Computer Lab 4

Martynas Lukosevicius, Alejo Perez Gomez, Zahra Jalil Pour

30/11/2020

Question 1: Computation with Metropolis - Hastings

The probability density function is :

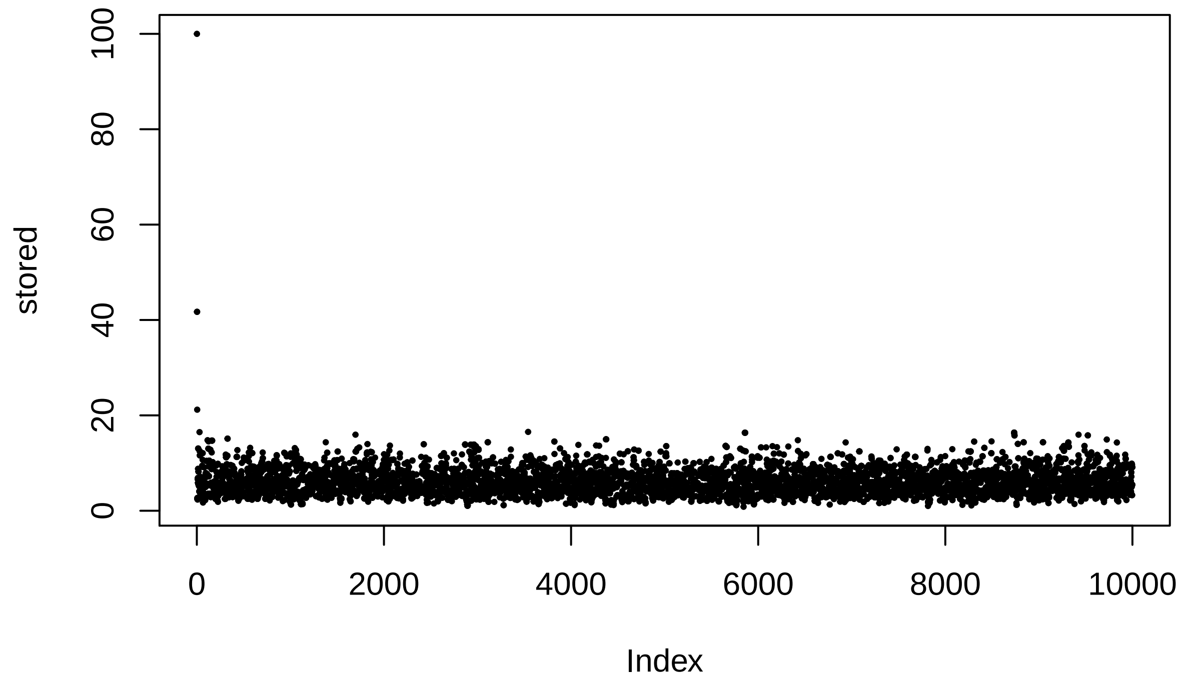
$$f(X) \sim x^5 e^{-x}, x > 0$$

1. Generating samples from target distribution, with metropolis-hastings algorithm, by using Log-normal distribution as proposal distribution.

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


### density of Log-normal function by sigma=1
##dlnorm gives the density
proposed <- function(x, mu){
  res <- dlnorm(x,log(mu), sd=1)
  return(res)
}
### rlnorm generates random deviates.
metro_hast <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){
    x <- stored[i-1]
    xprime <- rlnorm(1,log(x),1)

    ratio <- min(c(1,
                   target(xprime) * proposed(x, xprime)) / (target(x) * proposed(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  #return(stored)
  plot(stored,pch=19,cex=0.3)
  #hist(stored[2000:10000] , breaks = 30 )
  #plot(vN,stored,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",main="",
  #ylim=c(min(x-0.5,-5),max(5,x+0.5)))
}
metro_hast(100,10000)
```



Judging by the plot, the burn-in period consists in the first three samples, followed by a period of convergence around Y=6.

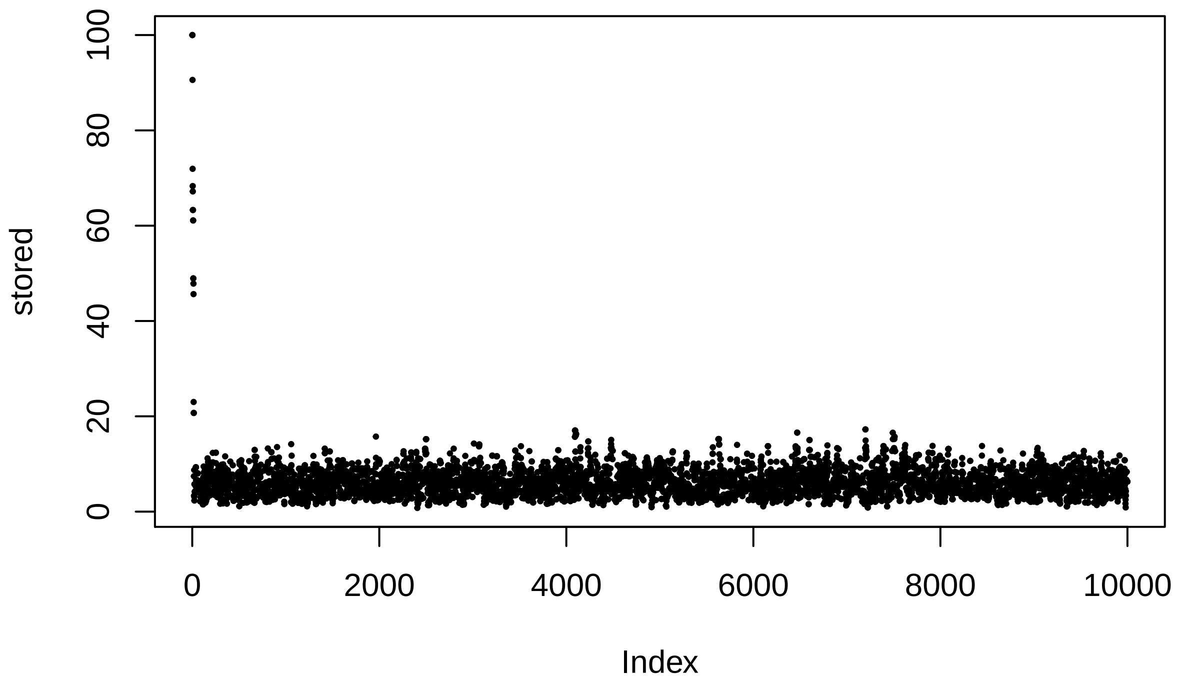
```
metro_hast2 <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rlnorm(1,log(x),1)  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * proposed(x, xprime)) / (target(x) * proposed(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  return(stored)
}
```

2. - performing step 1 by using Chi- Square distribution

$$\tilde{\chi}^2([X_t + 1])$$

```
####chi- square distribution
##The dchisq() function gives the density
p_chi_square <- function(x, df){
  return(dchisq(x, df=floor(x+1)))
}
metro_chi_hast <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rchisq(1,df=floor(x+1))  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * p_chi_square(x, xprime)) / (target(x) * p_chi_square(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  #return(stored)
  plot(stored,pch=19,cex=0.3)
  #hist(stored[5000:10000] , breaks = 30 )
  #plot(vN,stored,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",main="",ylim=c(min(x-0.5,-5),max(5,x+
}
```

```
metro_chi_hast(100,10000)
```



3. Comparing result of step1 and step2:

Our plots lead us to thing that our sampling by means of both techniques (normal-logarithm and chi-squared) converge to the same value and have a similar variance just looking at their oscillation. Both arrays present a similar standard deviation around 2.5 and mean value around 6. Therefore, we can conclude by stating both procedures lead to similar results.

4. Generating 10 MCMC sequences using the generator from step2, by using Gelman-Rubin method:

```r
### Generate 10 MCMC
p_chi_square <- function(x, df){
  return(dchisq(x, df=floor(x+1)))
}
metro_chi_hast2 <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rchisq(1,df=floor(x+1))  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * p_chi_square(x, xprime)) / (target(x) * p_chi_square(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  return(stored)
}
library(coda)
k <- 10
f1 <- mcmc.list()
for(i in 1:k){
  f1[[i]]<-as.mcmc(metro_chi_hast2(i, 10000))
}
```

```r
print(gelman.diag(f1))
```

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

5.

Estimate this function by sampling from step1 and step2:

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

	mean
log_norm	5.945054
chi_square	5.800837

6.

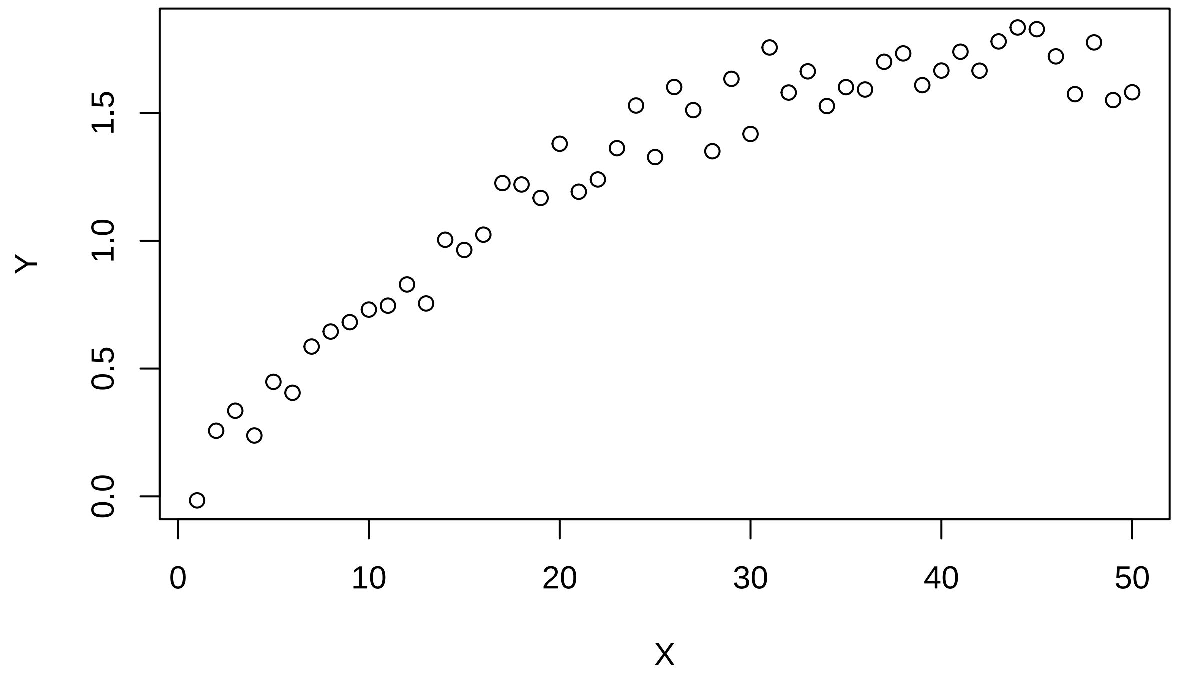
The value of the integral defined for a Gamma distribution is $\alpha\beta$, being these the parameters alpha and beta of such a kind of distribution. It coincides with its Expected Value. In our case, according to the function of probability density $f(X) \sim x^5 e^{-x}, x > 0$, where $\alpha = 6$ and $\beta = 1$, let $E[Y]$ be:

$$E[Y] = \alpha\beta = 6$$

It is the same as the obtained through Metropolis-Hastings Sampling in the previous two approaches.

Question 2. Gibbs sampling

1.



The scatter imaginary line traced by the scatter plot resembles a logarithm function. Therefore, a logarithmic model would probably fit the data.

2.

We know

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

The likelihood of $p(\vec{Y}|\vec{\mu})$ is the probability of observing our $\vec{Y}$ data given a set of parameters $\vec{\mu}$. It is defined like the product of our probability function all over the observed data:

$$\mathcal{L}(\vec{Y}|\vec{\mu}) = \prod_{i=1}^{n} p(\vec{Y}|\vec{\mu})$$

$$\mathcal{L}(\vec{Y}|\vec{\mu}) = (2\pi\sigma^2)^{-\frac{n}{2}} exp[-\frac{\sum_{i=2}^{n}(y_i - \mu_i)^2}{2\sigma^2}]$$

Our prior probability is defined like the following expression according to the chain rule:

$$p(\vec{\mu}) = (2\pi\sigma^2)^{-\frac{n}{2}} exp[-\frac{\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2}{2\sigma^2}]p(\mu_1)$$

where: $\sigma = 0.2$ and $p(\mu_1) = 1$

The posterior probability according to Bayes theorem follows the next equation:

$$p(\vec{\mu}|\vec{Y}) \propto \mathcal{L}(\vec{Y}|\vec{\mu}) * p(\vec{\mu})$$

$$p(\vec{\mu}|\vec{Y}) \propto exp[-\frac{\sum_{i=1}^{n}(y_i - \mu_i)^2}{2\sigma^2}] * exp[-\frac{\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2}{2\sigma^2}]$$

$$p(\vec{\mu}|\vec{Y}) \propto exp[-\frac{(y_1 - \mu_1)^2 + \sum_{i=2}^{n}[(\mu_i - \mu_{i-1})^2 + (y_i - \mu_i)^2]}{2\sigma^2}]$$

3.

Now we will develop a expression for $p(\mu_i|\vec{\mu}_{-i})$ first splitting between $p(\mu_1|\vec{\mu}_{-1,}, p(\mu_n|\vec{\mu}_{-n,}, \vec{Y})$ and the middle points. We leverage the property of conditional probability assuming independent events:

$$p(A|B) = \frac{p(A) \cap (B)}{p(B)} = \frac{p(A)(B)}{p(B)}$$

$$p(\mu_1|\vec{\mu}_{-1,}, \vec{Y}) \propto exp[-\frac{1}{2\sigma^2}[(\mu_1 - \mu_2)^2 + (\mu_1 - y_1)^2]]$$

$$p(\mu_1|\vec{\mu}_{-1,}, \vec{Y}) \propto exp[-\frac{1}{\sigma^2}[\mu_1 - \frac{\mu_2 + y_1}{2}]^2] \sim N(\frac{\mu_2 + y_1}{2}, \frac{\sigma^2}{2})$$

Let the procedure for the last point be applied:

$$p(\mu_n|\vec{\mu}_{-n,}, \vec{Y}) \propto exp[-\frac{1}{2\sigma^2}[(\mu_{n-1} - \mu_n)^2 + (\mu_n - y_n)^2]]$$

$$p(\mu_n|\vec{\mu}_{-n}, \vec{Y}) \propto exp[-\frac{1}{\sigma^2}[\mu_n - \frac{\mu_{n-1} + y_n}{2}]^2] \sim N(\frac{\mu_{n-1} + y_n}{2}, \frac{\sigma^2}{2})$$

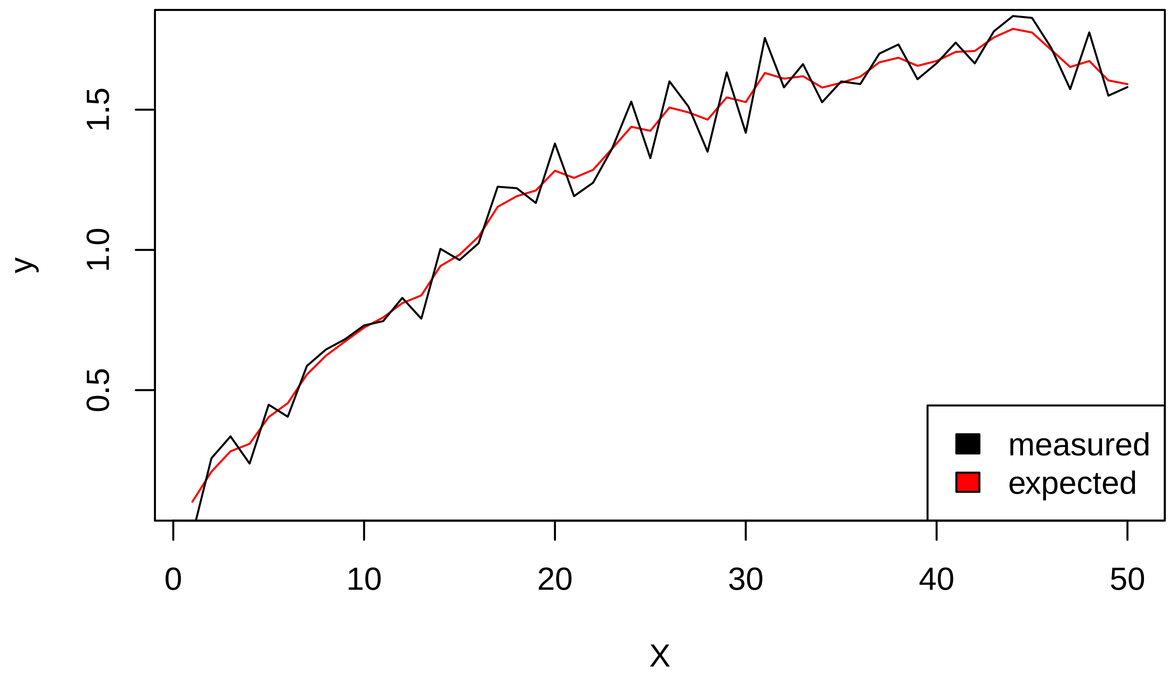
Finally, for the middle points:

$$p(\mu_i|\vec{\mu}_{-i}, \vec{Y}) \propto exp[-\frac{1}{2\sigma^2}[(\mu_{i-1} - \mu_i)^2 + (\mu_i - \mu_{i+1})^2 + (\mu_i - y_i)^2]$$

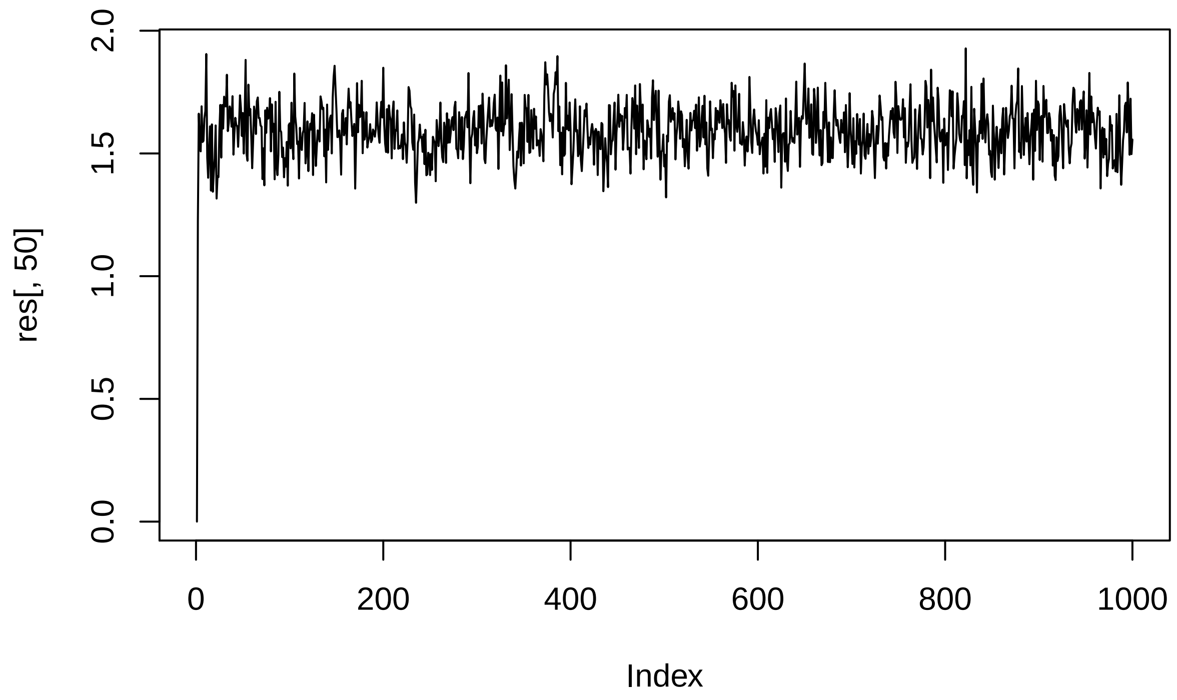
$$p(\mu_i|\vec{\mu}_{-i}, \vec{Y}) \propto exp[-\frac{3}{2\sigma^2}[\mu_i - \frac{\mu_{i-1} + \mu_{i+1} + y_i}{3}]^2] \sim N(\frac{\mu_{i-1} + \mu_{i+1} + y_i}{3}, \frac{\sigma^2}{3})$$

4.

```r
GibsSampler <- function(n,k){
  initx <- as.data.frame(t(rep(0,k)))
  for (i in 2:n) {
    mu <- unlist(initx[i-1, ])
    mu[1] <- rnorm(1,(mu[2]+Y[1])/2,0.2/2)
    for (j in 2:(k-1)) {
      mu[j] <- rnorm(1,(mu[j-1]+mu[j+1]+Y[j])/3,0.2/3)
    }
    mu[k] <- rnorm(1,(mu[k-1]+Y[k])/2,0.2/2)
    initx <- rbind(initx,mu)
  }
  return(initx)
}
```



Using this method it looks like the expected values plotted smooth the noise compared to the observed ones, leading us to observe with more clarity the correlation variables have underlying.



The plot does not present a clear burn out period since it goes directly from value 0 to the span it begins oscillating around. Then, convergence is not notoriously achieved as values of μ fluctuate around 1.5.

Appendix

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


### density of Log-normal function by sigma=1
##dlnorm gives the density
proposed <- function(x, mu){
  res <- dlnorm(x,log(mu), sd=1)
  return(res)
}
### rlnorm generates random deviates.
metro_hast <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){
    x <- stored[i-1]
    xprime <- rlnorm(1,log(x),1)

    ratio <- min(c(1,
```

```r
                   target(xprime) * proposed(x, xprime)) / (target(x) * proposed(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  #return(stored)
  plot(stored,pch=19,cex=0.3)
  #hist(stored[2000:10000] , breaks = 30 )
  #plot(vN,stored,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",main="",
  #ylim=c(min(x-0.5,-5),max(5,x+0.5)))
}
metro_hast(100,10000)

metro_hast2 <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rlnorm(1,log(x),1)  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * proposed(x, xprime)) / (target(x) * proposed(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  return(stored)
}
####chi- square distribution
##The dchisq() function gives the density
p_chi_square <- function(x, df){
  return(dchisq(x, df=floor(x+1)))
}
metro_chi_hast <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rchisq(1,df=floor(x+1))  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * p_chi_square(x, xprime)) / (target(x) * p_chi_square(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  #return(stored)
  plot(stored,pch=19,cex=0.3)
  #hist(stored[5000:10000] , breaks = 30 )
  #plot(vN,stored,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",main="",ylim=c(min(x-0.5,-5),max(5,x+
}
metro_chi_hast(100,10000)
### Generate 10 MCMC
p_chi_square <- function(x, df){
  return(dchisq(x, df=floor(x+1)))
}
metro_chi_hast2 <- function(startvalue, iteration){
  stored <- rep(startvalue, iteration)
  vN<-1:iteration
  for (i in 2:iteration){  ### I am not sure about 2
    x <- stored[i-1]
    xprime <- rchisq(1,df=floor(x+1))  # proposed a value for start and I am not sure about x=1
    ratio <- min(c(1,
                   target(xprime) * p_chi_square(x, xprime)) / (target(x) * p_chi_square(xprime, x)))
    accept <- (runif(1) <= ratio)
    stored[i] <- ifelse(accept, xprime, x)
  }
  return(stored)
}
library(coda)
k <- 10
f1 <- mcmc.list()
for(i in 1:k){
  f1[[i]]<-as.mcmc(metro_chi_hast2(i, 10000))
}

print(gelman.diag(f1))
log_norm <- metro_hast2(100,10000)
chi_square <- metro_chi_hast2(100,10000)

knitr::kable(setNames(c(mean(log_norm),mean(chi_square)), c("log_norm", "chi_square")), col.names = "mea


load("chemical.RData")  # read csv file
plot(X,Y)
GibsSampler <- function(n,k){
  initx <- as.data.frame(t(rep(0,k)))
  for (i in 2:n) {
    mu <- unlist(initx[i-1, ])
    mu[1] <- rnorm(1,(mu[2]+Y[1])/2,0.2/2)
    for (j in 2:(k-1)) {
      mu[j] <- rnorm(1,(mu[j-1]+mu[j+1]+Y[j])/3,0.2/3)
    }
    mu[k] <- rnorm(1,(mu[k-1]+Y[k])/2,0.2/2)
    initx <- rbind(initx,mu)
  }
  return(initx)
}
res <- GibsSampler(1000, length(X))

expectedmean <- unname(unlist(colSums(res/nrow(res))))
plot(X,expectedmean,type = "l", col="red",ylab = "y")
lines(X,Y)
legend("bottomright", c("measured", "expected"), fill=c("black", "red"))
plot(res[, 50], type="l")
```