computational stats_lab4_GROUP10.rmd

Yusur Al-Mter (yusal621), Lakshidaa Saigiridharan (laksa656)

2/20/2019

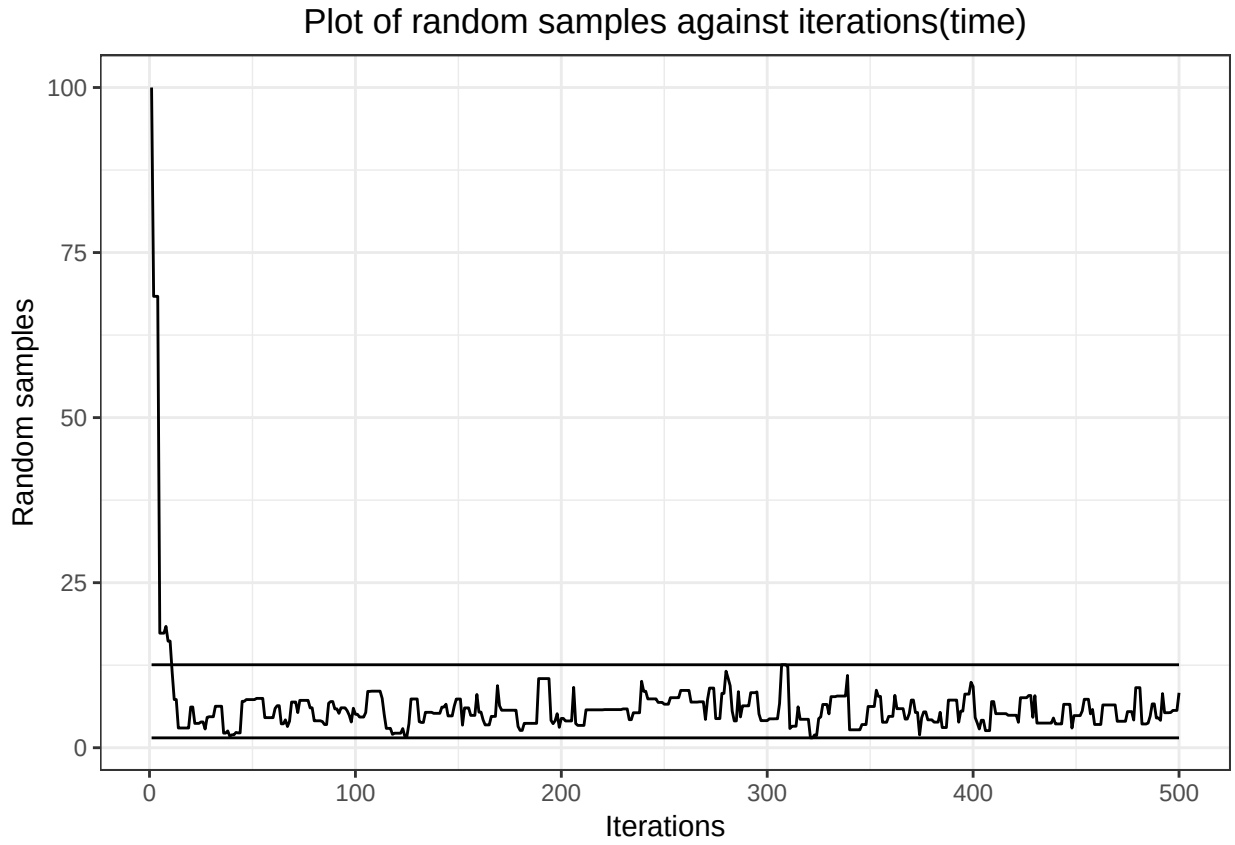
QUESTION 1: Computations with Metropolis-Hastings

TASK 1.1:

Consider the following probability density function:

$$f(x) = x^5 e^{-x}, \quad x > 0$$

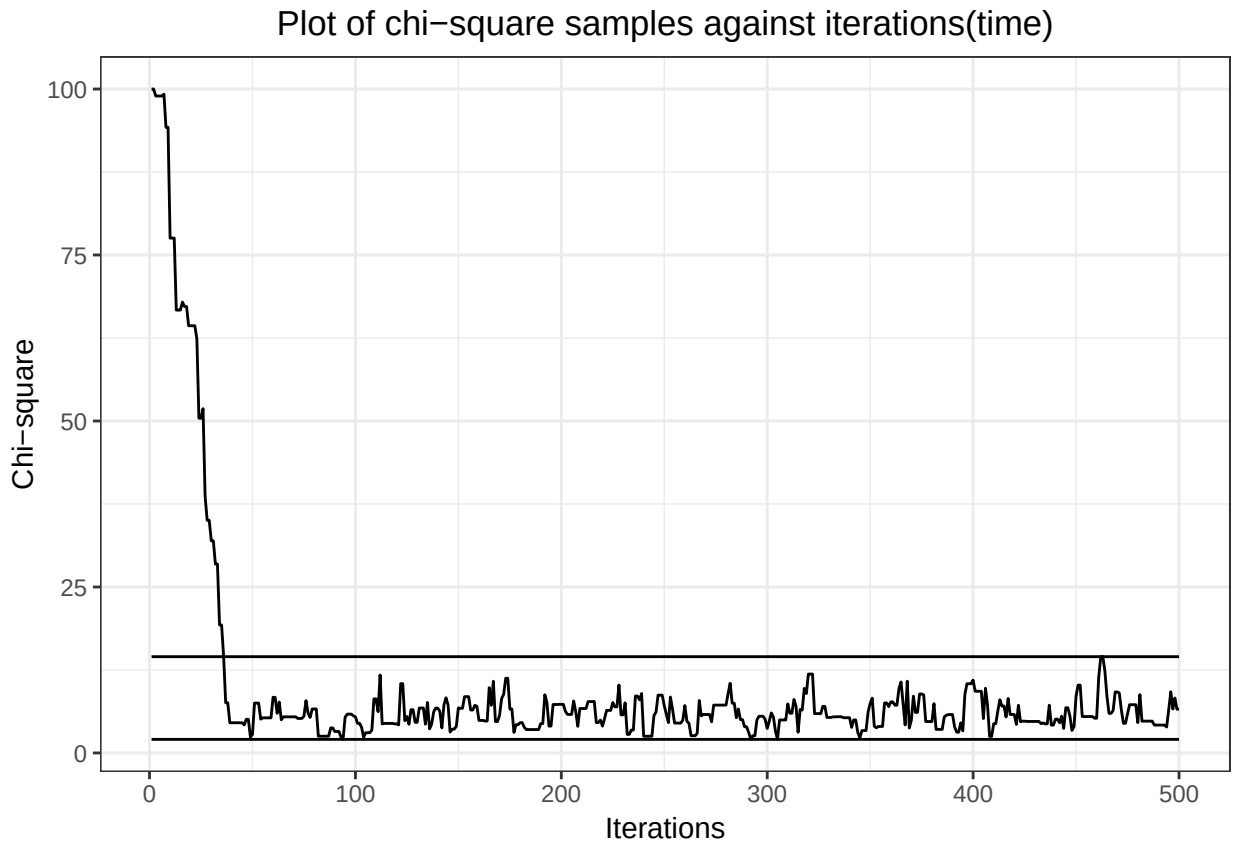
Use Metropolis-Hastings algorithm to generate samples from this distribution by using proposal distribution as log-normal LN(X_t,1), take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn-in period, what can be the size of this period?



Observing the plot, the sequence seems to converge to the target distribution quite rapidly and exhibits good mixing inspite of an extreme starting point. After running the algorithm a few times, most values are mixing well around 1 for a lower edge and 14 as an upper edge. As for the burn-in period, it depends mostly on the good choice of the initial starting point. In our case, I would say a good starting point would be around 30.

TASK 1.2:

Perform Step 1 by using the chi-square distribution Chi^2(Xt + 1) as a proposal distribution, where x is the floor function, meaning the integer part of x for positive x, i.e. 2.95 = 2



TASK 1.3:

Compare the results of Steps 1 and 2 and make conclusions.

By observing the behaviour of both plots and taking into account the starting point, one can say that the two proposal distributions converge quite rapidly. Although the chi-square comes from the same family as the target, it can be seen that the log normal distribution seems to converge faster. In general, one has no mathematical analysis of the Markov chain that tells where the good starting points are (nor how much burn-in is required to get to a good starting point). All decisions about starting points are based on the output of some preliminary runs that appear to have converged to stationarity. In this assignment, only by observing the plots, one can choose the burn-in period for the log normal case almost around 30 and for the Chi-square case almost around 50.

TASK 1.4:

Generate 10 MCMC sequences using the generator from Step 2 and starting points 1, 2, . . . , or 10. Use the Gelman-Rubin method to analyze convergence of these sequences.

```
## Gelman-Rubin method to analyze convergence for Chi-square sequences :
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.02       1.04
```

The Gelman and Rubin diagnostic is used to check the convergence of multiple mcmc chains run in parallel. It compares the within-chain variance to the between-chain variance. It provides estimate of how much variance could be reduced by running chains longer. The 'potential scale reduction factor' is calculated for each variable in x, together with upper and lower confidence limits. Approximate convergence is diagnosed when the upper limit is close to 1.

From our computations, it can be seen that the upper confidence interval for the chi-square generator is 1.04. Therefore, as the upper CI value is extremely close to 1, we can conclude that our MCMC sequences have converged well.

TASK 1.5

Estimate

$$\int_0^\infty x f(x) dx$$

using the samples from Steps 1 and 2.

```
## [1] 6.529159
```

```
## [1] 5.665162
```

TASK 1.6

The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

The generated target distribution is a gamma distribution, and the general probability density function for gamma distribution is as follows,

$$f(x) \sim Gamma(\alpha, \beta) = \frac{1}{\Gamma\alpha\ \beta^\alpha} x^{\alpha-1} e^{\frac{-x}{\beta}}$$

and in this case the values of α and β are 6 and 1, respectively. So, the actual expected value for the integral in 1.5 is,

$$E[X] = \alpha * \beta = 6 * 1 = 6$$

Comparing the range of expected values obatined in 1.5 after running it several times which is almost around 5.5 and 6.8 with the actual expected value; one can conclude that both results are in line

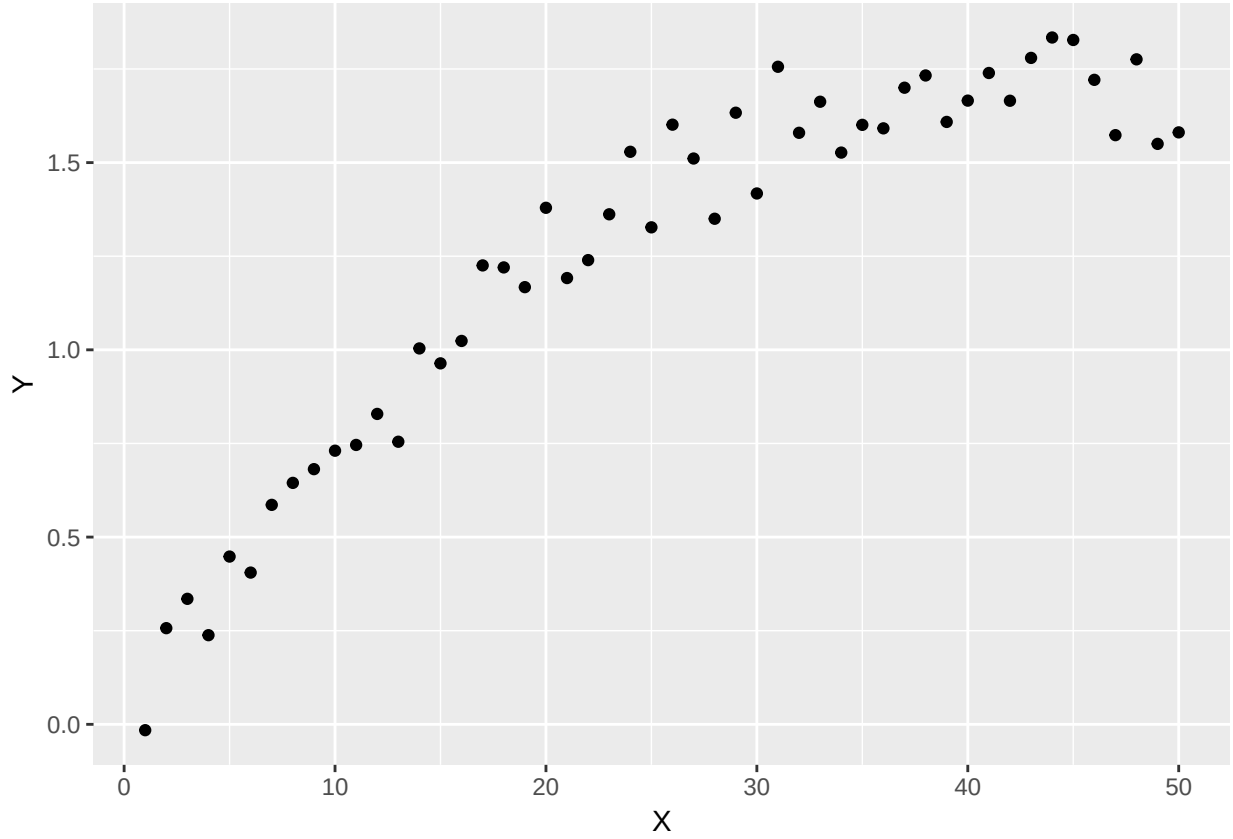
---

QUESTION 2: Gibbs sampling

A concentration of a certain chemical was measured in a water sample, and the result was stored in the data chemical.RData having the following variables: 1. X: day of the measurement 2. Y: measured concentration of the chemical. The instrument used to measure the concentration had certain accuracy; this is why the measurements can be treated as noisy. Your purpose is to restore the expected concentration values.

TASK 2.1:

Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?



From the plot obtained, it can be seen that the obatined scatterplot is similar to that of a logarithmic curve. Therefore, the model that best fits this case would be a logarithmic model.

TASK 2.2:

Having Y_i normally distributed, $Y_i \sim N(\mu_i, variance = 0.2)$, $\quad i = 1, 2, ..., n$, the likelihood function for $p(\vec{Y}|\vec{\mu})$ is,

$$L(\vec{Y}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{(}2\pi\sigma^2)} e^{-} \frac{1}{2\sigma^2}(Yi - \mu_i)^2$$

Then the likelihood becomes,

$$L(\vec{Y}|\vec{\mu}) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{-} \sum_{i=1}^{n} \frac{1}{2\sigma^2}(Y_i - \mu_i)^2$$

Having the following properties for $P(\vec{\mu})$

$$P(\mu_1) = 1$$

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

And using the chain rule for,

$$P(\vec{\mu}) = P(\mu_1)P(\mu_2|\mu_1)P(\mu_3|\mu_2)...P(\mu_n|\mu_{n-1})$$

The prior distribution will become,

$$P(\vec{\mu}) = \prod_{i=1}^{n-1} \frac{1}{\sqrt{(2\pi\sigma^2)}} e^{-} \frac{1}{2\sigma^2}(\mu_{i+1} - \mu_i)^2$$

$$P(\vec{\mu}) = \frac{1}{(2\pi\sigma^2)^{\frac{n-1}{2}}} e^{-} \frac{1}{2\sigma^2} \sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2$$

TASK 2.3:

Using Bayes' Theorem to get the posterior distribution represented by the following formula,

$$P(\vec{\mu}|\vec{Y}) = \frac{P(\vec{\mu})P(\vec{Y}|\vec{\mu})}{\int P(\vec{\mu})P(\vec{Y}|\vec{\mu})d\mu}$$

The above stated formula is the distribution of $\vec{\mu}$ conditional on the observed data $\vec{Y}$. Since the denominator is not a function of $\vec{\mu}$ and is a constant of proportionality to make the posterior integrate to one, we can write the posterior as being proportional to the prior times the likelihood,

$$P(\vec{\mu}|\vec{Y}) \propto P(\vec{\mu})P(\vec{Y}|\vec{\mu}) = P(\vec{\mu})L(\vec{Y}|\vec{\mu})$$

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

$$\propto exp(-\frac{1}{2\sigma^2}(\sum_{i=1}^{n-1}[(\mu_{i+1} - \mu_i)^2 + (Y_i - \mu_i)^2] + (Y_n - \mu_n)^2)) \qquad .....(*)$$

Thus, using eq(*), the conditional distriibution for i=1 is,

$$P(\mu_1|\mu_{-1}, Y) \propto exp(-\frac{1}{2\sigma^2}(\mu_1 - \mu_2)^2 + (\mu_1 - Y_1)^2))$$

$$\propto exp(-\frac{(\mu_1 - \frac{(\mu_2 + Y_1)}{2})^2}{\sigma^2})$$

Hence,

$$P(\mu_1|\mu_{-1}, Y) \sim N(\frac{(\mu_2 + Y_1)}{2}, \frac{\sigma^2}{2})$$

Again using eq(*), the conditional distribution for i=i is,

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

Hence,

$$P(\mu_i|\mu_{-i}, Y) \sim N(\frac{(\mu_{i-1} + \mu_{i+1} + Y_i)}{3}, \frac{2\sigma^2}{3})$$

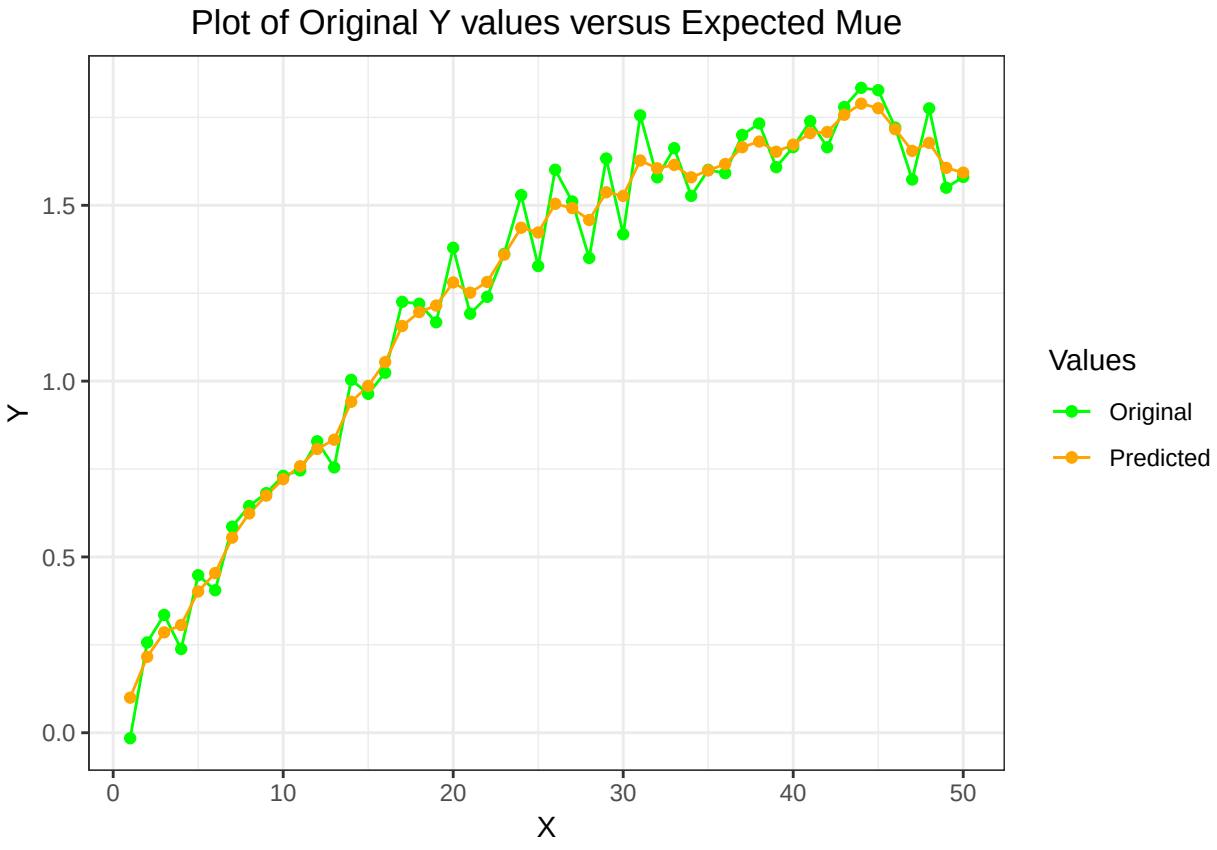
And the conditional distribution for i=n is,

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

Hence,

$$P(\mu_n|\mu_{-n}, Y) \sim N(\frac{\mu_{n-1} + Y_n}{2}, \frac{\sigma^2}{2})$$

TASK 2.4:

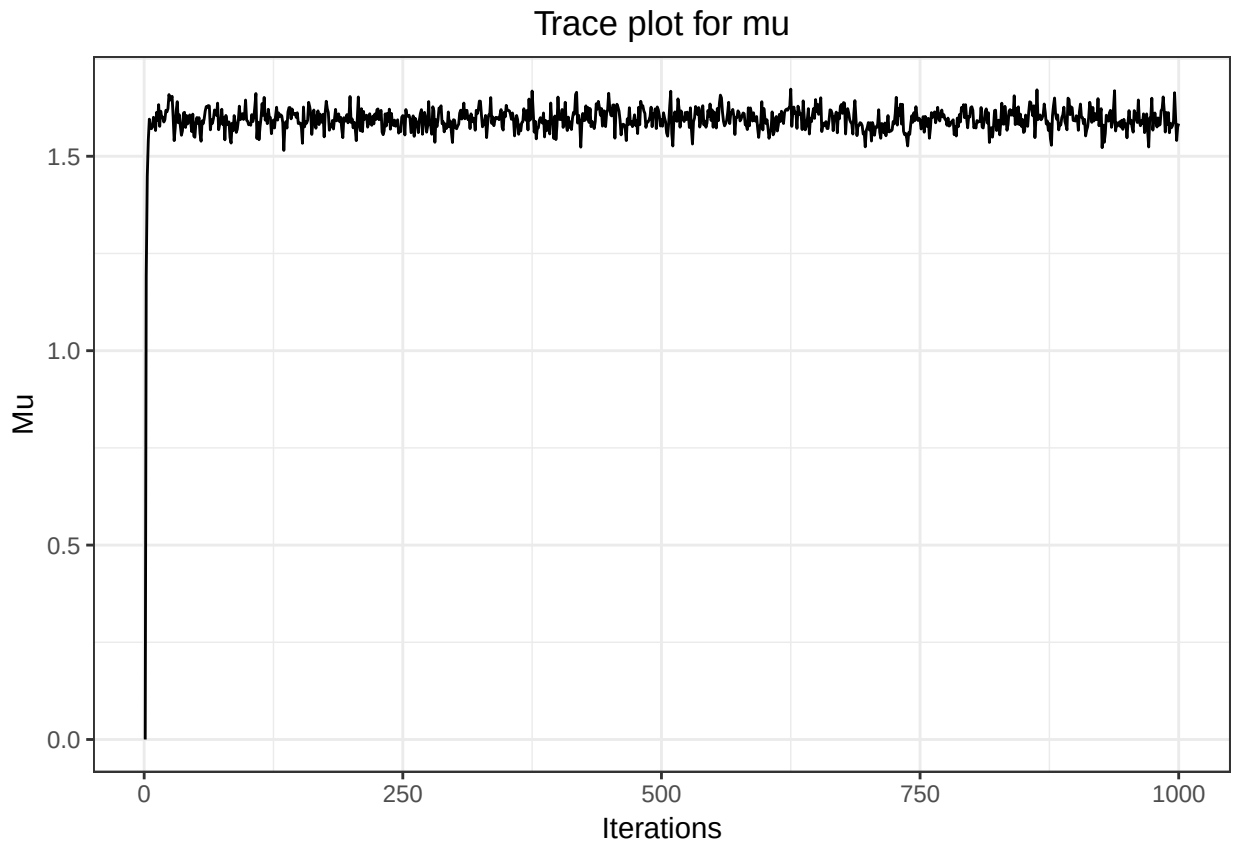


Our computations have managed to remove a part of the noise. This indicates that our function works well.

From the plot, it can be seen the predcited values are very close to the original values. Therefore, it does seem that the expected value of mu can catch the true underlying dependence between Y and X.

TASK 2.5:

Make a trace plot for Mn and comment on the burn-in period and convergence.



This trace plot shows that the sequence of μ_n values seem to converge to the target distribution very fast (within the first two iterations). By observation, the burn-in period can be seen to occur at around iterations 3 to 5.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
# Loading required R packages
library(ggplot2)
library(coda)


# QUESTION 1 : Computations with Metropolis-Hastings


# Task 1.1

f <- function(x){
  target<- (x^5 * exp(-x))/120
  return(target)
```

```
}
f.MCMC.MH<-function(nstep,X0,props){
    vN<-1:nstep
    vX<-rep(X0,nstep)

    for (i in 2:nstep){

        X<-vX[i-1]
        Y<-rlnorm(1,meanlog = log(X),sdlog = props)
        u<-runif(1)
        alpha<-min(c(1,(f(Y)*dlnorm(X,meanlog = log(Y),sdlog = props))/(f(X)*dlnorm(Y,meanlog = log(X),s

      if (u <=alpha){
          vX[i]<-Y          # accept move with probabily min(1,alpha)
         }
        else{
            vX[i]<-X        # otherwise "reject" move, and stay where we are
         }
    }
    return(vX)
}
test <- f.MCMC.MH(500,100,1)

# Plot of random samples against iterations(time)
ggplot() +
  geom_line(aes(x=1:500, y=test)) +
  geom_line(aes(x=1:500, y=max(test[30:500]))) +
  geom_line(aes(x=1:500, y=min(test[30:500]))) +
  xlab("Iterations") +
  ylab("Random samples") +
  ggtitle("Plot of random samples against iterations(time)") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")


# Task 1.2
# Generating samples from Chi square

f.MCMC.MH.CHI<-function(nstep,X0){
    vN<-1:nstep
    vX<-rep(X0,nstep)

    for (i in 2:nstep){
        X<-vX[i-1]
        Y<-rchisq(1, floor(X+1))
        u<-runif(1)
        alpha1<-min(c(1,(f(Y)*dchisq(X,df=floor(Y+1)))/(f(X)*dchisq(Y,df=floor(X+1)))))

      if (u <=alpha1){
          vX[i]<-Y          # accept move with probabily min(1,alpha1)
         }
        else{
           vX[i]<-X        # otherwise "reject" move, and stay where we are
```

```
        }
    }
    return(vX)
}
test1 <- f.MCMC.MH.CHI(500,100)

# Plot of random samples against iterations(time)
ggplot() +
  geom_line(aes(x=1:500, y=test1)) +
  geom_line(aes(x=1:500, y=max(test1[50:500]))) +
  geom_line(aes(x=1:500, y=min(test1[50:500]))) +
  xlab("Iterations") +
  ylab("Chi-square") +
  ggtitle("Plot of chi-square samples against iterations(time)") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")


# Task 1.4

set.seed(12345)
# Gelman-Rubin method for Chi square
seq_list_chi<- list()
x_vec_chi <- c()

for (i in seq(1,10)){
  x_vec_chi <-f.MCMC.MH.CHI(500,i)
  x_vec_chi <- as.mcmc(x_vec_chi)
  seq_list_chi [[i]]<- x_vec_chi
}

seq_list_chi <- as.mcmc.list(seq_list_chi)
cat("Gelman-Rubin method to analyze convergence for Chi-square sequences : ")
gelman.diag(seq_list_chi)


# Task 1.5

# By observing the plot for Log normal, the burn in period is approximately around 30
# expected value for Log normal
mean(f.MCMC.MH(500,100,1)[30:500])

# By observing the plot for Chi-square, the burn in period is approximately around 50
# expected value for Chi-square
mean(f.MCMC.MH.CHI(500,100)[50:500])

# QUESTION 2 : Gibbs sampling


# Task 2.1

load("chemical.RData")
chemical <- as.data.frame(cbind(X=X, Y=Y))
```

```r
ggplot(chemical) +
  geom_point(aes(x=X, y=Y))


# Task 2.4

mat<-matrix(0, nrow = 1000, ncol = 50)

for(ro in 2:1000)
{
  for (co in 1:50)
  {
    if(co==1){
      mat[ro,co] <- rnorm(1, (mat[ro-1,2]+Y[1])/2, (0.2)^2/2)
    }
    else if(co==50){
      mat[ro,co] <- rnorm(1, (mat[ro,49]+Y[50])/2, (0.2)^2/2)
    }
    else{
      mat[ro,co] <- rnorm(1, (mat[ro,co-1]+mat[ro-1,co+1]+Y[co])/3, (2*(0.2)^2)/3)
    }
  }
}

averages<-colMeans(mat)

cols <- c(Predicted = "orange", Original = "green")
ggplot() + geom_line(aes(x=X,y=Y, color="Original")) +
  geom_line(aes(x=X,y=averages,color="Predicted")) +
  geom_point(aes(x=X,y=Y, color="Original")) +
  geom_point(aes(x=X,y=averages,color="Predicted")) +
  scale_color_manual(name = "Values", values=cols) +
  ggtitle("Plot of Original Y values versus Expected Mue") +
  theme_bw() +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")


# Task 2.5

ggplot() +
  geom_line(aes(x=1:1000, y=mat[,50])) +
  theme_bw() +
  ggtitle("Trace plot for mu") +
  xlab("Iterations") +
  ylab("Mu") +
  theme(plot.title = element_text(hjust = 0.5), legend.position = "right")
```