

# Assessment1-ST5222

Name: Zhu Xu; User ID: E033798; Student ID: A0191344H

15 September 2018

## Problem:

We are given a Binomial distribution:

$$f(x) = \binom{n}{x} p^x (1-p)^{n-x} \quad x \in \{0, \dots, n\}$$

now,  $n = 100$ , and write a computer code to perform a Metropolis-Hastings algorithm with stationary distribution  $f(x)$ . Consider various  $p$  as well as proposals.

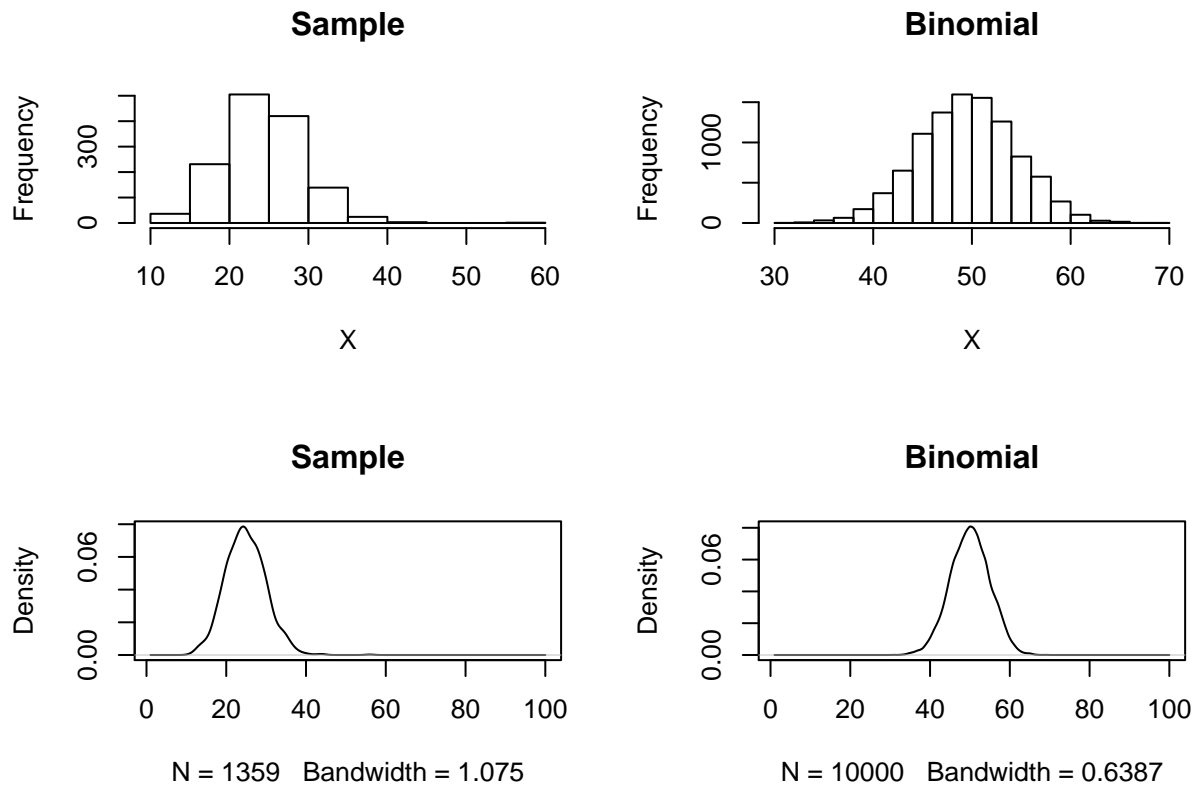
Here is my solutions.

## Solutions:

### 1. Proposal-Unif , Binomial distribution( $n=100$ , $p=0.25$ )

```
f_0 <- function(x){
  (factorial(100)/(factorial(x)*factorial(100-x)))*0.25^(x)*(1-0.25)^(100-x)
}
# define the Binomial distribution , n=100, p=0.25
q_0 <- function(x){1/100}
# define the proposal function q_0 (Uniform)
n <- 100
X <- numeric() # store samples
count <- 1
x_old <- 0 # initialize the Markov chain
for(i in 1:10000) # main loop to obtain samples
{
  x_new <- sample.int(101, 1)-1 # sample a integer from q_0, as a candidate
  u <- runif(1,0,1)
  acc_1 <- min((f_0(x_new)*q_0(x_old))/(f_0(x_old)*q_0(x_new)), 1)
  # calculate the acceptance prob
  if(u < acc_1) # accept
  {
    X[count] <- x_new
    count <- count +1
    x_old <- x_new
  }
  else # reject the sample
  {
    x_old <- x_old
  }
}
par(mfrow = c(2, 2))
hist(X, main = "Sample", xlab = "X")
XX <- rbinom(10000, 100, 0.5)
hist(XX, main = "Binomial", xlab = "X")
P <- density(X, n = 10000, from = 1, to = 100)
plot(P, main = "Sample")
```

```
B <- density(X, n = 10000, from = 1, to = 100)
plot(B, main = "Binomial")
```



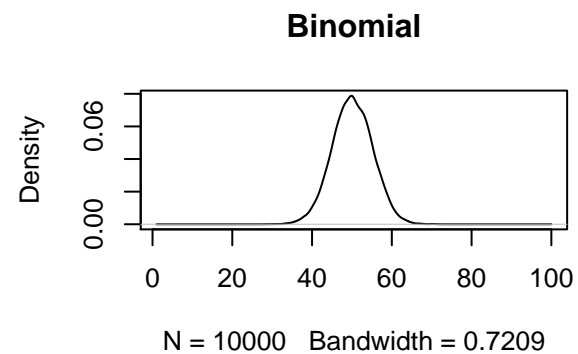
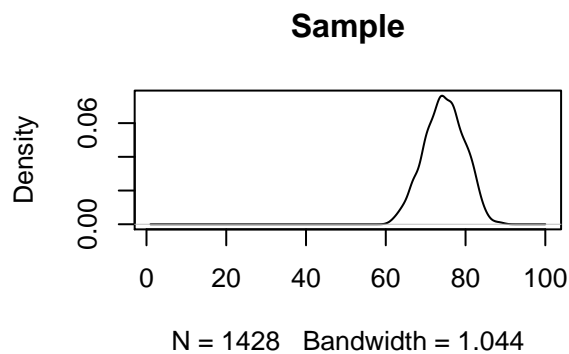
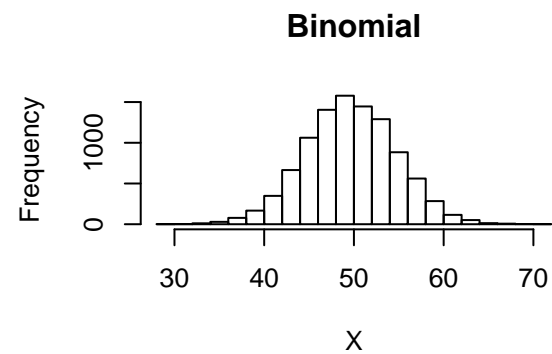
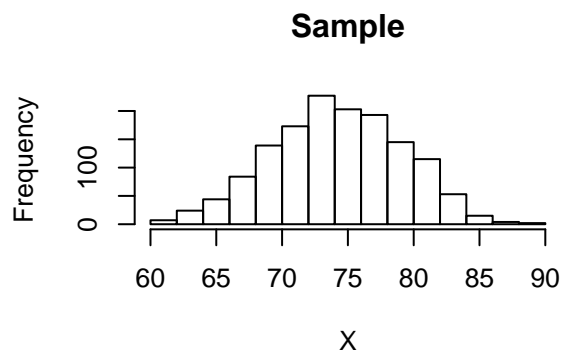
## 2.Proposal–Unif , Binomial distribution(n=100, p=0.75)

```
f_2 <- function(x){
  (factorial(100)/(factorial(x)*factorial(100-x)))*0.75^(x)*(1-0.75)^(100-x)
}
# define the Binomial distribution , n=100, p=0.75
q_0 <- function(x){1/100}
# define the proposal function q_0 (Uniform)
n <- 100
X <- numeric() # store samples
count <- 1
x_old <- 0 # initialize the Markov chain
for(i in 1:10000)
{
  x_new <- sample.int(101, 1)-1 # sample a integer from q_0, as a candidate
  u <- runif(1,0,1)
  acc_1 <- min((f_2(x_new)*q_0(x_old))/(f_2(x_old)*q_0(x_new)), 1)
  # calculate the acceptance prob
  if(u < acc_1) # accept
  {
    X[count] <- x_new
    count <- count +1
  }
}
```

```

    x_old <- x_new
  }
  else # reject
  {
    x_old <- x_old
  }
}
par(mfrow = c(2, 2))
hist(X, main = "Sample", xlab = "X")
XX <- rbinom(10000, 100, 0.5)
hist(XX, main = "Binomial", xlab = "X")
P <- density(X, n = 10000, from = 1, to = 100)
plot(P, main = "Sample")
B <- density(XX, n = 10000, from = 1, to = 100)
plot(B, main = "Binomial")

```



### 3.Proposal — unif 3, p=0.5

```

f_1 <- function(x){
  (factorial(100)/(factorial(x)*factorial(100-x)))*0.5^(x)*(1-0.5)^(100-x)
}
# define the Binomial distribution , n=100, p=0.5
q_0 <- function(x){1/100}
# define the proposal function q_0 (Uniform)
n <- 100

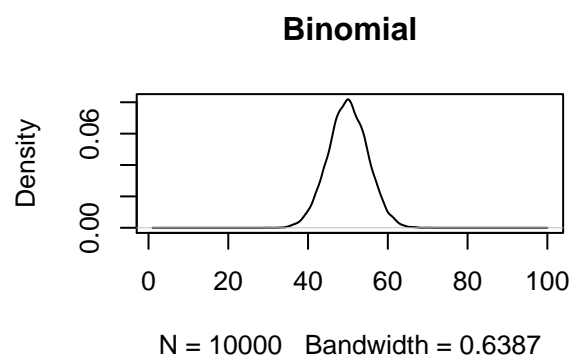
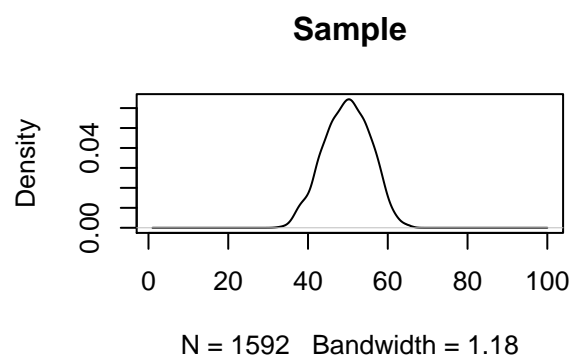
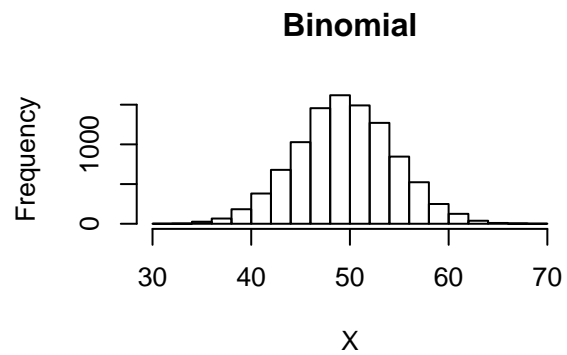
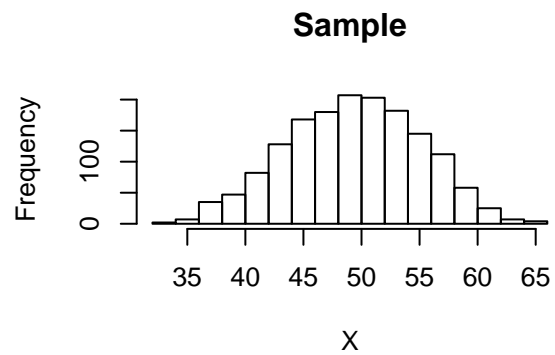
```

```

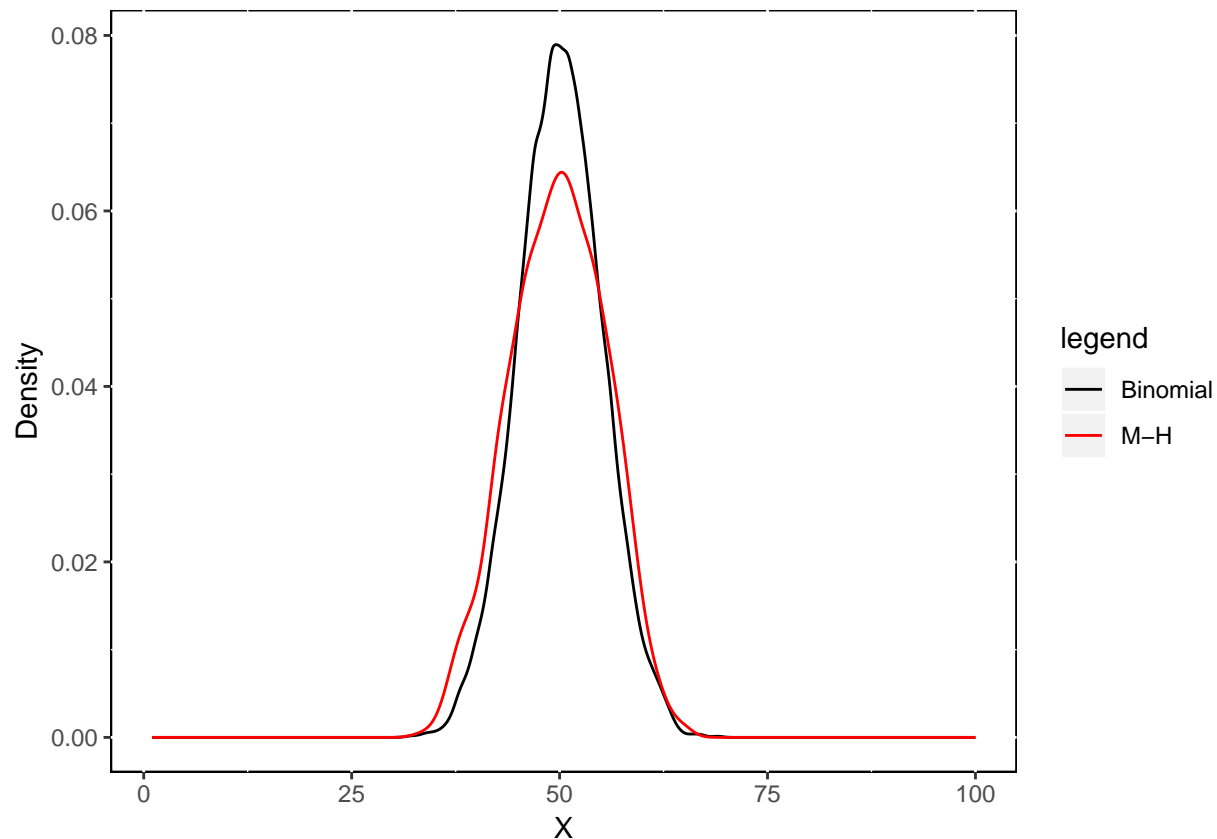
X <- numeric() # store samples
count <- 1
x_old <- 0 # initialize the Markov chain
for(i in 1:10000)
{
  x_new <- sample.int(101, 1)-1 # sample a integer from q_0, as a candidate
  u <- runif(1,0,1)
  acc_1 <- min((f_1(x_new)*q_0(x_old))/(f_1(x_old)*q_0(x_new)), 1)
  # calculate the acceptance prob
  if(u < acc_1) # accept
  {
    X[count] <- x_new
    count <- count +1
    x_old <- x_new
  }
  else # reject
  {
    x_old <- x_old
  }
}
par(mfrow = c(2, 2))
hist(X, main = "Sample", xlab = "X")
XX <- rbinom(10000, 100, 0.5)
hist(XX, main = "Binomial", xlab = "X")
P <- density(X, n = 10000, from = 1, to = 100)
plot(P, main = "Sample")
B <- density(XX, n = 10000, from = 1, to = 100)
plot(B, main = "Binomial")

p <- 0.5
real <- rbinom(10000, 100, p)
d <- density(real, n = 10000, from = 1, to = 100)
s <- density(X, n = 10000, from = 1, to = 100)
Data_real <- data.frame(num = d$x, y = d$y, legend = rep(c("Binomial "), 10000, 1))
Data_sample <- data.frame(num = s$x, y = s$y, legend = rep(c("M-H"), 10000, 1))
Data <- rbind(Data_real, Data_sample)
library(ggplot2)

```



```
g <- ggplot( Data, aes(x = num, y = y, group = legend) ) +
  geom_line( aes(color = legend), size = 0.5 ) +
  xlab("X") + ylab("Density") +
  scale_color_manual(values = c("black", "red")) +
  theme(panel.background = element_rect(fill = 'white', colour = 'black'))
g
```



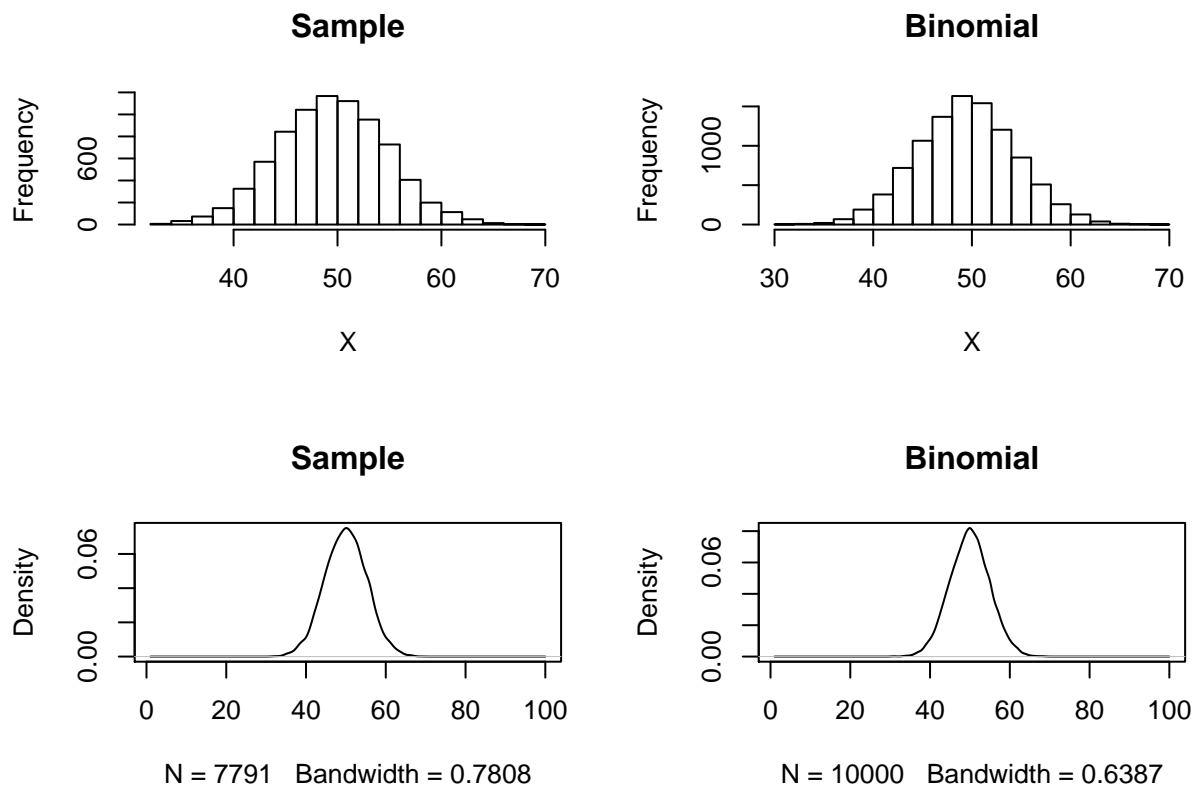
#### 4.Proposal — poisson lamdba=50

```
lambda <- 50
f_1 <- function(x){
  (factorial(100)/(factorial(x)*factorial(100-x)))*0.5^(x)*(1-0.5)^(100-x)
}
# define the Binomial distribution , n=100, p=0.5
q_1 <- function(x){(lambda^x)*exp(-lambda)/factorial(x)}
# define the proposal function q_1 (Poisson), lambda=np=50
n <- 100
X <- numeric() # store samples
count <- 1
x_old <- 0 # initialize the Markov chain
for(i in 1:10000)
{
  x_new <- rpois(1, lambda) # sample a candidate from q_1
  u <- runif(1,0,1)
  acc_1 <- min((f_1(x_new)*q_1(x_old))/(f_1(x_old)*q_1(x_new)), 1)
  # calculate the acceptance prob
  if(u < acc_1) # accept
  {
    X[count] <- x_new
    count <- count +1
    x_old <- x_new
  }
}
```

```

}
else # reject
{
  x_old <- x_old
}
}
par(mfrow = c(2, 2))
hist(X, main = "Sample", bty = "o", xlab = "X")
XX <- rbinom(10000, 100, 0.5)
hist(XX, main = "Binomial", xlab = "X")
P <- density(X, n = 10000, from = 1, to = 100)
plot(P, main = "Sample")
B <- density(XX, n = 10000, from = 1, to = 100)
plot(B, main = "Binomial")

```



```

p <- 0.5
real <- rbinom(10000, 100, p)
d <- density(real, n = 10000, from = 1, to = 100)
s <- density(X, n = 10000, from = 1, to = 100)
Data_real <- data.frame(num = d$x, y = d$y, legend = rep(c("Binomial "), 10000, 1))
# data of Binomial distribution
Data_sample <- data.frame(num = s$x, y = s$y, legend = rep(c("M-H Sample"), 10000, 1))
# data of sample
Data <- rbind(Data_real, Data_sample)
library(ggplot2) # use "ggplot2" package
g <- ggplot( Data, aes(x = num, y = y, group = legend) ) +

```

```
geom_line( aes(color = legend), size = 0.5 ) +
xlab("X") + ylab("Density") +
scale_color_manual(values = c("black", "red")) +
theme(panel.background = element_rect(fill = 'white', colour = 'black'))
g
```

