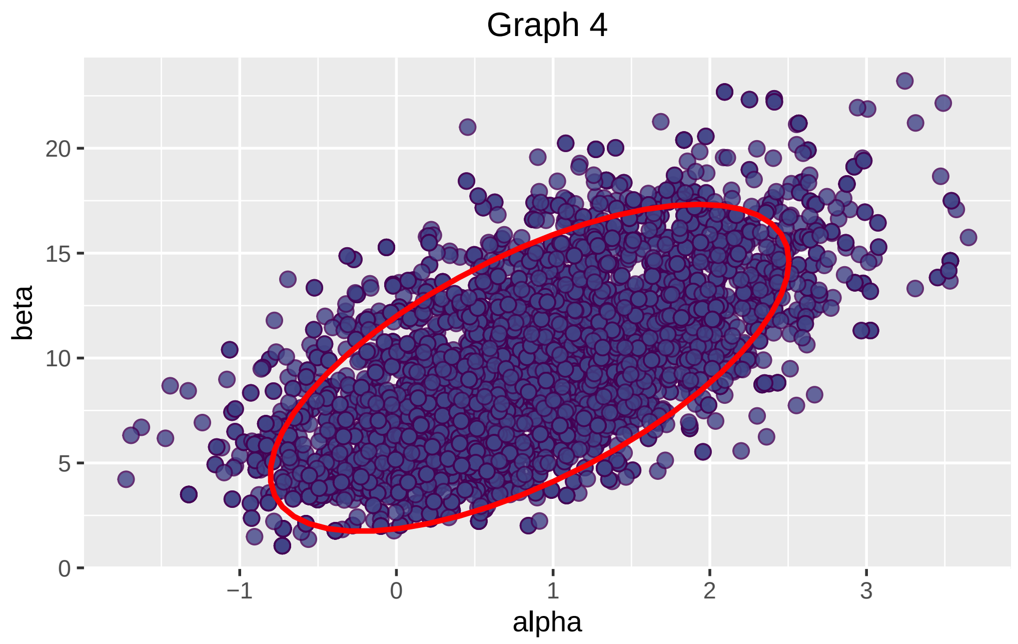
```
print( paste("Beta hat is: ",round(r_hat_beta,3) ))
```

```
[1] "Beta hat is:  1.055"
```

R_hat is computed with convergence diagnostic for a single variable as described in Gelman et al. (2013) with changes according to Vehtari et al. (2021).

At removning 600 warmup samples didnt give good results. There after I did some and trail and error testing until I came to removing 800 warmup samples which left me with 1200 sample. this game me good results

```
graph_2b <-mcmc_scatter(usable_df[800:samples, ,])
graph_2b +  stat_ellipse(level = 0.9, color = "red", linewidth = 1) + labs(title = "Graph 4")   +the
```



AI was used to help formulate the explanations.