Lab4_comp_stat

Gowtham(gowku593) & Biswas (bisku859)

3/21/2020

Question 1: Computations with Metropolis-Hastings :

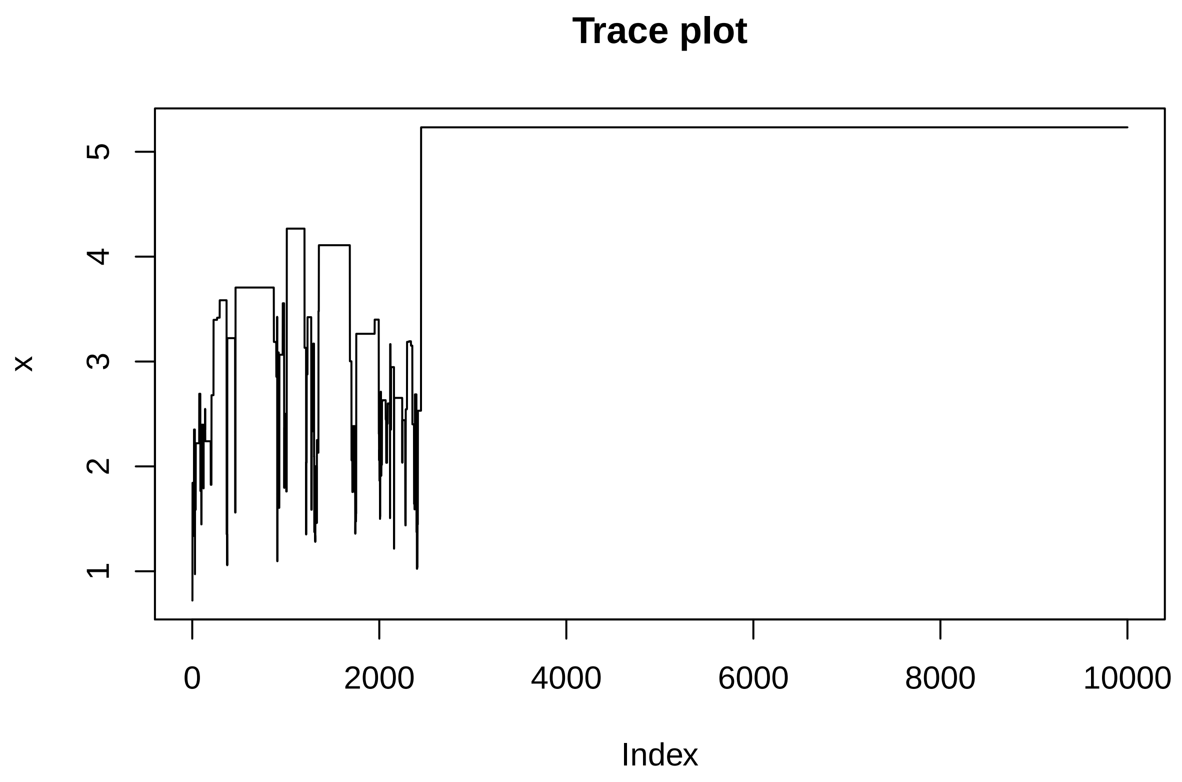
Consider the following probability density function: f(x)~x^{5e}-x , x > 0:

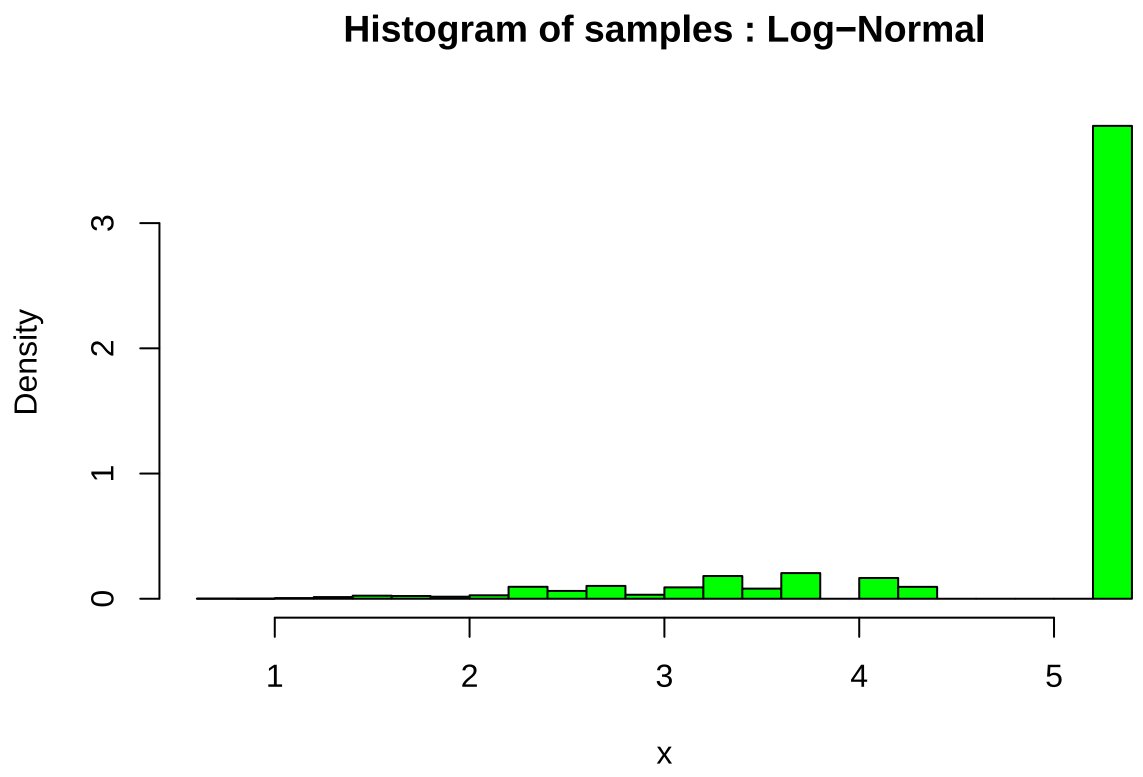
1.

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?

The probabilty distribution function is given. We shall use Markov Chain Monte Carlo method to determine sample data from an unknown distribution.

Herein we implement Metropolis-Hastings algorithm with 10,000 iterations and will proceed as per below sequence : 1. Initialize the starting point , X1 at t=1 or x[t=1] as random sample 2. Build up Y as a candidate point 3. Generate random U 4. Comparision of U and alpha and set x[t] accordingly 5. set t=t+1 and repeat the loop (step 2-5) until t = 10,000





From th trace plot , we observe that most of the time it remains flat or gets stuck which indicated that our chain does not move . It further suggests that it doesnt seems to converge even at a larger iteration.Actually, it remains stuck for most of the iterations after approx. 2200 iterations

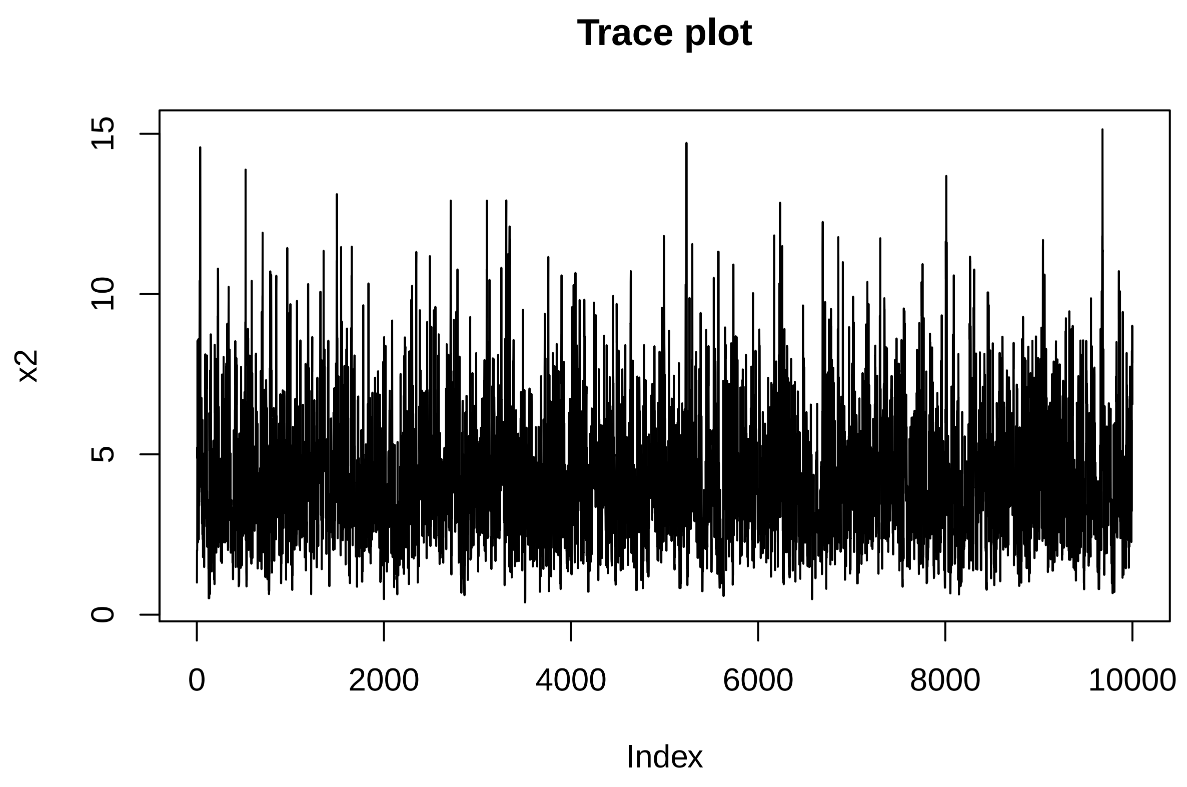
Since we can not see the convergence and movement of chain, it is therefore we can saftely say that there is no such burn-in period for the same.

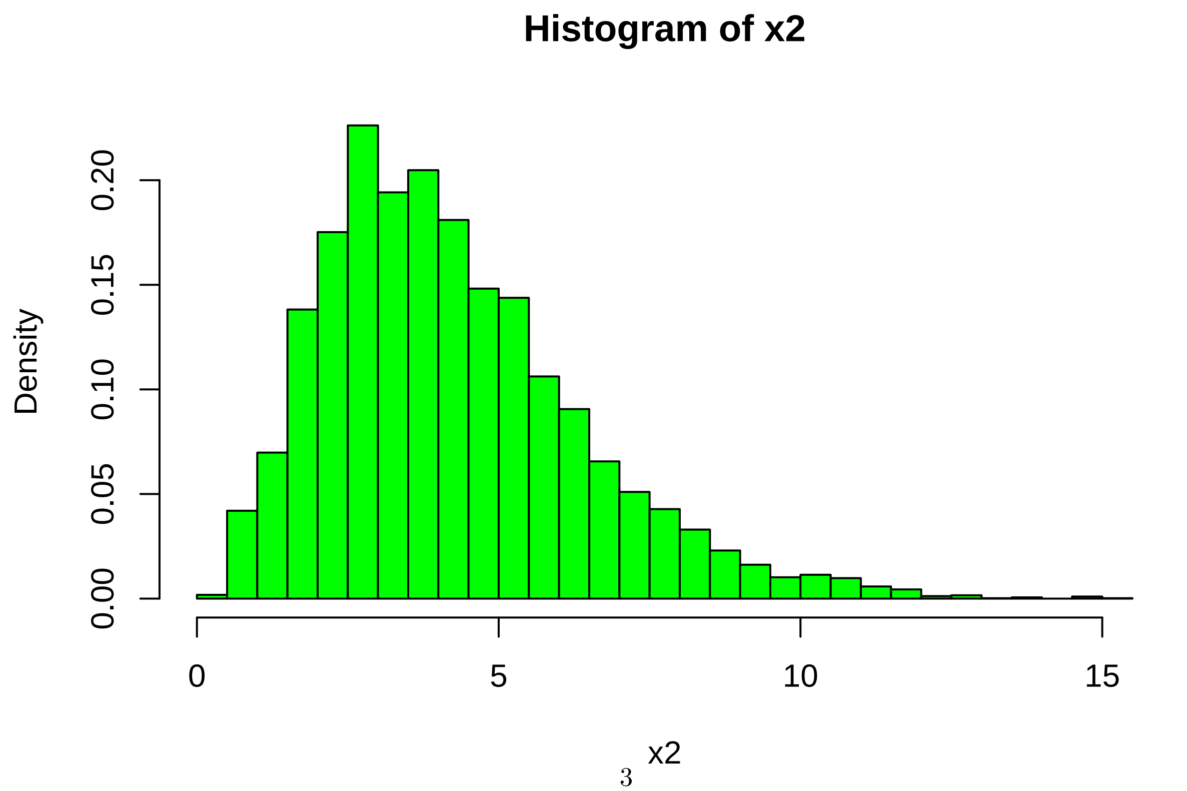
The histogram of the sampled data also doesn't look good too.

2. Perform Step 1 by using the chi{square distribution

$$\chi^2[X_t + 1]$$

as a proposal distribution, where [x] is the floor function, meaning the integer part of x for positive x, i.e.[2.95] = 2





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

The chi-square distribution trace plot of chain responds well to the movement of chain.It continuously moves without getting stuck. From the plot, we estimate some abnormal start and few iternations can be ommited . Hence, we can recommend a short burn-in period of around 20 iterations. The density distribution of histogram is also looking good and seems to follow curve well.

When compared with the trace plot of step 1, the chi-square plot (step 2) is definatively meaningful and respond to the chain (good mixing)with a very short burn-in period. In contrast to the same, we could not find convergence to the target distribution in step 1 even.

In short, chi-square as proposal distribution is more effective then log-normal distribution, to obtain samples from the given distribution.

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.

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

The point est. and upper confidence interval (C.I) is 1, which indicates that the sequence has convergerged . Normally, the range for convergence is between 1.0 to 1.2. The more it is near to 1, the better it reflects about convergence.

5. Estimate

$\int_0^\infty xf(x)\,dx$

using the samples from Steps 1

```
## Estimated Output from step 1 = 4.740649
```

using the samples from Steps 2

```
## Estimated Output from step 2 = 4.204923
```

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 probability density function using the shape-scale parametrization is: $f(x,k,\theta) = \frac{x^{k-1}e^{\frac{-x}{\theta}}}{\theta^k\Gamma(k)}$ for x>0 and $k,\theta > 0$,

substituting the value of $k = 6$ and $\theta = 1$, we get pdf $f(x,k = 6,\theta = 1) \propto x^5e^{-x}$ for x>0 The expected value of Gamma distribution is : $k\theta = 6 \times 1 = 6$

The theoritical expected value is 6.

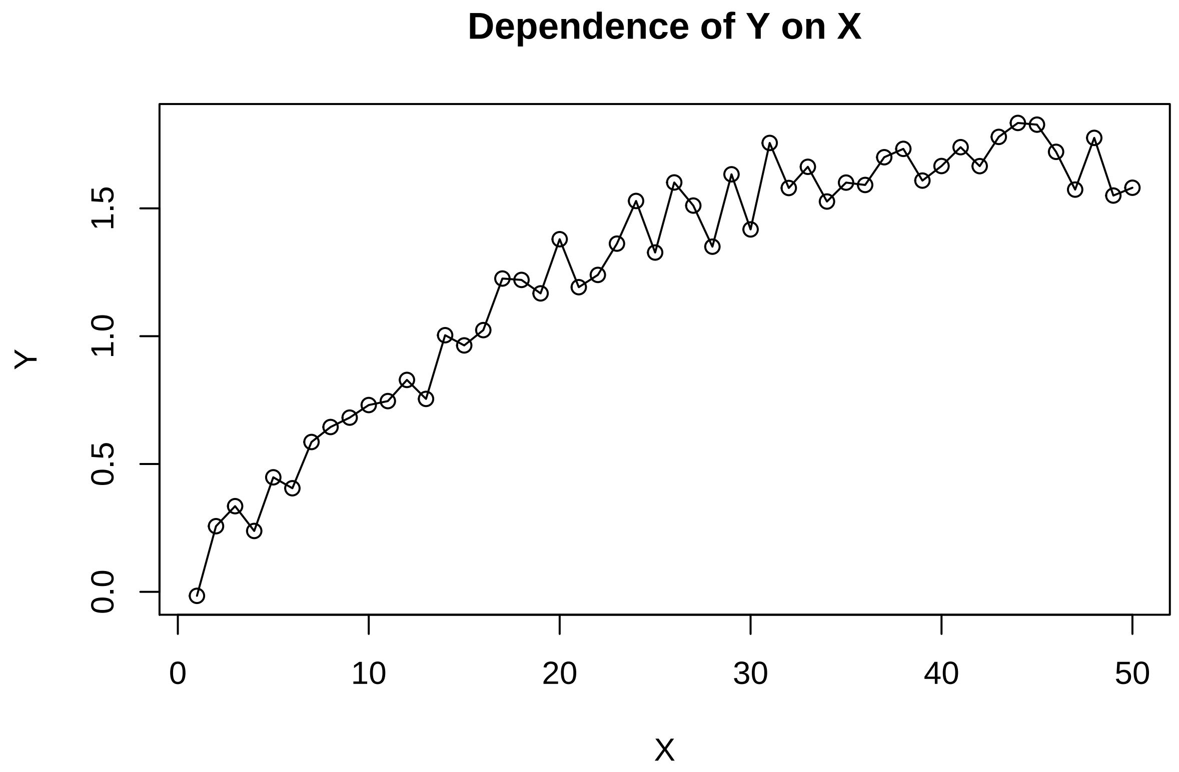
However, on estimation through samples, We are getting around 4.74 and 4.26 from step 1 and step 2 respectively.

We find that our estimation it is not following the theoritical value, even though the Gelman Rubin method suggests convergence of our chi-square proposal distribution from step 2.

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.

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



Looking at graph, we observe that X is incresing with increase in Y in general. We can also see a lot of fluctuation with the rising graph.

It can be concluded that a linear model shall not be able to capture the trend wisely as X slows down a bit after having a almost proportional(or linear) start against Y in beginning.

In such senario, a non -linear model(or probably logistic model) could be the optimum choice.

2. A researcher has decided to use the following (random walk) Bayesian model (n=number of observations,

$\mu = (\mu_1, \cdots, \mu_n)$ are unknown parameters): $Y_i \sim \mathcal{N}(\mu_i, variance = 0.2), i = 1, \cdots, n$ where the prior is : $p(\mu_1) = 1$ $p(\mu_{i+1} \mid \mu_i) = \mathcal{N}(\mu_i, 0.2), i = 1, ..., n1$

Present the formulae showing the likelihood $p(Y \mid \mu)$ and the prior $p(\mu)$. Hint : a chain rule can be used here $p(\mu) = p(\mu_1)p(\mu_2 \mid \mu_1)p(\mu_3 \mid \mu_2) \cdots p(\mu_n \mid \mu_{n1})$

a. The formulae for liklihood for the vector Y and μ is given by $p(Y \mid \mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e - \frac{(x-\mu)^2}{2\sigma^2}$ given $\mathcal{N}(\mu_i, \sigma^2 = 0.2)$

Putting the value of value of σ^2 in the formulae we get : $(2 \times 0.2\pi)^{\frac{-n}{2}} e(-\frac{1}{2\times 0.2}\sum_{i=1}^{n}(Y_i - \mu_i)^2)$ $\frac{1}{(0.4\pi)^{\frac{n}{2}}} e(-\frac{1}{0.4}\sum_{i=1}^{n}(Y_i - \mu_i)^2) \ldots$equation (1)

b. Prior $p(\mu)$ can be obtained by just substituting Y_i for μ_i and μ_i for μ_{i-1} in equation (1) above , as it depends of last value of μ: Given that : $p(\mu_1) = 1$ $p(\mu_2 \mid \mu_1) = (0.4\pi)^{\frac{-1}{2}} e(-\frac{1}{0.4}(\mu_2 - \mu_1)^2)$ $p(\mu_3 \mid \mu_2) = (0.4\pi)^{\frac{-1}{2}} e(-\frac{1}{0.4}(\mu_3 - \mu_2)^2) \cdots p(\mu_n \mid \mu_{n-1}) = (0.4\pi)^{\frac{-1}{2}} e(-\frac{1}{0.4}(\mu_n - \mu_{n-1})^2)$

Applying suggested chain rule in equation one , the outcome is as below :

$\frac{1}{(0.4\pi)^{\frac{n-1}{2}}} e(-\frac{1}{0.4}\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2)$ ….. equation (2)

## 3. Use Bayes' Theorem to get the posterior up to a constant proportionality, and then find out the distributions of $(\mu_i \mid \mu_{-i}, Y)$,where μ_{-i} is a vector containing all μ values except of μ_i. From Bayes theorem :

$p(\mu \mid Y) = \frac{p(Y|\mu)p(\mu)}{p(Y)}$ Now, a. $Posterior \propto Likelihood \times Prior$ or, $p(\mu \mid Y) \propto p(Y \mid \mu)p(\mu)$ Now, multiplying above equation (1 and (2), we can remove all constants and get posterior as proportional :

$\frac{1}{(0.4\pi)^{\frac{n}{2}}} e(-\frac{1}{0.4}\sum_{i=1}^{n}(Y_i - \mu_i)^2) \times \frac{1}{(0.4\pi)^{\frac{n-1}{2}}} e(-\frac{1}{0.4}\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2)$

or, $\frac{1}{(0.4\pi)^{\frac{n}{2}+\frac{n-1}{2}}} e[(-\frac{1}{0.4}(\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2 + \sum_{i=1}^{n}(Y_i - \mu_i)^2)]$

or, $\frac{1}{(0.4\pi)^{\frac{2n-1}{2}}} e[(-\frac{1}{0.4}(\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2 + \sum_{i=1}^{n}(Y_i - \mu_i)^2)]$

or, $\frac{1}{(0.4\pi)^{n-\frac{1}{2}}} e[(-\frac{1}{0.4}(\sum_{i=2}^{n}(\mu_i - \mu_{i-1})^2 + \sum_{i=1}^{n}(Y_i - \mu_i)^2)]$

In general we can also write it as : or, $\propto e[(-\frac{1}{0.4}(Y_i - \mu_i)^2) + \sum_{i=2}^{n}((\mu_i - \mu_{i-1})^2 + (Y_i - \mu_i)^2)] \ldots$ equation (3)

b. Using equation (3), the distributions of $(\mu_i \mid \mu_{-i}, Y)$ is given by : $p(\mu_i \mid \mu_{-i}, Y) = e - \frac{1}{2\times 0.2}[(\mu_1 - Y_1)^2) + (\mu_1 - \mu_2)^2)]$ On using hint B , the expression can be re-written as :

$e(-\frac{(\mu_1 - \frac{(Y_1+\mu_2)}{2})^2}{\frac{2\times 0.2}{2}})$

The normal distribution is given by :

$\sim \mathcal{N}(\frac{Y_1+\mu_2}{2}, 0.1) \ldots$ where, $\frac{\sigma^2}{2} = 0.1$

Similarly, the distributions of $(\mu_n \mid \mu_{-n}, Y)$ is given by : $= e - \frac{1}{2\times 0.2}[(\mu_n - Y_n)^2) + (\mu_n - \mu_{n-1})^2)]$ Using hint B we get and solving we get Normal distribution is given as:

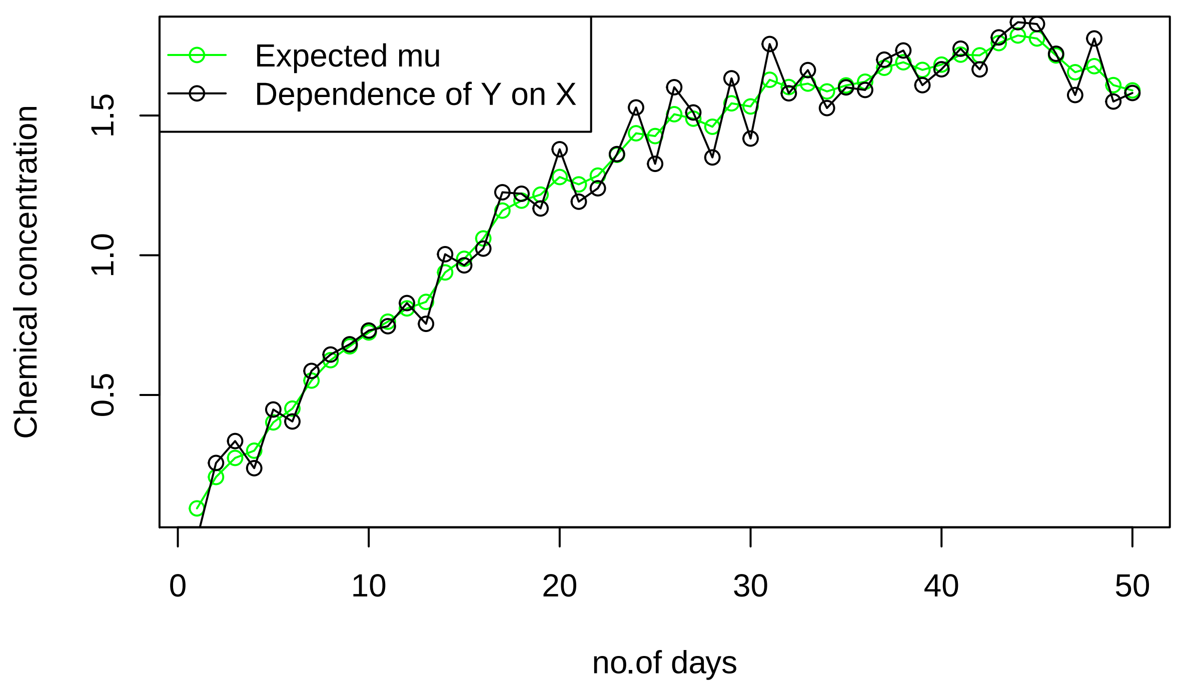
$\sim \mathcal{N}(\frac{Y_n+\mu_{n-1}}{2}, 0.1)$

Lastly, the distributions of $(\mu_i \mid \mu_{-i}, Y)$ is given by : $= e - \frac{1}{2\times 0.2}[(\mu_i - Y_i)^2) + (\mu_i - \mu_{i-1})^2 + (\mu_i - \mu_{i+1}))]$ Using hint C we get :

$\sim \mathcal{N}(\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{2}, \frac{0.2}{3})$

4. Use the distributions derived in Step 3 to implement a Gibbs sampler that uses

$\mu^0 = (0, \cdots, 0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of μ and then compute the expected value of μ by using a Monte Carlo approach. Plot the expected value of μ 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 μ can catch the true underlying dependence between Y and X?



The expected mu has been plotted as green on the graph while the original Y has been highlighted as black connecting data points.

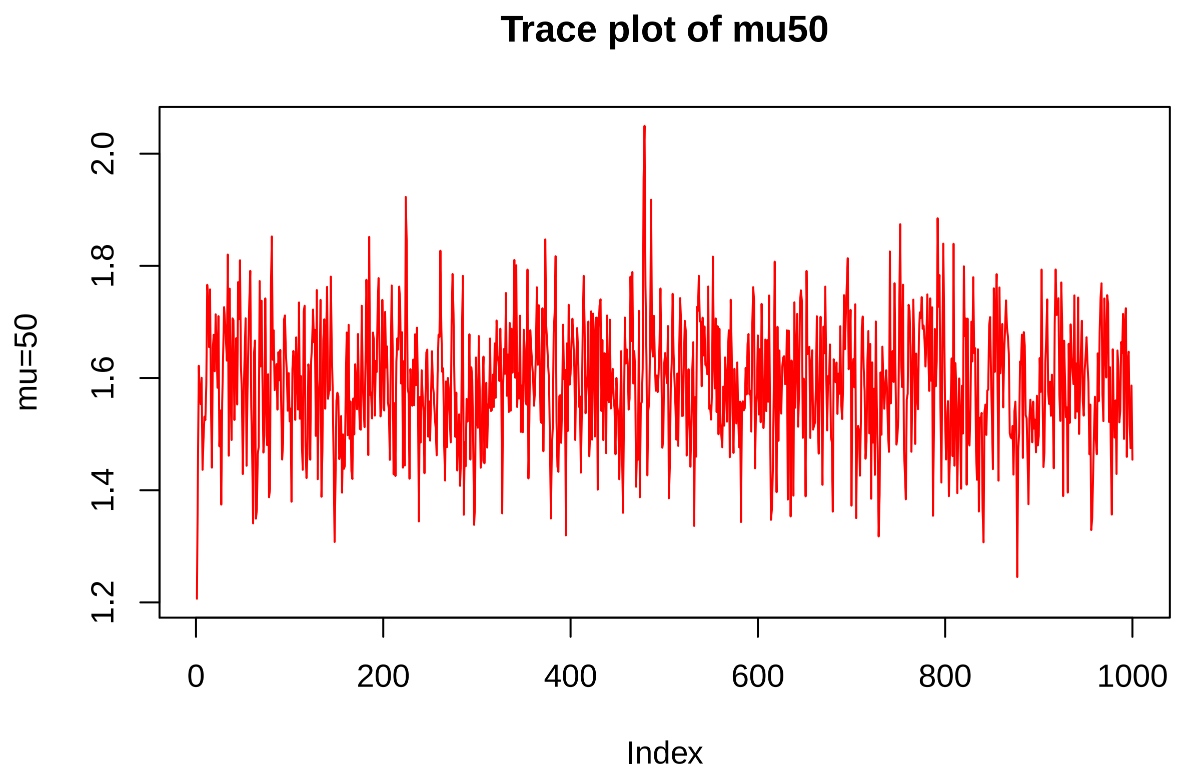
The resultant graph (expected mu) looks smooth when compared with the turbulant original Y on X plot. It is therefore, we can say that we have managed to remove the noise to a far extent.

Moreover, we have excluded first 30 values abnormal values because of burn-in period. This is to ensure the effectiveness of our model from the beginning itself.

Lastly, based on above observations, We can say that the expected value of mu (μ) is following the true underlying dependence between Y and X.

5. Make a trace plot for :

```
$\mu_{n}$ and comment on the burn-in period and convergence.
```



Looking at the trace plot of mu 50 , we can say that the graph started with a big difference and as algorithm progressed, it establizes itself within range. This initial stage of abnormal start is known burn-in period. We can further estimate here that this burn-in period is very short and the chain has converged. These initial burn-in period should be excluded from the analysis as it does not give right interpretation of model.

Even if a graph fails to capture burn-in period because of some reason, it is advisable to let go initial observations for further analysis purpose.

Code Appendix

```
knitr::opts_chunk$set(echo = FALSE)
library(readxl)
library(coda)

RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")

# Probability Distribution Function :
my_fun = function(x){
  if(x<0){
    return(0)}
  else {
    return(x^5 * exp(-x))
  x^5 * exp(-x)
```

```r
    }
}

#Creating rnorm normal generation function
my_norm=function(x){
rnorm(1,mean= x, sd=1)
}

#Creating rlog normal generation function
my_logn=function(y1){
rlnorm(1,meanlog =1)
}

#Creating rnorm normal generation function
my_rlnorm=function(x){
  rlnorm(1,meanlog =x, sdlog = 1)
}

#Creating dlnorm normal generation function
my_dlnorm=function(x,m){
  dlnorm(x,meanlog = m,sdlog = 1)
}

# Following steps as per lecture notes
# Initialize chain or x1 (length- 10000) and stating value = random number
t_max<-10000
x<-rep(0,t_max)
x[1]=runif(1) # starting value choosing randomly as per theory
t=2 # second position
while(t <= t_max){ # between 2 and 10,000
  #instant_x<- my_rnorm(x0) ;
  #Generate a candidate point
  Y<- my_rlnorm(x[t-1])
  #Generate U
  U <- runif(1)
  alpha<- min(1,((my_fun(Y)*my_dlnorm(x[t-1],Y))/(my_fun(x[t-1])*my_dlnorm(Y,x[t-1]))))
  if(U < alpha){
       x[t]= Y}
  else {x[t]=x[t-1]}
  t=t+1
}
x1<-x # saving for for later use
plot(x,type='l',main="Trace plot")
hist(x,probability = TRUE,breaks=20,
     main = "Histogram of samples : Log-Normal",col="green")


RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")

my_chi = function(x){
  x<-floor(x+1) # floor function for x
  return(rchisq(1, df=x))
```

```r
}

# Initialize chain or x (length- 1000) and stating value = x_0
chi_square<-function(x_0){
t_max<-10000
x<-rep(0,t_max)

x[1]=x_0# starting value based on input function
t=2
while(t <= t_max){
  #instant_x<- my_rnorm(x0) ;
  #Generate a candidate point
  Y<- my_chi(x[t-1])
  #Generate U
  U <- runif(1)
  alpha<- min(1,((my_fun(Y)*my_norm(x[t-1]))/(my_fun(x[t-1])*my_norm(Y))))
  if(U < alpha){
       x[t]= Y}
  else {x[t]=x[t-1]}
  t=t+1
}
x2<-x # saving for later use

return (x2)
}
x2<-chi_square(1)
plot(x2,type='l',main="Trace plot")
hist(x2,probability = TRUE,breaks=30,col="green")


# functions from step 2
RNGversion("3.6.2")
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")

t_max<-10000 # maximum iterations
# preparing list to save from loop , passing starting points
#(i) from 1 to 10 for chi_sqware function (from step 2)
sequence_chi<-mcmc.list()
for(i in 1:10){
  sequence_chi[[i]]<-as.mcmc(chi_square(i))
}
# to check convergence
print(gelman.diag(as.mcmc.list(sequence_chi)))




# As per lecture slides f(x)= g(x)*p(x) where integration over p(x)=1
# when comparing with step 1 , we get g(x)=x1
#As we know, Expected value of x1 is mean of x1 therefore -
mean_x1<-(sum(x1)/length(x1))
cat("Estimated Output from step 1 =", mean_x1)
```

```r
# neglecting first 100 for burn-in
mean_x2<-mean(chi_square(1)[-c(1:100)])
cat("Estimated Output from step 2 =", mean_x2)


chemical=load("chemical.RData")
#get(chemical)
plot(X,Y,type="o",main = "Dependence of Y on X")

data_values<-1000
n=length(Y)
# building a matrix of 1000 x 50
gibbs_matrix <- matrix(nrow=data_values,ncol=n)
mu_vector <- rep(0, 50)
for(j in 1:data_values){
for(i in 1:n){
# three conditions : i=1, i=50 , i=2 to 49 as discussed in algorithm,sd values taken from sd as well
  if(i == 1){mu_vector[i] <- rnorm(1, mean=( (mu_vector[i+1]+Y[i])/2 ), sd=(0.2/2))}
  else if(i == 50){mu_vector[i] <- rnorm(1, mean=( (mu_vector[i-1]+Y[i])/2 ), sd=(0.2/2))}
  else{mu_vector[i] <- rnorm(1, mean=( (mu_vector[i-1]+mu_vector[i+1]+Y[i])/3 ), sd=(0.2/3))}
             }
gibbs_matrix[j,] <- mu_vector
}

# expected_mu <- colMeans(gibbs_matrix), other than monte carlo approach
# Expected value of mu using Monte Carlo approach : to apply the mean
# function to the gibbs matrix (1000 x50)  and return the mean of each column,
# also discarded first 30 from data set as burn-in period estimation

expected_mu <-apply(X=gibbs_matrix[30:1000,],MARGIN=2,FUN=mean)

# Plotting the expecetd mu
plot(X, expected_mu, type='o', col='green',
     xlab="no.of days", ylab="Chemical concentration")
points(X,Y, type='o', col='black',lwd=1,pch=21)
legend("topleft", c("Expected mu","Dependence of Y on X"),
       col=c("green","black"),lwd=1,pch=21)

#traceplot of mu : the 50th column of the database
plot(gibbs_matrix[,50], type='l',
     ylab="mu=50", main="Trace plot of mu50",col="red")
```