STAT 206 Final: Due Monday, December 9, 5PM

Xin Feng(Vanessa)

General instructions: Final must be completed as a pdf file. Be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. **The final exam is open book/internet access, but absolutely no communicating with other humans.** Any questions you have must be directed to jflegal@ucr.edu.

Office hours via Zoom (Meeting ID: 951-827-2247) will be as follows:

- December 5, Thursday 9:00 a.m. - 10:00 a.m.
- December 6, Friday 1:00 p.m. - 2:00 p.m.
- December 9, Monday 9:00 a.m. - 10:00 a.m.

Part I - Rescaled Epanechnikov kernel

The rescaled Epanechnikov kernel is a symmetric density function given by

$$f(x) = \begin{cases} \frac{3}{4}(1-x^2) & \text{for } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

1. Check that the above formula is indeed a density function. When $|x| \leq 1$,

$$\begin{aligned} F(x) &= \int_{-1}^{1} \frac{3}{4}(1-x^2)dx \\ &= (\frac{3}{4}x - \frac{1}{4}x^3 + C)|_{-1}^{1} \\ &= (\frac{3}{4} - \frac{1}{4} + \frac{3}{4} - \frac{1}{4}) \\ &= 1 \end{aligned} \tag{2}$$

   Thus, this formular is a density function.

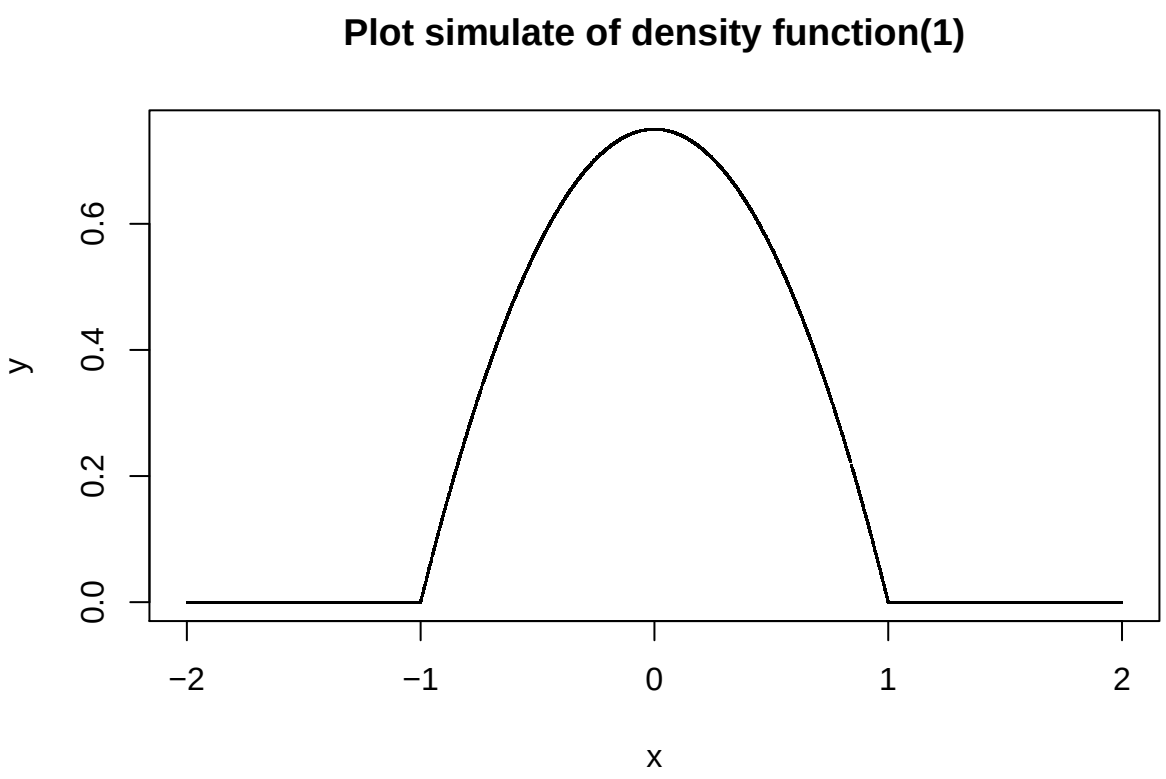
```
N <- 10000
x <- runif(N, min=-1, max=1)
func <- function(x) ifelse(x>=-1 & x<=1, 3/4*(1-x^2), 0)
result <- integrate(func, -1, 1)
cat("The result of integrate is", result$value, "with absolute error ", result$abs.error, "\n")
```

```
## The result of integrate is 1 with absolute error  1.110223e-14
```

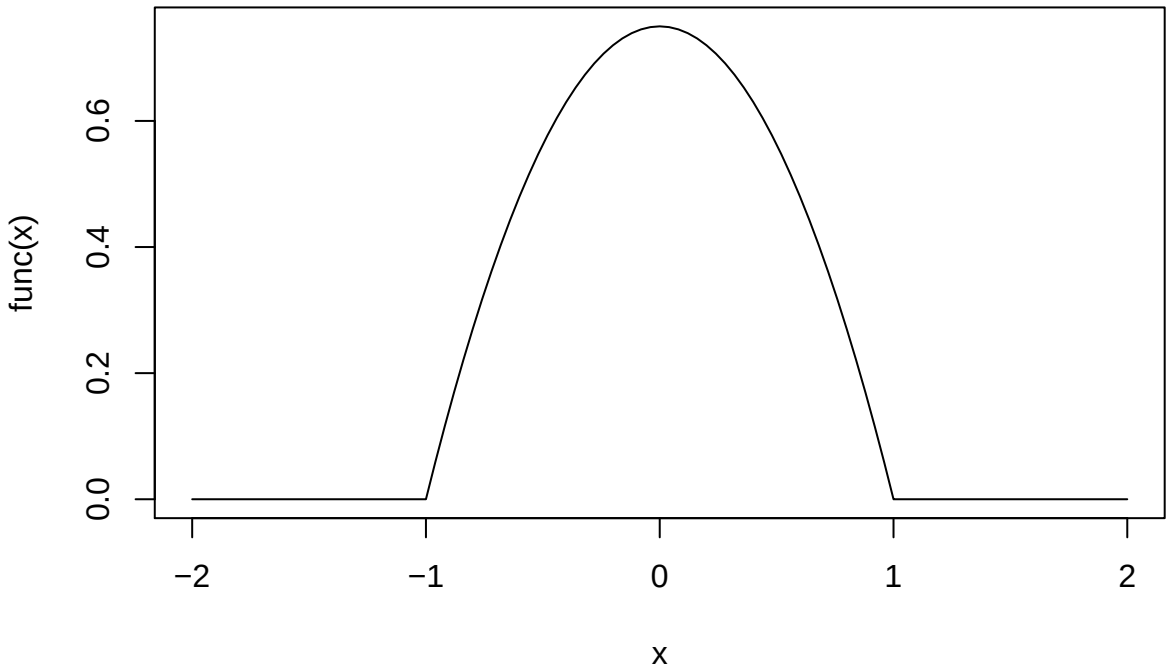
```
# Integrate of the formular equal to 1.
# It is a density function.
```

2. Produce a plot of this density function. Set the X-axis limits to be -2 and 2. Label your axes properly and give a title to your plot.

```r
# Method I
x <- runif(N, min=-2, max=2)
y <- func(x)
plot(x, y, main="Plot simulate of density function(1)", pch=".", xlab="x", xlim=c(-2,2))
```

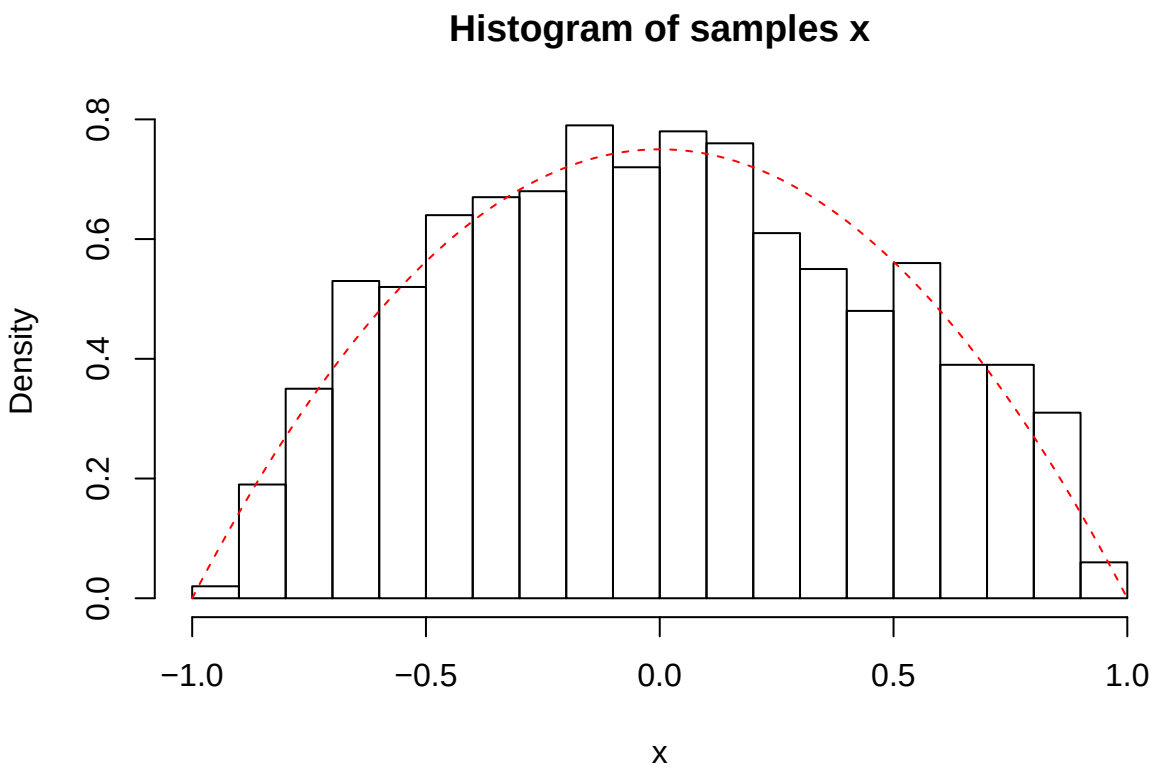



```r
# Method II
curve(func, main="Curve of density function(1)", xlab="x", xlim=c(-2,2))
```

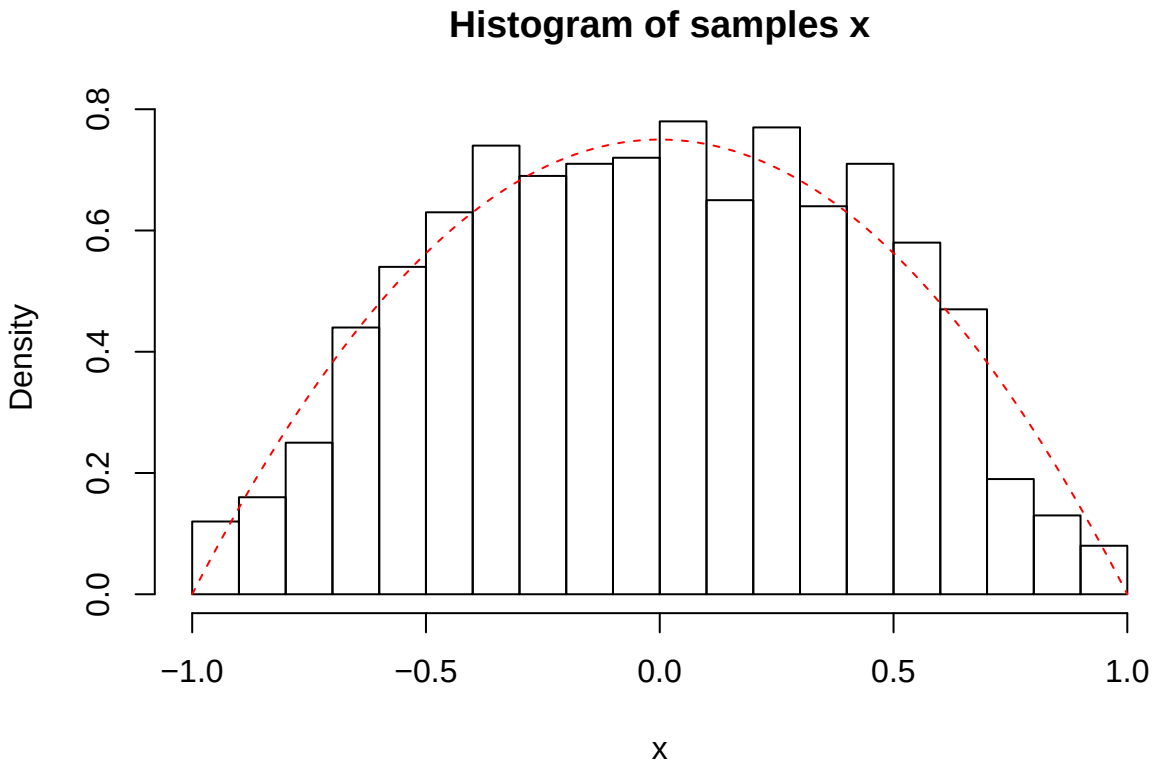


3. Devroye and Gy"orfi give the following algorithm for simulation from this distribution. Generate iid random variables $U_1, U_2, U_3 \sim U(-1, 1)$. If $|U_3| \geq |U_2|$ and $|U_3| \geq |U_1|$, deliver U_2, otherwise deliver U_3. Write a program that implements this algorithm in R. Using your program, generate 1000 values from this distribution. Display a histogram of these values.

```r
N <- 1000

# Method I
U_prime <- matrix(runif(3*N, -1, 1), ncol=3)
U <- apply(U_prime, 1, median)
hist(U, probability = T, nclass=20, xlab="x", main="Histogram of samples x")
curve(func, xlim=c(-1,1), col=2, lty=2, add=T)
```




```
# Method II
U_prime2 <- matrix(runif(3*N, 0, 1), ncol=3)
func_u <- function(x) x <- ifelse(x[3]>=x[2] & x[3]>=x[1], x[2], x[3])
U <- apply(U_prime2, 1, func_u)
U <- c(sample(U, 1/2*N), -sample(U, 1/2*N))
hist(U, probability = T, nclass=20, xlab="x", main="Histogram of samples x")
curve(func, xlim=c(-1,1), col=2, lty=2, add=T)
```



Theory of Method II: Consider the part of $0 \leq p \leq 1$, the sample x should satisfy $x \leq p$.
$U1, U2, U3 \sim unif(0,1)$, choose $x$from U1, U2, U3.
Thus, the probability of x satisfy $0 \leq x \leq p$ is p.
And x follow binomial distribution. The binomial probability as follows:

i) U1, U2, U3 all can be x: $C_3^3 p^3$

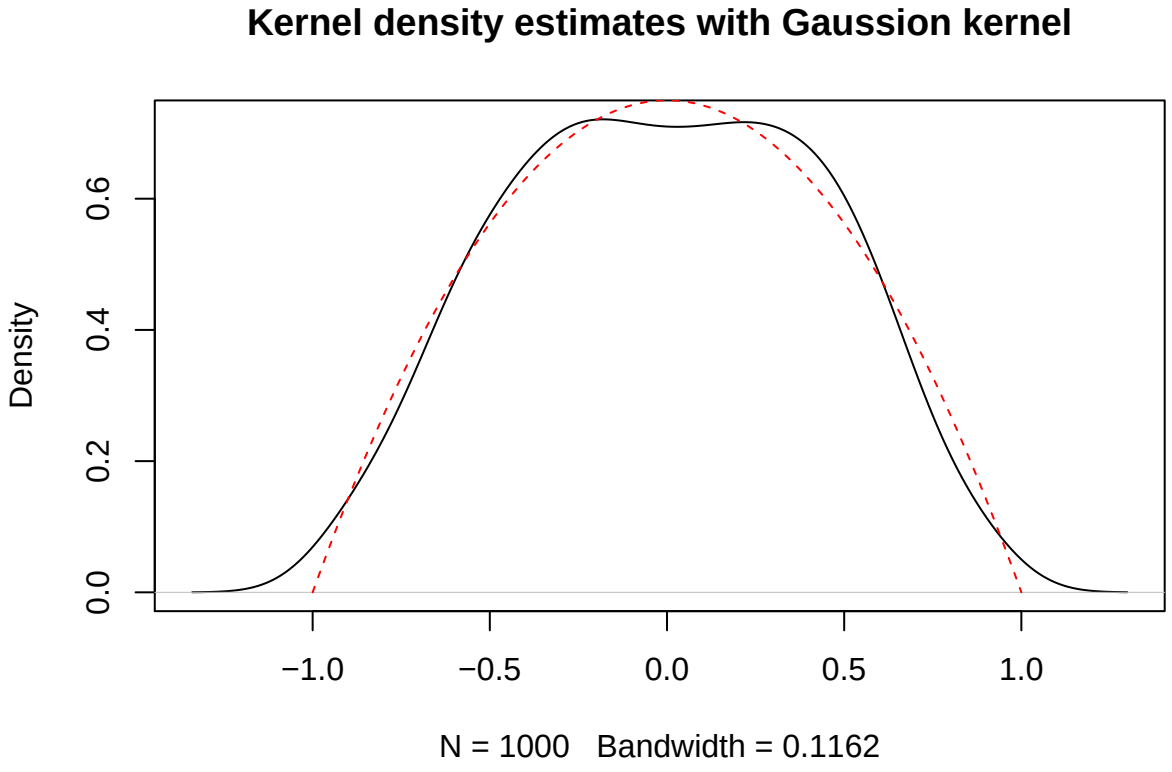
ii) Two of Ui can be x: $C_3^2 p^2(1-p)$

iii) One of Ui can be x: $C_3^1 p(1-p)^2$

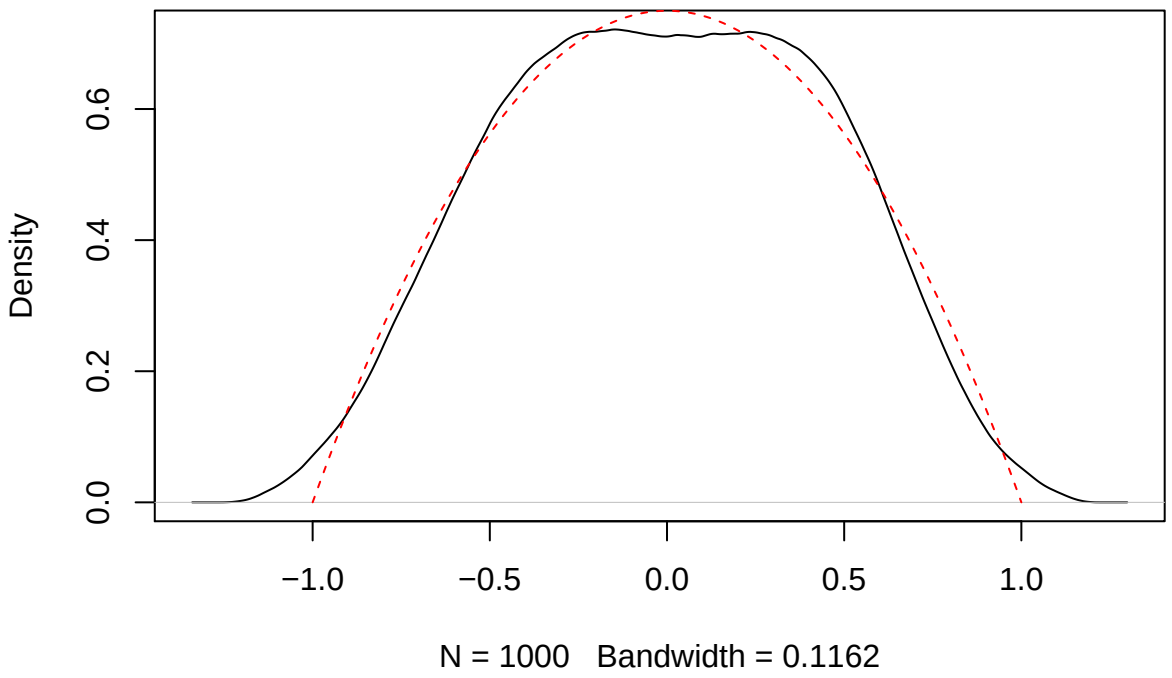
$F = C_3^3 p^3 + C_3^2 p^2(1-p) + C_3^1 p(1-p)^2 = \frac{3}{2}t - \frac{1}{2}t^3$
$f(x) = \frac{1}{2}F\prime = \frac{3}{4}(1-t^2)$

4. Construct kernel density estimates from your 1000 generated values using the Gaussian and Epanechnikov kernels. How do these compare to the true density?

```
y <- density(U, bw='SJ', kernel="gaussian")
plot(y, main="Kernel density estimates with Gaussion kernel")
curve(func, xlim=c(-1,1), col=2, lty=2, add=T)
```




```
y <- density(U, bw='SJ', kernel="epanechnikov")
plot(y, main="Kernel density estimates with Epanechnikov kernel")
curve(func, xlim=c(-1,1), col=2, lty=2, add=T)
```



```
# The kernel density estimates using Gaussian and Epanechnikov kernels are
# both close to the true density. Both kernels during the estimates are similar.
```

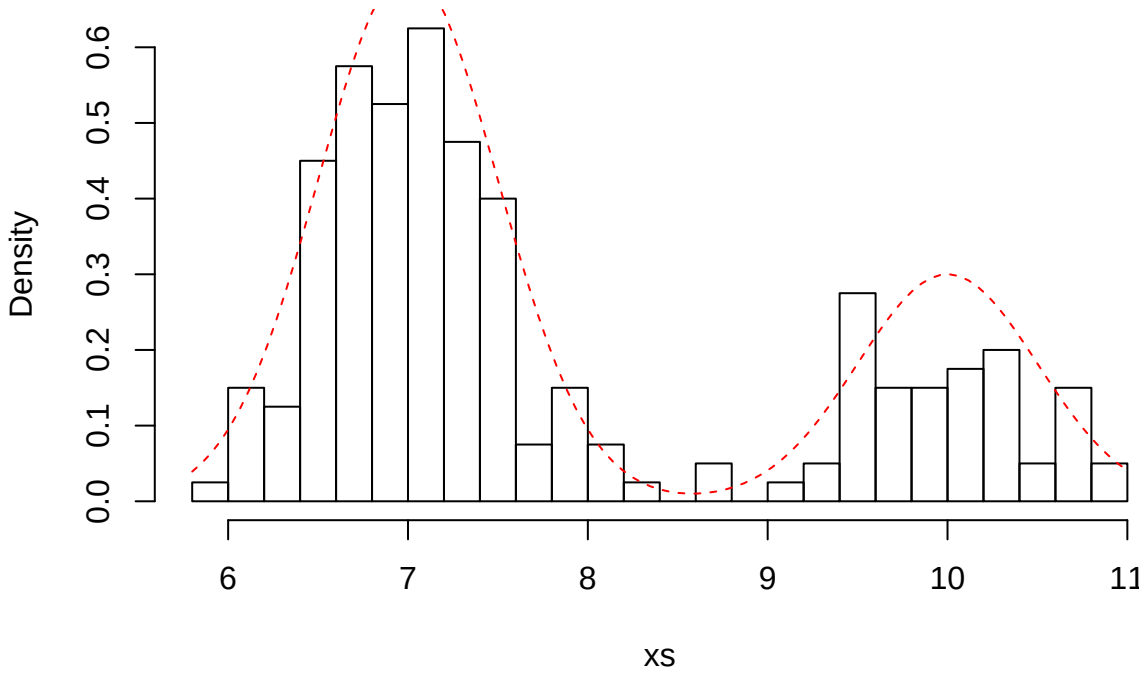
Part II - Metropolis Hastings

Suppose we have observed data $y_1, y_2, \ldots, y_{200}$ sampled independently and identically distributed from the mixture distribution

$$\delta N(7, 0.5^2) + (1 - \delta)N(10, 0.5^2).$$

5. Simulate 200 realizations from the mixture distribution above with $\delta = 0.7$. Method I: Rejection sampling

```
# Rejection sampling
N=200
C=1/(sqrt(2*pi)*0.5)
x <- runif(N)
func_n <- function(x, d=0.7, a1=7, a2=10, b=0.5){
  nm.func <- function(x, a, b) exp(-(x-a)^2/(2*b^2))
  d*nm.func(x,a1,b)+(1-d)*nm.func(x,a2,b)
}

# Method I
i <- 0
xs <- c()
while(i < N){
  x1 <- runif(1, 5, 12)
  y1 <- runif(1)
  if(y1<C*func_n(x1)){
    xs<-c(xs, x1)
    i <- i+1
  }
}
hist(xs, probability=TRUE, 30, main="Histogram of MH samples")
curve(func_n(x), add=T, lty=2, col=2)
```



Method II: Random walk MH sampler with $h \sim N(0, \sigma^2)(\sigma = 1)$

```
# Method II: random walk MH sampler
N <- 200
rw.chain <- function(x=8, n, delta=0.7, mu1=7, mu2=10, sigma=0.5) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- x[i-1]+rnorm(1, sd=1)
    u <- func_n(x.prime)/func_n(x[i-1])
    x[i] <- ifelse(runif(1) < u && x.prime > 0, x.prime, x[i-1])
  }
  return(x)
}
data1 <- rw.chain(n=N)
```

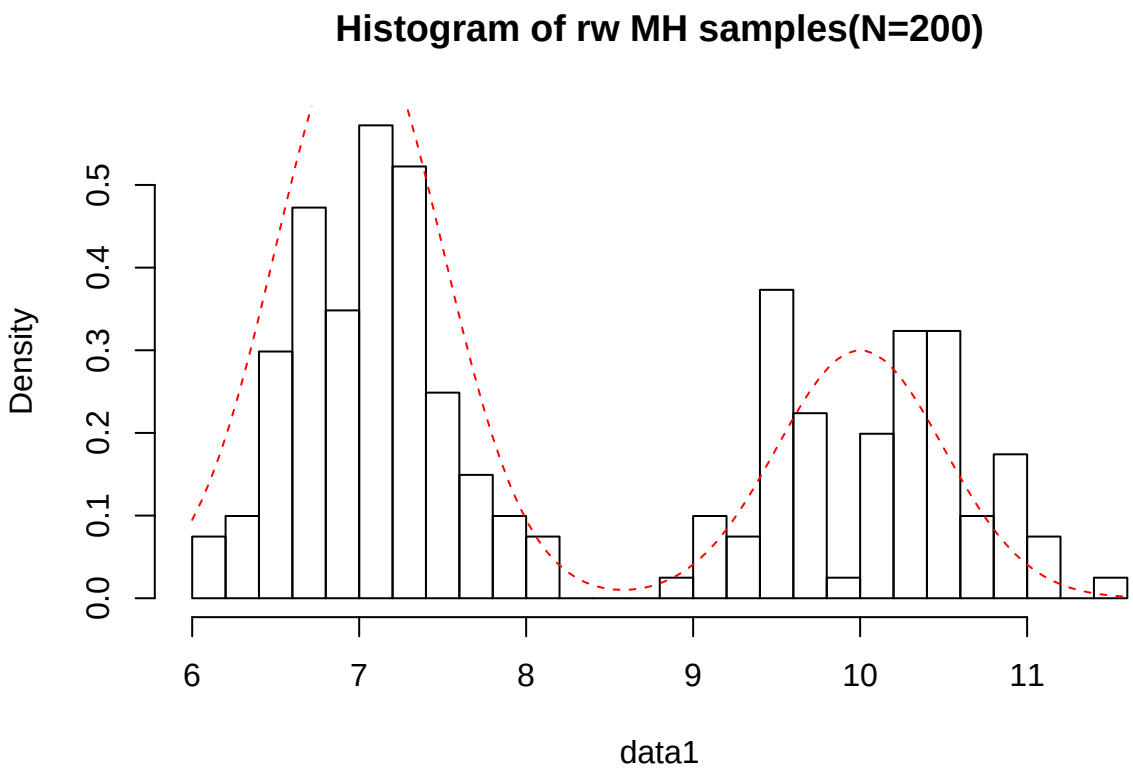
Theory of MH samples:

Because $x^* \sim Normal(0, \sigma^2)$ is symmetric zero mean random variables,

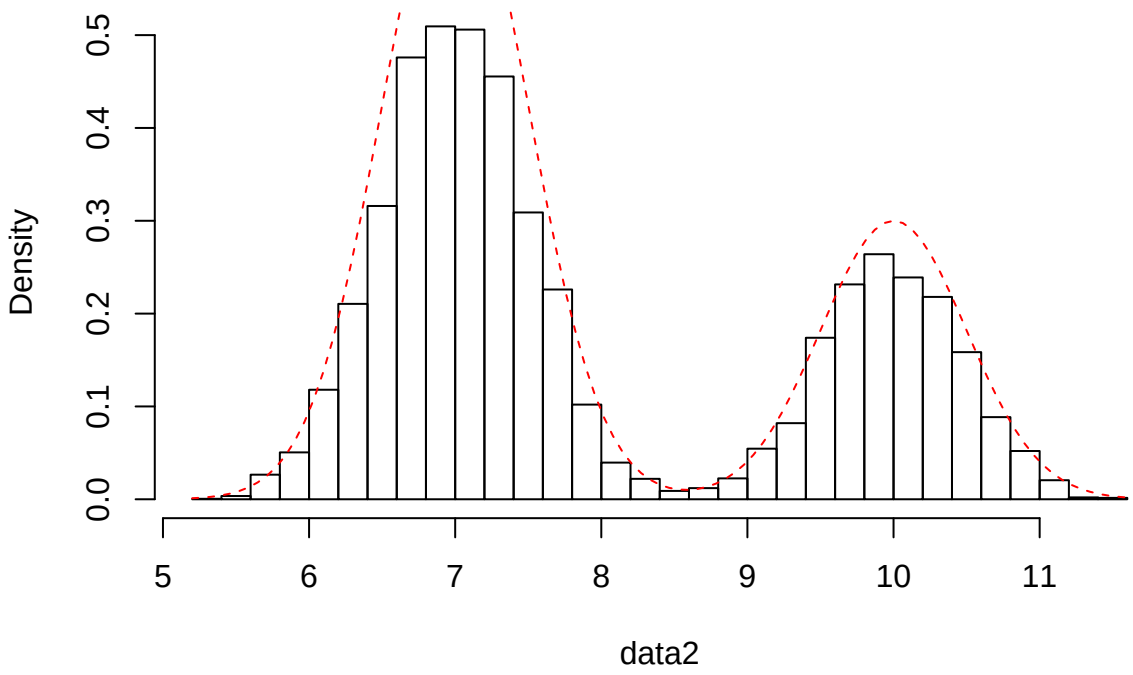
$$
\begin{aligned}
R(x_t, X^*) &= \frac{f(x^*)}{f(x_t)} \\
&= \frac{\frac{\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x^*-7)^2}{2*0.5^2}} + \frac{1-\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x^*-10)^2}{2*0.5^2}}}{\frac{\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x_t-7)^2}{2*0.5^2}} + \frac{1-\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x_t-10)^2}{2*0.5^2}}} \\
&= \frac{\delta e^{-\frac{(x^*-7)^2}{2*0.5^2}} + (1-\delta) e^{-\frac{(x^*-10)^2}{2*0.5^2}}}{\delta e^{-\frac{(x_t-7)^2}{2*0.5^2}} + (1-\delta) e^{-\frac{(x_t-10)^2}{2*0.5^2}}}
\end{aligned}
\tag{3}
$$

6. Draw a histogram of the data that also includes the true density. How close is the histogram to the true density?

```
hist(data1, probability=TRUE, 30, main="Histogram of rw MH samples(N=200)")
curve(func_n(x), add=T, lty=2, col=2)
```



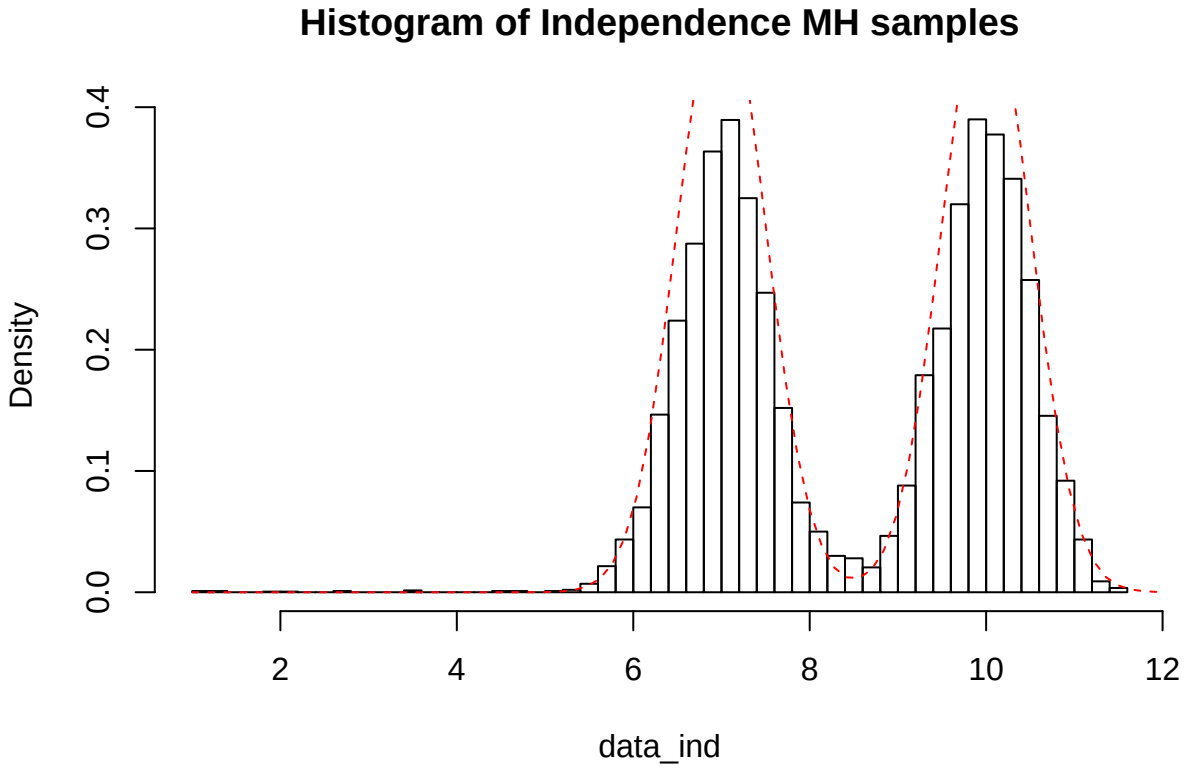

```
# Due to insufficient data damples, the histogram is barely follow the true density.

# When N=1e+4,
N <- 1e+4
data2 <- rw.chain(n=N)
hist(data2, probability=TRUE, 30, main="Histogram of rw MH samples(N=10000)")
curve(func_n(x), add=T, lty=2, col=2)
```




```
# Now the simulation seems further closer to the true density
```

7. Now assume δ is unknown with a Uniform(0,1) prior distribution for δ. Implement an independence Metropolis Hastings sampler with a Uniform(0,1) proposal.

```
ind.chain <- function(x=1, n){
  # if theta=1, then this is an iid sampler.
  m<- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- x[i-1]+runif(1, -1, 1)
    d <- runif(1)
    Rt <- func_n(x.prime, d=d)/func_n(x[i-1], d=d)
    x[i] <- ifelse(runif(1) < Rt, x.prime, x[i-1])
  }
  return(x)
}
data_ind <- ind.chain(n=10000)
hist(data_ind, probability = TRUE, 40, main="Histogram of Independence MH samples")
curve(func_n(x, d=0.5), add=T, xlim=c(1,15), lty=2, col=2)
```



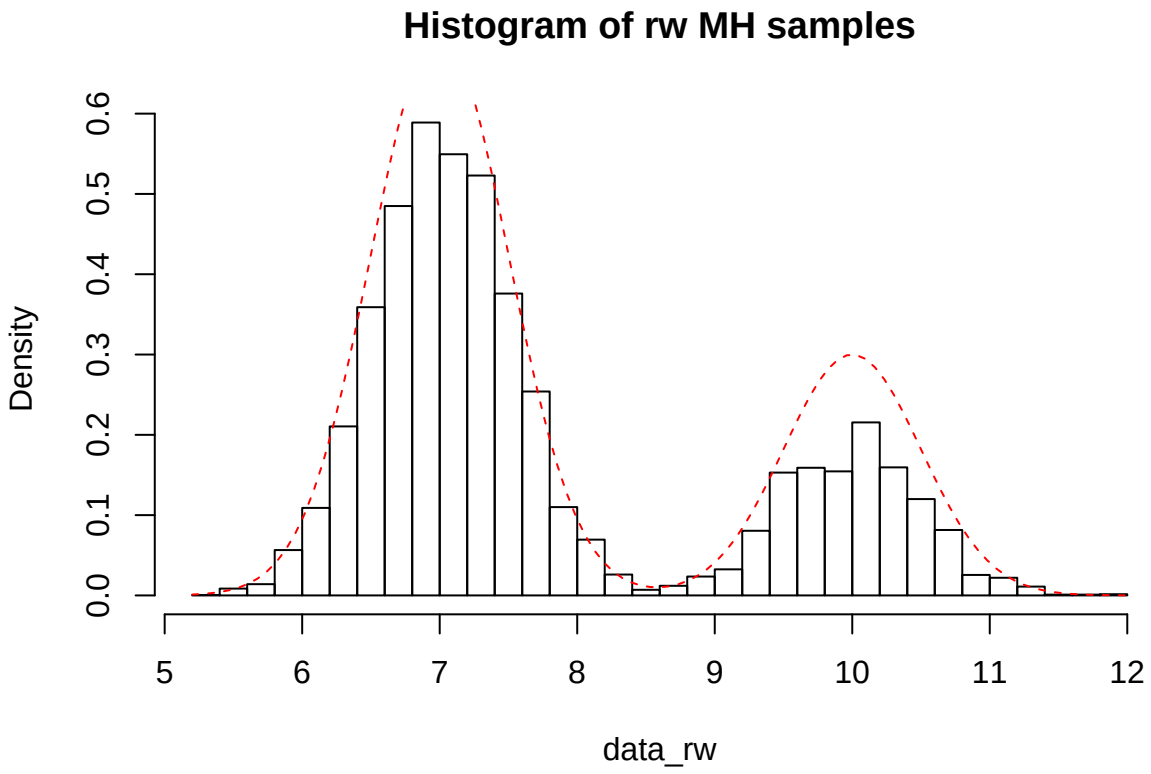
For normal distribution, 99.7% of the data are within $(\mu - 3\sigma, \mu + 3\sigma)$. That's we expect that most data to be concentrated in $(7 - 3 * 0.5, 10 + 3 * 0.5) = (5.5, 11.5)$. During the process of creating each sample, regenerate delta by sampling from uniform distribution. The samples will follow the distribution with $\delta = 0.5$, which is the expectation of $\delta \sim uniform(0, 1)$.

$$
\begin{aligned}
proposal &\sim Uniform(a, b) \\
g(x) = g(x^*) = g(x_t) &= \frac{1}{b-a} \\
R(x_t, X^*) &= \frac{f(x^*) * g(x_t)}{f(x_t) * g(x^*)} \\
&= \frac{\frac{\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x^*-7)^2}{2*0.5^2}} + \frac{1-\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x^*-10)^2}{2*0.5^2}}}{\frac{\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x_t-7)^2}{2*0.5^2}} + \frac{1-\delta}{\sqrt{2\pi}0.5} e^{-\frac{(x_t-10)^2}{2*0.5^2}}} \\
&= \frac{\delta e^{-\frac{(x^*-7)^2}{2*0.5^2}} + (1-\delta) e^{-\frac{(x^*-10)^2}{2*0.5^2}}}{\delta e^{-\frac{(x_t-7)^2}{2*0.5^2}} + (1-\delta) e^{-\frac{(x_t-10)^2}{2*0.5^2}}}
\end{aligned}
\tag{4}
$$

8. Implement a random walk Metropolis Hastings sampler where the proposal $\delta* = \delta^{(t)} + \epsilon$ with $\epsilon \sim$ Uniform$(-1, 1)$.

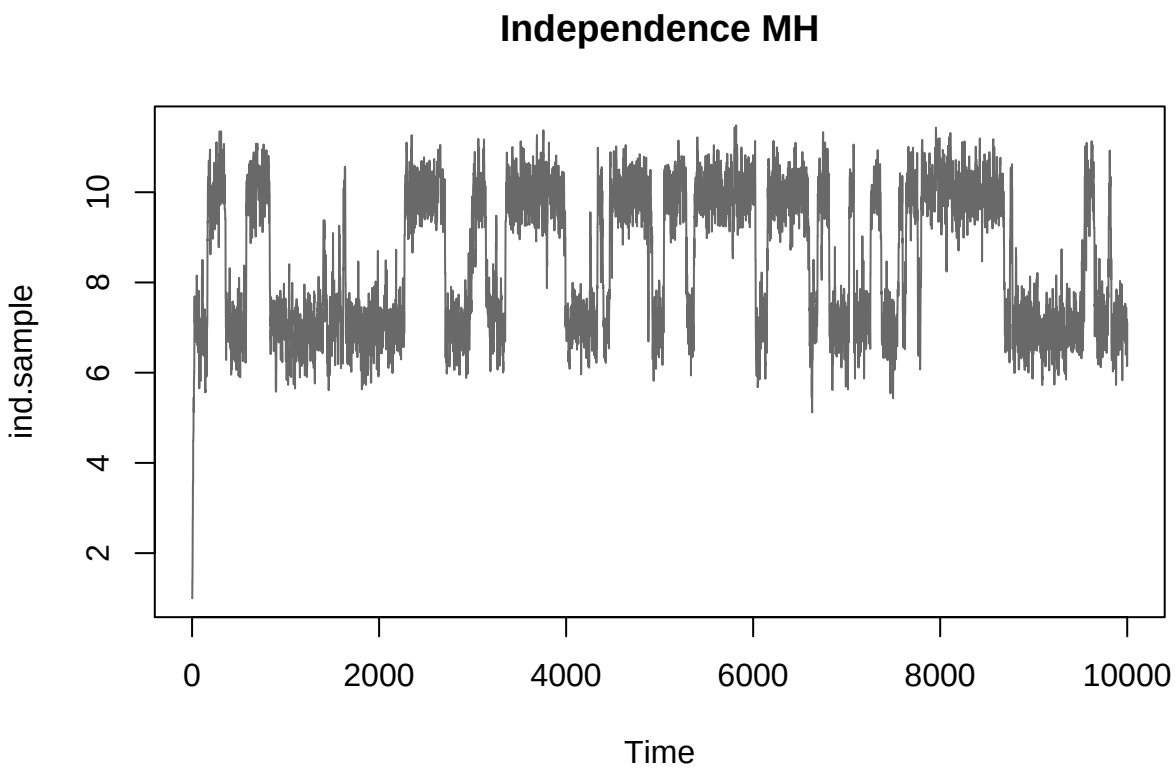
```
rw.chain.dt <- function(x=8, n) {
  m <- length(x)
  x <- append(x, double(n))
  for(i in (m+1):length(x)){
    x.prime <- x[i-1]+runif(1)
    d <- d + runif(1, -1, 1)
    u <- func_n(x.prime)/func_n(x[i-1])
    x[i] <- ifelse(runif(1) < u && x.prime > 0, x.prime, x[i-1])
```

```
  }
  return(x)
}
data_rw <- rw.chain(n=10000)
hist(data_rw, probability=TRUE, 40, main="Histogram of rw MH samples")
curve(func_n(x), add=T, lty=2, col=2)
```

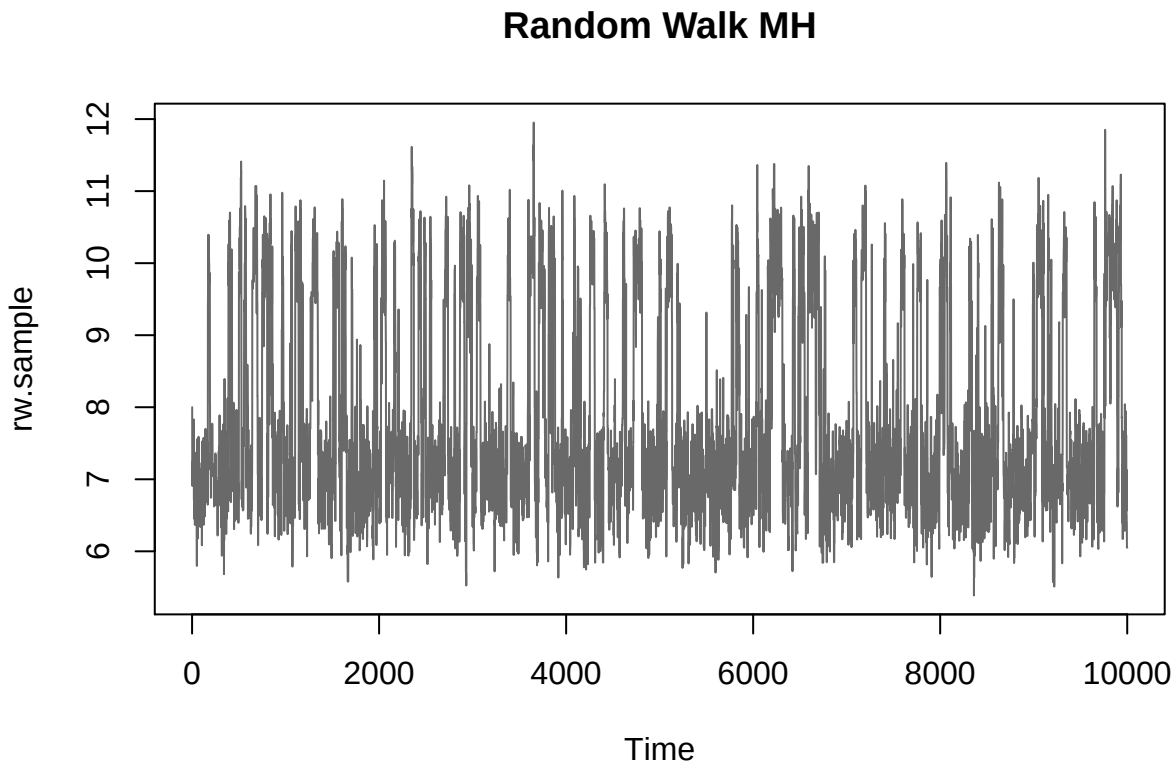


9. Comment on the performance of the independence and random walk Metropolis Hastings samplers including at least one relevant plot.

```
plot(data_ind, type="l", xlab="Time", ylab="ind.sample", main="Independence MH", col="dimgrey")
```




```r
plot(data_rw, type="l", xlab="Time", ylab="rw.sample", main="Random Walk MH", col="dimgrey")
```




```r
# According to the plot above, the second one has more random samples.
# Thus, a random walk MH sampler is better.
```

Part III - Maximum Likelihood

Suppose $X_1, \ldots, X_n$ is a random sample from the following density

$$f(X|s) = \sqrt{\frac{s}{2\pi}} \frac{\exp\{-s/(2X)\}}{X^{3/2}} I(X > 0),$$

where s is an unknown parameter.

10. What is the log-likelihood function?

$$\begin{aligned} l(x|s) &= \prod_{i=1}^{n} f(X|s) \\ &= (\frac{s}{2\pi})^{\frac{n}{2}} \prod_{i=1}^{n} x_i^{-\frac{3n}{2}} e^{\sum_{i=1}^{n} \frac{s}{2x_i}} \end{aligned} \tag{5}$$

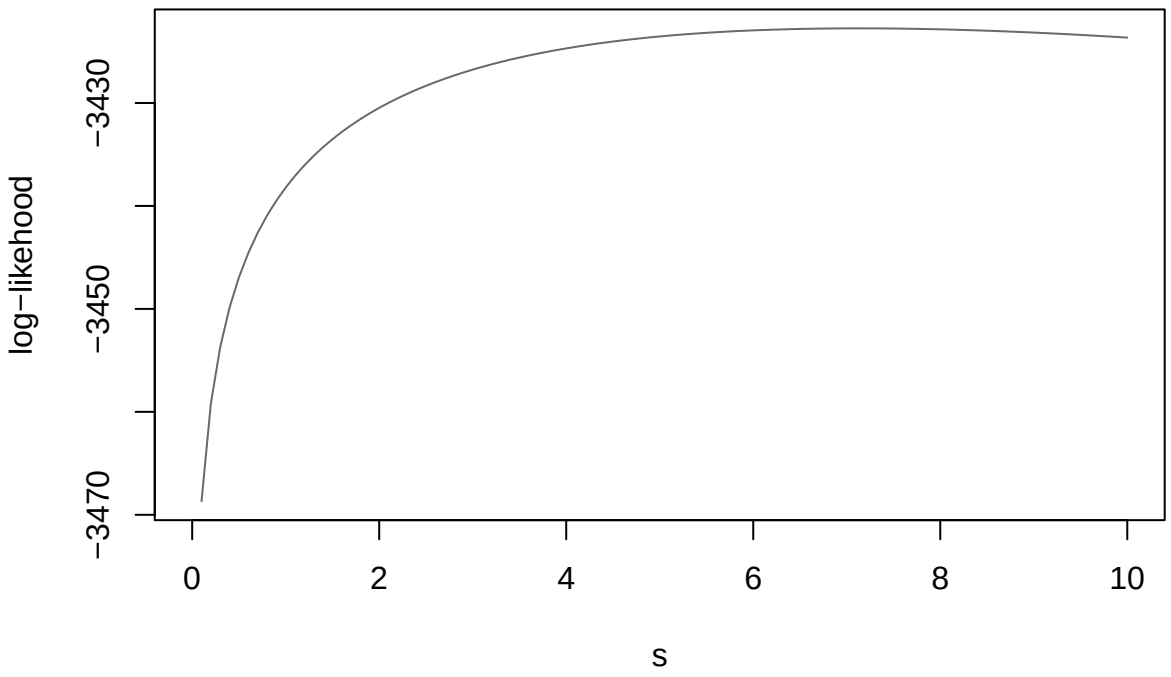
$$\begin{aligned} L(x|s) &= log(l(x|s)) \\ &= \frac{n}{2} log(\frac{s}{2\pi}) - \frac{3n}{2} \sum_{i=1}^{n} log x_i - \frac{s}{2} \sum_{i=1}^{n} \frac{1}{x_i} \end{aligned} \tag{6}$$

11. Suppose we want to numerically calculate the MLE of s using Newton's method. Write a general program to do this. Run your program using the dataset available at [http://faculty.ucr.edu/~jflegal/206/finaldata.txt] starting from .2, 1, 5, and 10.

```
# Newton's Method
# Newton's method is an iterative method for finding the roots of a
# differentiable function f, which are solutions to the equation f(x) = 0.
# curve of likely-hood
library(numDeriv)

# Log-likelyhood function
func.logsum <- function(s, data)
  length(data)/2*log(s/(2*pi))-3*length(data)/2*sum(log(data))-s/2*sum(1/data)

# Data
finaldata <- read.table("http://faculty.ucr.edu/~jflegal/206/finaldata.txt")
# View(finaldata)
summary(finaldata$V2)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##   0.9103   6.3647  17.3175  46.2591  39.6485 259.5928
```

```
X <- finaldata$V2

# Curve of log-likelihood function
gra.logsum <- function(s, ...)
  return(sapply(s, func.logsum, ...))
curve(gra.logsum(s=x, data=finaldata$V2), from=0, to=10,
      main="Log-likehood function", xlab="s", ylab="log-likehood", col="dimgrey")
```




```
nt.log.lh <- function(data){
  log_lh <- function(s)
    result <- func.logsum(s, data)
return(log_lh)
}

# Estimate s
func.nt <- function(f, s_0, e, N){
  s <- c(s_0)
  for(i in 2:N){
    data_deri <- genD(f, s[i-1])
    para_p <- data_deri[[1]][1]
    para_h <- data_deri[[1]][2]
    s_1 <- s[i-1]-para_p/para_h
    if(is.na(s_1)) next
    s <- c(s,s_1)
    if (abs(s[i]-s[i-1])< e) break
  }
  return(s[i])
}


est_s_0 <- func.nt(nt.log.lh(X),0.2,1e-5,1e+3)
cat("s0=0.2, est_s=", est_s_0, "\n")
```

```
## s0=0.2, est_s= 7.130362
```

```
est_s_1 <- func.nt(nt.log.lh(X),1,1e-5,1e+3)
cat("s0=1, est_s=", est_s_1, "\n")
```

```
## s0=1, est_s= 7.130362
```

```
est_s_5 <- func.nt(nt.log.lh(X),5,1e-5,1e+3)
cat("s0=5, est_s=", est_s_5, "\n")
```

```
## s0=5, est_s= 7.130362
```

```
est_s_10 <- func.nt(nt.log.lh(X),10,1e-5,1e+3)
cat("s0=10, est_s=", est_s_10, "\n")
```

```
## s0=10, est_s= 7.130362
```

12. Let $\hat{s}$ be the MLE obtained above. Construct an approximate 95% confidence interval for s, centered at $\hat{s}$.

```
# Extimate s for different s_0.
s_hat <- est_s_10
s_0 <- seq(from=0.1, to=10, length.out=100)
est_s <- vector("numeric", length=0)
for(i in 1:100){
  est_s[i] <- func.nt(nt.log.lh(X), s_0[i], 1e-5, 1e+3)
}
sd_s <- sd(est_s)
cat("95% confidence interval of s is (", s_hat-sd_s*1.96, ", ", s_hat+sd_s*1.96, ").\n")
```

```
## 95% confidence interval of s is ( 7.130362 ,  7.130362 ).
```

```
# Estimate s by bootstrap.
est_s <- vector("numeric", length=0)
n <- length(X)
for(i in 1:100){
  Xstar <- sample(X, n, replace=TRUE)
  est_s[i] <- func.nt(nt.log.lh(Xstar), s_hat, 1e-5, 1e+3)
}
sd_s <- sd(est_s)
cat("95% confidence interval of s is (", s_hat-sd_s*1.96, ", ", s_hat+sd_s*1.96, ").\n")
```

```
## 95% confidence interval of s is ( 1.78432 ,  12.4764 ).
```