com_stats_lab04_group14

Akshath Srinivas, Samira Goudarzi

2022-12-03

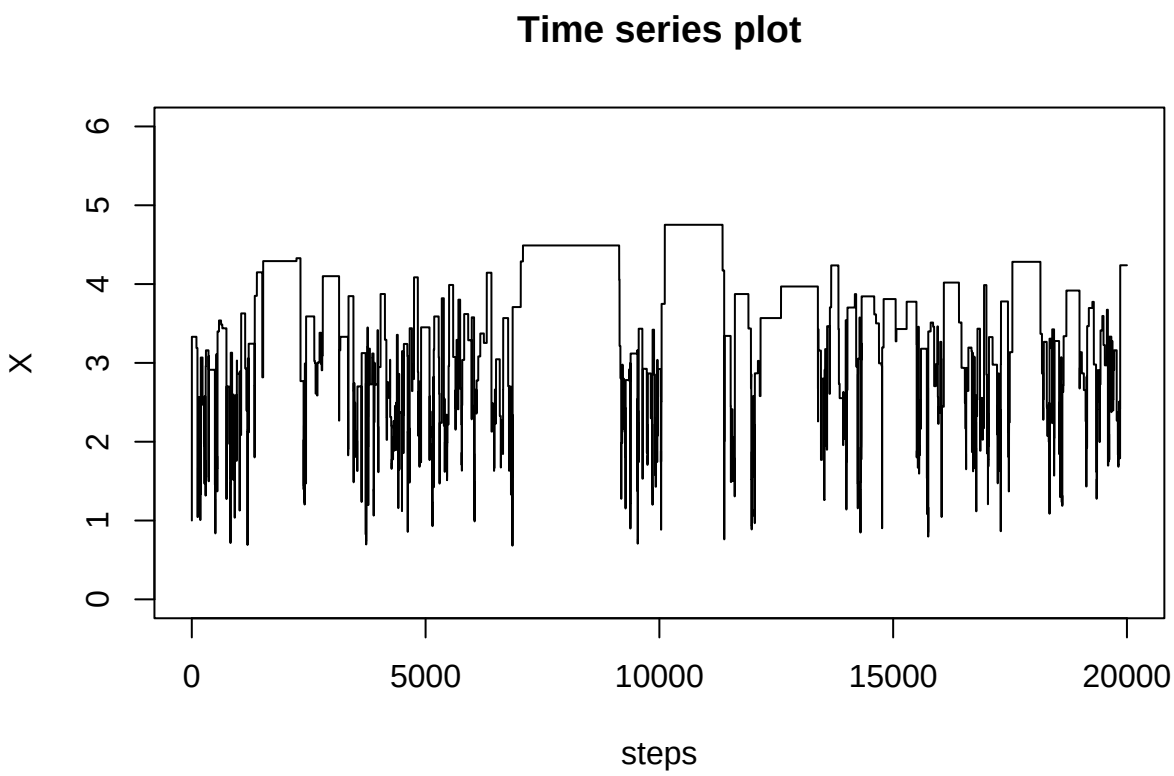
Lab 4

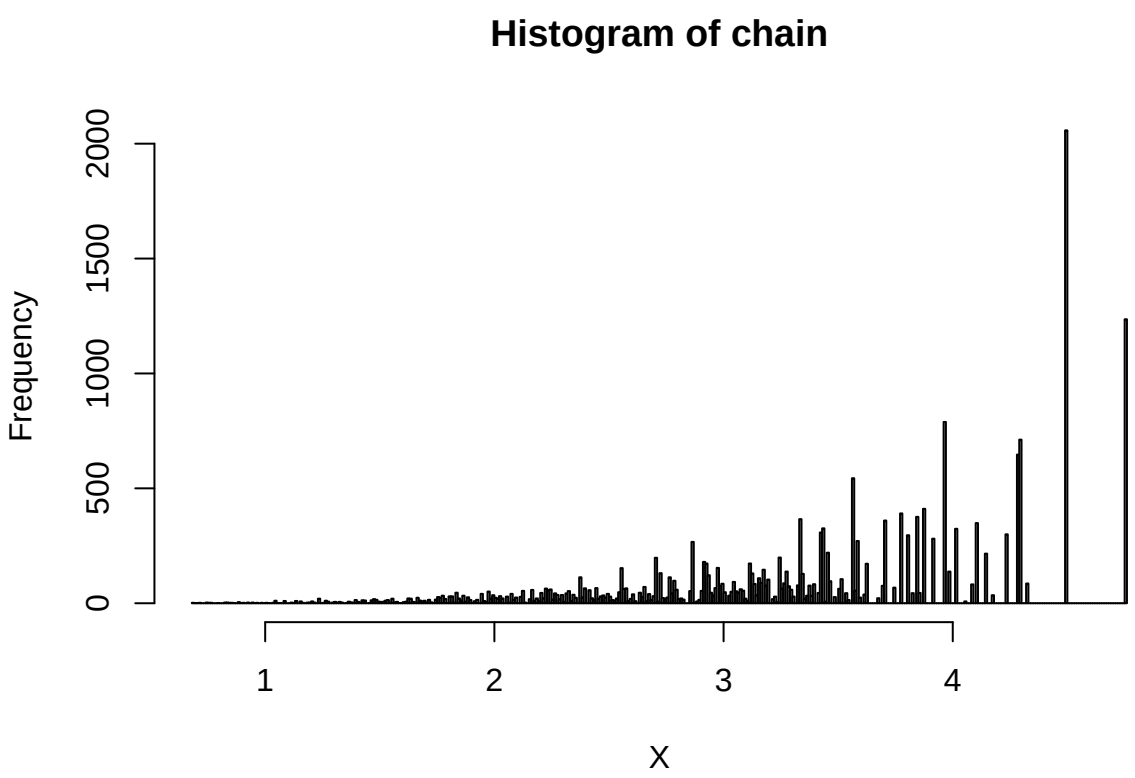
Question 1: Computations with Metropolis–Hastings

subquestion 1

Given Target density function

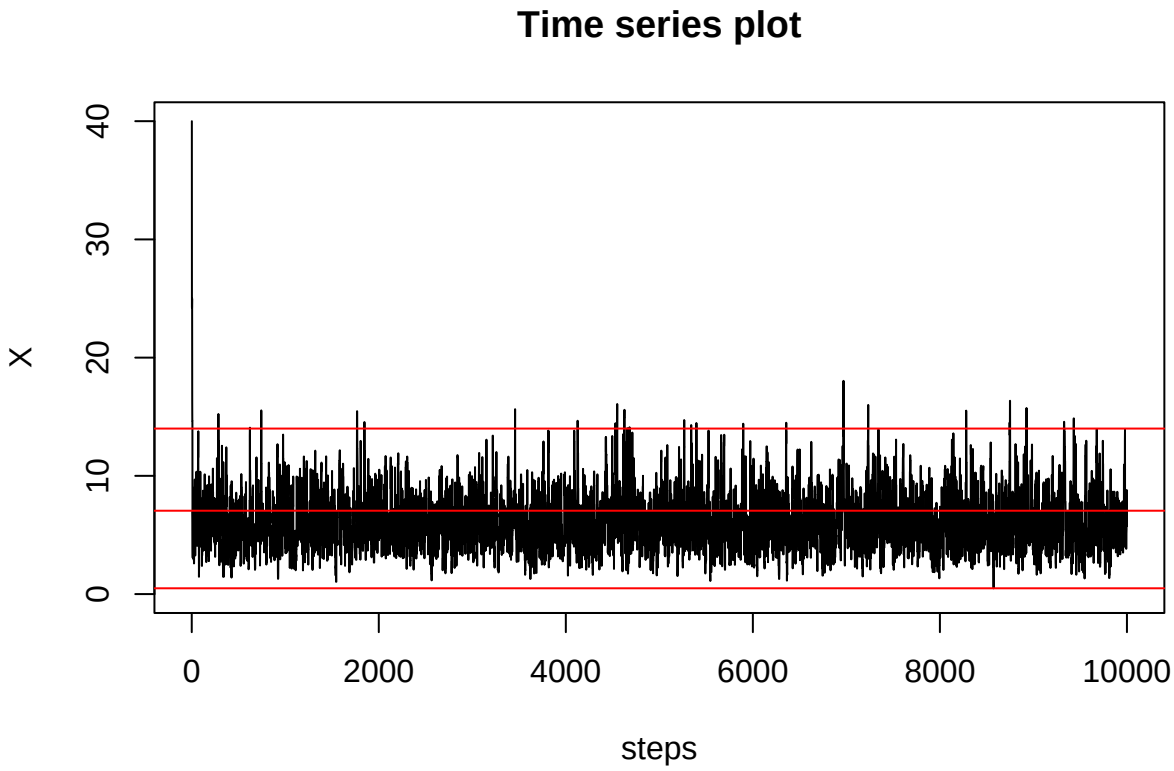
$$f(x) \propto x^5 e^{-x} = \int_0^\infty \frac{x^5 e^{-x}}{120} dx$$

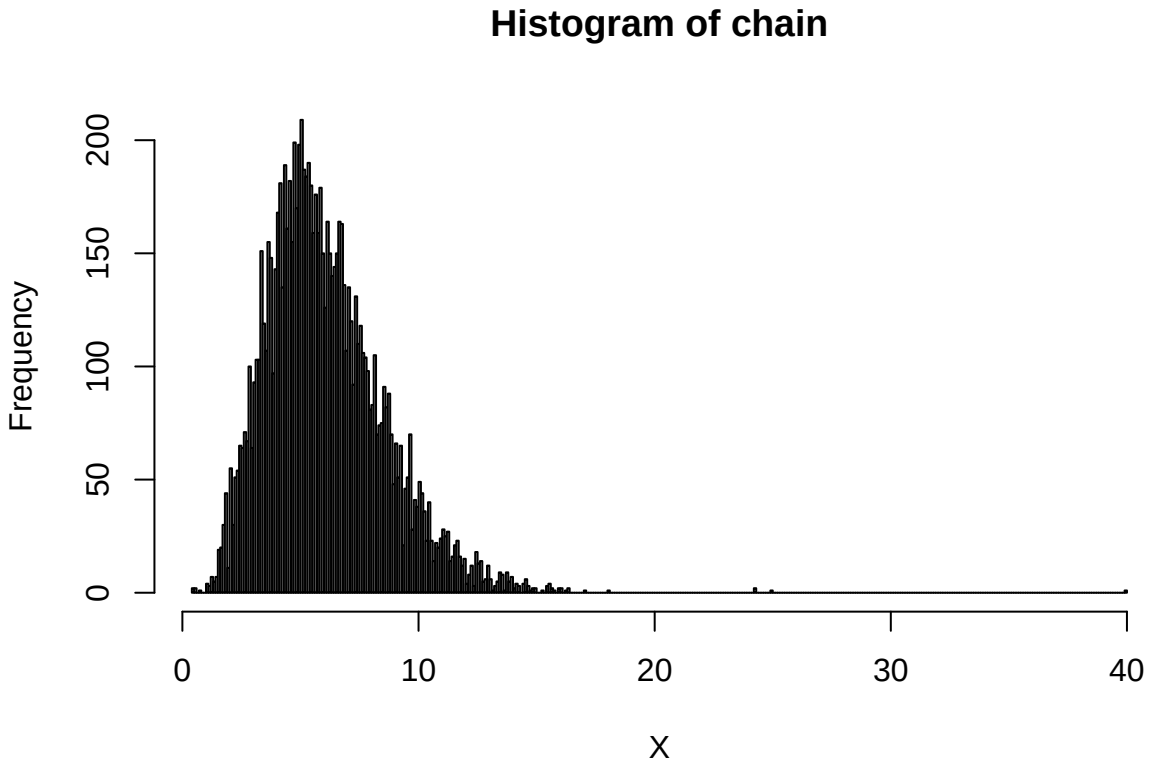




According to markov chain theory, We can guess the chain will finally converge at some point. But from the plot we cannot observe the convergence and will not be able to calculate the size of burn-in period.

subquestion 2





subquestion 3

The algorithms in step 1 and 2 uses different proposal distribution to get the sample values.

From the first sub question time series plot and histogram we can conclude say that algorithm drops lot of x values(drop rate is more) and have high values using **log-normal** distribution, and the histogram does not match the target distribution.

Using **chi-squared** distribution as proposal distribution will normalize the data which avoids high values and reduce dropout rate. The histogram shown above also matches with the target distribution. This chain converges faster than the chain in sub question1 which we can see from the time series plot and also we can see the burn in period visually in the plot.

subquestion 4

Convergence monitoring: Gelman-Rubin method

step 1: We have generated 10(K) MCMC sequences of length 10000(n) with different starting points. To obtain reliable Gelman-Rubin factor, the MCMC chains should be started from different points in the parameter space to get over dispersed values.

step 2: We computed between and within sequence variances by below equations.

$$B = \frac{n}{(m-1)} \sum_{i=1}^{m} (\bar{\mu} - \mu)^2$$

$$s_i^2 = \frac{1}{(n-1)} \sum_{j=1}^{m} (\theta_{ij} - \mu_i)^2$$

$$W = \frac{1}{m} \sum_{i=1}^{m} s_i^2$$

step 3: Overall variance estimate is given by

$$\hat{var}(\theta) = (\frac{n-1}{n})W + \frac{1}{n}B$$

step 3: Gelman–Rubin factor can be calculated by

$$\sqrt{\hat{R}} = \sqrt{\frac{\hat{var}(\theta)}{W}}$$

After solving all the equations(see the code in appendix) we got the Gelman-Rubin factor value as below.The closer the $\hat{R}$ to 1, we can say convergence is achieved. The value we got above is close to 1. So, we can conclude that convergence is attained. Values much larger than 1 indicate lack of convergence.

```
## Gelman-Rubin factor calculated: 0.99995
```

We created the MCMC list with 10 MCMC chains with different starting point and inputted this list to the gelman.daiag() function, we get the psrf values as below which includes potential scale reduction factor(labelled point est.) and Upper Confidence interval(labelled upper C.I). In practice, these values <1.1 or 1.2 are considered as acceptable and conclude that convergence is achieved.

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

subquestion 5

The equation to estimate in this question is expected value of step 1 and 2 algorithm samples.

$$E[f(x)] = \int_0^{+\infty} xf(x) = \frac{1}{n}\sum_{i=1}^{\infty} f(x_i)$$

So, we have to find out mean of the samples from step 1 and step 2.

```
## Mean or Expected value for step 1 algorithm samples: 3.509073
```

```
## Mean or Expected value for step 2 algorithm samples: 5.977739
```

subquestion 6

The Probability density function of gamma distribution is given by.

$$\frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{\frac{-x}{\theta}}$$

$$k, \theta > 0$$

k is shape parameter and θ is scale parameter

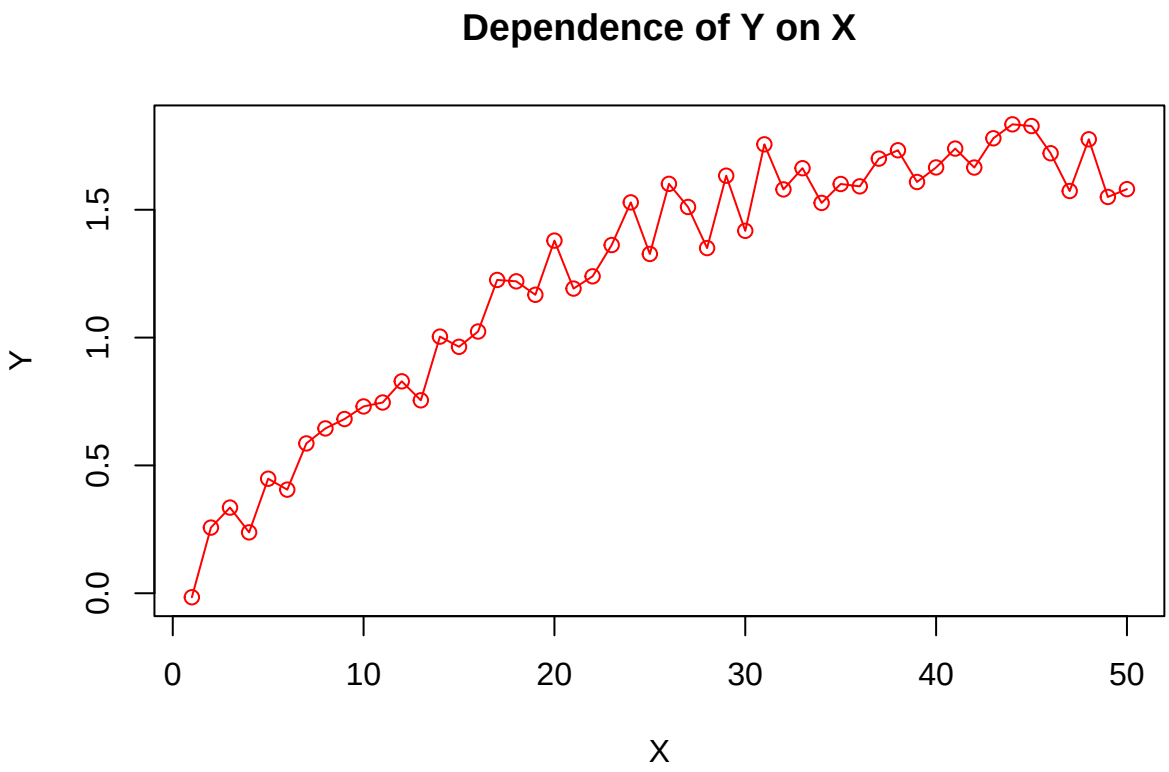
Mean of the Gamma distribution is given by $k\theta$

The density function $f(x) == \int_0^{\infty} \frac{x^5 e^{-x}}{120} dx$ which we took as target distribution can be obtained by substituting $\theta = 1$,$k = 6$ and $\Gamma(6) = 120$ to the above gamma PDF equation. Also we know that mean of the distribution can be calculated by $k\theta$, after substituting the k and θ we get the mean as 6 which matches the mean of the proposal distribution (chi-squared in sub question 2) calculated above.

Question 2: Gibbs Sampling

subquestion 1

Below we can see the plot for dependence of Y on X, by observing the plot we can say that linear model would be reasonable for our dataset.



subquestion 2

Given that

$$Y_i \sim N(\mu_i, variance = 0.2)$$

Where

$$i = 1, 2, ..., n$$

The likelihood is given by.

$$P(Y|\mu) = \frac{1}{(\sigma\sqrt{2\pi})^n} \exp(-\frac{\sum_{i=1}^{n}(y_i - \mu_i)^2}{2\sigma^2})$$

The prior is given by.

$$p(\mu_1) = 1$$
$$P(\mu_{i+1}|\mu_i) = N(\mu_i, 0.2)$$
$$i = 1, 2, ..., n$$

By chain rule we get.

$$P(\mu) = P(\mu_1)\prod_{i=1}^{n-1} P(\mu_{i+1}|\mu_i) = \frac{1}{(2\pi\sigma^2)^{n-1}}\exp(-\frac{\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2}{2\sigma^2})$$

subquestion 3

Using Bayes Theorem m to get the posterior up to a constant proportionality.

$$P(\mu|Y) \propto P(Y|\mu).P(\mu)$$
$$= \frac{1}{(\sigma\sqrt{2\pi})^n}\exp(-\frac{\sum_{i=1}^{n}(y_i-\mu_i)^2 + \sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2}{2\sigma^2})$$

For i=1.

$$P(\mu_1|Y,\mu_{-1}) \propto \exp(-\frac{(y_1-\mu_1)^2+(\mu_2-\mu_1)^2}{2\sigma^2}) = \exp(-\frac{(\mu_1-y_1)^2+(\mu_1-\mu_2)^2}{2\sigma^2})$$

From Hint B we can write this as

$$P(\mu_1|Y,\mu_{-1}) \propto \exp(\frac{-1}{(2\frac{\sigma^2}{2})}(\mu_1 - \frac{y_1+\mu_2}{2})^2)$$

Therefore μ_1 has a normal distribution with mean $= \frac{\mu_2+y_1}{2}$ and variance $= \frac{\sigma^2}{2} = \frac{0.2}{2} = 0.1$

For i=n.

$$P(\mu_n|Y,\mu_{-n}) \propto \exp(-\frac{(y_n-\mu_n)^2+(\mu_n-\mu_{n-1})^2}{2\sigma^2}) = \exp(-\frac{(\mu_n-y_n)^2+(\mu_n-\mu_{n-1})^2}{2\sigma^2})$$

From Hint b we can write this as.

$$P(\mu_n|Y,\mu_{-n}) \propto \exp(\frac{-1}{(2\frac{\sigma^2}{2})}(\mu_n - \frac{y_n+\mu_{n-1}}{2})^2)$$

Therefore μ_n has normal distribution with mean $= \frac{\mu_{n-1}+y_n}{2}$ and variance $= \frac{\sigma^2}{2} = \frac{0.2}{2} = 0.1$

For other part of i (neither 1 nor n), we get.

$$P(\mu_i|Y,\mu_{-i}) \propto \exp(-\frac{(\mu_i-y_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_i-\mu_{i+1})^2}{2\sigma^2})$$

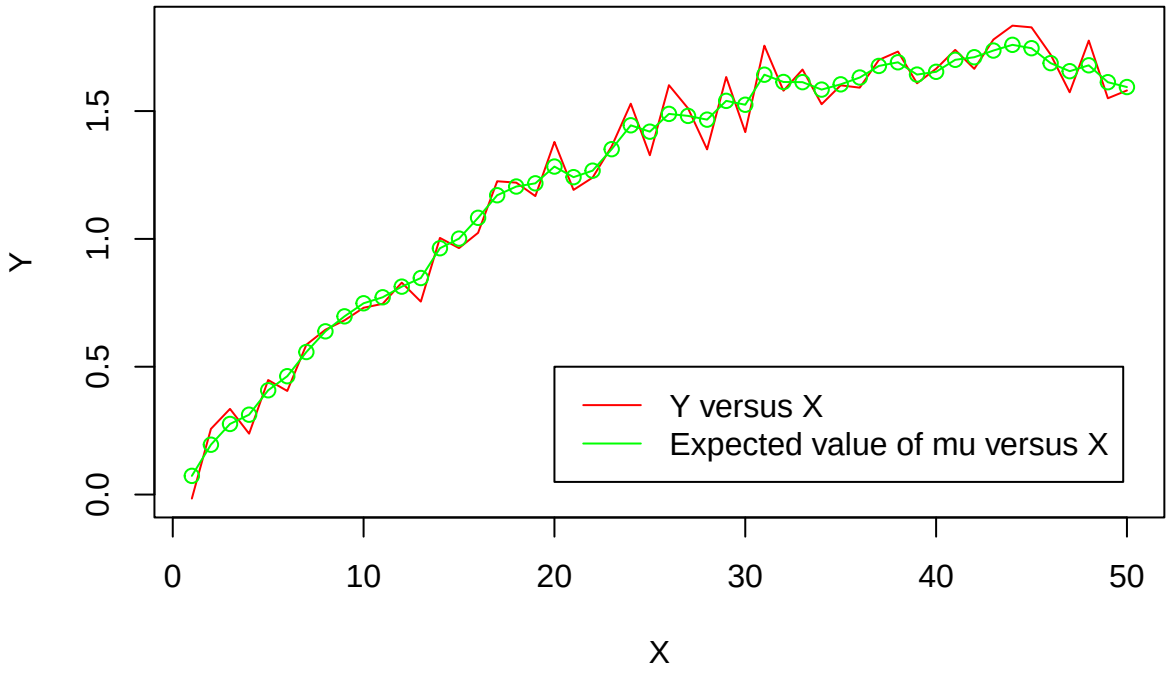
From Hint C, we can write it as below.

$$P(\mu_i|Y,\mu_{-i}) \propto \exp(\frac{-1}{(2\frac{\sigma^2}{3})}(\mu_i - \frac{\mu_{i-1}+\mu_{i+1}+y_i}{3})^2)$$

where mean $= \frac{\mu_{i-1}+\mu_{i+1}+y_i}{3}$ and variance $= \frac{\sigma^2}{3} = \frac{0.2}{3}$

subquestion 4

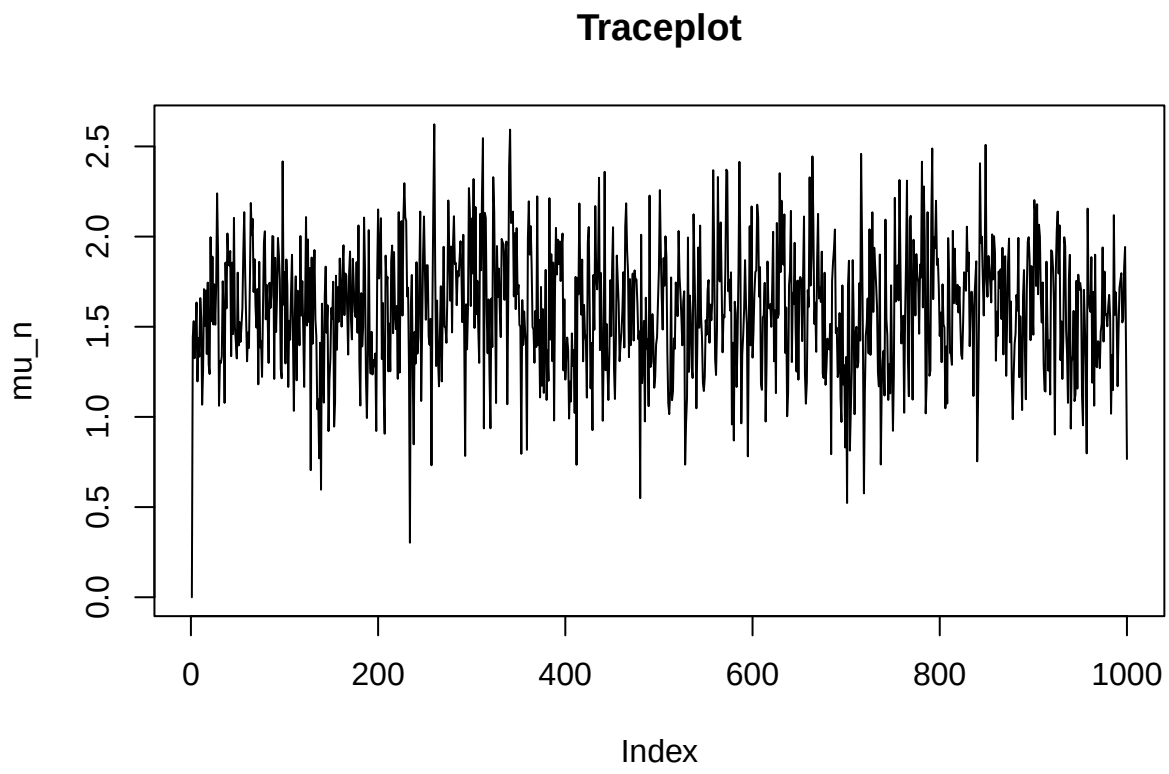
using the distributions derived for 3 conditions above, we implemented Gibbs sampler.



When the green dots is compared to red line in the plot, we can say that we have managed to remove the noise and also we can conclude from the plot that expected values of μs can catch the true underlying dependence between Y and X.

subquestion 5

Trace plot of μ_n can be seen below.



We can see the convergence and burn in period in trace plot visually.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

##########question 1###############

# my target distribution
my_target_function<-function(x){
  fn<-((x^5)*(exp(-x)))/120
  return(fn)
}

# metropolis hasting algorithm by using log normal as proposal distribution
m_h_algo<-function(x0,steps){
  n_steps<-1:steps
  x_n<-rep(x0,steps)
  for(i in 2:steps){
    x<-x_n[i-1]
    y<-rlnorm(1,meanlog = x,sdlog = 1)
    u<-runif(1)
    acc<-min(c(1,((my_target_function(y)*dlnorm(x,meanlog = y,sdlog = 1))/
                  (my_target_function(x)*dlnorm(y,meanlog = x,sdlog = 1)))))
```

```
    if(u<=acc){
      x_n[i]<-y
    }else{
      x_n[i]<-x
    }
  }
  return(list(x_n,n_steps))
}
y<-m_h_algo(1,20000)

# ploting the sample values
plot(y[[2]],y[[1]],pch=19,cex=0.3,col="black",xlab="steps",ylab="X",ylim=c(0,6),type='l',main='Time seri
# abline(h)
hist(y[[1]],breaks=400,main='Histogram of chain',xlab='X')



# metropolis hasting algorithm by using chi squared as proposal distribution
m_h_algo_chi<-function(x0,steps){
  n_steps<-1:steps
  x_n<-rep(x0,steps)
  for(i in 2:steps){
    x<-x_n[i-1]
    y<-rchisq(1,floor(x+1))
    u<-runif(1)
    acc<-min(c(1,((my_target_function(y)*dchisq(x,floor(y+1)))/
                    (my_target_function(x)*dchisq(y,floor(x+1))))))
    if(u<=acc){
      x_n[i]<-y
    }else{
      x_n[i]<-x
    }
  }
  return(list(x_n,n_steps))
}
x<-m_h_algo_chi(40,10000)

#plotting the sample values
plot(x[[2]],x[[1]],pch=19,cex=0.3,col="black",xlab="steps",ylab="X",ylim=c(0,40),type='l',main = 'Time s
abline(h=min(x[[1]]),col='red')
abline(h=mean(x[[1]])+1,col='red')
abline(h=14,col='red')
hist(x[[1]],breaks=400,main='Histogram of chain',xlab='X')


# creating a data frame and storing 10 MCMC sequences
df<-data.frame(matrix(nrow=10000,ncol=0))
for(i in 1:10){
  x<-m_h_algo_chi(i,10000)
  df<-cbind(df,x[[1]])
}
```

```r
# Gelman Rubin test
m_mean<-c()
variances<-c()
for (i in 1:ncol(df)){
  m_mean<-c(m_mean,mean(df[,i]))
  mubar<-mean(m_mean)
  variances<-c(variances,var(df[,i]))
  w<-mean(variances)
  # n=1000, k=10
  b<- (10000/9)*(sum(m_mean-mubar)**2)
  v<-(1-(1/10000))*w + (1/10000)*b
  hatR<- sqrt(v/w)
}

cat('Gelman-Rubin factor calculated:',hatR)


#using gelman.diag function
X0_vals <- seq(1,10,1)
mc_list <- list()
for (i in 1:length(X0_vals)) {
  mc_list[[i]] <- m_h_algo_chi(x0 = X0_vals[i], steps = 5000)[[1]]
}

options(warn = -1)
library(coda)
mc_list <- lapply(1:length(mc_list), function(x){as.mcmc(mc_list[[x]])})

gelman.diag(mc_list)


#calculating expected values or mean

cat('Mean or Expected value for step 1 algorithm samples:',(mean(y[[1]])))
cat('Mean or Expected value for step 2 algorithm samples:',(mean(x[[1]])))


################ Question 2 #############

load("C:/Users/Varun/Desktop/LiU - STIMA/Computational Statistics/Lab4/chemical.RData")

plot(X,Y,type="o",col="red", main="Dependence of Y on X")


# gibbs sampler to obtain 1000 values.

my_mat <- matrix(nrow=1000,ncol=50)
my_mat[,1:50]<-0

for (j in 2:1000){
  my_mat[j,1]<- rnorm(1,(my_mat[j-1,2]+Y[1])/2, sd=sqrt(0.1))
  for(i in 2:49){
    my_mat[j,i]<- rnorm(1,(my_mat[j,(i-1)]+my_mat[j-1,(i+1)]+Y[i])/3,sqrt(0.2/3))
```

```
  }
  my_mat[j,50]<- rnorm(1,(my_mat[j,49]+Y[50])/2, sd=sqrt(0.1))
}

mu_hat <- colMeans(my_mat)
plot(y=Y,x=X, type = "l",col="red")
points(mu_hat, col='green', type = "o")
legend(x = 20, y = 0.5,
       legend = c('Y versus X', 'Expected value of mu versus X'),
       col = c('red', 'green'), lty = c(1,1))


plot(my_mat[,50],type='l',main='Traceplot',ylab='mu_n')
```