STA 534 Final Submission

Michael Pena

Spring 2024

Contents

Problem 1 (a) [5 points] 2
Problem 1(b) [3 Points] 2
Problem 2(a) [2 points] 2
Problem 2(b) [3 points] 2
Problem 2 (c) [5 points] 3
Problem 3 6
Problem 3(a) [3 points] 6
Problem 3 (b) [5 points] 6
Problem 3 (c) [5 points] 7
Problem 3 (d) [3 points] 10
Problem 3 (e) [5 points] 11
Problem 4 13
Problem 4 (a) [3 points] 13
Problem 4 (b) [5 points] 14
Problem 4 (c) [3 points] 15

Problem 1 (a) [5 points]

```r
set.seed(2024)
x <- rnorm(10000)

# render weight
w = sin(sin(x))/dnorm(x)
# make weights zero when oob
for(i in 1:10000){
  if(x[i] < 0 | x[i] > (pi/2)){
    w[i] = 0
  }
}
W = w/sum(w)

# sample
x <- sample(x,size = 5000,prob = W,replace = T)
var(x)
```

```
## [1] 0.1459689
```

Problem 1(b) [3 Points]

Suppose that you want to obtain an estimate of the standard error for your estimate in part (a). Explain mathematically what you need to compute, and what quantities would be difficult to compute and why. [2 points for what you need to compute, and 1 point for pointing out possible difficulties.]

Problem 2(a) [2 points]

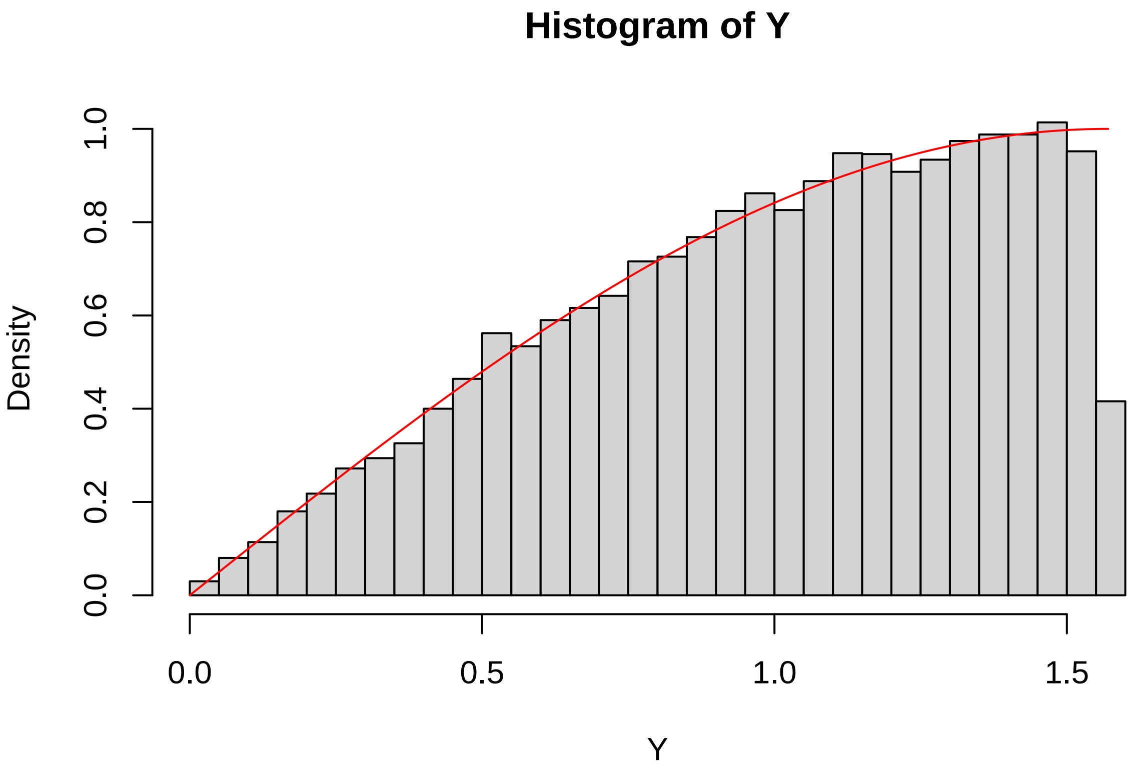
It will be first necessary to find the inverse of the CDF for $Y \sim G$

$$G_X(x) = \int_0^x sin(x)dx = \Big[- cos(x) \Big]_0^x = -cos(x) + 1 G_X^{-1}(y) = arccos(1-y) = x$$

here if $x \in (0, \pi/2)$ then $y \in (0,1)$ so we don't need any restrictions on our image x. We can simply input values for y_i from (0,1) to return an appropriate x_i

Problem 2(b) [3 points]

```r
#render samples
set.seed(2024)
Y <- acos(1 - runif(10000))
# render graphic
hist(Y, freq = FALSE, breaks = 50)
curve(sin(x), 0, pi/2, type = "l", add = TRUE, col="red")
```

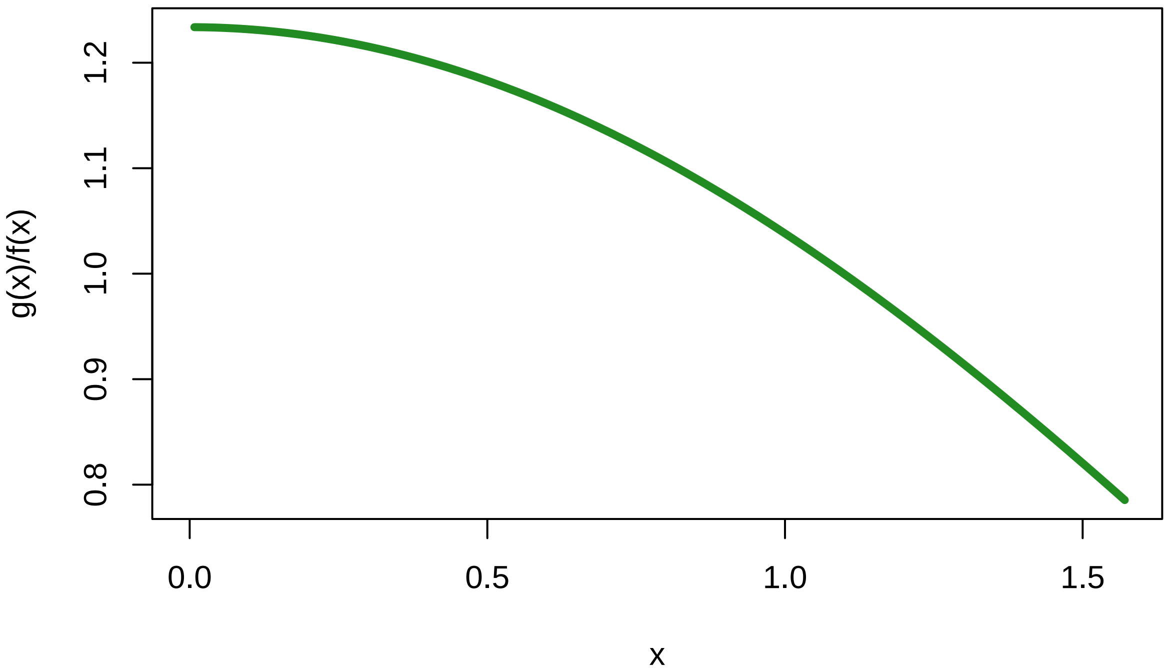


Problem 2 (c) [5 points]

What alpha makes $sin(x)/\alpha \geq 8x/\pi^2$ this true?

$$\alpha \leq \frac{\pi^2 sin(x)}{8x}$$

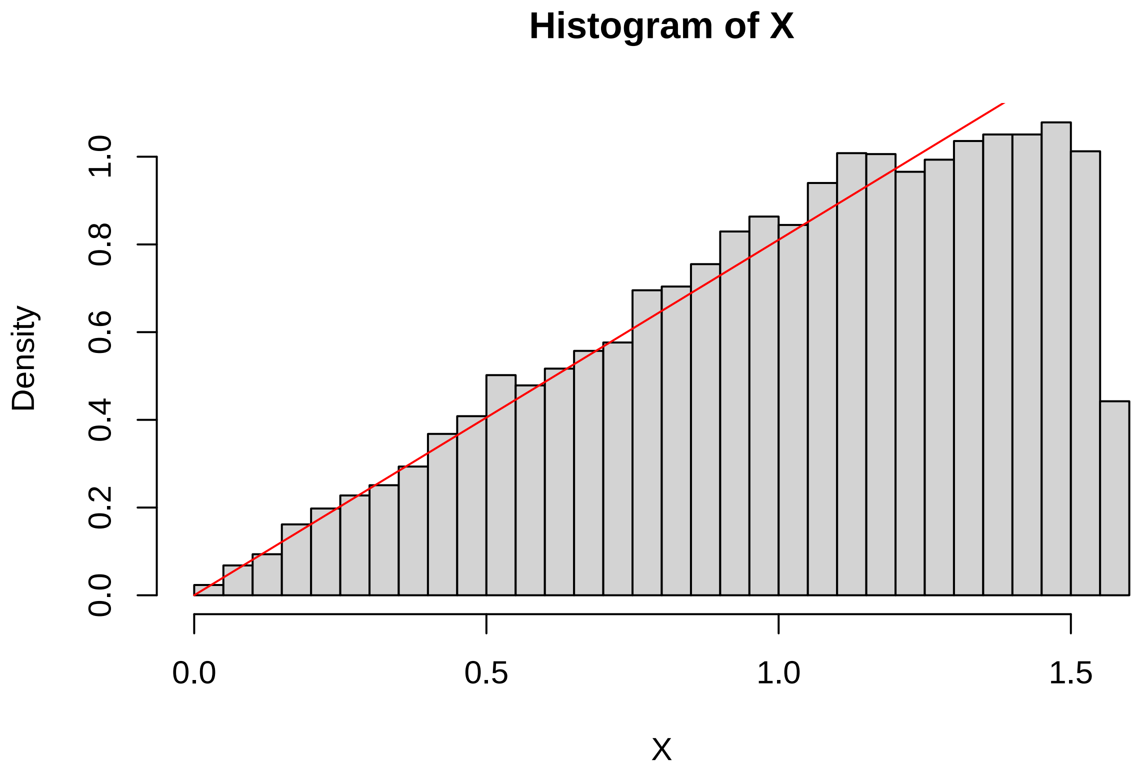
```
# render graph
alF <- function(X){
  (pi^2*sin(X))/(8*X)
}
Xseq <- seq(0,pi/2,length = 200)
plot(Xseq,alF(Xseq),type = "l", col = "forestgreen", lwd = 4,
     xlab = "x", ylab = "g(x)/f(x)")
```



smallest $alpha$ is at $x = \pi/2$

```
# render function
f <- function(x){8*x/pi^2}
# define alpha and vectors
alpha = sin(pi/2)/f(pi/2)
ey = sin(Y)/alpha
fy = f(Y)
Ry = fy/ey
# Accept-Reject
U <- runif(10000,0,alpha)
X <- Y[U < Ry]
```

```
# render graphics
hist(X, freq = FALSE, breaks = 50)
curve(8*x/pi^2, 0, pi/2, type = "l", add = TRUE, col="red")
```




```
# show the amount accepted
sprintf("%f of the 10000 values that were generated were accepted",length(X)/10000)
```

```
## [1] "0.940300 of the 10000 values that were generated were accepted"
```

Problem 3

Consider the trivariate random vector (Y_1, Y_2, Y_3) with the density

$$f(y_1, y_2, y_3) = c \times \exp\left\{-(y_1 + y_2 + y_3 + y_1 y_2 + 2y_2 y_3 + 4y_1 y_3)\right\}; \quad y_1 \geq 0, y_2 \geq 0, y_3 \geq 0.$$

where $c > 0$ is the normalizing constant so that the density integrates to 1.

Problem 3(a) [3 points]

we know that

$$f(y_1, y_2, y_3) = c \cdot e^{-(y_1+y_2+y_3+y_1 y_2+2y_2 y_3+4y_1 y_3)}$$

$$f(Y_1|Y_2, Y_3) = \frac{c \cdot e^{-(y_1+y_2+y_3+y_1 y_2+2y_2 y_3+4y_1 y_3)}}{c \cdot e^{-(y_2+y_3+2y_2 y_3)}} = e^{-y_1(1+y_2+4y_3)} f(Y_2|Y_1, Y_3) = \frac{c \cdot e^{-(y_1+y_2+y_3+y_1 y_2+2y_2 y_3+4y_1 y_3)}}{c \cdot e^{-(y_1+y_3+4y_1 y_3)}} = e^{-y_2(1+y_1}$$

the above are kernels for the below densities

$$Y_1 \sim Exp(\frac{1}{1 + y_2 + 4y_3}) Y_2 \sim Exp(\frac{1}{1 + y_1 + 2y_3}) Y_3 \sim Exp(\frac{1}{1 + 2y_2 + 4y_1})$$

Problem 3 (b) [5 points]

```
Gibbs <- function(N,Y0,nburn,see.d){
 # render intials
 Yi <- matrix(0,N,3)
 set.seed(see.d)

 # render loop
 for(i in 1:N){
  # render new Yi's
  Yi[i,1] <- rexp(1,rate = 1 + Y0[2] + 4*Y0[3])
  Yi[i,2] <- rexp(1,rate = 1 + Y0[1] + 2*Y0[3])
  Yi[i,3] <- rexp(1,rate = 1 + 2*Y0[2] + 4*Y0[1])

  # pass this vector back into Y0
  Y0 <- Yi[i,]
 }

 # burn off
 Yi <- Yi[(nburn+1):N,]

 # form mu and Sigma
 MU <- c(mean(Yi[,1]),mean(Yi[,2]),mean(Yi[,3]))
 v <- matrix(c(1,1,
              1,2,
              1,3,
              2,1,
              2,2,
              2,3,
              3,1,
              3,2,
              3,3), ncol = 2, nrow=9, byrow = T)
```

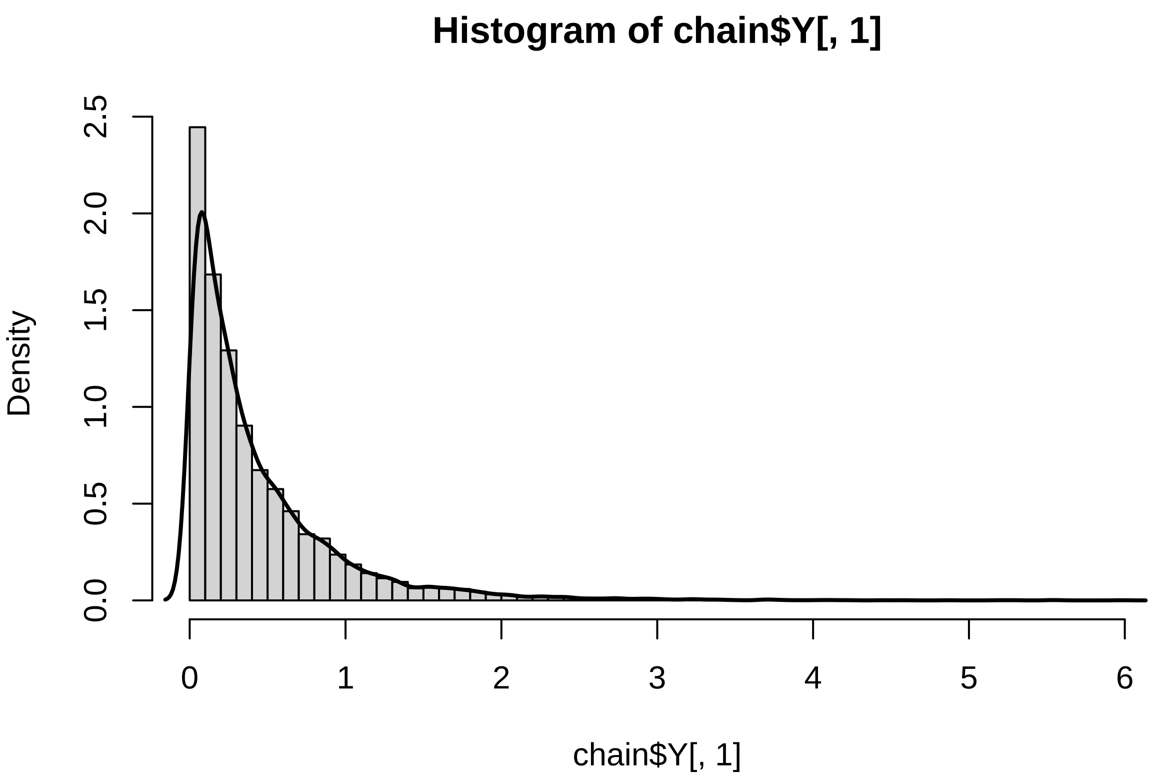
```
 SIG <- matrix(0,3,3)
 for(i in 1:9){
  SIG[v[i,1],v[i,2]] <- cov(Yi[,v[i,1]],Yi[,v[i,2]])
 }
 return(list(Y = Yi, mu = MU, Sigma = SIG))
}

#
y0 <- c(1,1,1)
Gibbs(10000,y0,1000,2024) -> chain
chain$mu
```

```
## [1] 0.4381503 0.5459522 0.3621343
```

```
chain$Sigma
```

```
##            [,1]       [,2]       [,3]
## [1,] 0.26786143 0.02809556 0.01087738
## [2,] 0.02809556 0.36578126 0.01497441
## [3,] 0.01087738 0.01497441 0.22849207
```

```
# graphics
hist(chain$Y[,1],prob = T,breaks = 50)
lines(density(chain$Y[,1]), col =1, lwd =2)
```



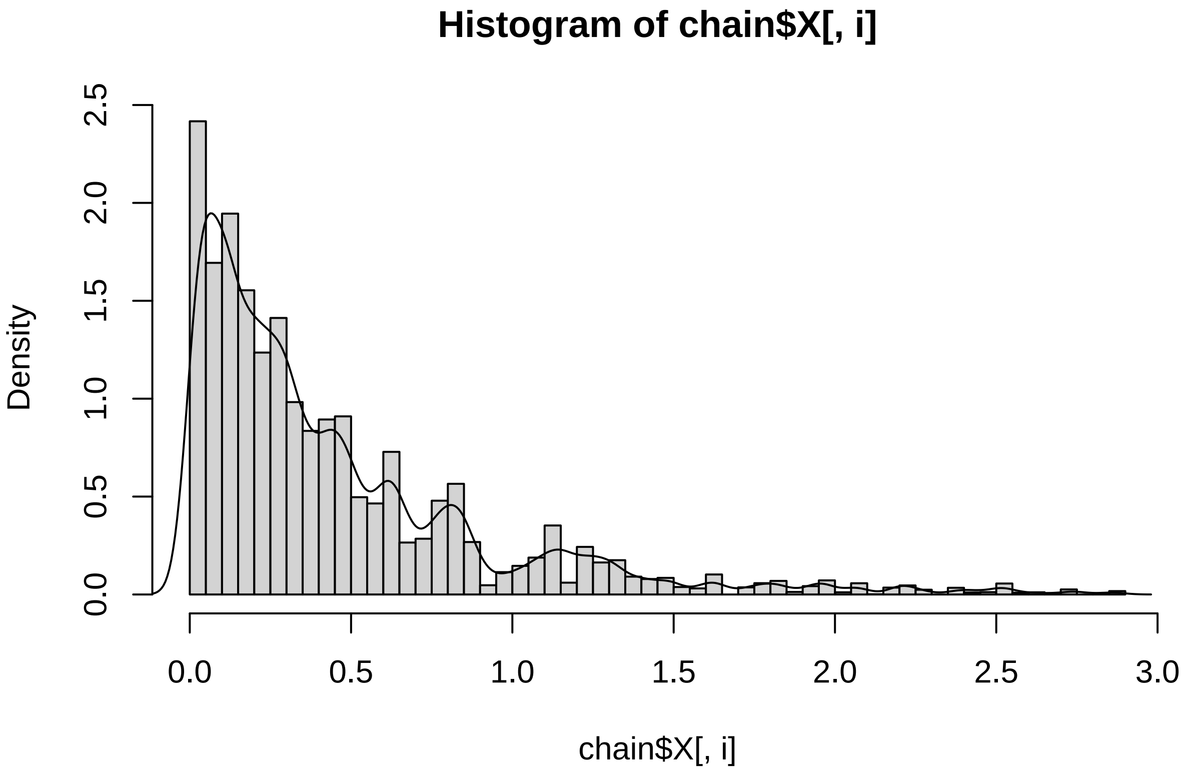
Problem 3 (c) [5 points]

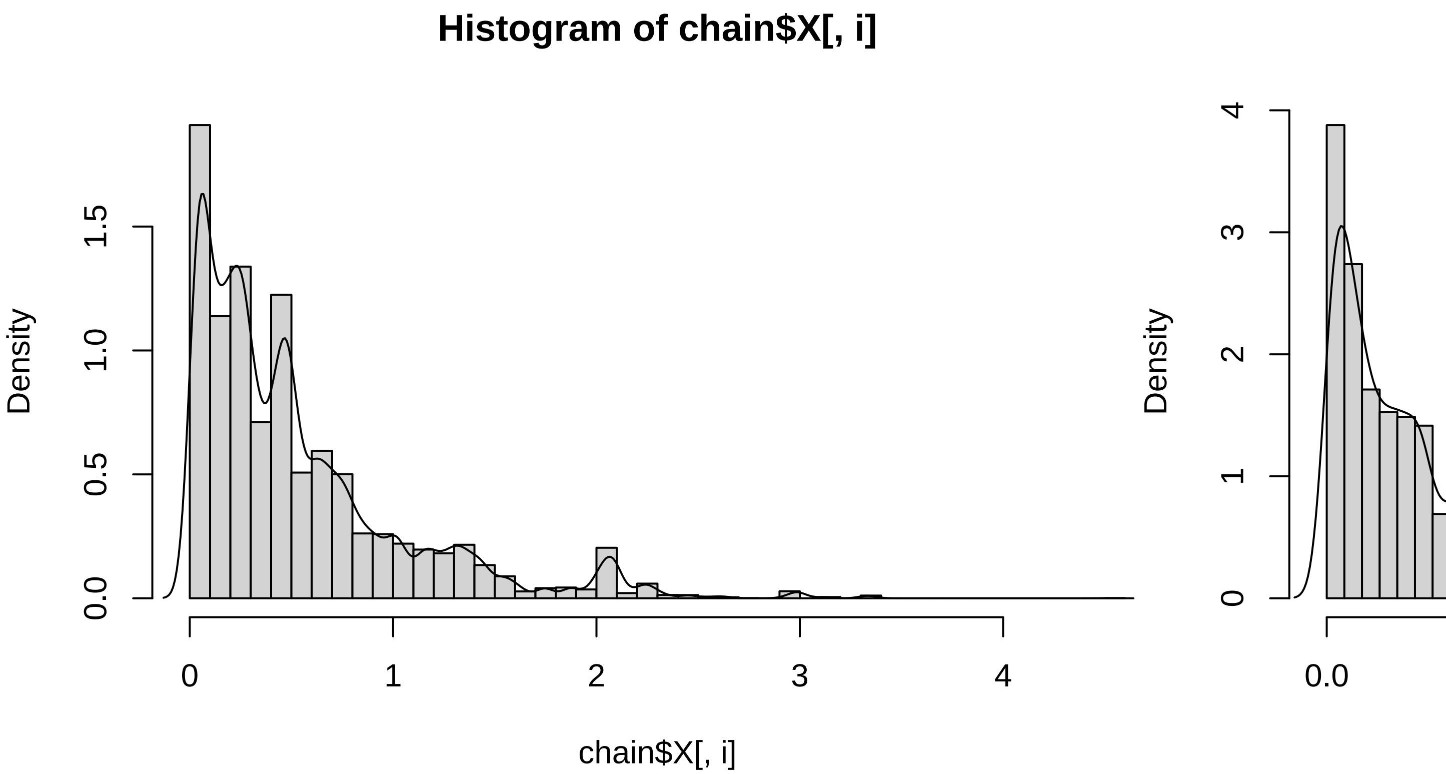
```
P <- function(X){
  # get xi's
  x1 <- X[1]
```

```
  x2 <- X[2]
  x3 <- X[3]
  # input c
  c = 6.33741006308
  # define math
  c * exp(-(x1 + x2 + x3 + x1*x2 + 2*x2*x3 + 4*x1*x3))
}

MH <- function(N,X0 = c(1,1,1),S = 2*diag(3),see.d = 2024, nburn = 5000){
 # initialize store vector
 X <- matrix(0,N,3)
 X.bar <- X

 # set seed
 set.seed(see.d)

 # make first iteration
 X[1,] <- mvrnorm(1,X0,S)
 accept = 0

 for(i in 2:N){
  # sample from mvrnorm
  x1 = mvrnorm(1,X[i-1,],S)
  # stay in support
  if(x1[1] < 0 | x1[2] < 0 | x1[3] < 0){
    alpha = 0
  } else{
  u = runif(1)
  A = x1[1] + x1[2] + x1[3] + x1[1]*x1[2] + 2*x1[2]*x1[3] + 4*x1[1]*x1[3]
  B = X[i-1,1] + X[i-1,2] + X[i-1,3] + X[i-1,1]*X[i-1,2] + 2*X[i-1,2]*X[i-1,3] + 4*X[i-1,1]*X[i-1,3]
  alpha = exp(-A+B)
  }


  # check condition
  if(u <= alpha){
   X[i,] = x1
   accept = accept + 1
  } else {
   X[i,] = X[i-1,]
  }

  # calc Xbar for iteration
  X.bar[i,] = mean(X[1:i,])
 }

 # burn items
 X = X[(nburn+1):N,]
 X.bar = X.bar[(nburn+1):N,]

 # return stats
 list(X=X, means = X.bar, ratio = accept/N)
}
```

```r
# render asks
chain <- MH(50000)
```

```r
# doing the graphics
for(i in 1:3){
 hist(chain$X[,i],breaks = 50, prob = T)
 lines(density(chain$X[,i]))
}
```

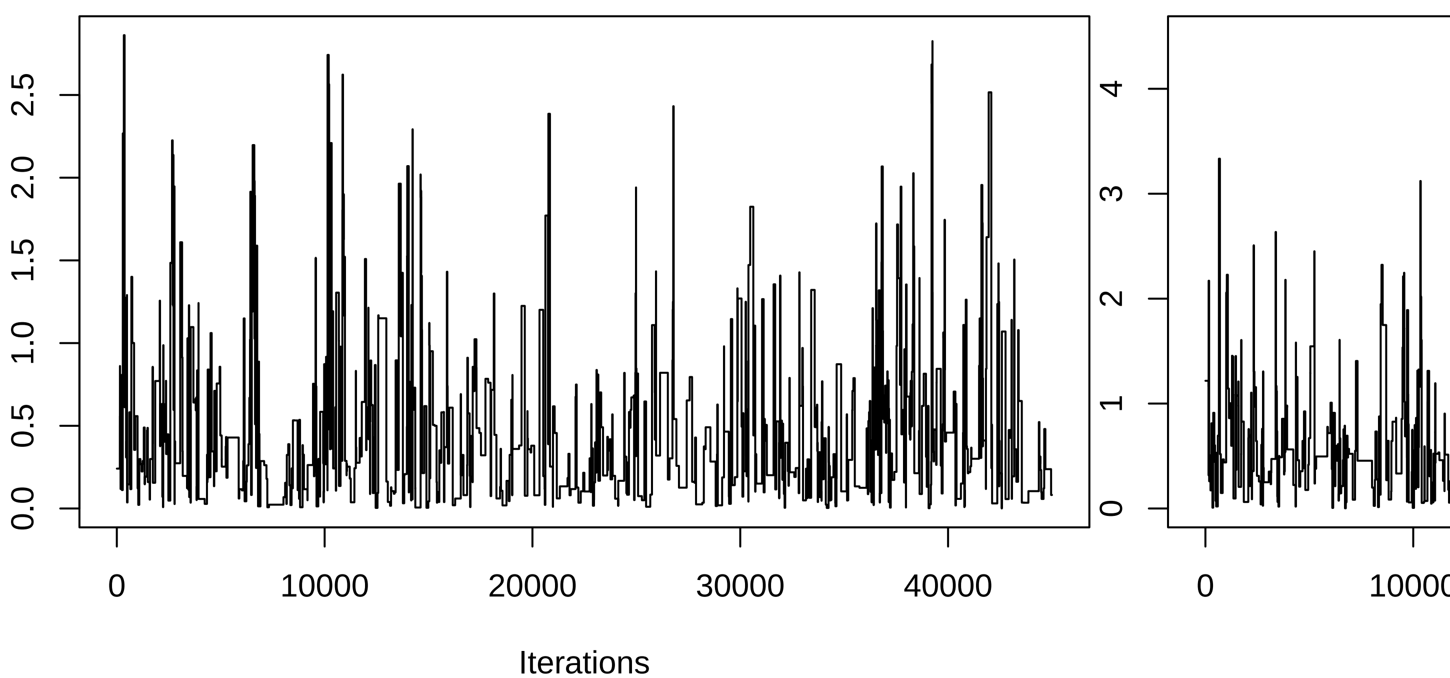


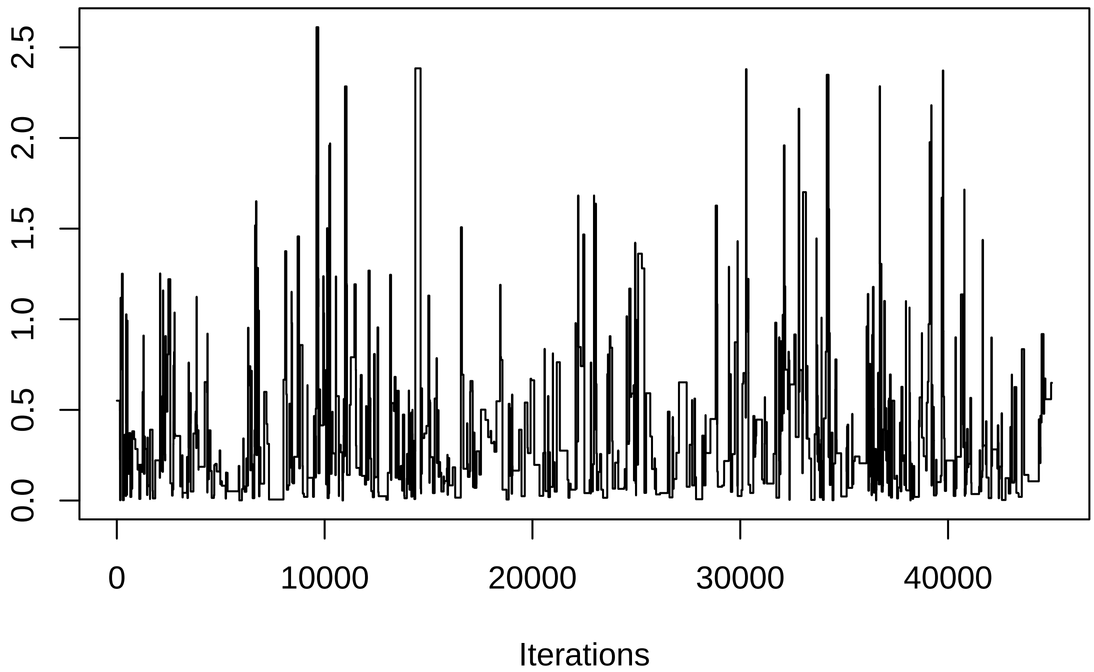


Problem 3 (d) [3 points]

Use the library `coda` to print the summary of the chain, and the trace plots for the chain (use the function `traceplot()`. Explain why the chain is not mixing well.

```
# render graphics
traceplot(mcmc(chain$X))
```



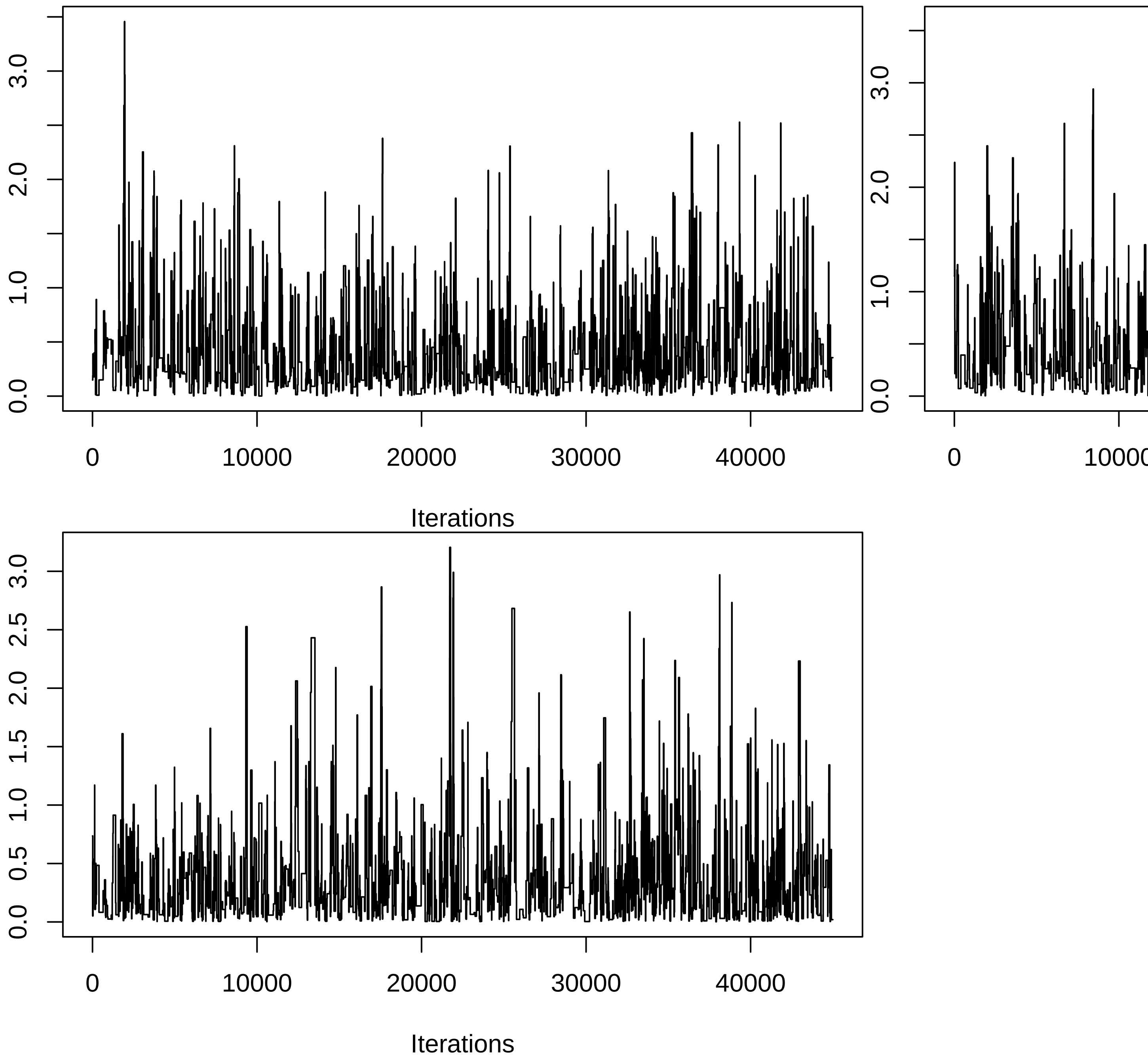


Problem 3 (e) [5 points]

```
D <- diag(3)*1
chain2 <- MH(50000,S = D)
summary(mcmc(chain2$X))
```

```
##
## Iterations = 1:45000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 45000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##        Mean     SD Naive SE Time-series SE
## [1,] 0.3986 0.4171 0.001966        0.01574
## [2,] 0.5085 0.5207 0.002454        0.02386
## [3,] 0.3801 0.4782 0.002254        0.02420
##
## 2. Quantiles for each variable:
##
##          2.5%    25%    50%    75% 97.5%
## var1 0.008651 0.1213 0.2500 0.5422 1.613
## var2 0.012160 0.1432 0.3470 0.7159 1.940
## var3 0.005414 0.0779 0.2236 0.4819 1.964
```

```
traceplot(mcmc(chain2$X))
```




```
chain2$ratio
```

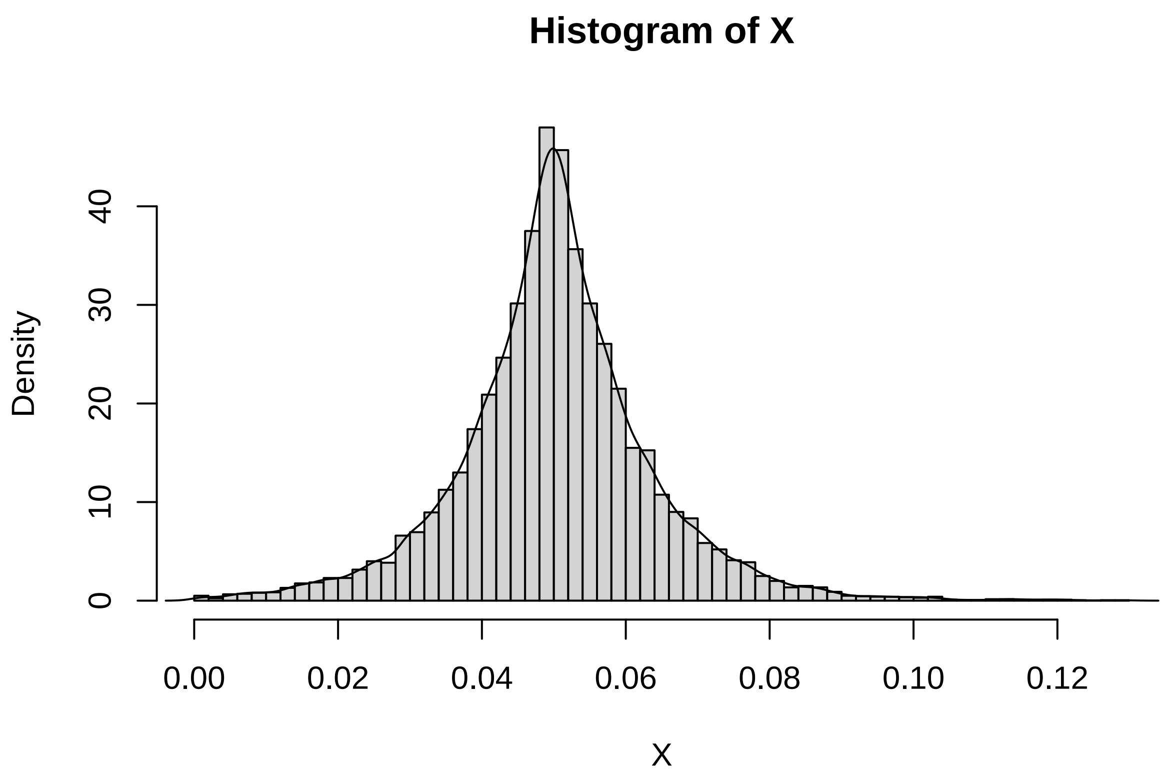
```
## [1] 0.03594
```

Problem 4

Problem 4 (a) [3 points]

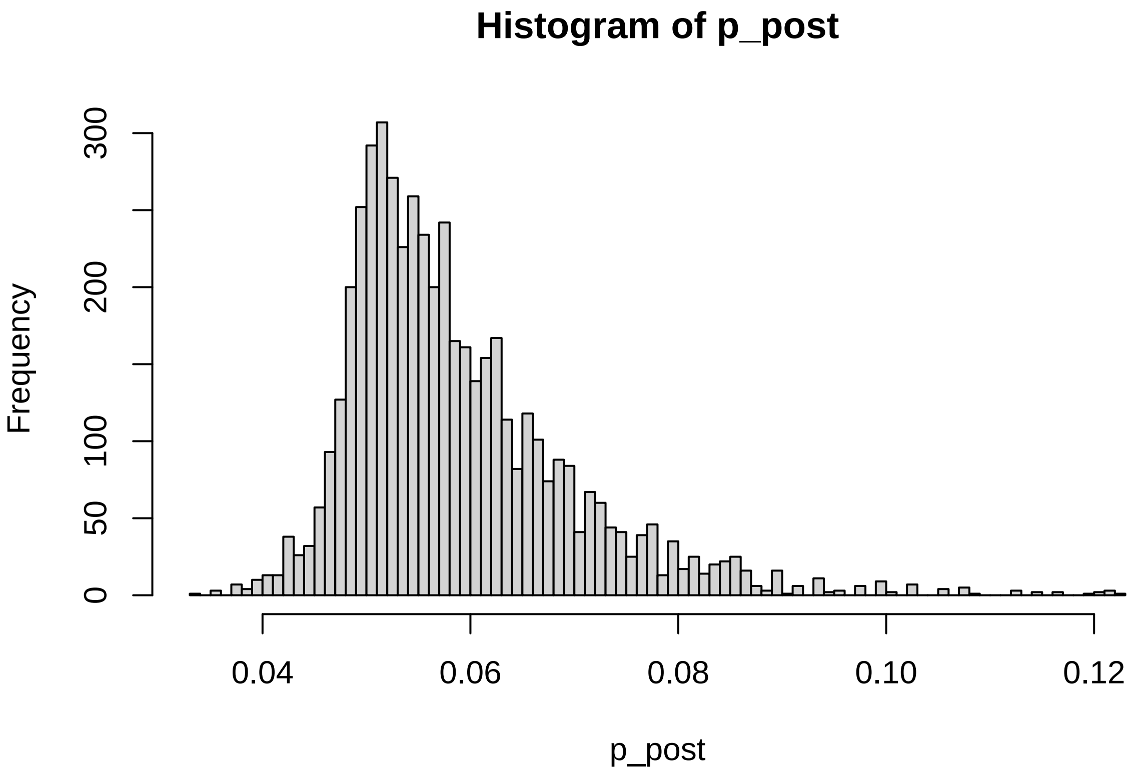
```r
# render of the prior
f <- function(theta){
  C <- 50/(1 - (exp(-95)+exp(-5))/2)
  C*exp(-100 * abs(theta - 0.05))
}

# generating from exponential
set.seed(2024)
X <- rexp(11000, rate = 100)

# address absolute value
for(i in 1:11000){
  u <- runif(1)
  if(u < 0.5){
   X[i] = -1*X[i]
  } else {
   # nothing
  }
}

# shift to the right by 0.05
X <- 0.05 + X

# clip from supports
X = X[X <= 1 & X >= 0]

# truncate
X <- X[1:10000]
```

```r
# render graphics
hist(X,breaks = 50,prob =T)
lines(density(X))
```



Problem 4 (b) [5 points]

```
set.seed(2024)
# making weights
W <- f(X)/dexp(X,rate = 100)
W.stand <- W/sum(W)
p_prior <- sample(X,size = 5000, replace = T, prob = W.stand)

# from SIR_Jim Albert.R
w = (p_prior^4*(1-p_prior)^40)
w = w/sum(w)
p_post = sample(p_prior,size=5000,replace=TRUE,prob=w)

# render graphics
hist(p_post,freq = T,breaks = 100)
```



Problem 4 (c) [3 points]

```r
# from SIR_Jim Albert.R
py = mean(dbinom(4,size = 50, p_post))
#statement
sprintf("The probability that Kevin Mitchell makes 4 home runs when he is at bat 50 times is %f",py)
```

```
## [1] "The probability that Kevin Mitchell makes 4 home runs when he is at bat 50 times is 0.161072"
```