Computer Lab 4

Group-19 - Dinesh and Umamaheswarababu

2022-12-06

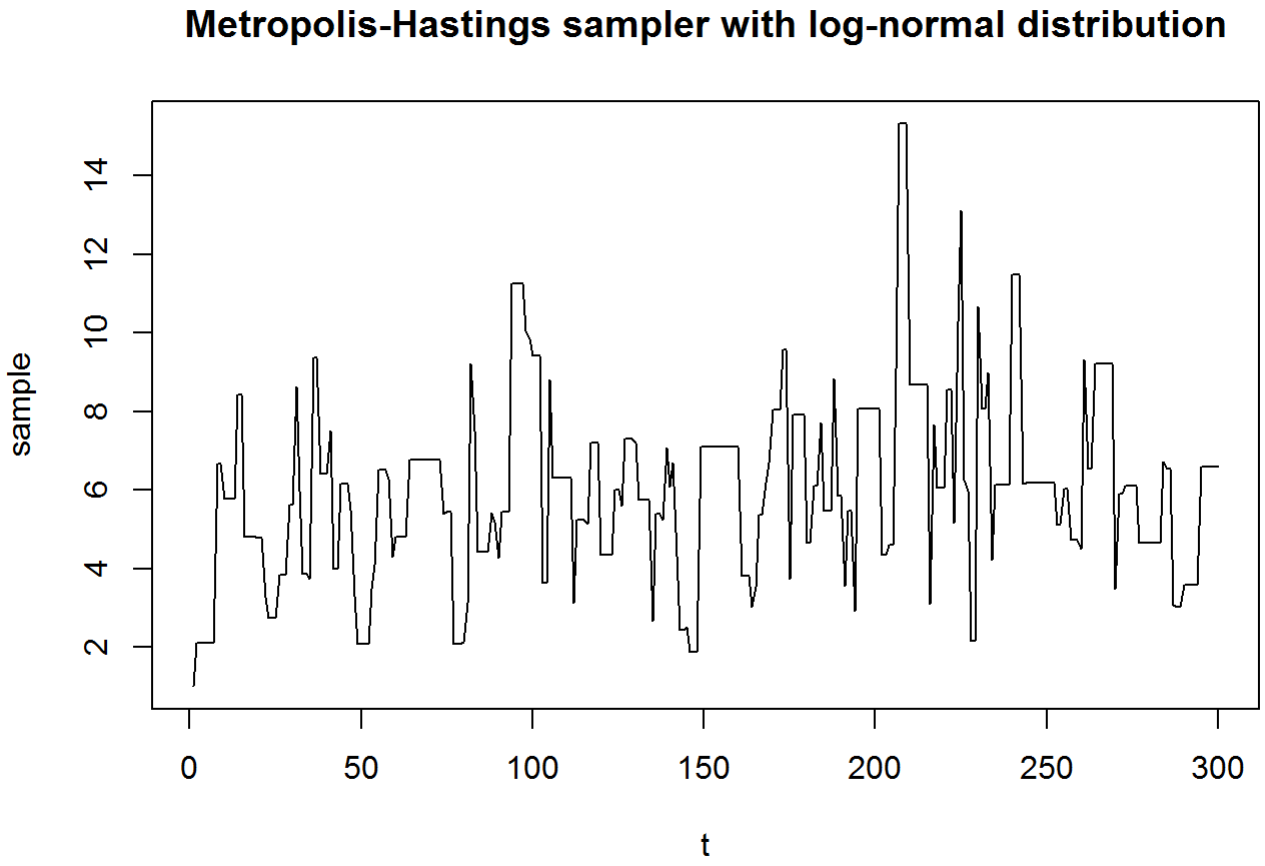
Statement of Contribution : Question 1 was mostly done by Umamaheswarababu and Question 2 was mostly done by Dinesh

Question 1: Computations with Metropolis–Hastings

Question 1.1

Metropolis-Hastings algorithm with Log-normal distribution

```
#target function
f_x<-function(x){
  return(x^5 * exp(-x))
}

#Metropolis-Hastings algorithm with proposal distribution as log-normal

metro_hast_ln<-function(x_0,t_max){

  x_vec<-rep(x_0,t_max)
  for (i in 2:t_max){
    x<-x_vec[i-1]
    x_star<- rlnorm(1,meanlog = log(x),sdlog = 1)
    alpha<-min(c(1,(f_x(x_star)*dlnorm(x,meanlog = log(x_star),sdlog = 1))/(f_x(x)*dlnorm(x_sta
r,meanlog = log(x),sdlog = 1))))
    u<-runif(1,0,1)
    if(u<alpha){
      x_vec[i]<-x_star
    }else{
      x_vec[i]<-x
    }
  }
  return(x_vec)
}

#ploting time series of samples with starting point as 1 and 300 iterations
plot(1:300, metro_hast_ln(1,300), type<-"l",xlab = "t", ylab = "sample",main = "Metropolis-Hasti
ngs sampler with log-normal distribution")
```

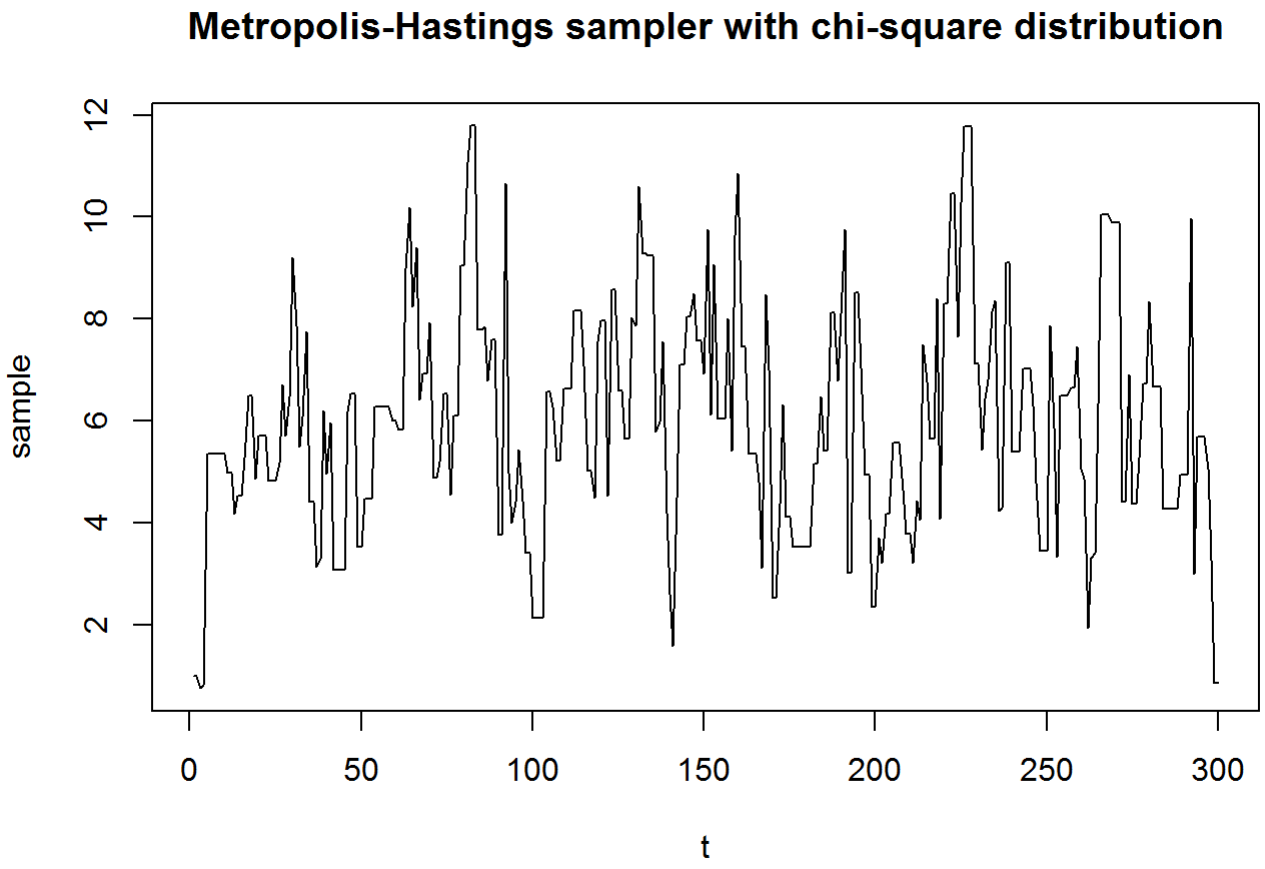


From the above plot we can say the the graph converges quickly and the burning period is very small(approximately 0-5).

Question 1.2

Metropolis-Hastings algorithm with chi-square distribution

```
metro_hast_chisq<-function(x_0,t_max){

  x_vec<-rep(x_0,t_max)
  for (i in 2:t_max){
    x<-x_vec[i-1]
    x_star<- rchisq(1,df=floor(x+1))
    alpha<-min(c(1,(f_x(x_star)*dchisq(x,floor(x_star+1)))/(f_x(x)*dchisq(x_star,floor(x+1)))))
    u<-runif(1,0,1)
    if(u<alpha){
      x_vec[i]<-x_star
    }else{
      x_vec[i]<-x
    }
  }
  return(x_vec)
}

#ploting time series of samples with starting point as 1 and 300 iterations
plot(1:300, metro_hast_chisq(1,300), type<-"l",xlab = "t", ylab = "sample",main = "Metropolis-Ha
stings sampler with chi-square distribution")
```



Question 1.3

Both the samplers have similar kind of burning period (0-5) and sampling with chi-square distribution converges quicker than sampling with log-normal distribution. From this we can conclude that chi-square distribution would be a better choice than log-normal as proposed distribution to sample f(x) using Metropolis-Hastings algorithm.

Question 1.4

Generating 10 MCMC sequences using the generator from step 2

```
# Generating 10 MCMC sequences with starting point 1,2,..10 and 10000 iterations
library(coda)
```

```
## Warning: package 'coda' was built under R version 4.2.2
```

```
s1<-metro_hast_chisq(1,10000)
s2<-metro_hast_chisq(2,10000)
s3<-metro_hast_chisq(3,10000)
s4<-metro_hast_chisq(4,10000)
s5<-metro_hast_chisq(5,10000)
s6<-metro_hast_chisq(6,10000)
s7<-metro_hast_chisq(7,10000)
s8<-metro_hast_chisq(8,10000)
s9<-metro_hast_chisq(9,10000)
s10<-metro_hast_chisq(10,10000)

MCMC_list<-mcmc.list(as.mcmc(s1),as.mcmc(s2),as.mcmc(s3),as.mcmc(s4),as.mcmc(s5),as.mcmc(s6),as.
mcmc(s7),as.mcmc(s8),as.mcmc(s9),as.mcmc(s10))

#Gelman–Rubin method
gelman.diag(MCMC_list)
```

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

Since both the potential scale reduction factors are equal to 1, we can say that our sampler for the given length(1 to 10) is convergent.

Question 1.5

Since $f(x)$ is a probability density function, the integral $\int_0^\infty x f(x)$ can be considered as mean of f(x). The value of the integral can be found from the samples derived in step1 and step2.

From step1 (sample1)

```
mean(metro_hast_ln(1,10000))
```

```
## [1] 6.025757
```

From step2 (sample2)

```
mean(metro_hast_chisq(1,10000))
```

```
## [1] 6.039027
```

Question 1.6

The probability density function of gamma distribution is given by

$f(x;\alpha,\beta) = [\frac{1}{\Gamma(\alpha)\beta^{\alpha}}]x^{\alpha-1}e^{-y/\beta}$

If we compare our target function with the above standard beta function we get $\alpha = 6$ and $\beta = 1$.

The mean of target function $f(x)$ is given by $\alpha\beta = 6 * 1 = 6$

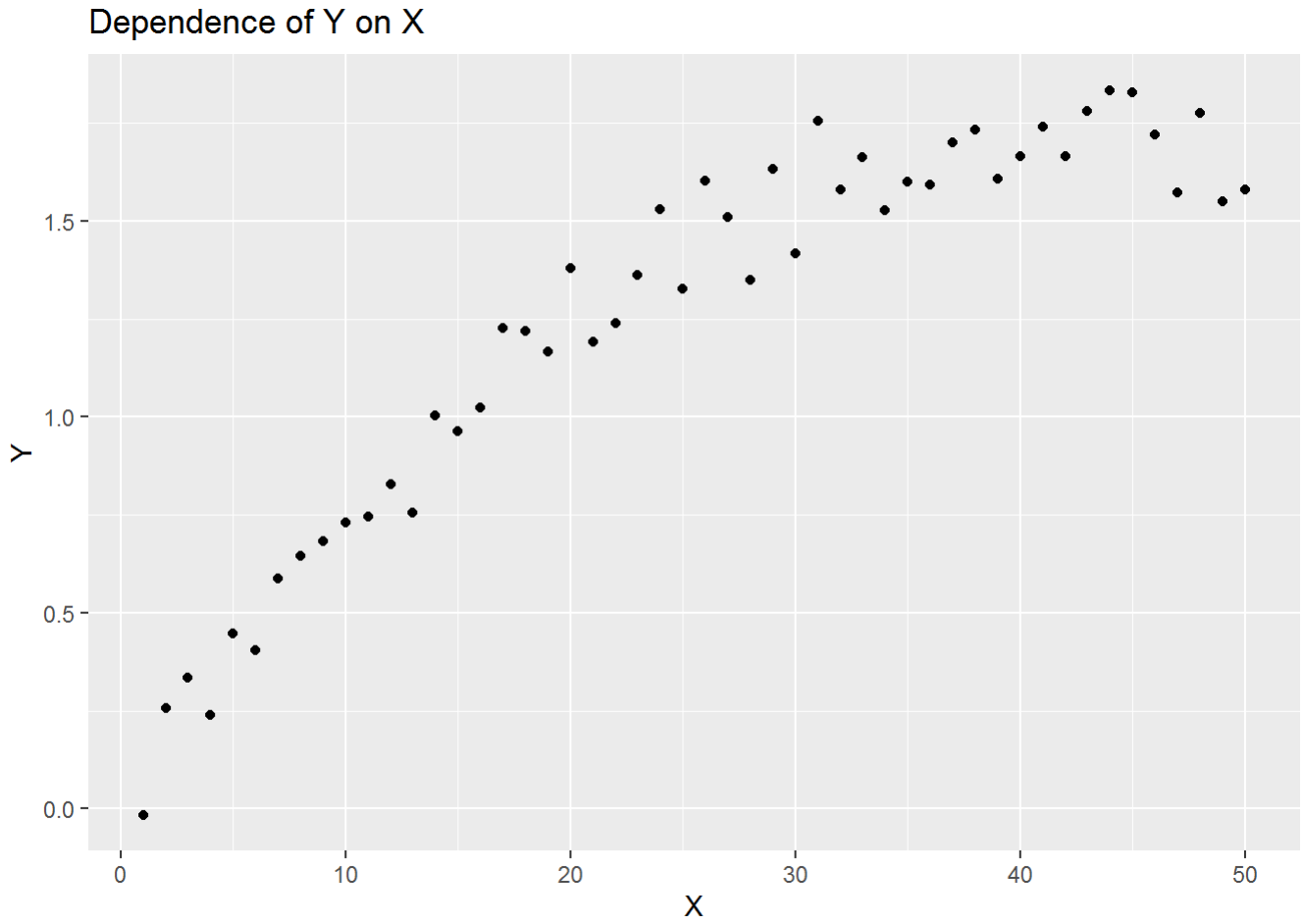
The mean $\alpha\beta = 6$ is close to the value we got in step 5.

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: * X: day of the measurement * 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.

Q 2.1 Import the data to R and plot the dependence of Y on X.

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

#plot the dependence of Y on X
library(ggplot2)
plot_1<-ggplot()+geom_point(aes(X,Y))+ggtitle("Dependence of Y on X")
plot_1
```



What kind of model is reasonable to use here?

By seeing the scatter plot, we can say that the logarithmic model or linear regression model is reasonable to use here.

Q 2.2 A researcher has decided to use the following (random-walk) Bayesian model (n=number of observations, $\hat{\mu} = (\mu_1, \ldots, \mu_n)$ are unknown parameters):

$$Y_i \sim \mathcal{N}(\mu_i, 0.2), i = 1, \ldots, n$$

where the prior is

$$P(\mu_1) = 1$$

$$P(\mu_{i+1} \mid \mu_i) \sim \mathcal{N}(\mu_i, 0.2), i = 1, \ldots, n-1$$

Present the formulae showing the likelihood $P(\vec{Y} \mid \vec{\mu})$ and the prior $P(\vec{\mu})$.

$Y_i \sim \mathcal{N}(\mu_i, 0.2), i = 1, \ldots, n$

$$f(Y_i) = \frac{1}{\sqrt{(2\pi\sigma^2)}} exp(^- \frac{1}{2\sigma^2}(Yi - \mu_i)^2)$$

In order to obtain the likelihood function, we can:

$$\begin{aligned} P(\vec{Y} \mid \vec{\mu}) &= \prod_{i=1}^{n} f(Y_i) \\ &= \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} exp^- \sum_{i=1}^{n} \frac{1}{2\sigma^2}(Y_i - \mu_i)^2 \end{aligned}$$

We can obtain the prior function using chain rules:

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

The prior distribution will become,

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

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

Q 2.3 Use Bayes' Theorem to get the posterior up to a constant proportionality, and then find out the distributions of $\left(\mu_i \mid \vec{\mu}_{-i}, \vec{Y}\right)$, where $\vec{\mu}_{-i}$ is a vector containing all μ values except of μ_i.

Bayes' Theorem

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

Posterior is directly proportionl to the prior times the likelihood.

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

From the Bayes' Theorem we can get the posterior as below:

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

So we calculate $(\mu_1 \mid \vec{\mu}_{-1}, \vec{Y})$

$$\begin{aligned} P(\mu_1 \mid \vec{\mu}_{-1}, \vec{Y}) &\propto exp(-\frac{1}{2\sigma^2}((Y_1-\mu_1)^2+(\mu_2-\mu_1)^2)) \\ &= exp(-\frac{1}{2\sigma^2}(\mu_1-Y_1)^2+(\mu_1-\mu_2)^2) \\ &\propto exp(-\frac{(\mu_1-\frac{(Y_1+\mu_2)}{2})^2}{\sigma^2}) \\ &\propto N(\frac{Y_1+\mu_2}{2}, \frac{\sigma^2}{2}) \end{aligned}$$

Then we calculate $(\mu_n \mid \vec{\mu}_{-n}, \vec{Y})$

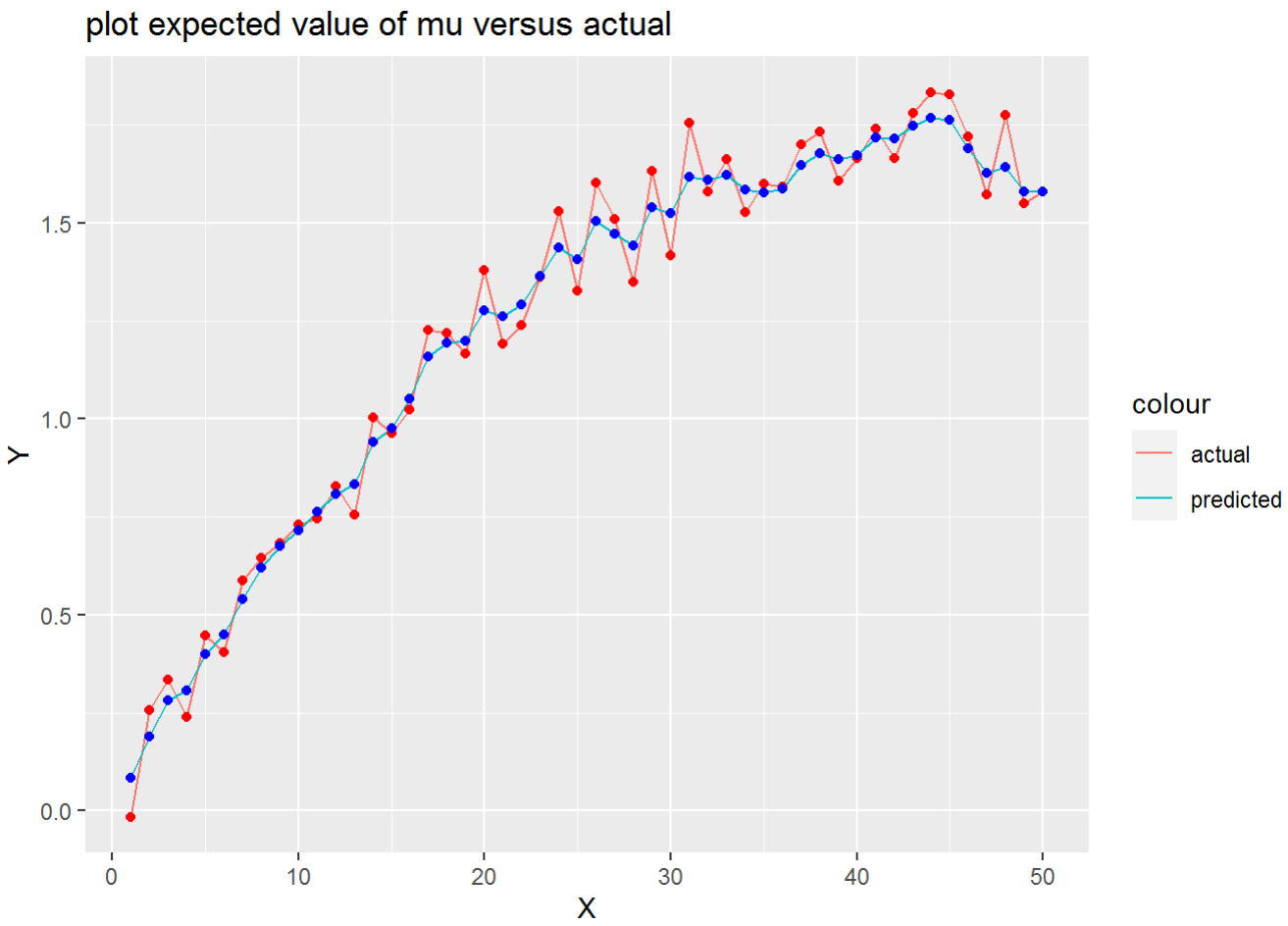
$$\begin{aligned} P(\mu_n \mid \vec{\mu}_{-n}, \vec{Y}) &\propto exp(-\frac{1}{2\sigma^2}((Y_n-\mu_n)^2+(\mu_n-\mu_{n-1})^2)) \\ &= exp(-\frac{1}{2\sigma^2}((\mu_n-Y_n)^2+(\mu_n-\mu_{n-1})^2)) \\ &\propto exp(-\frac{(\mu_n-\frac{(Y_n+\mu_{n-1})}{2})^2}{\sigma^2}) \\ &\propto N(\frac{Y_n+\mu_{n-1}}{2}, \frac{\sigma^2}{2}) \end{aligned}$$

Then we calculate $(\mu_i \mid \vec{\mu}_{-i}, \vec{Y}), i = 2, 3, \ldots, n-1$

$$\begin{aligned} P(\mu_i \mid \vec{\mu}_{-i}, \vec{Y}) &= exp(-\frac{1}{2\sigma^2}(\sum_{i=1}^{n}(Y_i-\mu_i)^2+\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2)) \\ &\propto exp(-\frac{1}{2\sigma^2}((Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2)) \\ &= exp(-\frac{1}{2\sigma^2}((\mu_i-Y_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_i-\mu_{i+1})^2)) \\ &\propto exp(-\frac{(\mu_i-\frac{(\mu_{i-1}+\mu_{i+1}+Y_i)}{3})^2}{\frac{2\sigma^2}{3}}) \\ &\propto N(\frac{(\mu_{i-1}+\mu_{i+1}+Y_i)}{3}, \frac{2\sigma^2}{3}) \end{aligned}$$

Q 2.4 Use the distributions derived in Step 3 to implement a Gibbs sampler that uses $\vec{\mu}_0 = (0, \ldots, 0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of $\vec{\mu}$ and then compute the expected value of $\vec{\mu}$ by using a Monte Carlo approach. Plot the expected value of $\vec{\mu}$ versus X and Y versus X in the same graph. Does it seem that you have managed to remove the noise? Does it seem that the expected value of $\vec{\mu}$ can catch the true underlying dependence between Y and X?

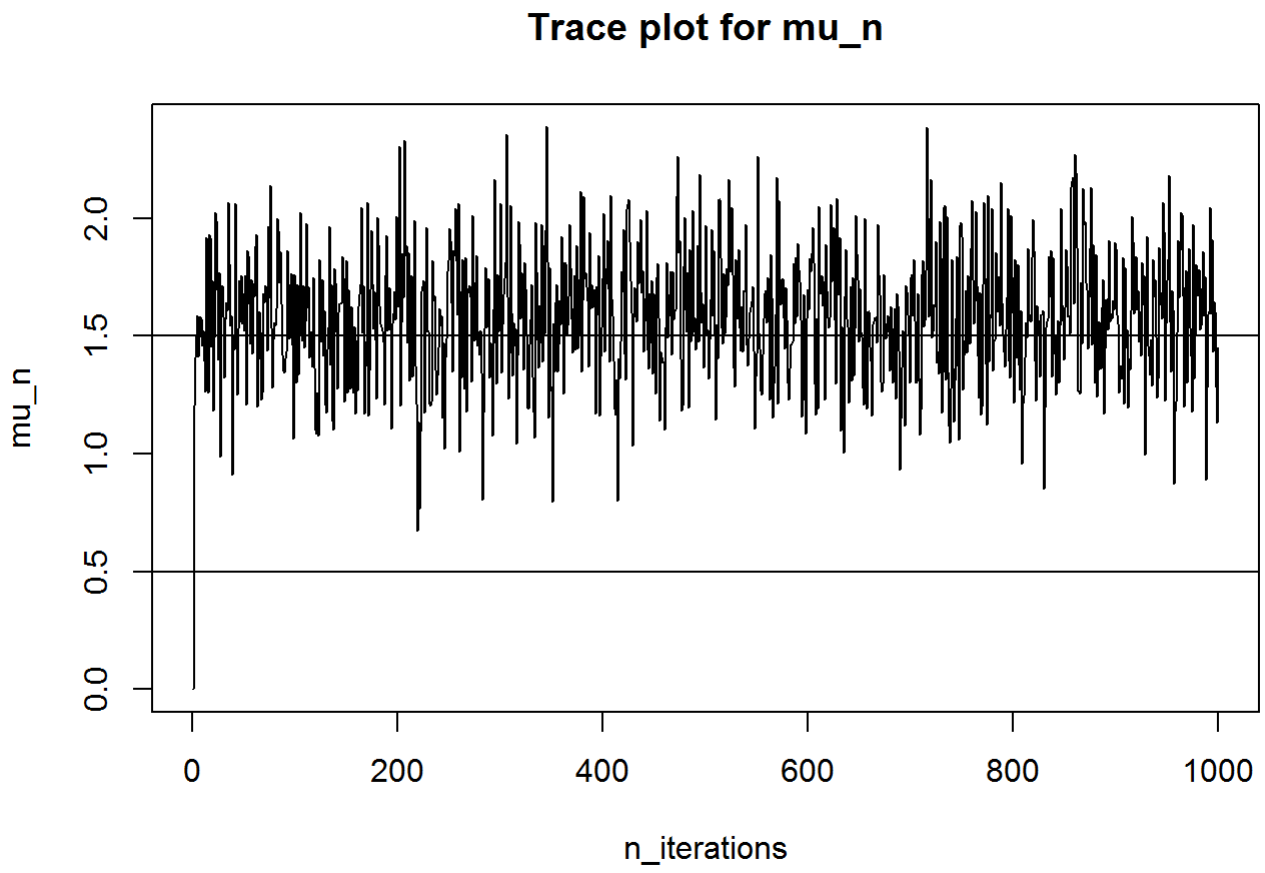
```
gibbs_fun<-function(n_val,d_y,var)
{
  d<-length(d_y)
  mX<-matrix(0, nrow = n_val, ncol = d)
  for(i in 2:n_val)
  {
    for (j in 1:d)
    {
      if(j==1){
        mX[i,j] <- rnorm(1, mean=(mX[i-1,2]+Y[1])/2, sd=sqrt(var)/2)
      }
      else if(j==50){
        mX[i,j] <- rnorm(1, mean=(mX[i,j-1]+Y[50])/2, sd=sqrt(var)/2)
      }
      else{
        mX[i,j] <- rnorm(1, mean = (mX[i,j-1]+mX[i-1,j+1]+Y[j])/3, sd=(2*sqrt(var))/3)
      }
    }
  }
  mX
}

n_val<-1000
d_y<-Y
var<-0.2
mX<-gibbs_fun(n_val,d_y,var)

avg<-colMeans(mX)

#plot expected value of mu versus actual
plot_2<-ggplot(data,aes(x=X,y=Y, col= "actual"))+geom_line()+xlab("X")+ylab("Y")+ggtitle("plot e
xpected value of mu versus actual")+geom_line(aes(x=X,y=avg, col="predicted"))+geom_point(aes(X,
Y),col="red")+geom_point(aes(X,avg),col="blue")
plot_2
```



In the plot, we can see that the points in the new plot are obviously much smoother than those in the actual one. So we can say we have managed to remove the noise. This proves that our function is effective. Also we can see that the predicted values of plot are almost similar to the actual plot.Therefore, it does seem that the expected value of mu can catch the true underlying dependence between Y and X.

Q 2.5 Make a trace plot for μ_n and comment on the burn-in period and convergence.

```
plot(x=1:1000, y=mX[,50],type="l",col="black", lwd=1, xlab="n_iterations",ylab="mu_n")
title("Trace plot for mu_n")
abline(h=1.5)
abline(h=2.5)
abline(h=0.5)
```



As we can see from the trace plot,the burn-in period for μ_n is $[0.5, 2.5]$. And convergence of the plot is 1.5.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#target function
f_x<-function(x){
  return(x^5 * exp(-x))
}

#Metropolis-Hastings algorithm with proposal distribution as log-normal

metro_hast_ln<-function(x_0,t_max){

  x_vec<-rep(x_0,t_max)
  for (i in 2:t_max){
    x<-x_vec[i-1]
    x_star<- rlnorm(1,meanlog = log(x),sdlog = 1)
    alpha<-min(c(1,(f_x(x_star)*dlnorm(x,meanlog = log(x_star),sdlog = 1))/(f_x(x)*dlnorm(x_sta
r,meanlog = log(x),sdlog = 1))))
    u<-runif(1,0,1)
    if(u<alpha){
      x_vec[i]<-x_star
    }else{
      x_vec[i]<-x
    }
  }
  return(x_vec)
}

#ploting time series of samples with starting point as 1 and 300 iterations
plot(1:300, metro_hast_ln(1,300), type<-"l",xlab = "t", ylab = "sample",main = "Metropolis-Hasti
ngs sampler with log-normal distribution")
metro_hast_chisq<-function(x_0,t_max){

  x_vec<-rep(x_0,t_max)
  for (i in 2:t_max){
    x<-x_vec[i-1]
    x_star<- rchisq(1,df=floor(x+1))
    alpha<-min(c(1,(f_x(x_star)*dchisq(x,floor(x_star+1)))/(f_x(x)*dchisq(x_star,floor(x+1)))))
    u<-runif(1,0,1)
    if(u<alpha){
      x_vec[i]<-x_star
    }else{
      x_vec[i]<-x
    }
  }
  return(x_vec)
}

#ploting time series of samples with starting point as 1 and 300 iterations
plot(1:300, metro_hast_chisq(1,300), type<-"l",xlab = "t", ylab = "sample",main = "Metropolis-Ha
stings sampler with chi-square distribution")
# Generating 10 MCMC sequences with starting point 1,2,..10 and 10000 iterations
library(coda)
s1<-metro_hast_chisq(1,10000)
```

```
s2<-metro_hast_chisq(2,10000)
s3<-metro_hast_chisq(3,10000)
s4<-metro_hast_chisq(4,10000)
s5<-metro_hast_chisq(5,10000)
s6<-metro_hast_chisq(6,10000)
s7<-metro_hast_chisq(7,10000)
s8<-metro_hast_chisq(8,10000)
s9<-metro_hast_chisq(9,10000)
s10<-metro_hast_chisq(10,10000)

MCMC_list<-mcmc.list(as.mcmc(s1),as.mcmc(s2),as.mcmc(s3),as.mcmc(s4),as.mcmc(s5),as.mcmc(s6),as.
mcmc(s7),as.mcmc(s8),as.mcmc(s9),as.mcmc(s10))

#Gelman–Rubin method
gelman.diag(MCMC_list)
mean(metro_hast_ln(1,10000))
mean(metro_hast_chisq(1,10000))

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

#plot the dependence of Y on X
library(ggplot2)
plot_1<-ggplot()+geom_point(aes(X,Y))+ggtitle("Dependence of Y on X")
plot_1

gibbs_fun<-function(n_val,d_y,var)
{
  d<-length(d_y)
  mX<-matrix(0, nrow = n_val, ncol = d)
  for(i in 2:n_val)
  {
    for (j in 1:d)
    {
      if(j==1){
        mX[i,j] <- rnorm(1, mean=(mX[i-1,2]+Y[1])/2, sd=sqrt(var)/2)
      }
      else if(j==50){
        mX[i,j] <- rnorm(1, mean=(mX[i,j-1]+Y[50])/2, sd=sqrt(var)/2)
      }
      else{
        mX[i,j] <- rnorm(1, mean = (mX[i,j-1]+mX[i-1,j+1]+Y[j])/3, sd=(2*sqrt(var))/3)
      }
    }
  }
  mX
}

n_val<-1000
d_y<-Y
var<-0.2
mX<-gibbs_fun(n_val,d_y,var)
```

```
avg<-colMeans(mX)

#plot expected value of mu versus actual
plot_2<-ggplot(data,aes(x=X,y=Y, col= "actual"))+geom_line()+xlab("X")+ylab("Y")+ggtitle("plot e
xpected value of mu versus actual")+geom_line(aes(x=X,y=avg, col="predicted"))+geom_point(aes(X,
Y),col="red")+geom_point(aes(X,avg),col="blue")
plot_2
plot(x=1:1000, y=mX[,50],type="l",col="black", lwd=1, xlab="n_iterations",ylab="mu_n")
title("Trace plot for mu_n")
abline(h=1.5)
abline(h=2.5)
abline(h=0.5)
```