Lab 6 (Bayesian Estimation and MCMC)

Shuyuan Wang, sw3449

11/21/2019

Instructions

Make sure that you upload the PDF (or HTML) output after you have knitted the file. The files you upload to the Canvas page should be updated with commands you provide to answer each of the questions below. You can edit this file directly to produce your final solutions.

Goals

This lab has two goals. The first goal is to use the **Accept-Reject** algorithm to simulate from a mixture of two normals. The second goal is to utilize Bayesian methods and the the famous **Markov Chain Monte Carlo** algorithm to estimate the mixture parameter δ.

Background: (Mixture)

A mixture distribution is the probability distribution of a random variable that is derived from a collection of other random variables (Wiki). In our case we consider a mixture of two normal distributions. Here we assume that our random variable is governed by the probability density $f(x)$, defined by

$$\begin{aligned} f(x) &= f(x;\mu_1,\sigma_1,\mu_2,\sigma_2,\delta) \\ &= \delta f_1(x;\mu_1,\sigma_1) + (1-\delta) f_2(x;\mu_2,\sigma_2) \\ &= \delta \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp -\frac{1}{2\sigma_1^2}(x-\mu_1)^2 + (1-\delta)\frac{1}{\sqrt{2\pi\sigma_2^2}} \exp -\frac{1}{2\sigma_2^2}(x-\mu_2)^2, \end{aligned}$$

where $-\infty < x < \infty$ and the parameter space is defined by $-\infty < \mu_1, \mu_2 < \infty$, $\sigma_1, \sigma_2 > 0$, and $0 \leq \delta \leq 1$. The **mixture parameter** δ governs how much mass gets placed on the first distribution $f(x;\mu_1,\sigma_1)$ and the complement of δ governs how much mass gets placed on the other distribution $f_2(x;\mu_2,\sigma_2)$.

To further motivate this setting, consider simulating $n = 10,000$ heights from the population of both males and females. Assume that males are distributed normal with mean $\mu_1 = 70$[in] and standard deviation $\sigma_1 = 3$[in] and females are distributed normal with mean $\mu_2 = 64$[in] and standard deviation $\sigma_2 = 2.5$[in]. Also assume that each distribution contributes equal mass, i.e., set the mixture parameter to $\delta = .5$. The distribution of males is governed by

$$f_1(x;\mu_1,\sigma_1) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp -\frac{1}{2\sigma_1^2}(x-\mu_1)^2, \quad -\infty < x < \infty,$$

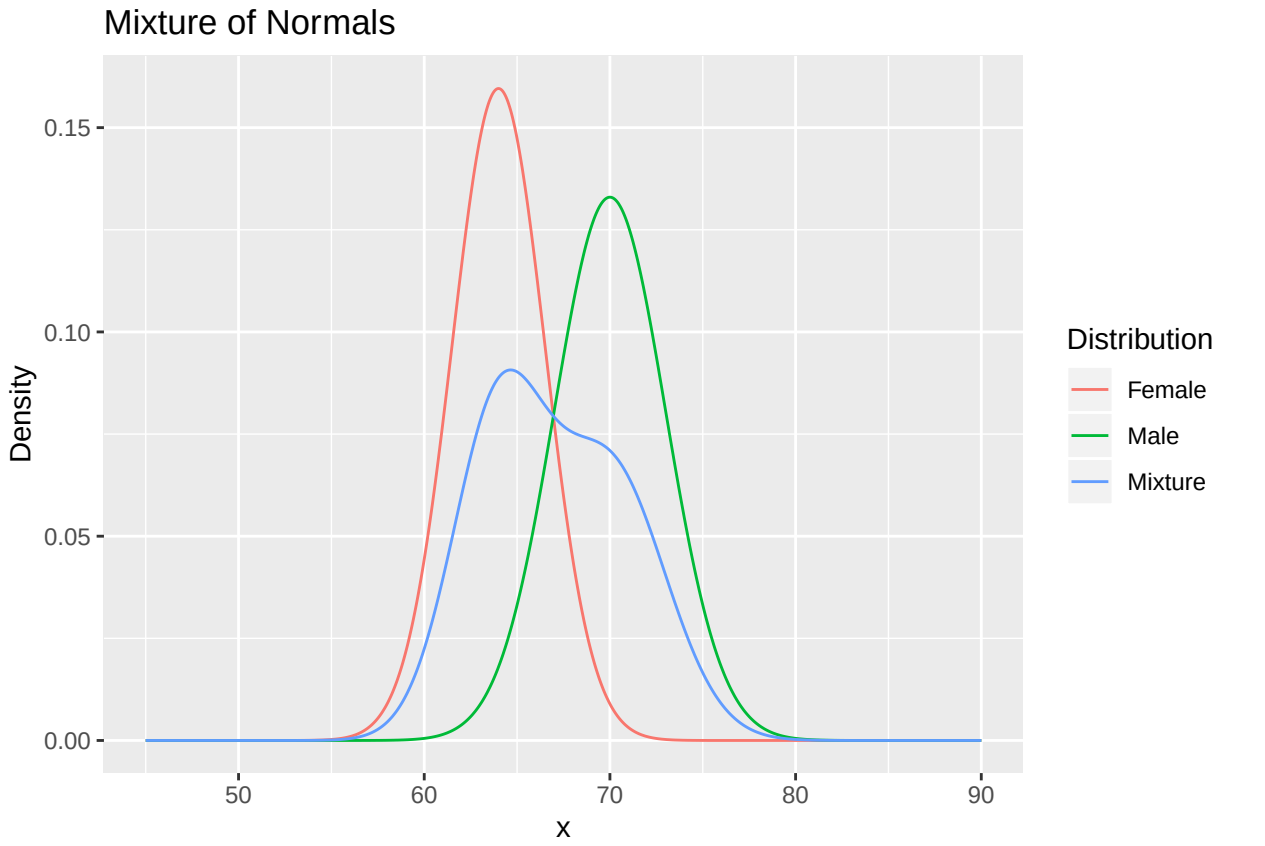
and the distribution of females is governed by

$$f_2(x;\mu_2,\sigma_2) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp -\frac{1}{2\sigma_2^2}(x-\mu_2)^2, \quad -\infty < x < \infty.$$

Below shows the pdf of $f_1(x;\mu_1,\sigma_1)$, $f_2(x;\mu_2,\sigma_2)$ and the mixture $f(x)$ all on the same plot.

```
x <- seq(45,90,by=.05)
n.x <- length(x)
f_1 <- dnorm(x,mean=70,sd=3)
f_2 <- dnorm(x,mean=64,sd=2.5)
f <- function(x) {
```

```
  return(.5*dnorm(x,mean=70,sd=3) + .5*dnorm(x,mean=64,sd=2.5))
  }

plot_df <-  data.frame(x=c(x,x,x),
                       Density=c(f_1,f_2,f(x)),
                       Distribution=c(rep("Male",n.x),rep("Female",n.x),rep("Mixture",n.x))
                       )

library(ggplot2)
ggplot(data = plot_df) +
   geom_line(mapping = aes(x = x, y = Density,color=Distribution))+
   labs(title = "Mixture of Normals")
```



Part I: Simulating a Mixture of Normals

The first goal is to simulate from the mixture distribution

$$\delta f_1(x;\mu_1,\sigma_1) + (1-\delta) f_2(x;\mu_2,\sigma_2),$$

where $\mu_1 = 70, \sigma_1 = 3, \mu_2 = 64, \sigma_2 = 2.5, \delta = 2.5$. We use the accept-reject algorithm to accomplish this task.

First we must choose the "easy to simulate" distribution $g(x)$. For this problem choose $g(x)$ to be a Cauchy distribution centered at 66 with scale parameter 7.

```
g <- function(x) {
  s=7
  l=66
  return(1/(pi*s*(1+((x-l)/s)^2)))
  }
```

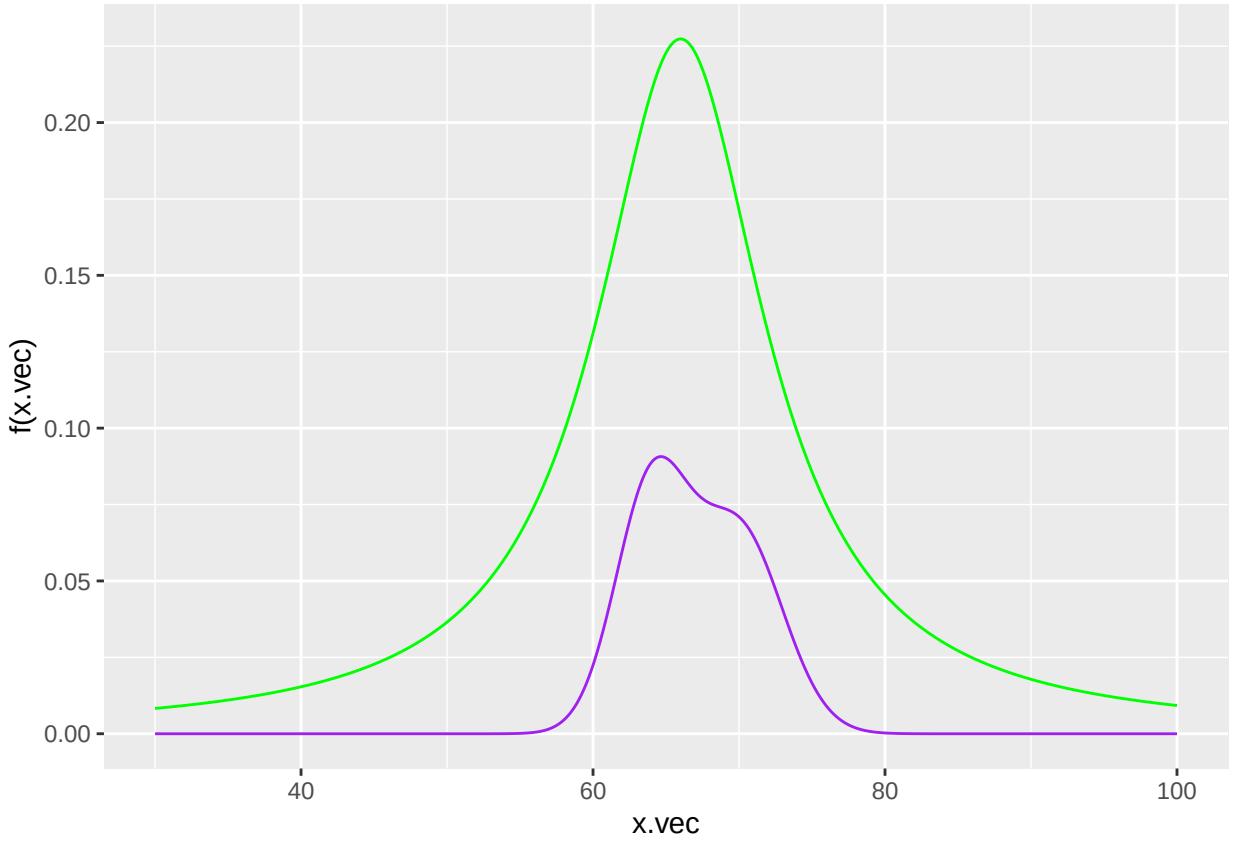
Perform the following tasks

1) Identify a **suitable** value of *alpha* such that your envelope function $e(x)$ satisfies

$$f(x) \leq e(x) = g(x)/\alpha, \quad \text{where} \quad 0 < \alpha < 1.$$

Note that you must choose α so that $e(x)$ is close to $f(x)$. There is not one unique solution to this problem. The below plot shows how $\alpha = .20$ creates an envelope function that is too large. Validate your choice of *alpha* with with a graphic similar to below.

```
# Choose alpha
alpha <- .20

# Define envelope e(x)
e <- function(x) {
  return(g(x)/alpha)
  }

# Plot
x.vec <- seq(30,100,by=.1)
ggplot() +
   geom_line(mapping = aes(x = x.vec, y = f(x.vec)),col="purple")+
   geom_line(mapping = aes(x = x.vec, y = e(x.vec)),col="green")
```

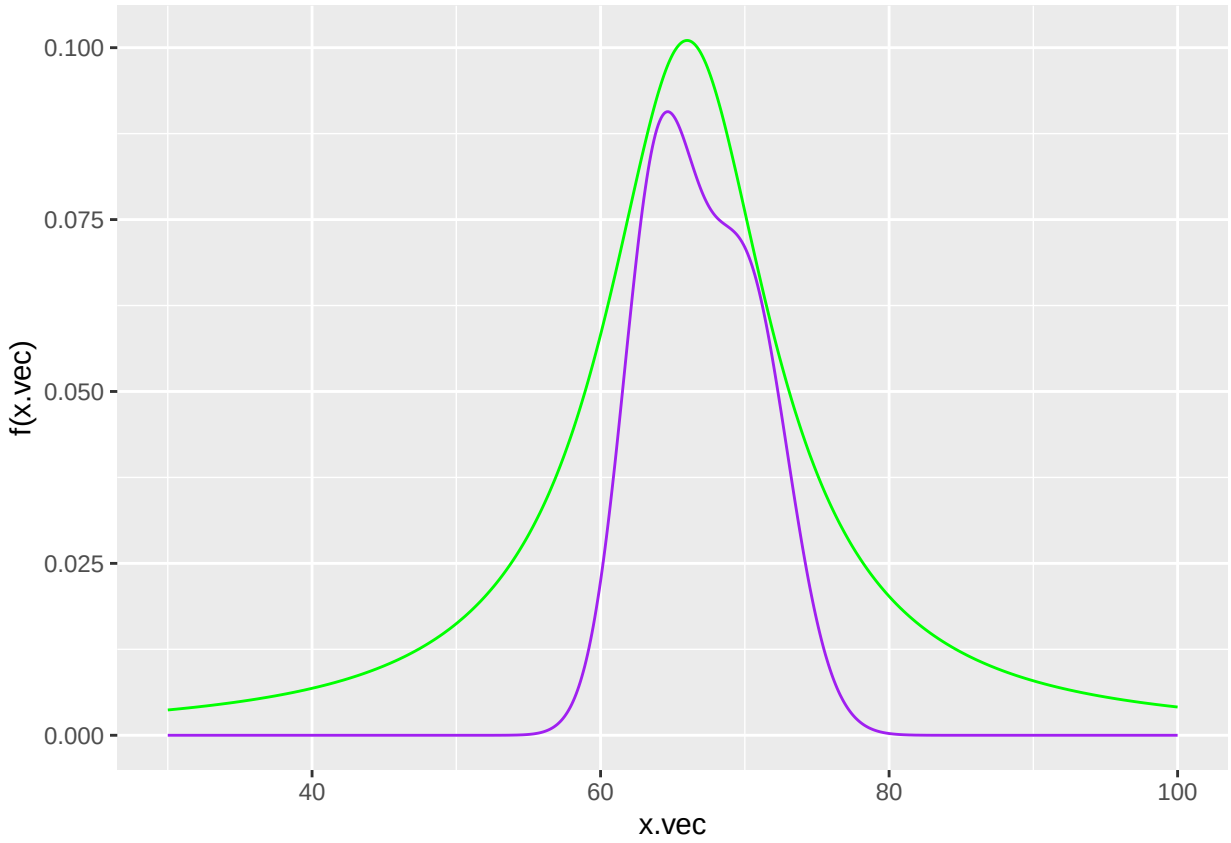



```
# Is g(x)>f(x)?
all(e(x.vec)>f(x.vec))
```

```
## [1] TRUE
```

Solution

```r
## Solution goes here -------
# My chosen alpha is 0.45
alpha <- .45

# Define envelope e(x)
e <- function(x) {
  return(g(x)/alpha)
  }

# Plot
x.vec <- seq(30,100,by=.1)
ggplot() +
   geom_line(mapping = aes(x = x.vec, y = f(x.vec)),col="purple")+
   geom_line(mapping = aes(x = x.vec, y = e(x.vec)),col="green")
```




```r
# Is g(x)>f(x)?
all(e(x.vec)>f(x.vec))
```

```
## [1] TRUE
```

2) Write a function named **r.norm.mix()** that simulates **n.samps** from the normal-mixture $f(x)$. To accomplish this task you will wrap a function around the accept-reject algorithm from the lecture notes. Also include the acceptance rate, i.e., how many times did the algorithm accept a draw compared to the total number of trials performed. Your function should return a list of two elements: (i) the simulated vector mixture and (ii) the proportion of accepted cases. Run your function **r.norm.mix()** to simulate 10,000 cases and display the first 20 values. What's the proportion of accepted cases? Compare this number to your chosen α and comment on the result. The code below should help you get started.

Solution

```
r.norm.mix <- function(n.samps) {
  n <- 0 # counter for number samples accepted
  m <- 0 # counter for number of trials
  samps   <- numeric(n.samps) # initialize the vector of output
  while (n < n.samps) {
    m <- m+1
    y <- rcauchy(1,location=66,scale=7) # random draw from Cauchy
    u <- runif(1)
    if (u<f(y)/e(y)){
      n <- n+1
      samps[n] <- y
    }
  }
return(list(x=samps,alpha.hat=n.samps/m))
}
sim <- r.norm.mix(n.samps=10000)
sim$x[1:20]
```

```
##  [1] 66.88113 71.44985 73.68978 73.27514 73.51811 66.64911 71.72277
##  [8] 65.57313 61.88905 68.67182 67.44386 64.07978 65.73029 63.41316
## [15] 63.94099 57.02918 68.69076 72.87826 59.48070 71.64597
```

```
sim$alpha.hat
```

```
## [1] 0.4452955
```

```
alpha
```

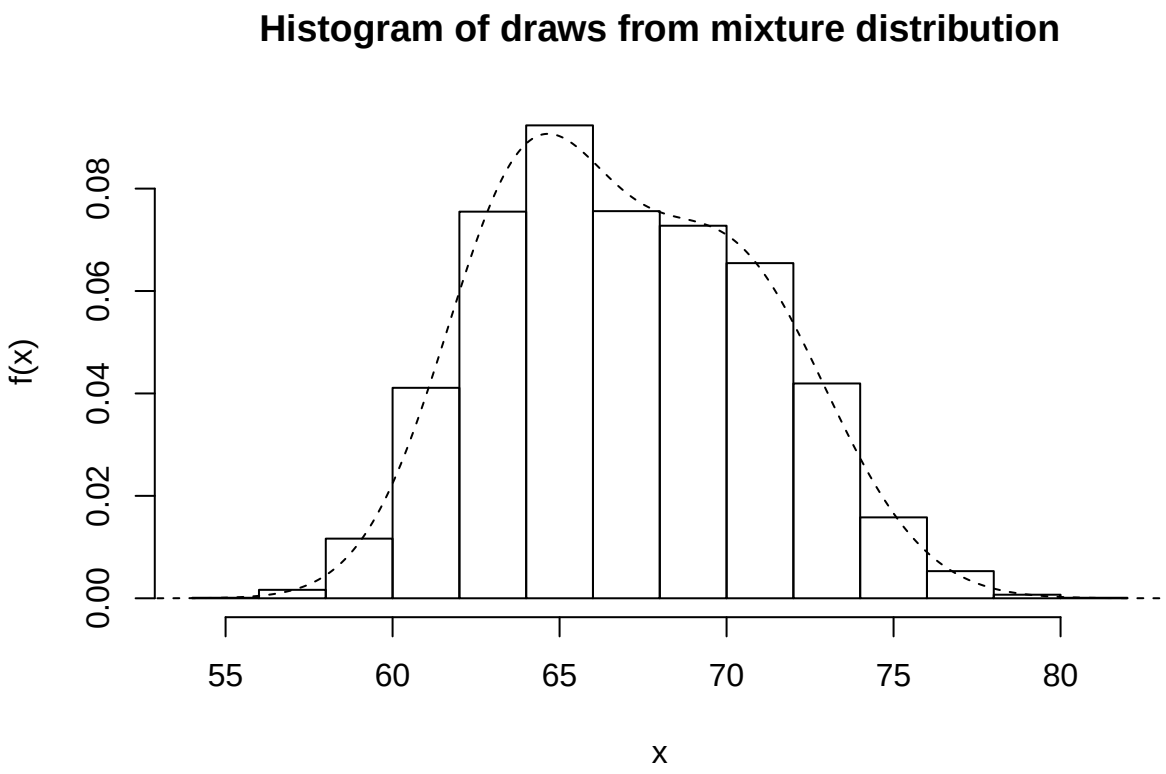
```
## [1] 0.45
```

The proportion of accepted cases is 0.4434 and it's close to my chosen α.

3) Using **ggplot** or **base R**, construct a histogram of the simulated mixture distribution with the true mixture pdf $f(x)$ overlayed on the plot.

Solution

```
## Solution goes here -------
x <- seq(30,100,by=0.01)
hist(sim$x,prob=T,ylab='f(x)',xlab='x',
     main='Histogram of draws from mixture distribution')
lines(x,f(x),lty=2)
```



Part II: Bayesian Statistics and MCMC

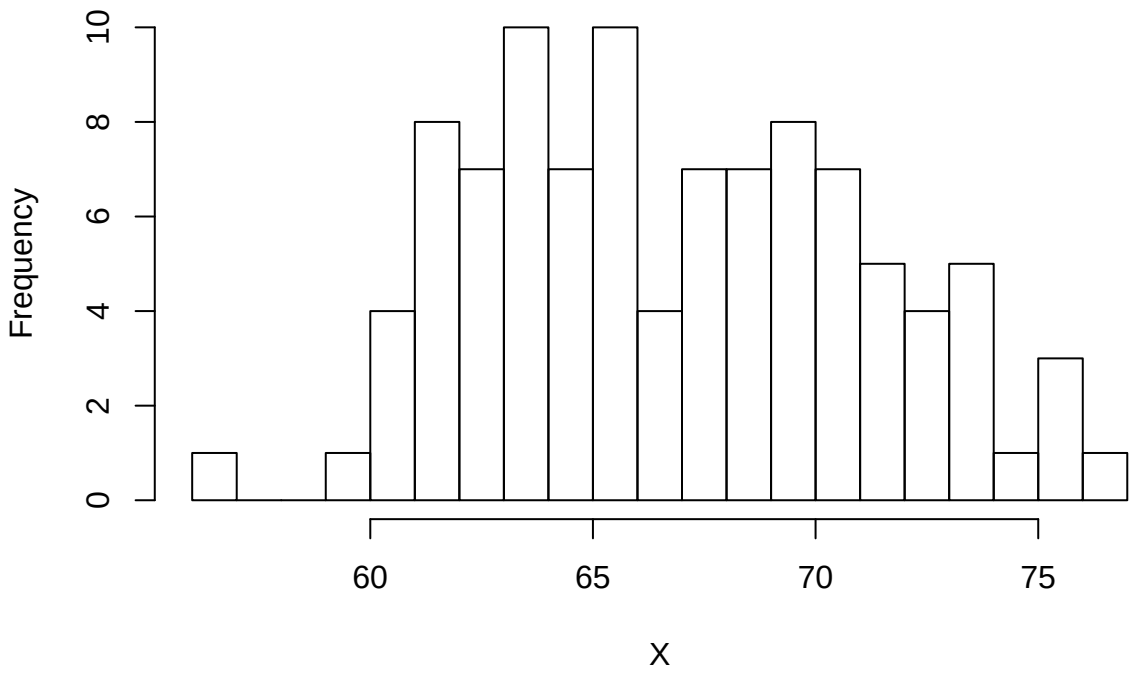
Suppose that the experimenter collected 100 cases from the true mixture-normal distribution $f(x)$. To solve problems (4) through (8) we analyze one realized sample from our function **r.norm.mix()**. In practice this dataset would be collected and not simulated. Uncomment the below code to simulate our dataset **x**. If you failed to solve Part I, then read in the csv file **mixture_data.csv** posted on Canvas.

Solution

```
# Simulate data
set.seed(1983)
x <- r.norm.mix(n.samps=100)$x
head(x)
```

```
## [1] 71.66666 63.91096 67.06554 65.49516 70.34363 65.69982
```

```
hist(x,breaks=20,xlab="X",main="")
```




```
# Or read data
#x <- read.csv("mixture_data.csv")$x
#head(x)
#hist(x,breaks=20,xlab="X",main="")
```

Further, suppose that we know the true heights and standard deviations of the two normal distributions but the mixture parameter δ is unknown. In this case, we know $\mu_1 = 70$, $\sigma_1 = 3$, $\mu_2 = 64$, $\sigma_2 = 2.5$. The goal of this exercise is to utilize **maximum likelihood** and **MCMC Bayesian techniques** to estimate mixture parameter δ.

Maximum likelihood Estimator of Mixture Parameter

4) Set up the likelihood function $L(\delta|x_1, \ldots, x_{100})$ and define it as **mix.like()**. The function should have two inputs including the parameter **delta** and data vector **x**. Evaluate the likelihood at the parameter values **delta=.2**, **delta=.4,** and **delta=.6**. Note that all three evaluations will be very small numbers. Which delta ($\delta = .2, .4, .6$) is the most likely to have generated the dataset **x**?
Solution

```
## Solution goes here -------
mix.like <- function(delta,x){
  return(prod(delta*dnorm(x,mean=70,sd=3) + (1-delta)*dnorm(x,mean=64,sd=2.5)))
}
mix.like(0.2,x)
```

```
## [1] 8.547304e-130
```

```
mix.like(0.4,x)
```

```
## [1] 1.214575e-124
```

```
mix.like(0.6,x)
```

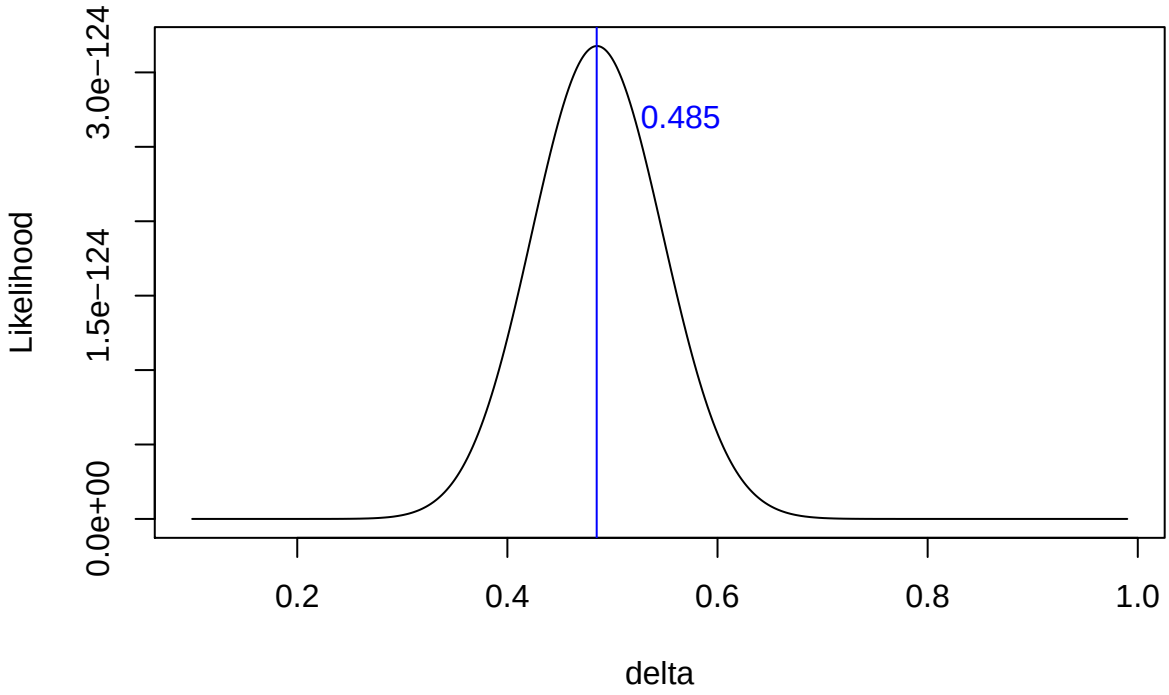
```
## [1] 5.735698e-125
```

$\delta = .4$ is the most likely to generate the dataset **x**.

5) Compute the maximum likelihood estimator of mixture parameter δ. To accomplish this task, apply

your likelihood function **mix.like()** across the vector **seq(.1,.99,by=.001)**. The solution to this exercise is given below.

```
delta <- seq(.1,.99,by=.001)
MLE.values <- sapply(delta,mix.like,x=x)
delta.MLE <- delta[which.max(MLE.values)]
plot(delta,MLE.values,ylab="Likelihood",type="l")
abline(v=delta.MLE,col="blue")
text(x=delta.MLE+.08,y=mix.like(delta=.45,x=x),paste(delta.MLE),col="blue")
```



MCMC

6) Run the Metropolis-Hastings algorithm to estimate mixture parameter δ. In this exercise you will assume a Beta($\alpha = 10, \beta = 10$) prior distribution on mixture parameter δ. Some notes follow:

- Run 20000 iterations. I.e., simulate 20000 draws of $\delta^{(t)}$
- Proposal distribution Beta($\alpha = 10, \beta = 10$)
- Independence chain with Metropolis-Hastings ratio:

$$R(\delta^{(t)}, \delta^*) = \frac{L(\delta^*|x_1, \ldots, x_{100})}{L(\delta^{(t)}|x_1, \ldots, x_{100})}$$

Display the first 20 simulated cases of $\delta^{(t)}$.

Solution

```
## Solution goes here -------
theta_1 <- rbeta(1,10,10) # draw theta_(0)
n.samps <- 20000 # number of iterations
theta_vec <- rep(NA,(n.samps+1))
theta_vec[1] <- theta_1

# MCMC loop
for (t in 1:n.samps){
```

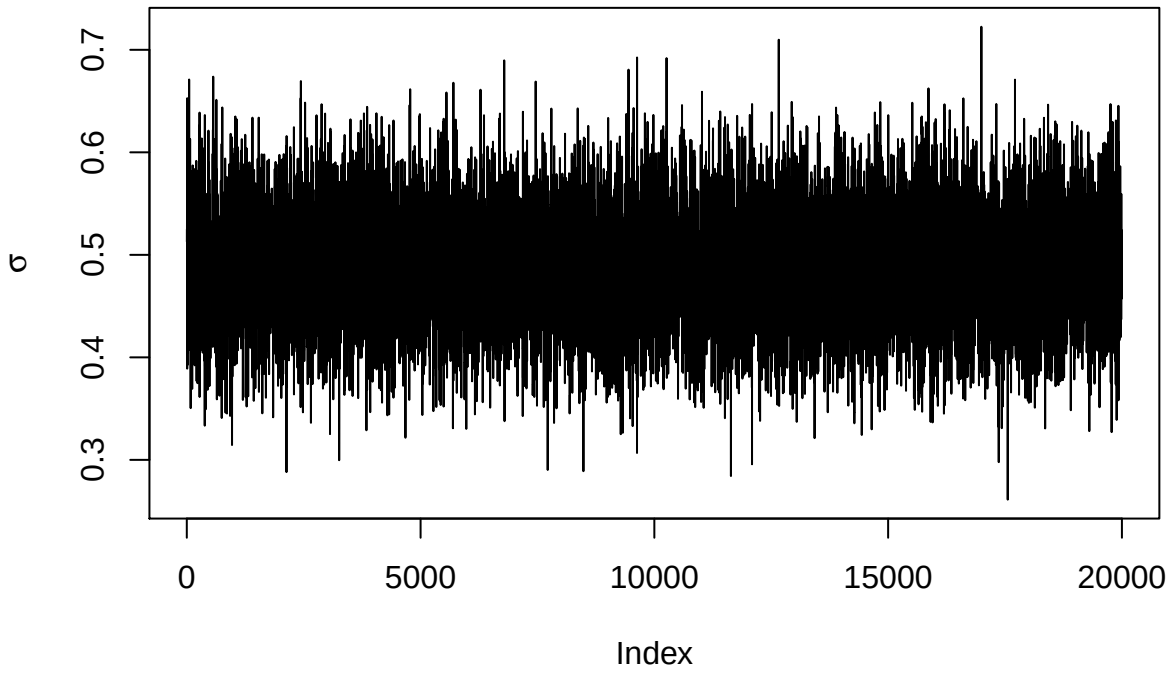
```
  theta_star <- rbeta(1,10,10) # draw theta* from proposal distribution
  theta_t <- theta_vec[t] # theta_(t)
  # compute MH ratio
  MH_ratio <- mix.like(theta_star,x)/mix.like(theta_t,x)
  # select new case
  prob_vec <- c(min(MH_ratio,1),1-min(MH_ratio,1))
  theta_vec[t+1] <- sample(c(theta_star,theta_t),1,prob=prob_vec)
}

theta_vec[1:20]
```

```
##  [1] 0.5128524 0.5248402 0.5248402 0.5248402 0.5548534 0.4293076 0.4115912
##  [8] 0.3889568 0.6528069 0.6114566 0.3923558 0.3923558 0.5482251 0.4231360
## [15] 0.4231360 0.4047930 0.5116155 0.5116155 0.4229895 0.5795368
```

7) Construct a lineplot of the simulated Markov chain from exercise (6). The vertical axis is the simulated chain $\delta^{(t)}$ and the horizontal axis is the number of iterations.

Solution

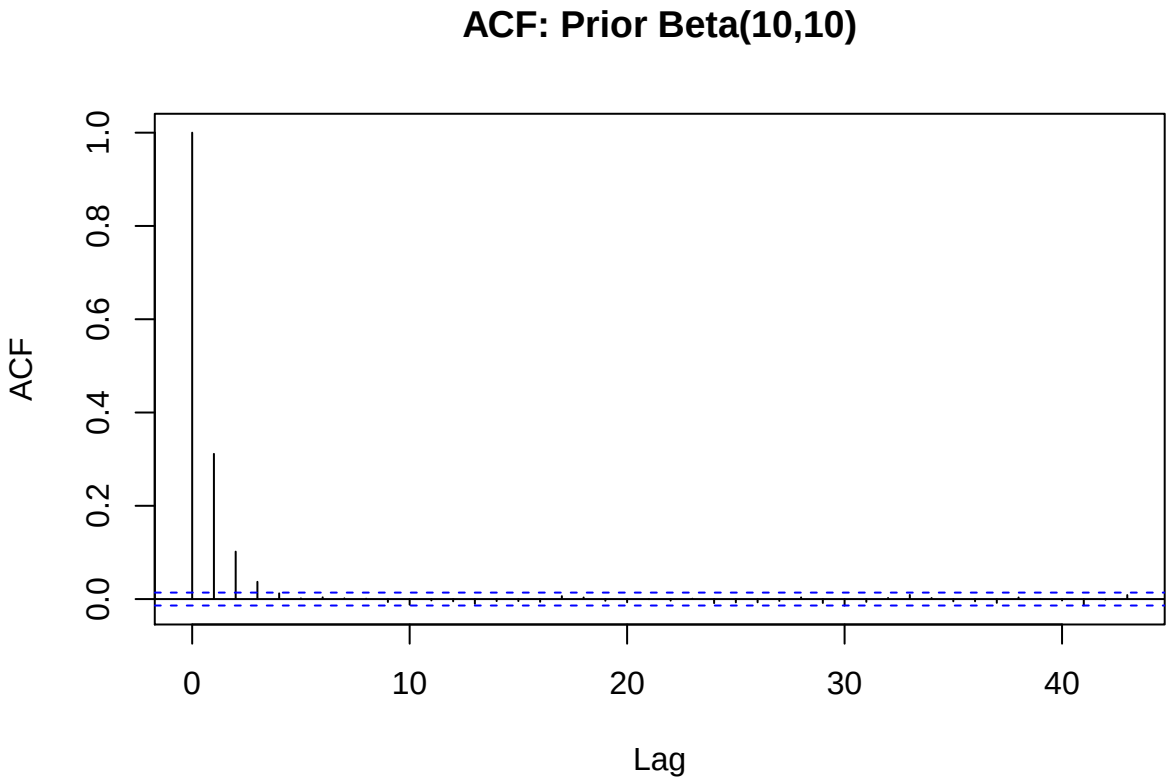
```
## Solution goes here -------
plot(theta_vec,col='white',ylab=expression(sigma))
lines(theta_vec)
```



8) Plot the empirical autocorrelation function of your simulated chain $\delta^{(t)}$. I.e., run the function **acf()**. A quick decay of the chain's autocorrelations indicate good mixing properties.

Solution

```
acf(theta_vec,main="ACF: Prior Beta(10,10)")
```



9) Compute the empirical Bayes estimate $\hat{\delta}_B$ of the simulated posterior distribution $\pi(\delta|x_1,\ldots,x_n)$. To solve this problem, simply compute the sample mean of your simulated chain $\delta^{(t)}$ after discarding a 20% burn-in.

Solution

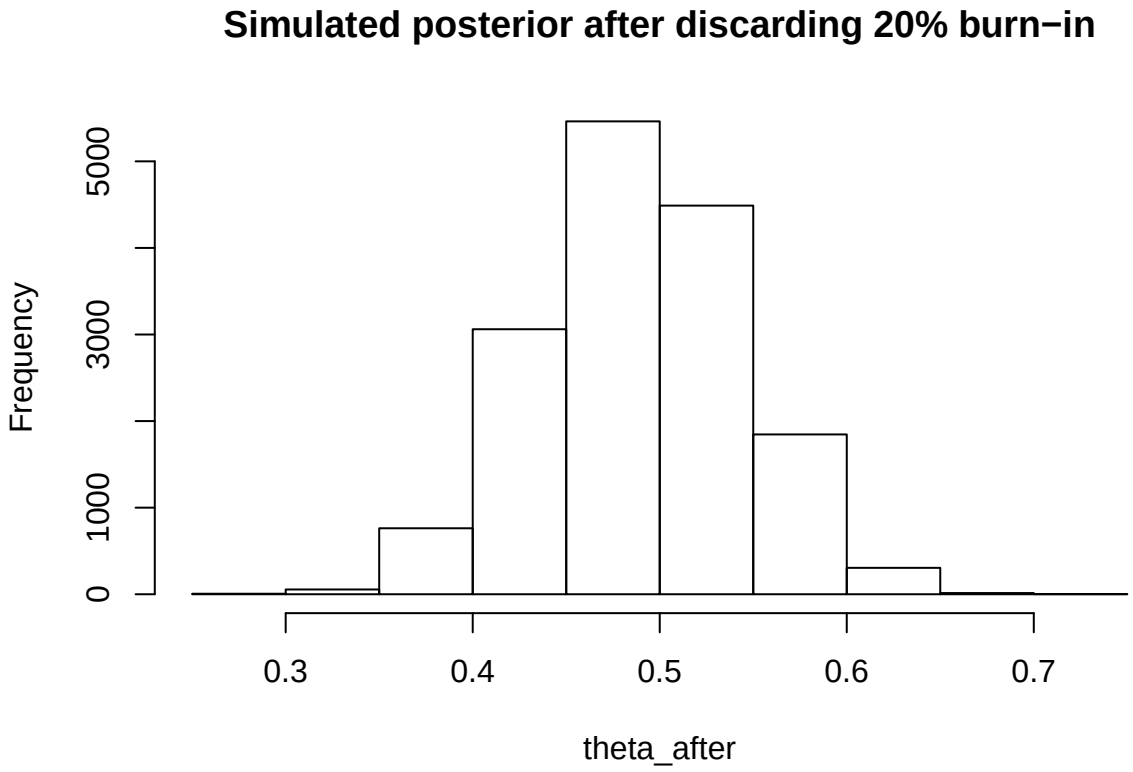
```
## Solution goes here -------
theta_after <- theta_vec[round(0.2*length(theta_vec)):length(theta_vec)]
mean(theta_after)
```

```
## [1] 0.488827
```

10) Construct a histogram of the simulated posterior $\pi(\delta|x_1,\ldots,x_n)$ after discarding a 20% burn-in.

Solution

```
## Solution goes here -------
hist(theta_after,main='Simulated posterior after discarding 20% burn-in')
```



11) Run the Metropolis-Hastings algorithm to estimate the mixture parameter δ using a Beta($\alpha = 15, \beta = 2$) prior distribution on mixture parameter δ. Repeat exercises 6 though 10 using the updated prior.

Solution

```
## Solution goes here -------
theta_1 <- rbeta(1,15,2) # draw theta_(0)
n.samps <- 20000 # number of iterations
theta_vec <- rep(NA,(n.samps+1))
theta_vec[1] <- theta_1

# MCMC loop
for (t in 1:n.samps){
  theta_star <- rbeta(1,15,2) # draw theta* from proposal distribution
  theta_t <- theta_vec[t] # theta_(t)
  # compute MH ratio
  MH_ratio <- mix.like(theta_star,x)/mix.like(theta_t,x)
  # select new case
  prob_vec <- c(min(MH_ratio,1),1-min(MH_ratio,1))
  theta_vec[t+1] <- sample(c(theta_star,theta_t),1,prob=prob_vec)
}

theta_vec[1:20]
```

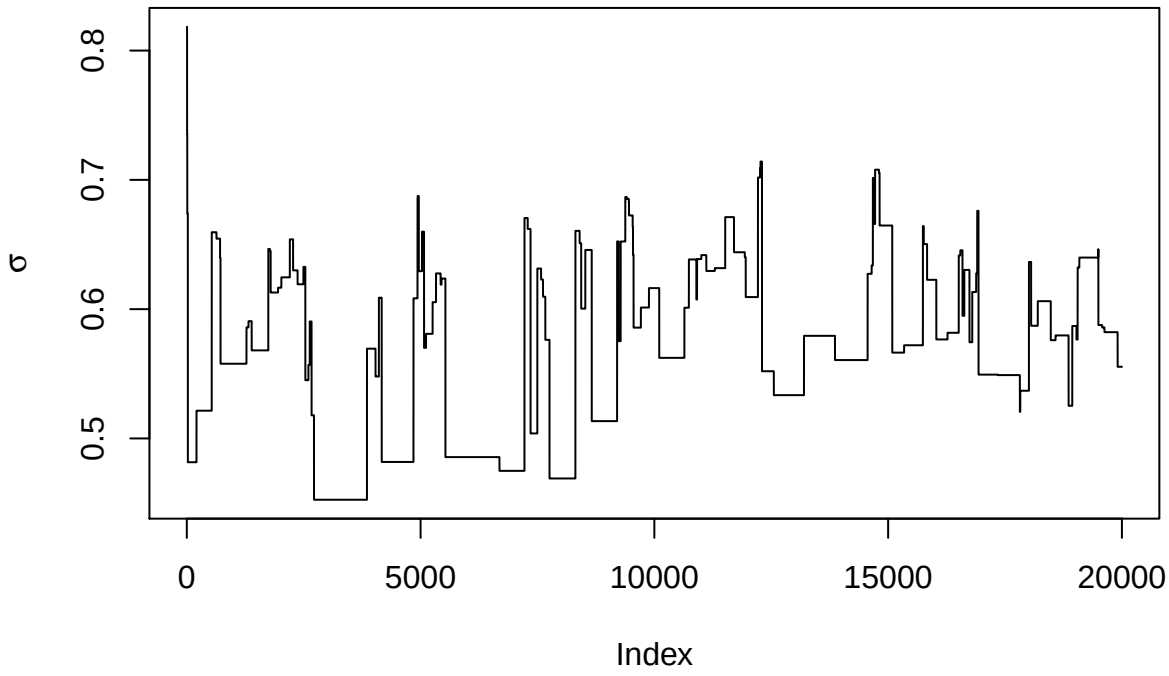
```
##  [1] 0.8183595 0.8183595 0.8183595 0.8183595 0.8183595 0.8183595 0.7356681
##  [8] 0.7356681 0.7356681 0.7356681 0.6740104 0.6740104 0.6740104 0.6740104
## [15] 0.6740104 0.6740104 0.6740104 0.6740104 0.6740104 0.6740104
```

lineplot:

Construct a lineplot of the simulated Markov chain from exercise (6). The vertical axis is the simulated chain $\delta^{(t)}$ and the horizontal axis is the number of iterations.

Solution

```
## Solution goes here -------
plot(theta_vec,col='white',ylab=expression(sigma))
lines(theta_vec)
```

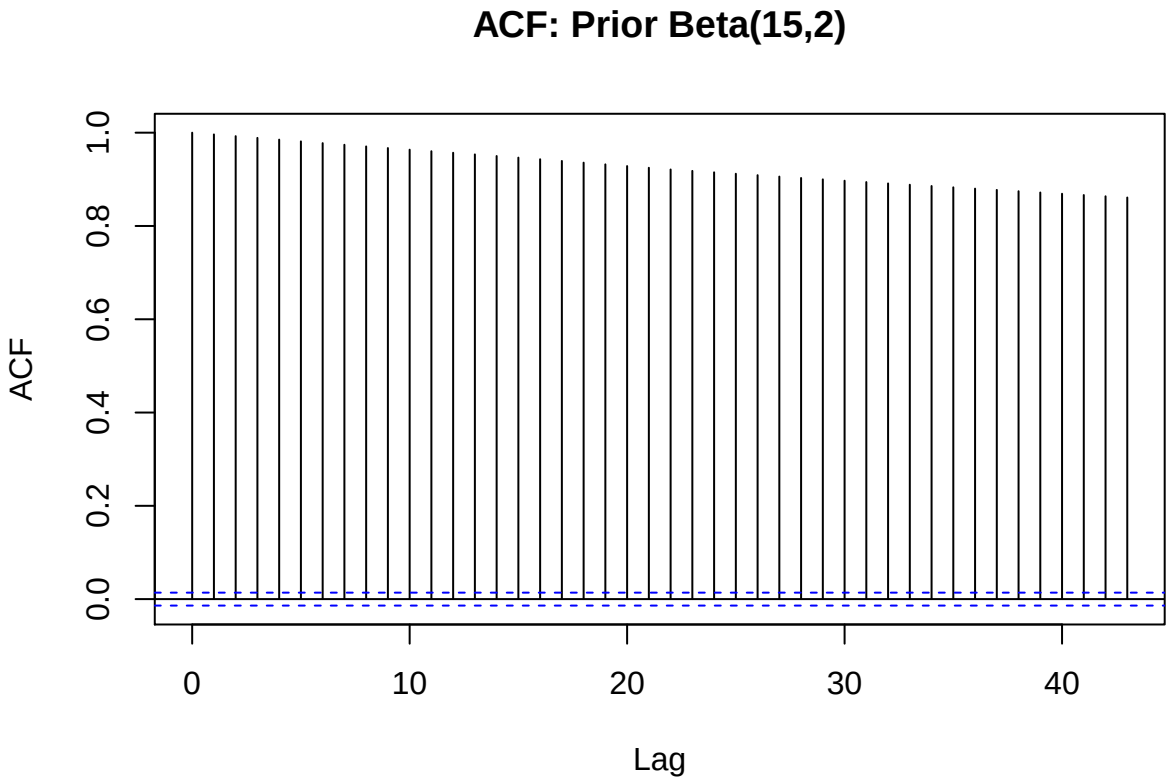


ACF:

Plot the empirical autocorrelation function of your simulated chain $\delta^{(t)}$. I.e., run the function **acf()**. A slow decay of the chain's autocorrelations indicate poor mixing properties.

Solution

```
## Solution goes here -------
acf(theta_vec,main="ACF: Prior Beta(15,2)")
```



Bayes estimate:

Compute the empirical Bayes estimate $\hat{\delta}_B$ of the simulated posterior distribution $\pi(\delta|x_1, \ldots, x_n)$. To solve this problem, simply compute the sample mean of your simulated chain $\delta^{(t)}$ after discarding a 20% burn-in. Your answer should be close to the MLE.

Solution

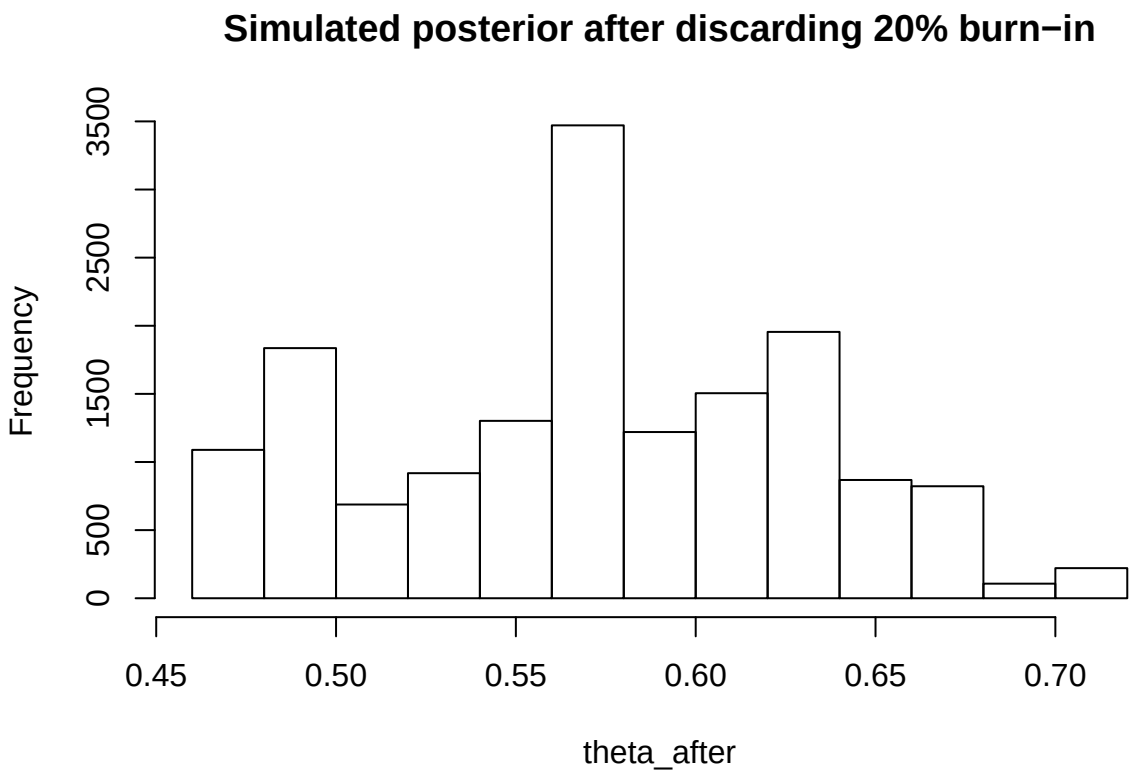
```
## Solution goes here -------
theta_after <- theta_vec[round(0.2*length(theta_vec)):length(theta_vec)]
mean(theta_after)
```

```
## [1] 0.5713894
```

Posterior: Construct a histogram of the simulated posterior $\pi(\delta|x_1, \ldots, x_n)$ after discarding a 20% burn-in.

Solution

```
## Solution goes here -------
hist(theta_after,main='Simulated posterior after discarding 20% burn-in')
```


Simulated posterior after discarding 20% burn−in
Frequency
0
500
1500
2500
3500
0.45
0.50
0.55
0.60
0.65
0.70
theta_after