BIOSTAT 213 Final Project - MCMC Binana Distribution Simulation

Xinyang Li, 605352032

Problem 3: 2D Banana Distribution

```
library(coda) # MCMC library
```

(i) Describe how to simulate Banana distribution:

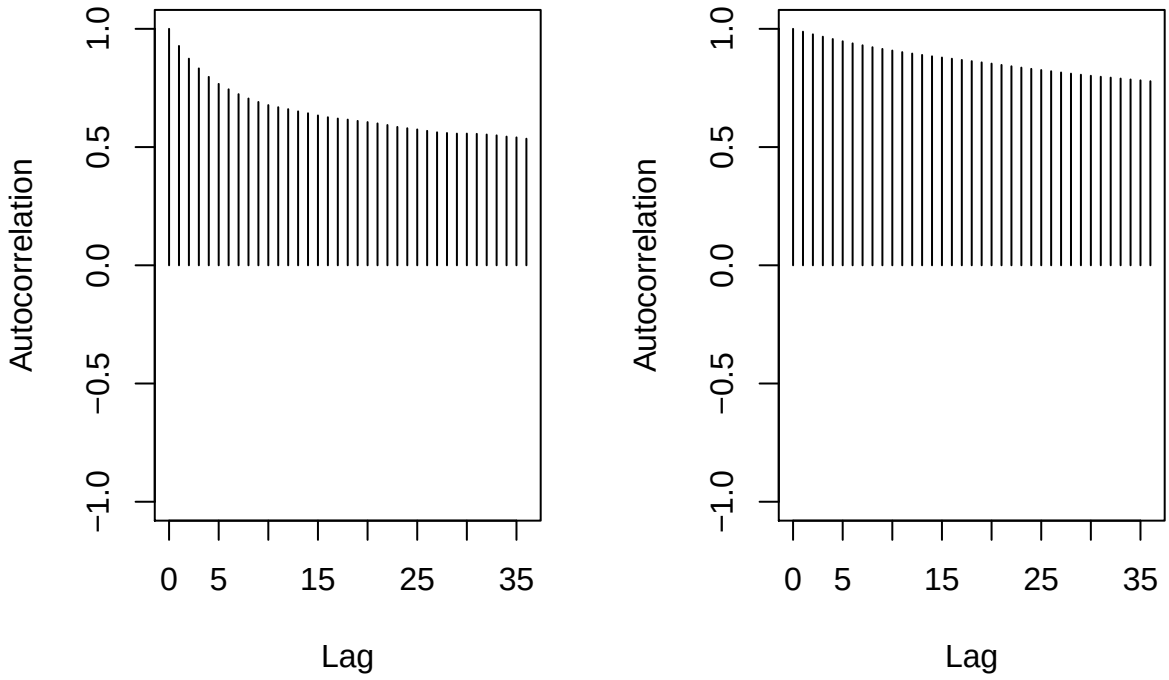
We are going to use the Metropolis-Hashing algorithm to simulate the banana distribution. First, The Metropolis-hasing algorithm works by simulating the Markov Chain with stationary distribution $\pi(x_1, x_2)$. In detail, to simulate the banana distribution, we first initialize $(x_1, x_2) = (0, 0)$ and iteratively update the value. For each iteration, we define an accept probability $A = min(1, \frac{banana(x_1', x_2')}{banana(x_1, x_2)})$, where x_1' and x_2' are updated and x_1 and x_2 are not. With probability A, we accept the proposed value, and update $y = (x_1', x_2')$, otherwise, we update $(x_1, x_2) = (x_1', x_2')$. Repeat the procedure for the number of defined iterations.

(ii) Construct Markov Chain: The belows are the implementation for the previous explanation of the Metropolis-hasing algorithm.

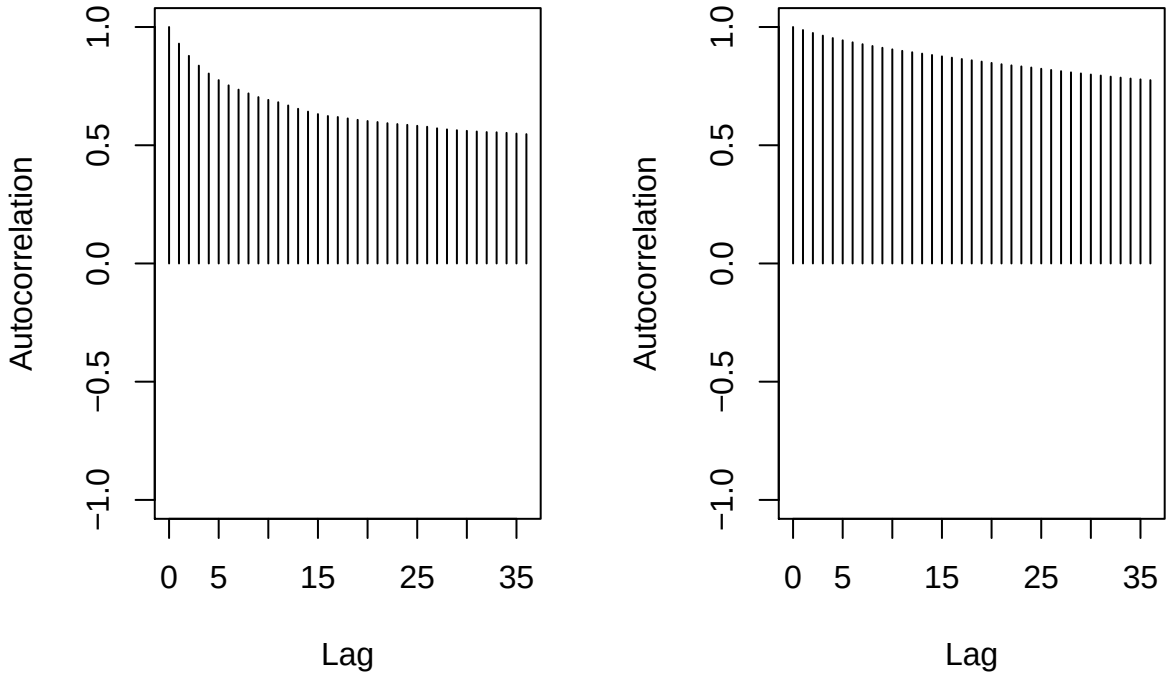
```
metropolis_hashing <- function(w, n) { # w = proposal jump size, n = # of iterations
  # initialize (x1, x2) = (0, 0)
  x1 <- 0
  x2 <- 0
  metro <- matrix(NA, n, 2)
  metro[1, ] <- c(x1, x2)

  for (i in 2:n) { # 10000 iterations
    # update (x1, x2)
    currentx1 <- x1 + rnorm(1,0,sqrt(w))
    currentx2 <- x2 + rnorm(1,0,sqrt(w))
    # pdf for banana distirbution using updated and un-updated x1 and x2
    banana <- exp(- x1^2 / 2) * exp(-(x2 - 2*(x1^2-5))^2 / 2)
    banana_curr <- exp(- currentx1^2 / 2) * exp(-(currentx2 - 2*(currentx1^2-5))^2 / 2)
    # accept probability A
    A <- min(1, banana_curr / banana)
    if (runif(1) < A) {
      x1 <- currentx1
      x2 <- currentx2
    }
    metro[i, ] <- c(x1, x2)
  }
  metro
}
```

(iii) Jump size $w = 0.5, 1, 2$, 10000 iterations.

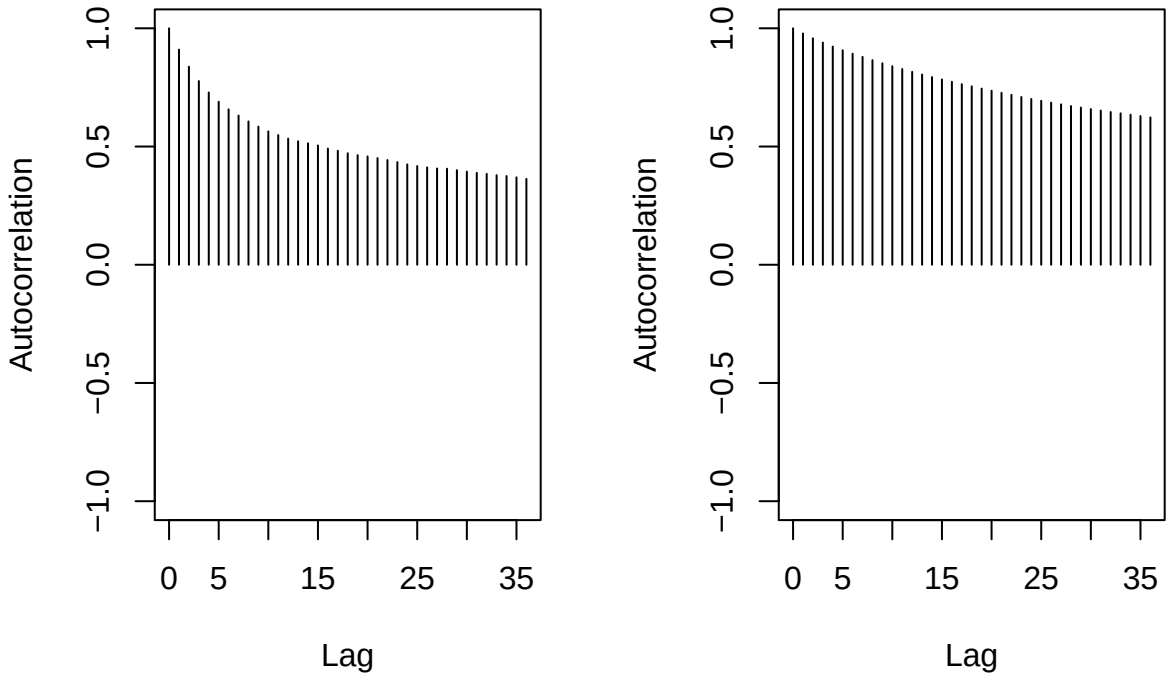
```
set.seed(123)
metro1 <- metropolis_hashing(0.5, 10000)
metro2 <- metropolis_hashing(1, 10000)
metro3 <- metropolis_hashing(2, 10000)

autocorr.plot(as.mcmc(metro1))
```




```
autocorr.plot(as.mcmc(metro2))
```



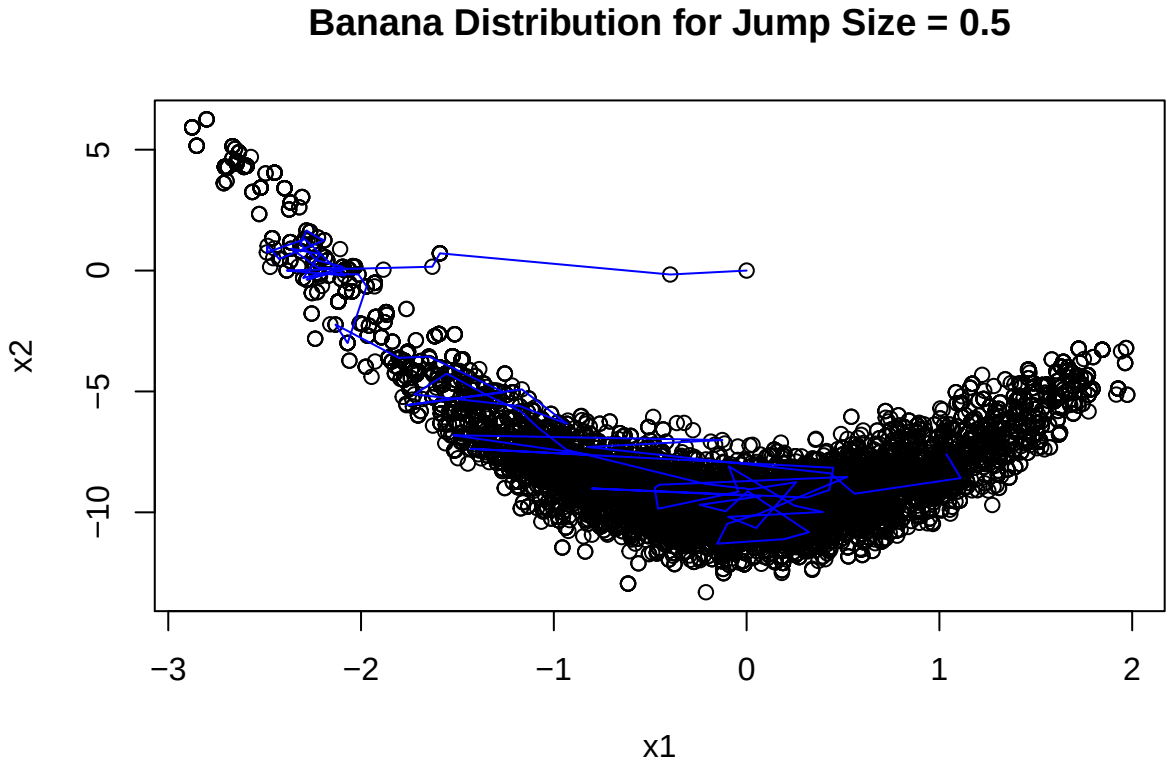
```
autocorr.plot(as.mcmc(metro3))
```



The autocorrelation measures the correlation of elements of a time series with the size of the Lag between the elements. By comparing the three pairs of plots, we found that as the jump size becomes larger, the autocorrelation converges faster. Also, the autocorrelation for x_1 converges faster than that for x_2, for all jump sizes, due to random generation.

(iv) When w = 0.5, plot the path of chain for the first 200 iterations.

```
iter <- metro1[1:200, ] # first 200 iterations for w = 0.5
plot(metro1, xlab="x1", ylab = "x2", main = "Banana Distribution for Jump Size = 0.5")
lines(iter, col = 'blue')
```


Banana Distribution for Jump Size = 0.5
x2
5
0
−5
−10
−3
−2
−1
0
1
2
x1