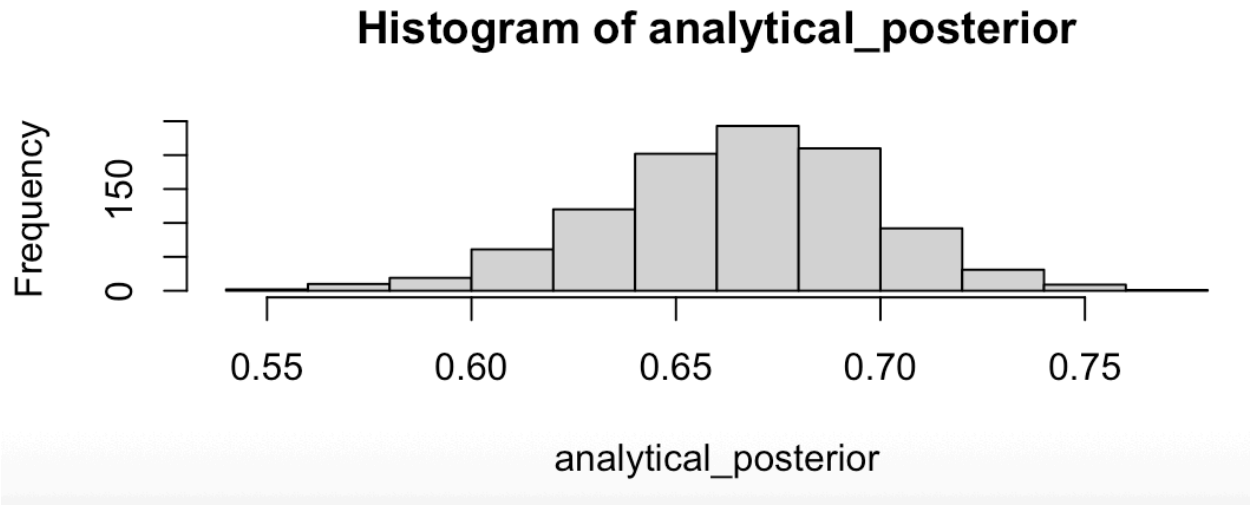
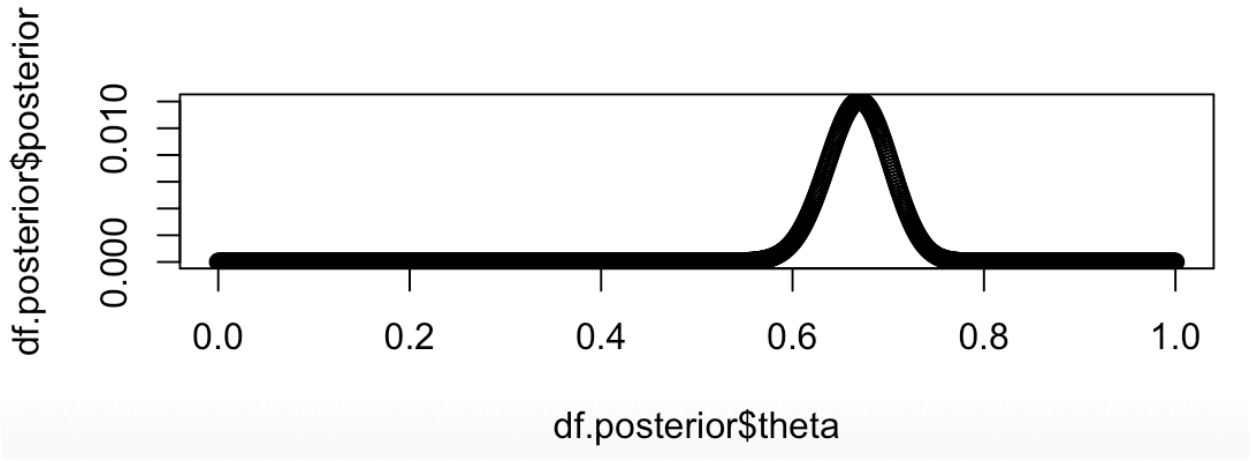
Name: Ishaan Jain
Roll Number: 230483
Part 1
1.1)

```
library(ggplot2)
#1.1
analytical_posterior <- rbeta(1000, 135, 67)
hist(analytical_posterior)
```



1.2)

```
y <- c(10, 15, 15, 14, 14, 14, 13, 11, 12, 16)
theta_grid <- seq(from=0, to=1, length=1000)
df.posterior <- data.frame(matrix(ncol=3, nrow=length(theta_grid)))
colnames(df.posterior) <- c('theta', 'lkl', 'prior')
for (i in 1:length(theta_grid)){
  lkl <- prod(dbinom(y, 20, theta_grid[i]))
  prior <- dbeta(theta_grid[i],1,1)
  df.posterior[i,] <- c(theta_grid[i], lkl, prior)
}
df.posterior$ML <- rep(sum(df.posterior$lkl*df.posterior$prior), 1000)
df.posterior <- df.posterior %>%
  mutate(posterior = lkl*prior/ML)
plot(df.posterior$theta, df.posterior$posterior)
```



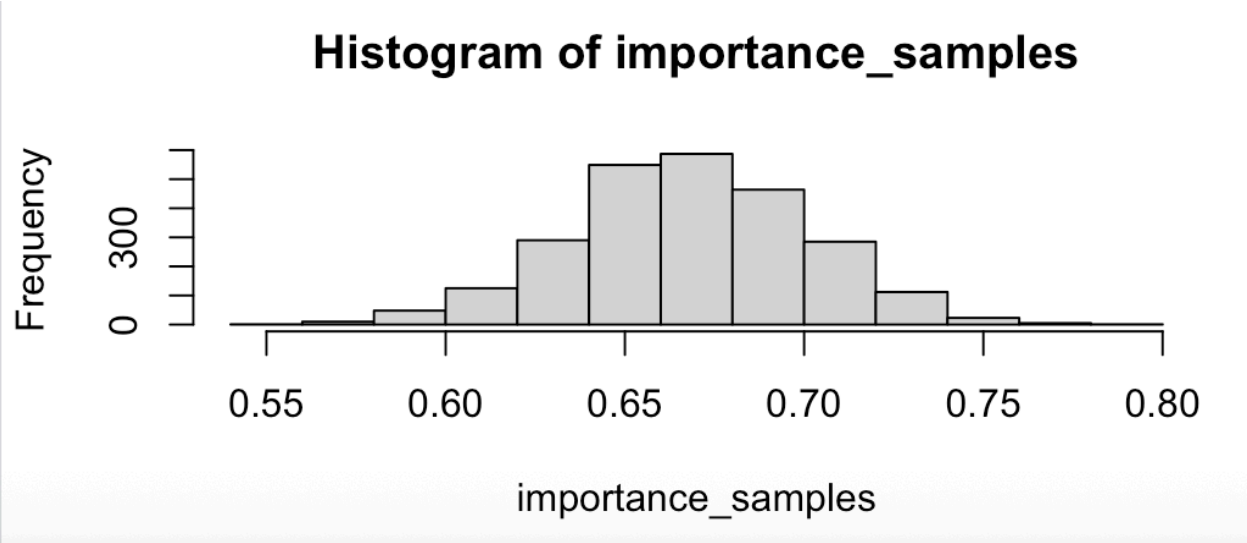
1.3)

```
df.mc <- data.frame(matrix(ncol=2, nrow=1000))
colnames(df.mc) <- c('theta_mc', 'lkl')
for (i in 1:1000){
  theta_i <- rbeta(1, 1, 1)
  lkl <- prod(dbinom(y, 20, theta_i))
  df.mc[i,] <- c(theta_i, lkl)
}
ML <- mean(df.mc$lkl)
ML
```

```
1.335742e-10
```

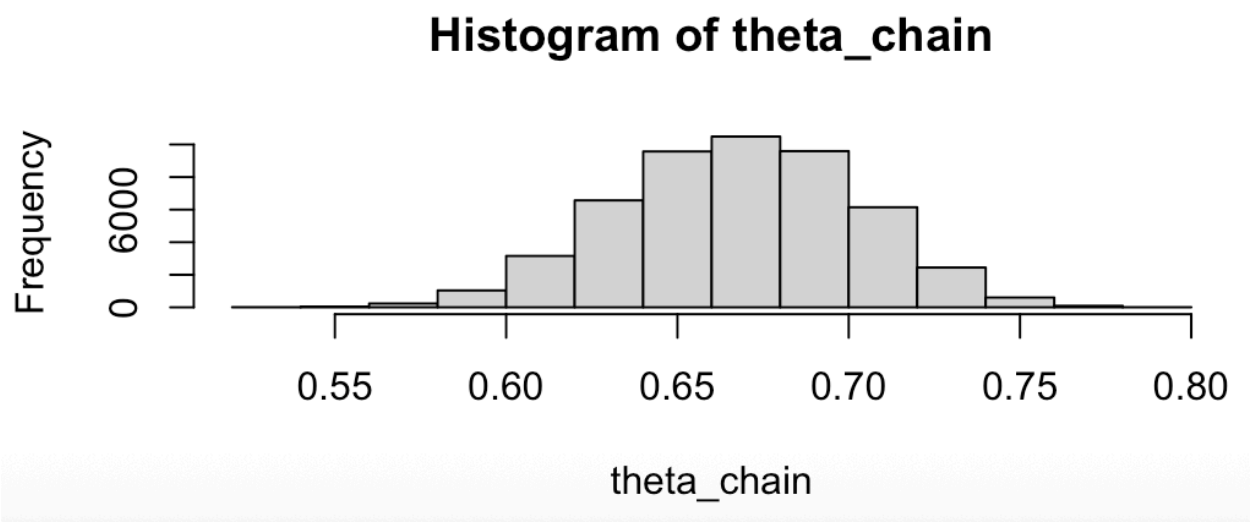
1.4

```
#1.4
proposed <- rbeta(10000, 5, 3)
df.importance <- data.frame(proposed=proposed)
df.importance$lkl <- NA
for (i in 1:10000){
  df.importance$lkl[i] <- prod(dbinom(y, 20, proposed[i]))
}
df.importance$prior <- dbeta(df.importance$proposed,1,1)
df.importance$proposal <- dbeta(df.importance$proposed, 5, 3)
df.importance$weights <- (df.importance$lkl)*
                         (df.importance$prior)/
                         (df.importance$proposal)
df.importance$weights <- df.importance$weights/sum(df.importance$weights)
importance_samples <- sample(proposed, size=2500, replace=TRUE,
                             prob=df.importance$weights)
hist(importance_samples)
```



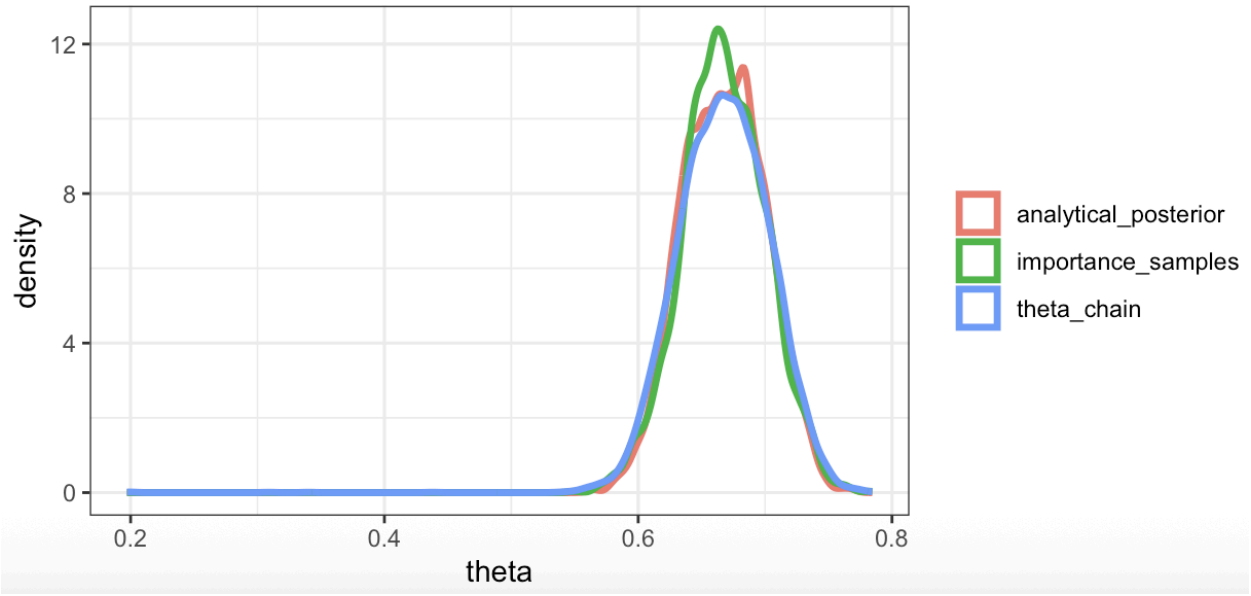
1.5)

```r
#1.5
nsamp <- 50000
theta_chain <- rep(NA, nsamp)
theta_chain[1] <- rbeta(1, 1, 1)
i <- 1
step <- 0.08
while (i < nsamp){
  proposal_theta <- rnorm(1, theta_chain[i], step)
  if (proposal_theta>0 & proposal_theta<1){
    post_new <- prod(dbinom(y, 20, proposal_theta))*
                dbeta(proposal_theta, 1, 1)
    post_prev <- prod(dbinom(y, 20, theta_chain[i]))*
                 dbeta(theta_chain[i], 1, 1)
    hastings_ratio <- (post_new*dnorm(theta_chain[i], proposal_theta, step))/
                      (post_prev*dnorm(proposal_theta, theta_chain[i], step))
    p_str <- min(hastings_ratio, 1)
    if (p_str > runif(1,0,1)){
      theta_chain[i+1] <- proposal_theta
      i <- i+1
    }
  }
}
hist(theta_chain)
```



1.6)

```
#1.6
posteriors <- data.frame(analytical_posterior, importance_samples, theta_chain)
ggplot(melt(posteriors), aes(x=value,colour=variable))+
  geom_density(linewidth=1.2) + theme_bw() + xlab("theta") +
  theme(legend.title=element_blank(), legend.position = "right")
```

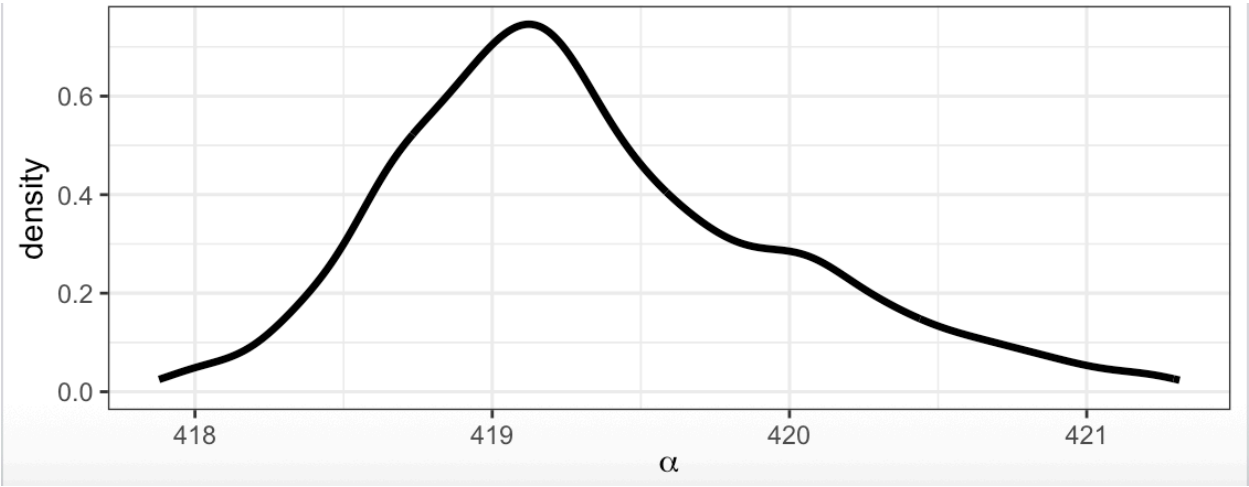


Part 2)

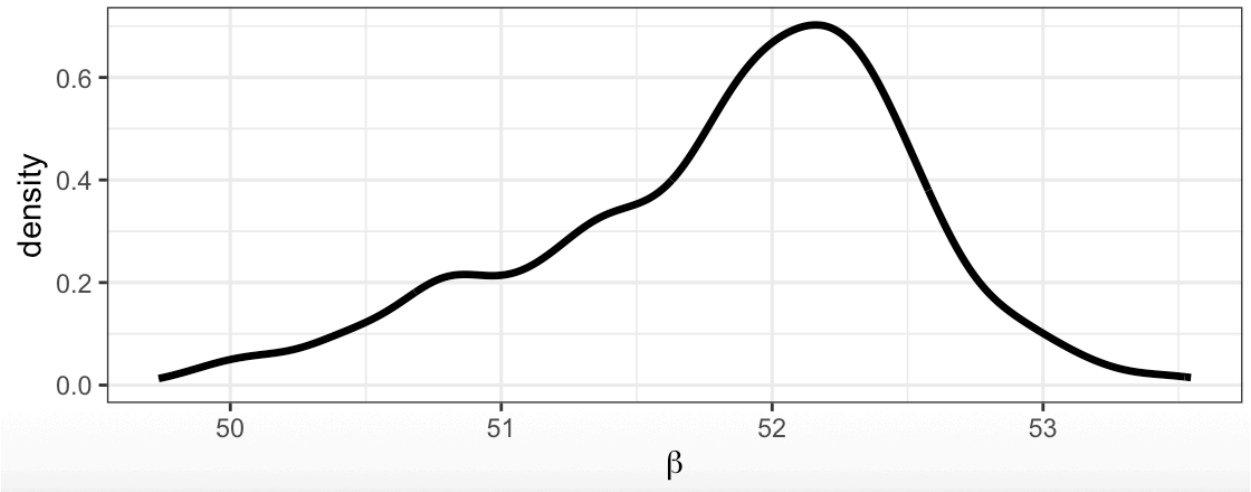
2.1)

```
library(truncnorm)
library(ggplot2)
dat <- read.csv("/Users/ishaanjain/CGS698/CGS698/Assn3/word-recognition-times.csv",
                sep=",",header = T)[,-1]

#2.1
for (i in 1:4000){
  if (dat$type[i]=='word'){
    dat$type[i] = 0
  }else{
    dat$type[i] = 1
  }
}
dat$type <- as.numeric(dat$type)

nsamp <- 4000
alpha_chain <- rep(NA, nsamp)
beta_chain <- rep(NA, nsamp)

alpha_chain[1] <- rnorm(1, 400, 50)
beta_chain[1] <- rtruncnorm(n=1, a=0, b=Inf, mean=0, sd=50)

i <- 1
reject <- 0
step <- 0.1

while(i < nsamp){
  proposal_alpha <- rnorm(1, alpha_chain[i], step)
  proposal_beta <- rtruncnorm(1, a=0, b=Inf, mean=beta_chain[i], sd=step)

  mu_new <- proposal_alpha + proposal_beta*dat$type
  mu_old <- alpha_chain[i] + beta_chain[i]*dat$type
```

```
  post_new <- sum(dnorm(dat$RT, mu_new, 30, log=TRUE))+
    dnorm(proposal_alpha, 400, 50,log=TRUE)+
    log(dtruncnorm(x=proposal_beta, a=0, b=Inf,mean=0,sd=50))
  post_prev <- sum(dnorm(dat$RT, mu_old, 30,log=TRUE))+
    dnorm(alpha_chain[i], 400, 50, log=TRUE)+
    log(dtruncnorm(x=beta_chain[i], a=0, b=Inf, mean=0, sd=50))

  hastings_ratio <-
    exp((post_new+dnorm(alpha_chain[i],proposal_alpha,step,log=TRUE)+
           log(dtruncnorm(x=beta_chain[i],mean=proposal_beta,sd=step,a=0,b=Inf)))-
          (post_prev+dnorm(proposal_alpha,alpha_chain[i],step,log=TRUE)+
             log(dtruncnorm(x=proposal_beta,mean=beta_chain[i],sd=step,a=0,b=Inf))))
  p_str <- min(hastings_ratio, 1)
  if (p_str > runif(1,0,1)){
    alpha_chain[i+1] <- proposal_alpha
    beta_chain[i+1] <- proposal_beta
    i <- i+1
  }else{
    reject <- reject + 1
  }
}

posteriors <- data.frame(alpha_chain, beta_chain)
ggplot(posteriors[-(1:2000),], aes(x=alpha_chain))+
  theme_bw() + geom_density(linewidth=1.2) + xlab(expression("alpha"))
```




```
ggplot(posteriors[-(1:2000),], aes(x=beta_chain))+
  theme_bw() + geom_density(linewidth=1.2) + xlab(expression(beta))
```



2.2)

```
> quantile(alpha_chain, probs=c(0.025, 0.975))
    2.5%    97.5%
404.5054 420.8113
> quantile(beta_chain, probs=c(0.025, 0.975))
    2.5%    97.5%
34.78947 52.94026
```

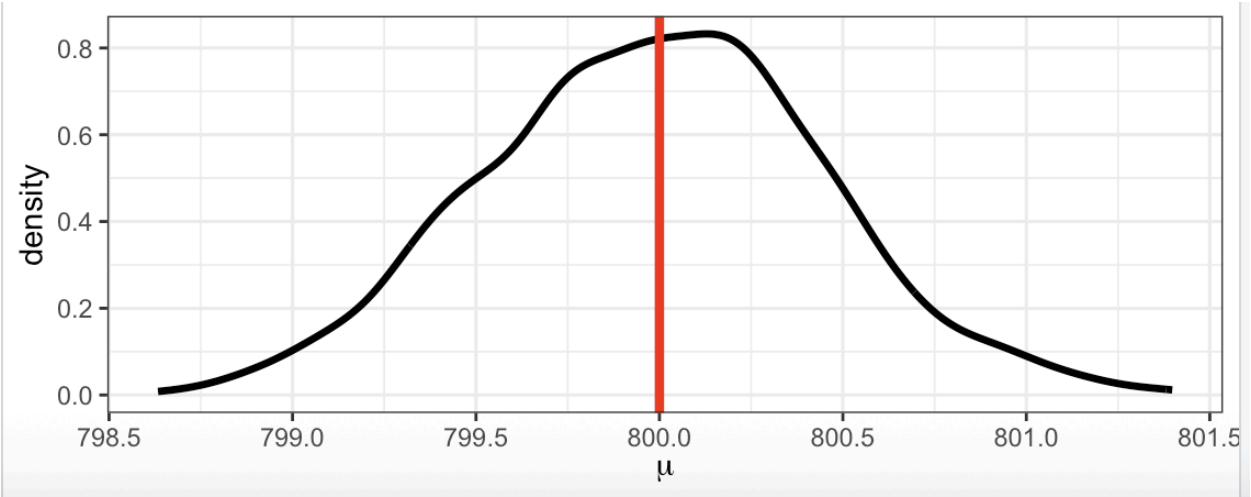
Part 3)

```
true_mu <- 800
true_var <- 100 #sigma^2
y <- rnorm(500,mean=true_mu,sd=sqrt(true_var))
hist(y)
sigma <- sqrt(true_var)
mu <- true_mu

#Gradient functions
gradient <- function(mu,sigma,y,n,m,s,a,b){
  grad_mu <- (((n*mu)-sum(y))/(sigma^2))+((mu-m)/(s^2))
  grad_sigma <- (n/sigma)-(sum((y-mu)^2)/(sigma^3))+((sigma-a)/(b^2))
  return(c(grad_mu,grad_sigma))
}
#Potential energy function
V <- function(mu,sigma,y,n,m,s,a,b){
  nlpd <- -(sum(dnorm(y,mu,sigma,log=T))+dnorm(mu,m,s,log=T)+dnorm(sigma,a,b,log=T))
  nlpd
}
```

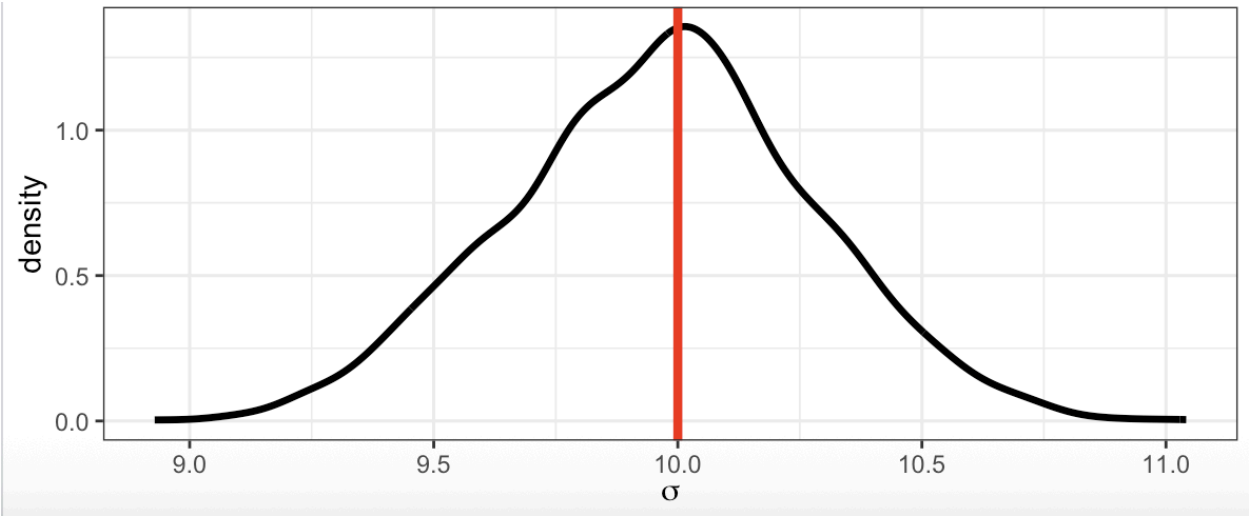
```r
#HMC sampler
HMC <- function(y,n,m,s,a,b,step,L,initial_q,nsamp,nburn){
  mu_chain <- rep(NA,nsamp)
  sigma_chain <- rep(NA,nsamp)
  reject <- 0
  #Initialization of Markov chain
  mu_chain[1] <- initial_q[1]
  sigma_chain[1] <- initial_q[2]
  #Evolution of Markov chain
  i <- 1
  while(i < nsamp){
    q <- c(mu_chain[i],sigma_chain[i])
    p <- rnorm(length(q),0,1) # Current position of the particle
    # Generate random momentum at the current position
    current_q <- q
    current_p <- p
    current_V = V(current_q[1],current_q[2],y,n,m,s,a,b) # Current potential energy
    current_T = sum(current_p
                    ^2)/2 # Current kinetic energy
    # Take L leapfrog steps
    for(l in 1:L){
      # Change in momentum in 'step/2' time
      p <- p-((step/2)*gradient(q[1],q[2],y,n,m,s,a,b))
      # Change in position in 'step' time
      q <- q + step*p
      # Change in momentum in 'step/2' time
      p <- p-((step/2)*gradient(q[1],q[2],y,n,m,s,a,b))
    }
    proposed_q <- q
    proposed_p <- p
    proposed_V = V(proposed_q[1],proposed_q[2],y,n,m,s,a,b) # Proposed potential energy
    proposed_T = sum(proposed_p
                     ^2)/2 # Proposed kinetic energy
    accept.prob <- min(1,exp(current_V+current_T-proposed_V-proposed_T))
    # Accept/reject the proposed position q
    if(accept.prob>runif(1,0,1)){
      mu_chain[i+1] <- proposed_q[1]
      sigma_chain[i+1] <- proposed_q[2]
      i <- i+1
    }else{
      reject <- reject+1
    }
  }
  posteriors <- data.frame(mu_chain,sigma_chain)[-(1:nburn),]
  posteriors$sample_id <- 1:nrow(posteriors)
  posteriors
}
```

3.1)

```
#3.1
df.posterior <- HMC(y=y,n=length(y), # data
                    m=1000,s=20,a=10,b=2, # priors
                    step=0.02, # step-size
                    L=12, # no. of leapfrog steps
                    initial_q=c(1000,11), # Chain initialization
                    nsamp=6000, # total number of samples
                    nburn=2000) # number of burn-in samples
library(ggplot2)
ggplot(df.posterior[-(1:2000),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))+
  geom_vline(xintercept=800,size=1.5,color="red")
```




```
ggplot(df.posterior[-(1:2000),],aes(x=sigma_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(sigma))+
  geom_vline(xintercept=10,size=1.5,color="red")
```

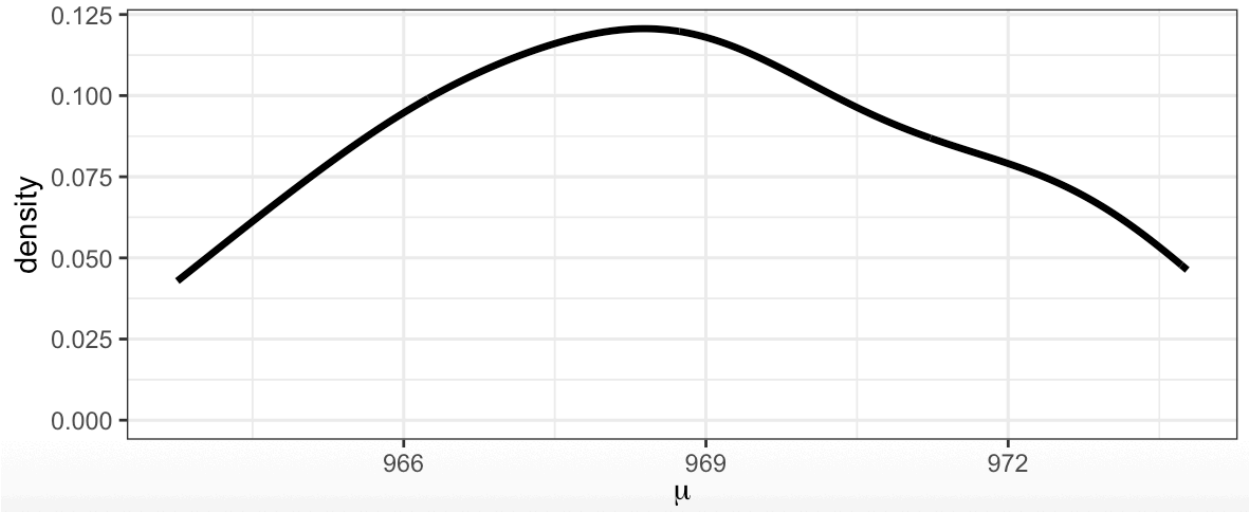


3.2)

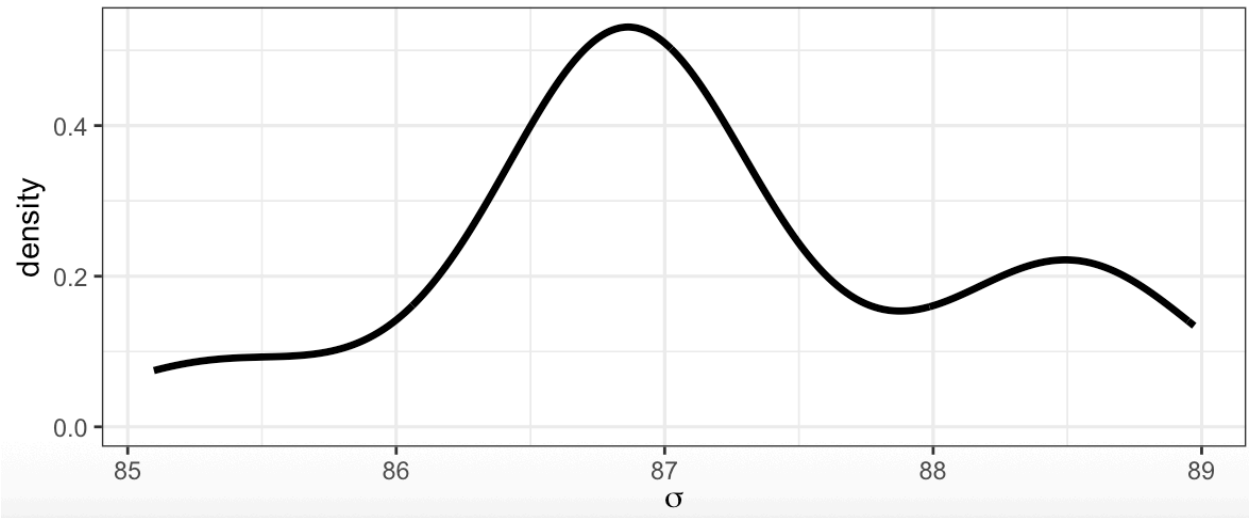
Upon **decreasing** the number of samples (nsamp), the model becomes **less accurate** and **mu** appears to **converge** at a **higher** value than **800**, while **sigma** appears to **converge** at a **higher** value than **50**. After a certain point, there is **no point** in **increasing nsamp** as we get **marginal increase** in **accuracy** despite **high increase in nsamp**. So **nsamp ~ 5000** is a decent value to get **decently accurate** readings.

a) **nsamp = 100**

```
df.posterior <- HMC(y=y,n=length(y), # data
                    m=1000,s=20,a=10,b=2, # priors
                    step=0.02, # step-size
                    L=12, # no. of leapfrog steps
                    initial_q=c(1000,11), # Chain initialization
                    nsamp=100, # total number of samples
                    nburn=33) # number of burn-in samples
library(ggplot2)
ggplot(df.posterior[-(1:33),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))
```

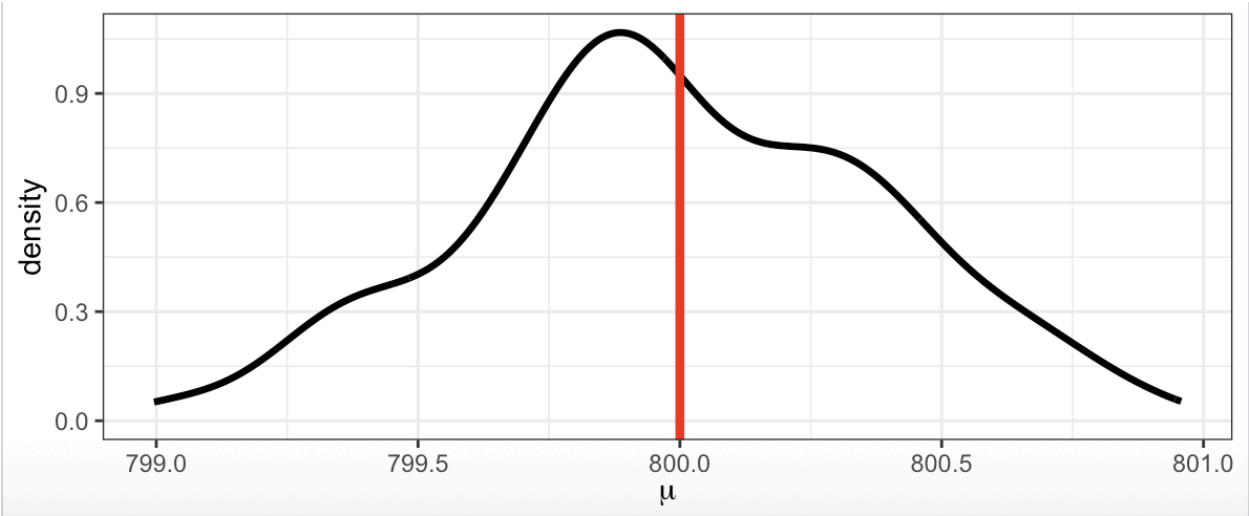



```
ggplot(df.posterior[-(1:33),],aes(x=sigma_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(sigma))
```

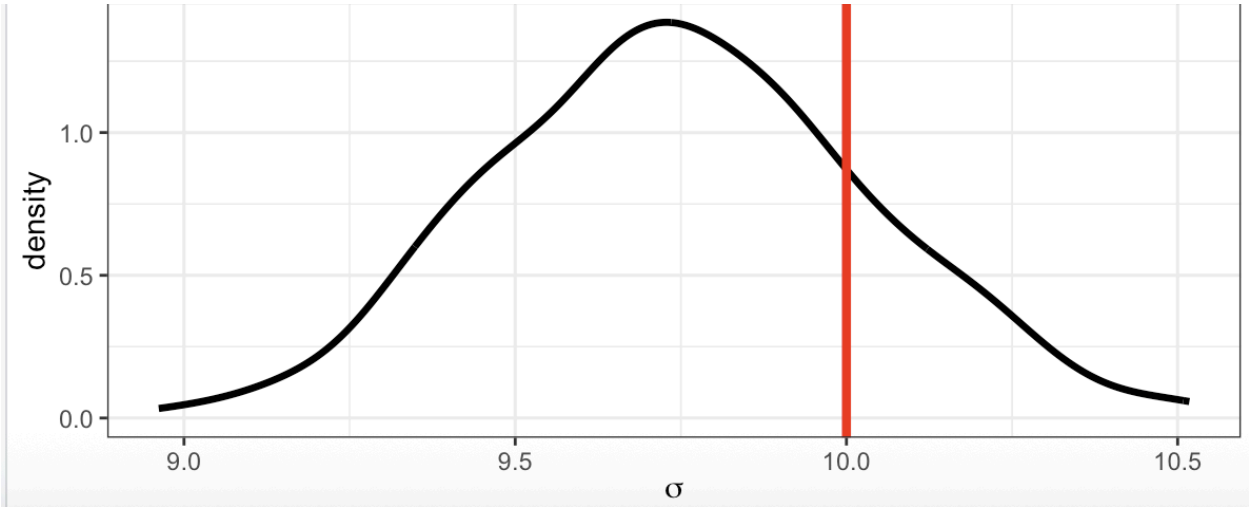


b) nsamp = 1000

```
ggplot(df.posterior[-(1:333),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))+
  geom_vline(xintercept=800,size=1.5,color="red")
```

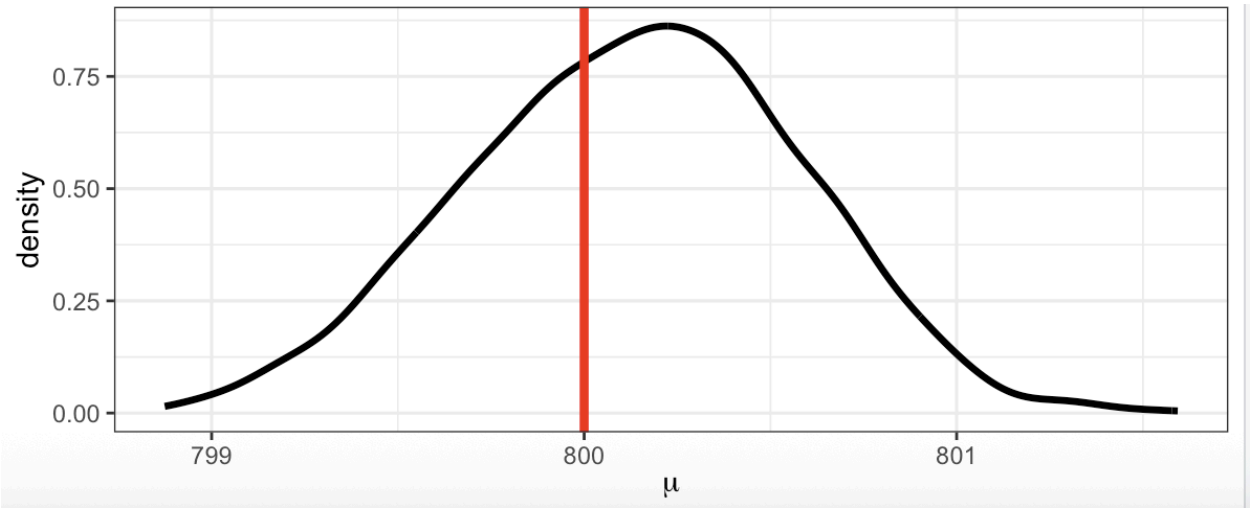



```
ggplot(df.posterior[-(1:333),],aes(x=sigma_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(sigma))+
  geom_vline(xintercept=10,size=1.5,color="red")
```

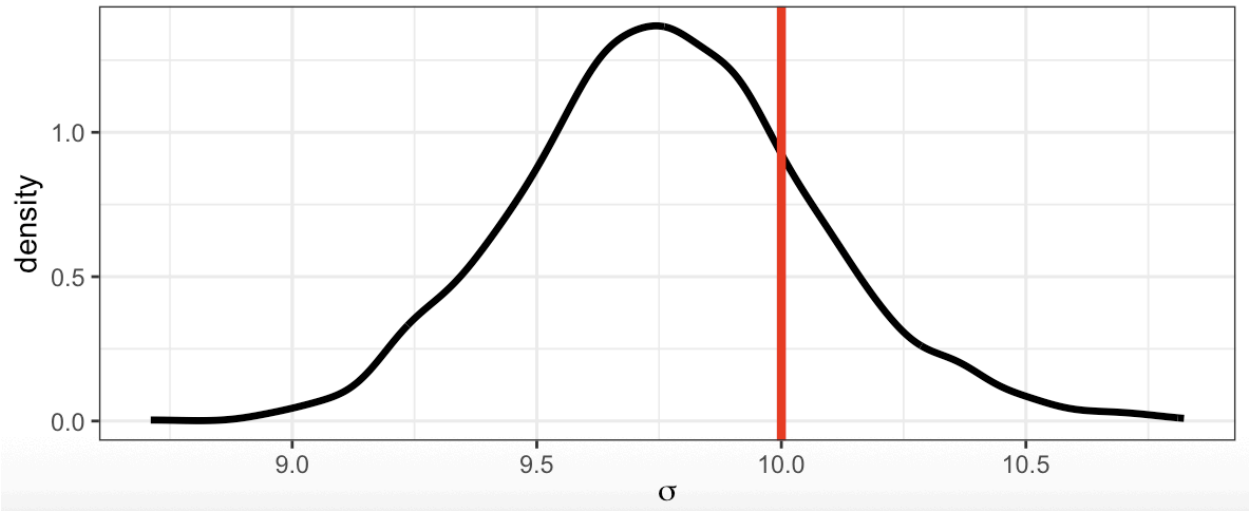


c) nsamp = 6000

```
ggplot(df.posterior[-(1:2000),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))+
  geom_vline(xintercept=800,size=1.5,color="red")
```




```
ggplot(df.posterior[-(1:2000),],aes(x=sigma_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(sigma))+
  geom_vline(xintercept=10,size=1.5,color="red")
```

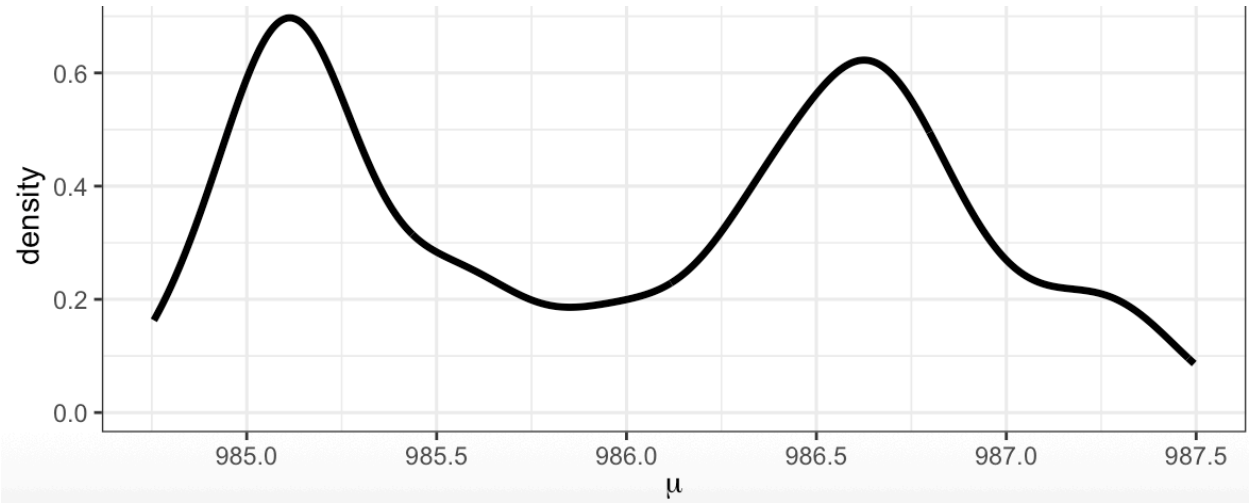


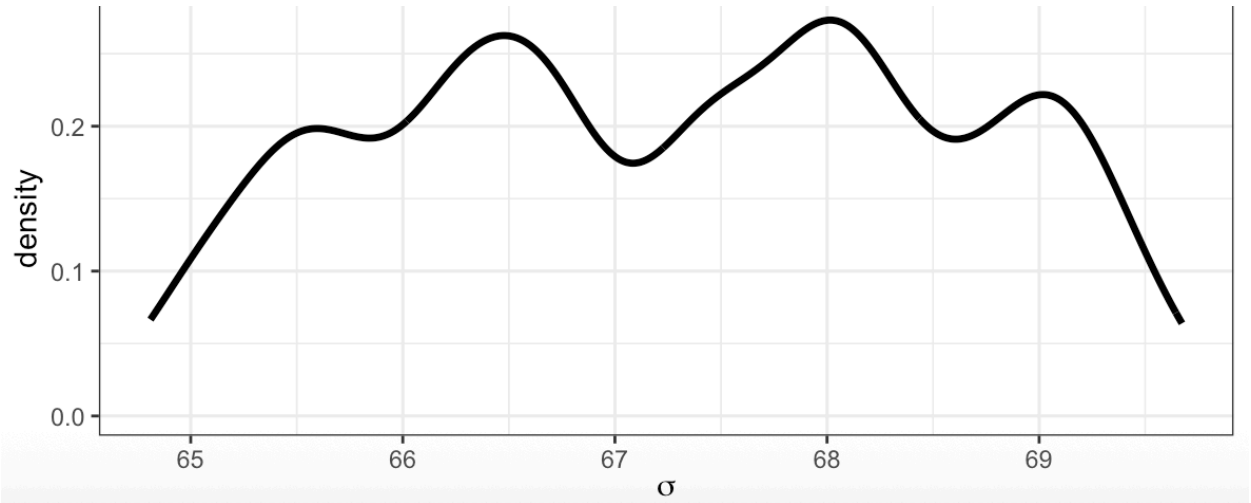
3.3)

Lower step-size tends to make HMC less efficient, while high step-size leads to model failure.

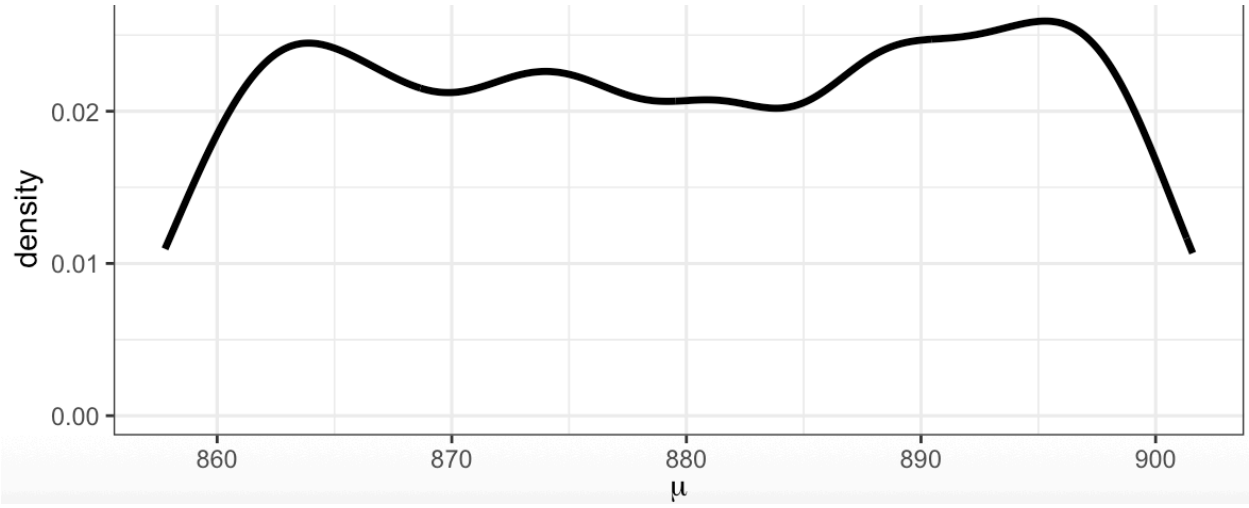
a) step=0.001

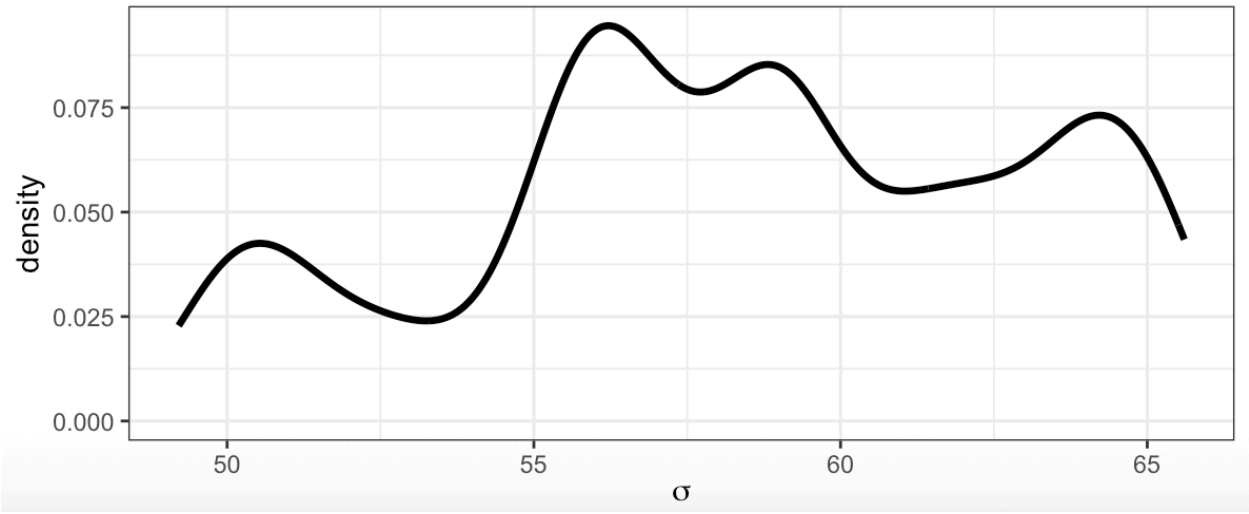
```r
df.posterior <- HMC(y=y,n=length(y), # data
                    m=1000,s=20,a=10,b=2, # priors
                    step=0.001, # step-size
                    L=12, # no. of leapfrog steps
                    initial_q=c(1000,11), # Chain initialization
                    nsamp=6000, # total number of samples
                    nburn=2000) # number of burn-in samples
library(ggplot2)
ggplot(df.posterior[-(1:2000),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))
```



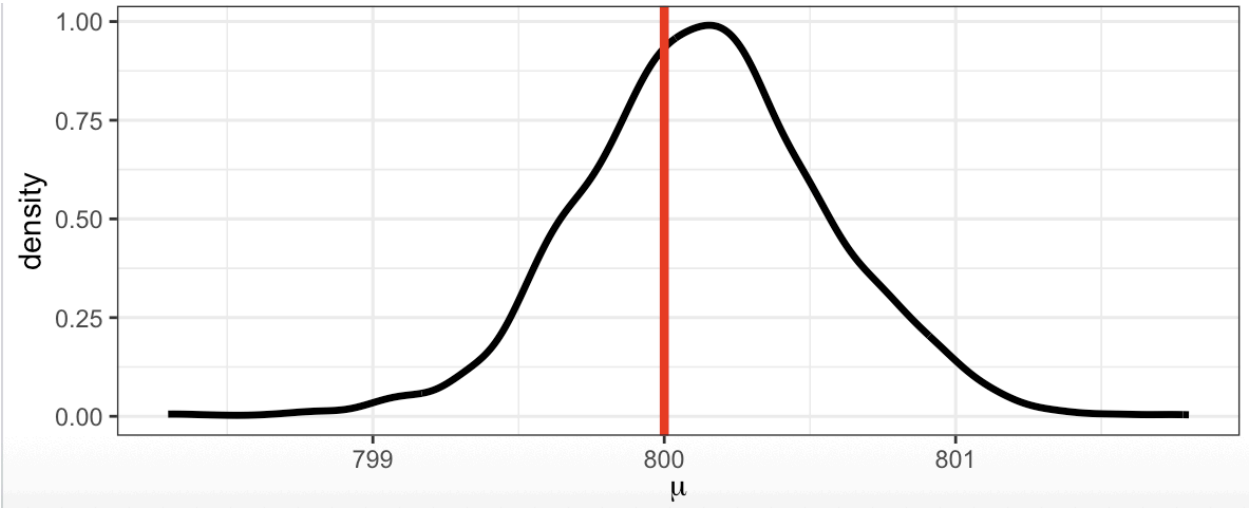


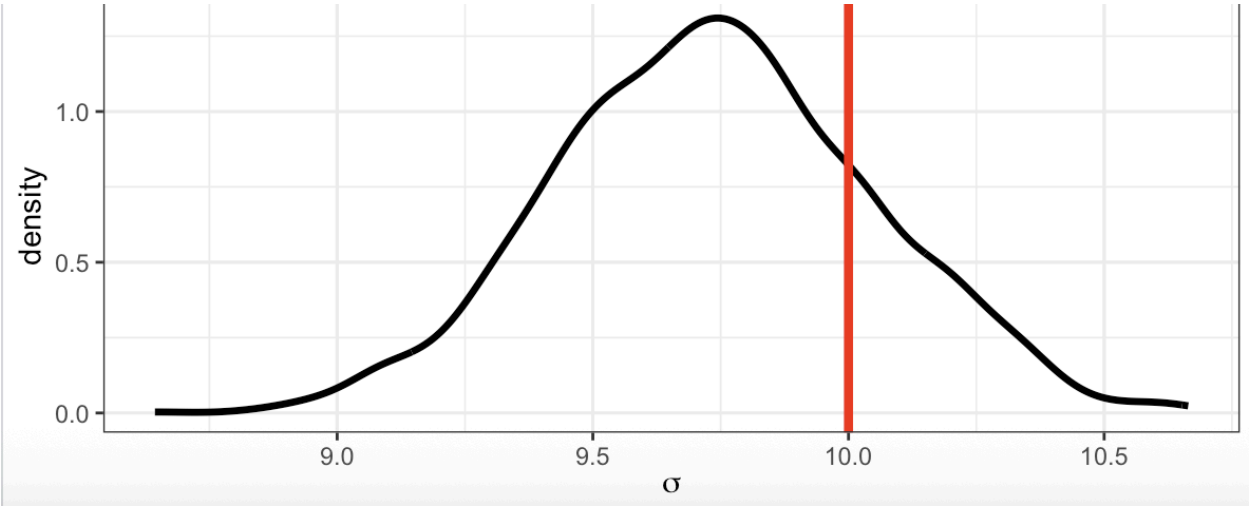
b) step = 0.005





c) step = 0.02



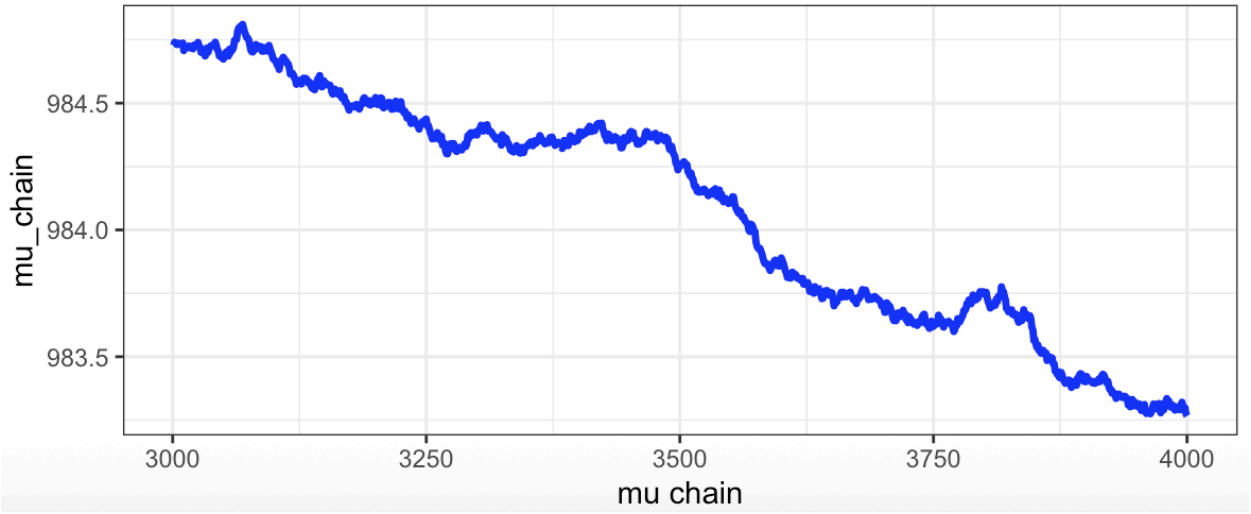


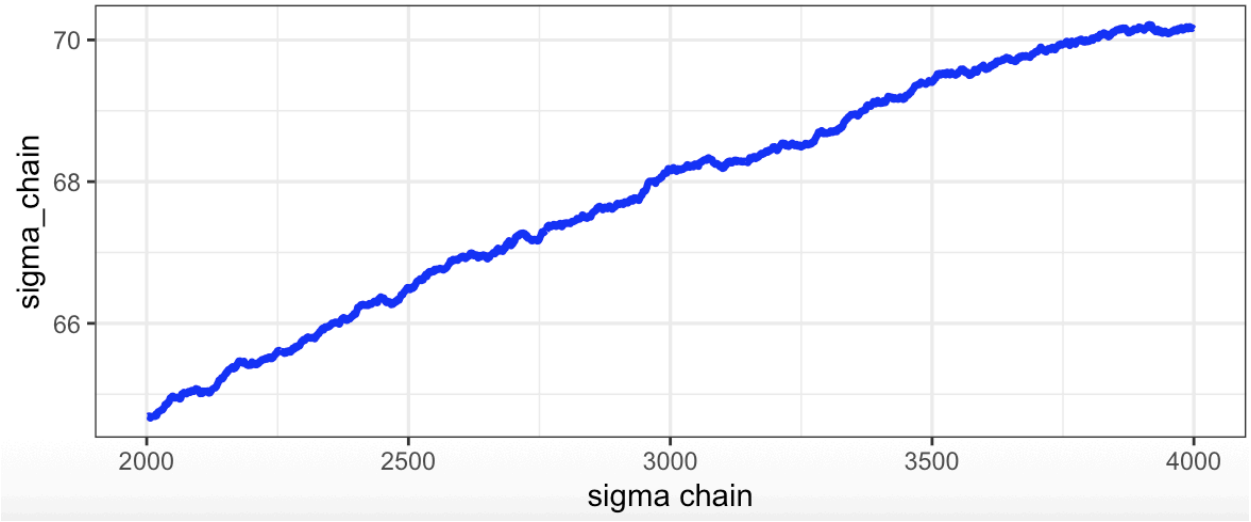
3.4)

When the step-size is too small, the mu_chain drifts downward very slowly, and is unable to explore the posterior fully, while the sigma_chain drifts both downward and upward, not exploring posterior fully either. Only at ideal step size does it properly explore the posterior chain.

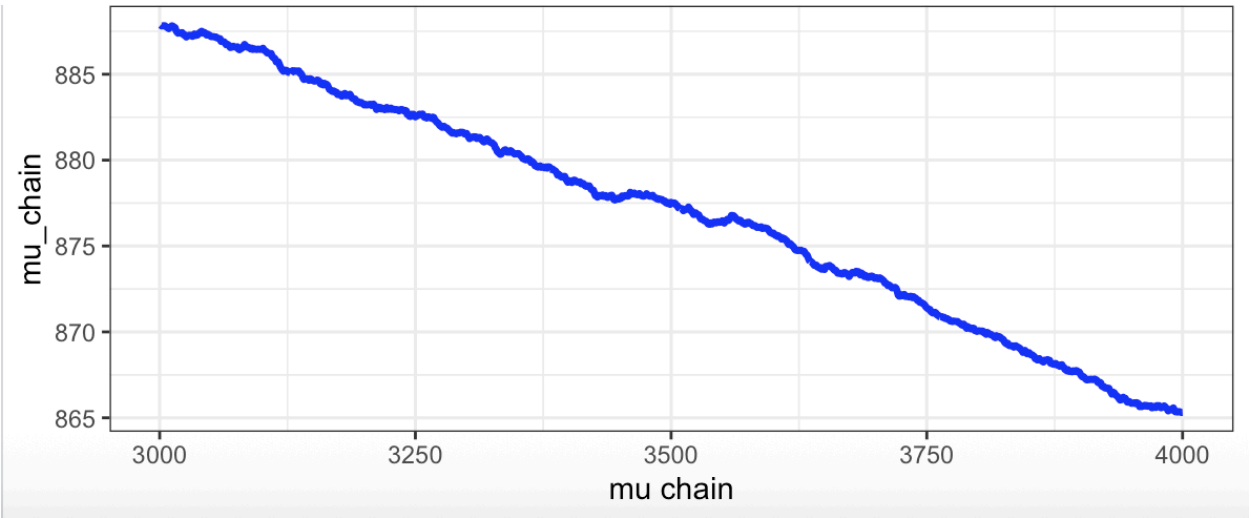
a) **Step size = 0.001**

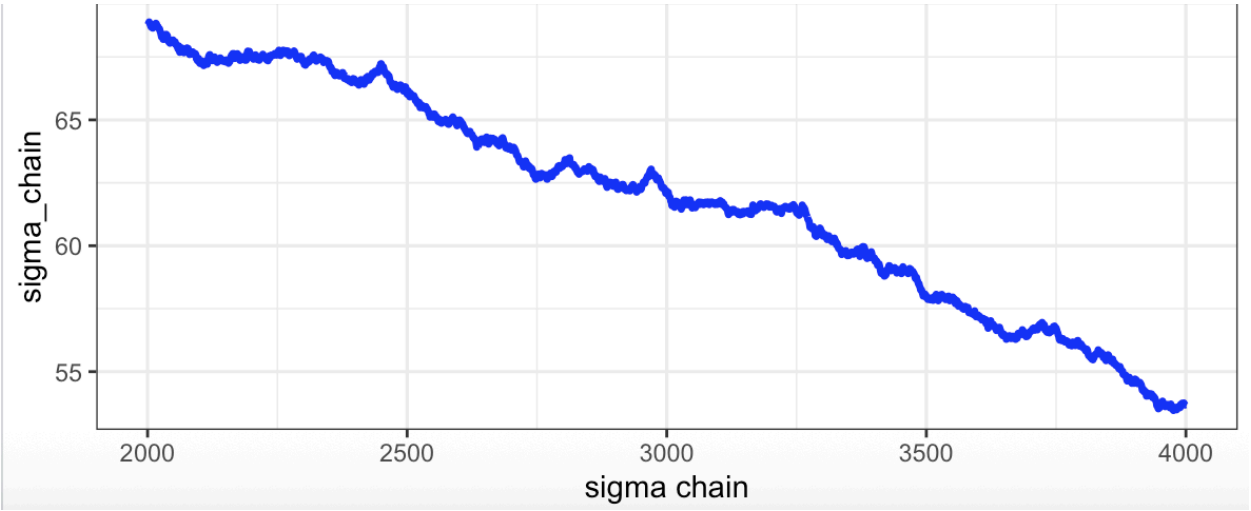
```
#3.4
df.posterior <- HMC(y=y,n=length(y), # data
                    m=1000,s=20,a=10,b=2, # priors
                    step=0.001, # step-size
                    L=12, # no. of leapfrog steps
                    initial_q=c(1000,11), # Chain initialization
                    nsamp=6000, # total number of samples
                    nburn=2000) # number of burn-in samples
#Inspect Chains
df.posterior$id <- 1:4000
ggplot(df.posterior[-(1:3000),],aes(x=id,y=mu_chain))+
  geom_line(size=1.2,color="blue")+
  theme_bw()+xlab("mu chain")
```



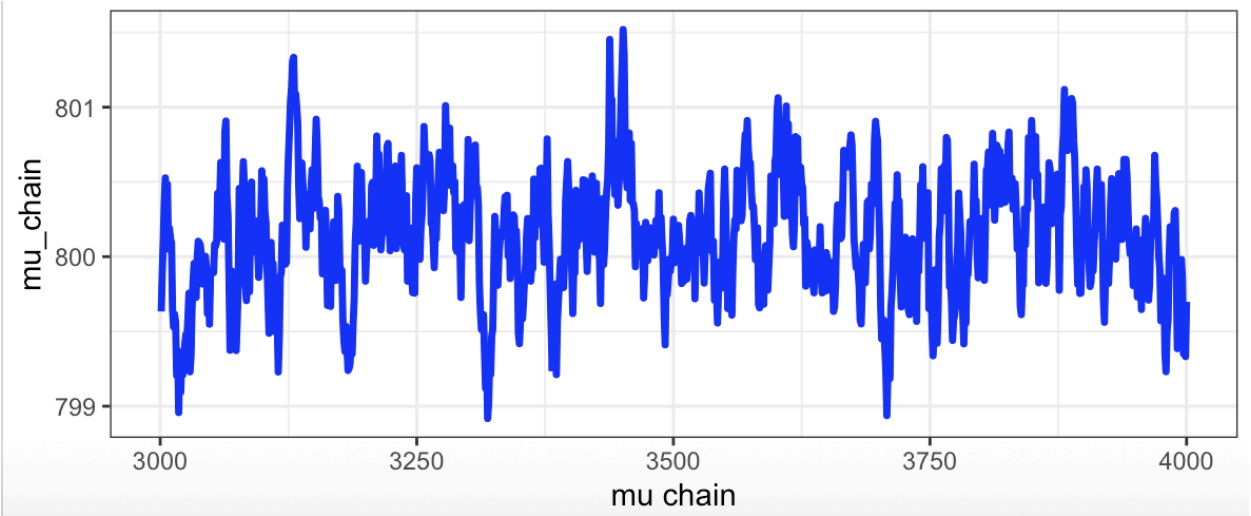


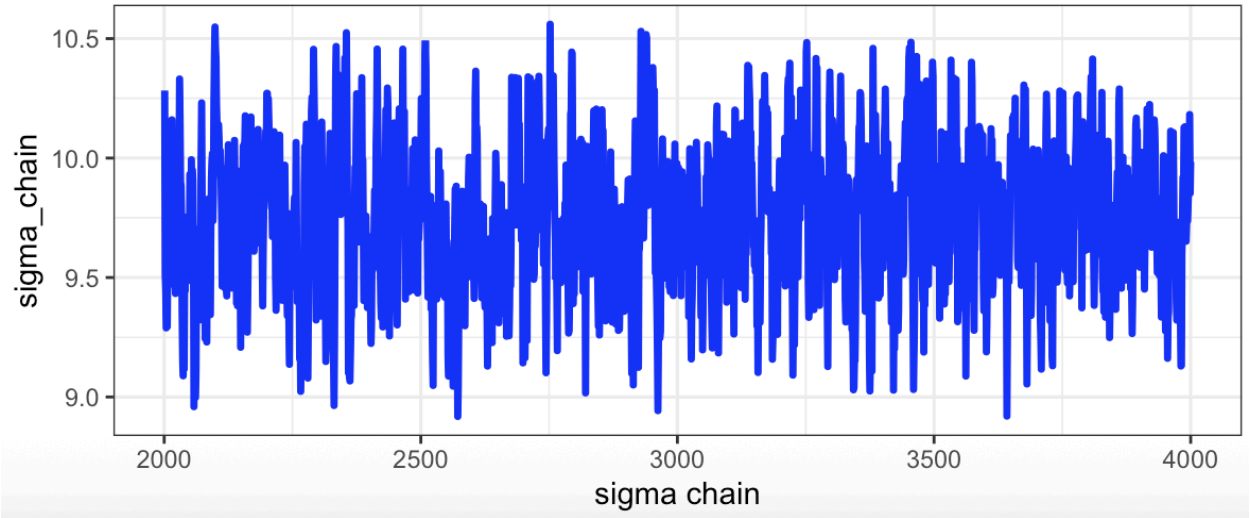
a) Step size = 0.005





b) Step size = 0.02

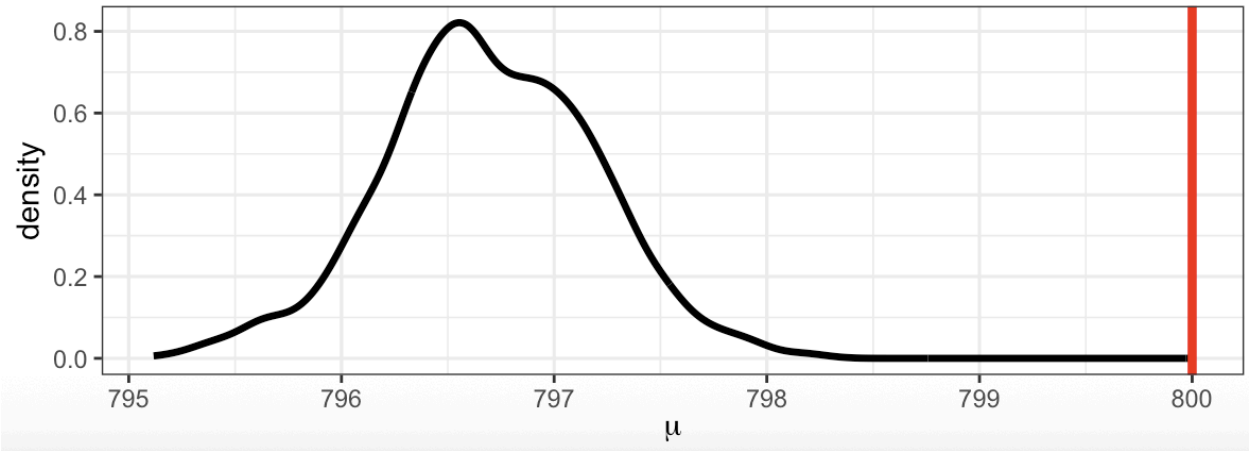




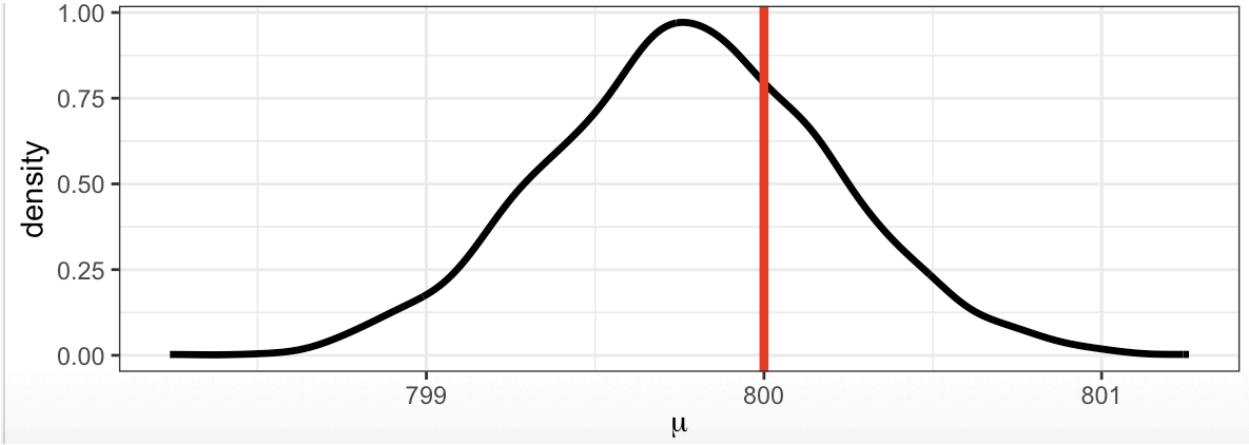
3.5)

```
df.posterior <- HMC(y=y,n=length(y), # data
                    m=400,s=5,a=10,b=2, # priors
                    step=0.02, # step-size
                    L=12, # no. of leapfrog steps
                    initial_q=c(1000,11), # Chain initialization
                    nsamp=6000, # total number of samples
                    nburn=2000) # number of burn-in samples
library(ggplot2)
ggplot(df.posterior[-(1:2000),],aes(x=mu_chain))+
  geom_density(linewidth=1.2)+theme_bw()+xlab(expression(mu))+
  geom_vline(xintercept=800,size=1.5,color="red")
```

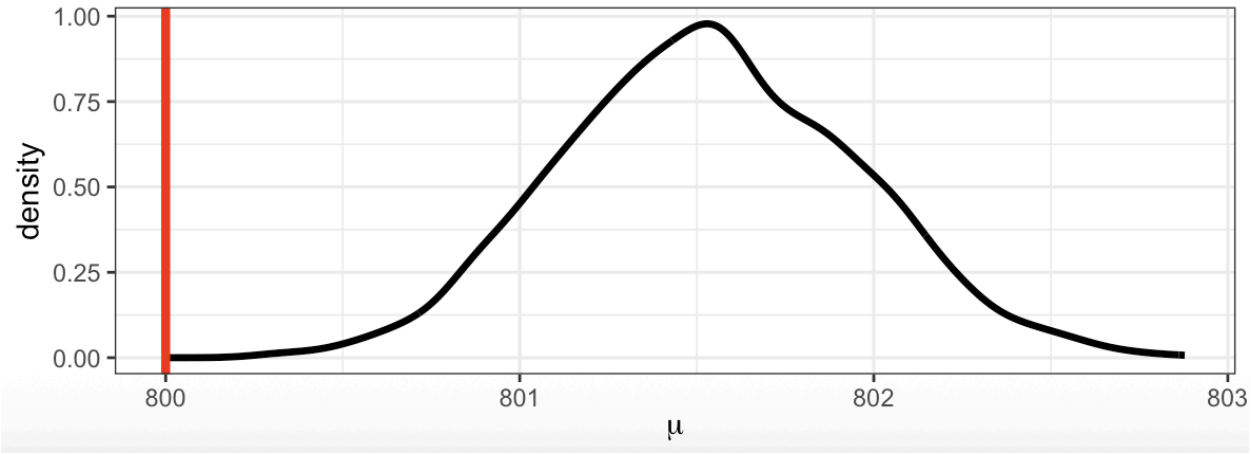
a) μ~Normal(m = 400, s = 5)



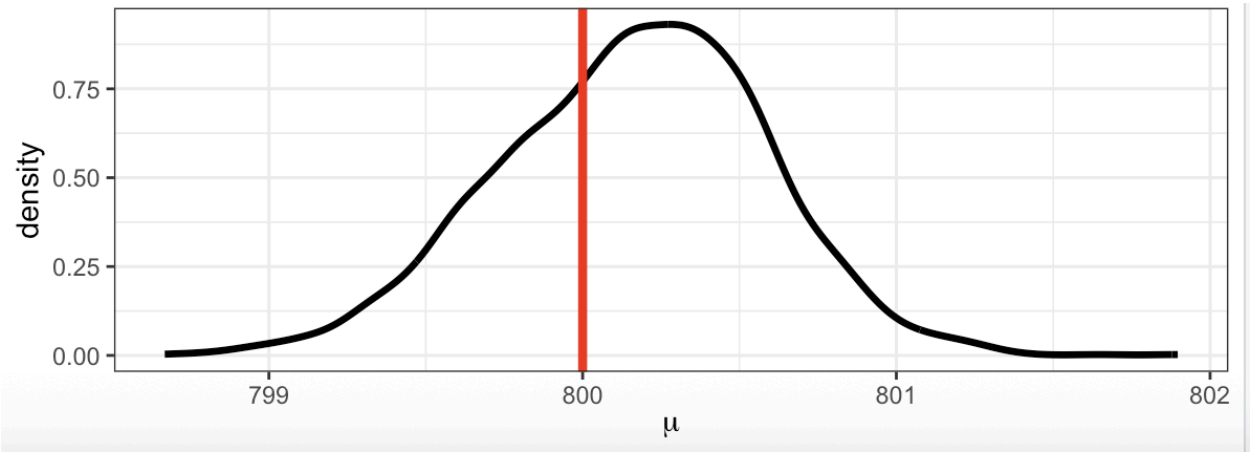
b) μ~Normal(m = 400, s = 20)



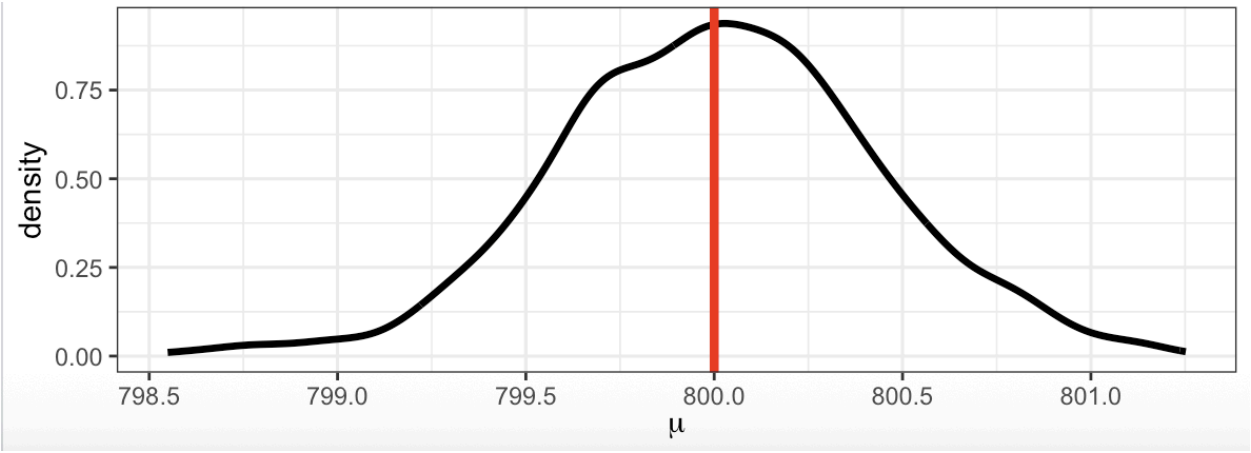
c) μ~Normal(m = 1000, s = 5)



d) μ~Normal(m = 1000, s = 20)



e) μ~Normal(m = 1000, s = 100)



Upon increasing the value of s, the distribution becomes more centered, and the peak is closer to 800.