Lab 4

Group 6

11/30/2020

Question 1: Computations with Metropolis Hastings

Target density :

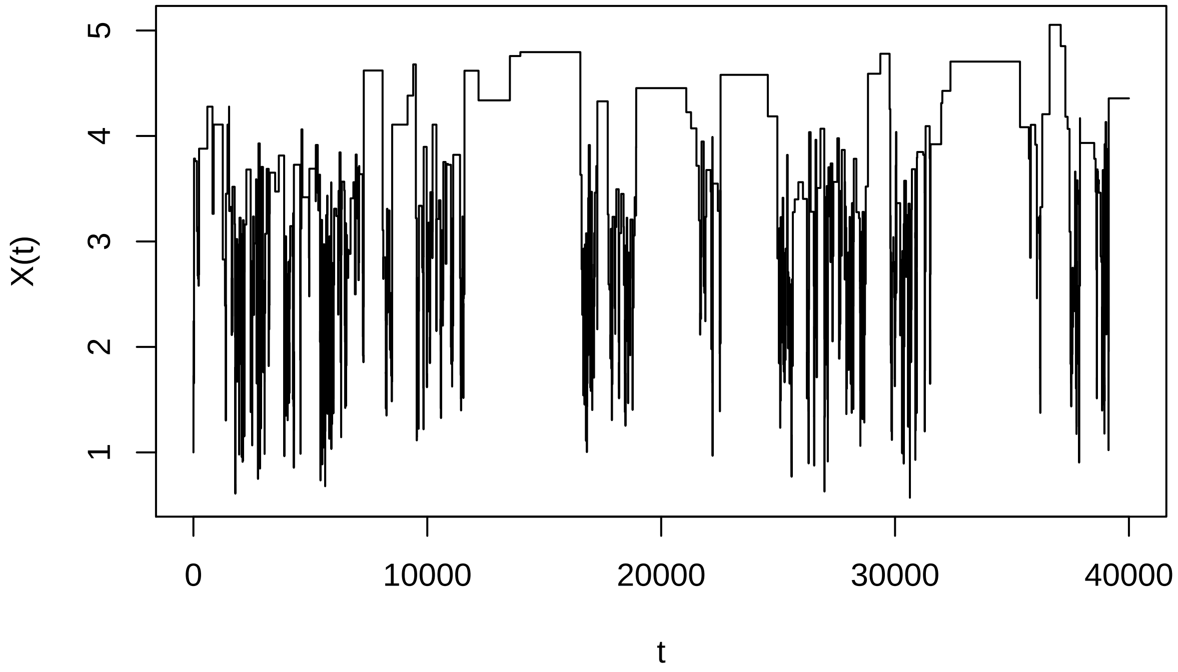
```
f = function(x){
  res = x^5 * exp(-x)
  return(res)
}
```

1

Using Metropolis Hastings algorithm to generate samples from given distribution by using proposal distribution as log normal LN(Xt; 1), taking 1 as starting point.

```
propose_lognorm_MCMC<-function(nstep,X0){
  vN<-1:nstep
  vX<-rep(X0,nstep);
  for (i in 2:nstep){
    X<-vX[i-1]
    Y<-rlnorm(1,mean=X,sd=1)
    u<-runif(1)
    a<-min(c(1,(f(Y)*dlnorm(X,meanlog = Y,sdlog = 1))/(f(X)*dlnorm(Y,meanlog = X,sdlog = 1))))
    if(u<=a){vX[i] = Y} else {vX[i] = X}
  }
  return(vX)
}
set.seed(12345)
lognorm = propose_lognorm_MCMC(40000,1)

plot(1:40000,lognorm,pch=19,cex=0.3,type = "l",col="black",xlab="t",ylab="X(t)",main="")
```



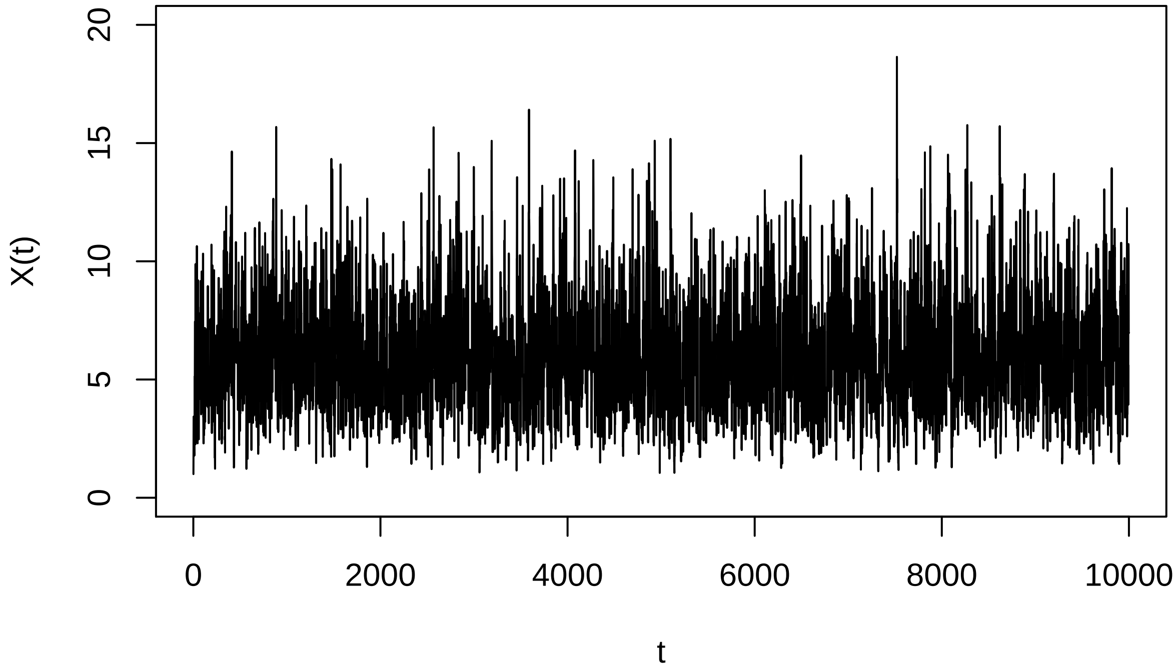
The chain takes different amount of time to converge depending on the selected starting point , this period is called as burn-in period. In the above graph we can see that the chain did not converge. Hence there is no burn in period.

2

Using the chi square distribution $\chi^2([X_t + 1])$ as a proposal distribution, where $[x]$ is the floor function.

```
propose_chi_MCMC<-function(nstep,X0){
  vN<-1:nstep
  vX<-rep(X0,nstep);
  for (i in 2:nstep){
    X<-vX[i-1]
    Y<-rchisq(1,df=floor(X+1))
    u<-runif(1)
    a<-min(c(1,(f(Y)*dchisq(X,floor(Y+1)))/(f(X)*dchisq(Y,floor(X+1)))))
    if(u<=a){vX[i] = Y} else {vX[i] = X}
  }
  return(vX)
}
set.seed(12345)
chi = propose_chi_MCMC(10000,1)

plot(1:10000,chi,pch=19,cex=0.3,type = "l",col="black",xlab="t",ylab="X(t)",main="",ylim=c(0,20))
```



In the above chain we can observe that chain converges in the very begining , ie to say it follows a similar pattern fluctuating around a mean value. Convergence seem to have achieved near the starting point , so we can consider first few iterations as burn in period.

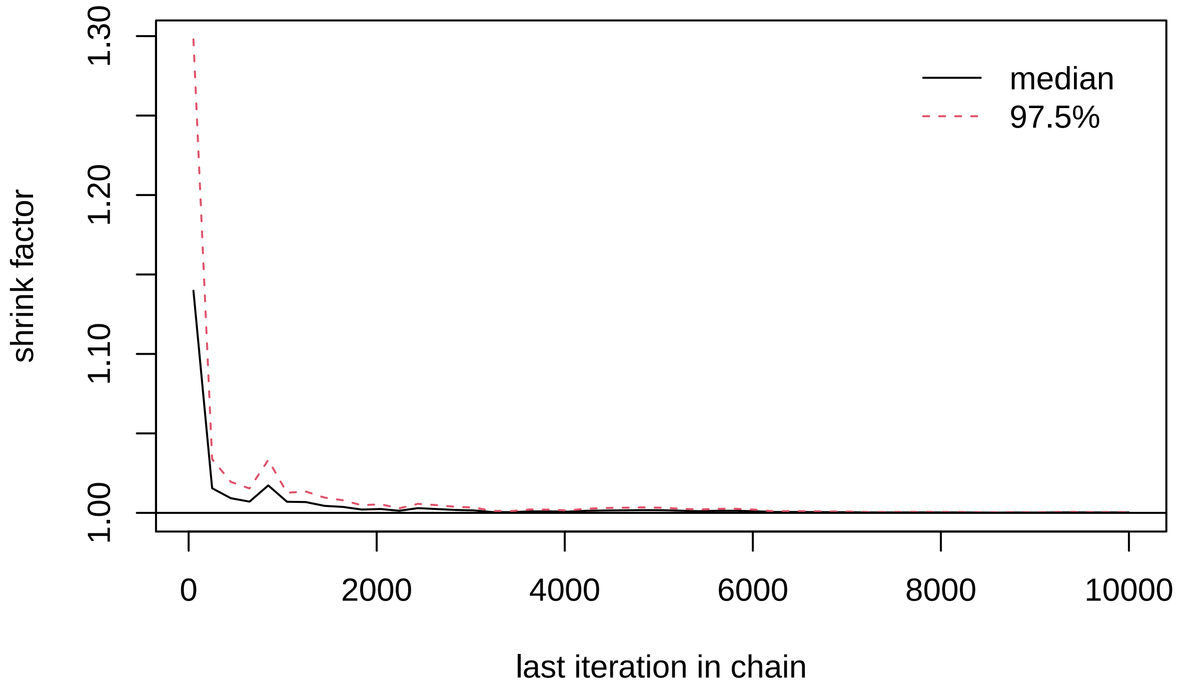
3

In conclusion we can say that chi squared is the better choice as a proposal distribution for the given target distribution, this can be justified by observing the time series plot for both proposal distributions, we could not find a proper convergence for log normal distribution , where as in sampling with chi squared convergence was very fast with a small burnin period.

4

```r
# reference : http://ugrad.stat.ubc.ca/R/library/coda/html/gelman.diag.html
library(coda)
set.seed(12345)
mcmc_data = mcmc.list()
for(i in 1:10){
  mcmc_data[[i]] = as.mcmc(propose_chi_MCMC(10000,i))
}

gelman.plot(mcmc_data)
```



```r
gelman.diag(mcmc_data)
```

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

Two ways to estimate the variance of stationary distribution 1. The mean of the emperical variance within each chain W 2. The emperical variance from all chain combined, this can be expressed as

$$\sigma^2 = \frac{(n-1)W}{n} + \frac{B}{n}$$

where n is number of iteration and $\frac{B}{n}$ is emperical between chain variance.

Assumption : Target distribution is normal. A bayesian probaility interval can be constructed using t distribution with

$$\hat{\mu} = samplemeanofallchainscombined$$

variance

$$\hat{V} = \sigma^2 + \frac{B}{mn}$$

degrees of freedom as

$$d = \frac{2\hat{V}}{var(\hat{V})}$$

The convergence diagnostic iteself is given by :

$$R = \sqrt{\frac{(d+3)\hat{V}}{(d+1)W}}$$

The value substantially above 1 indicates that the lack of convergence , we have obtained this factor as 1 confirming the convergence.

5

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

Step 1 : Mean of generated value in step 1 is given by

$$\frac{1}{n}\sum_i x_i$$

where x is generated values with log normal as proposal density.

```
mean(lognorm)
```

```
## [1] 3.854309
```

Step 2 : Here x is generated values with chi square as proposal density.

```
mean(chi)
```

```
## [1] 6.000143
```

6

The probability density function of gamma distribustion is given by

$$f(x;\alpha,\beta) = \left[\frac{1}{\Gamma(\alpha)\beta^\alpha}\right] x^{\alpha-1} exp^{\frac{-x}{\beta}}$$

$$0 < x < \infty$$

Comparing given density function and gamma function , we can find alpha and beta as follows

$$\alpha - 1 = 5$$

ie

$$\alpha = 6$$

$$\frac{1}{\beta} = 1$$

ie

$$\beta = 1$$

Mean or expected value for gamma distribution is given by

$$\mu = \alpha\beta$$

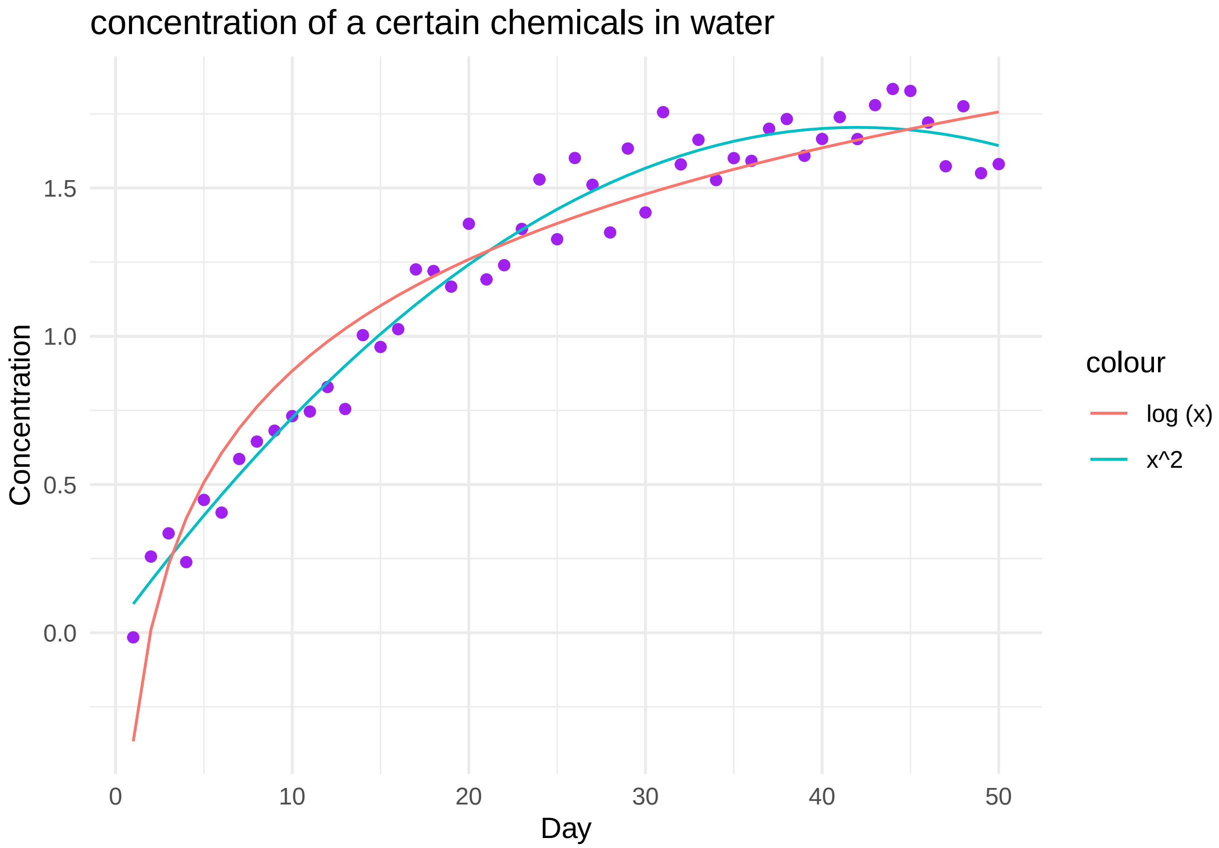
ie 6*1 = 6

Our estimation in step 2 is 6.0001431 which is almost 6. This indicates that chi square is a good choice for proposal distribution for the given target distribution.

Question 2: Gibbs sampling

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

What kind of model is reasonable to use here?



The model is noisy so a simple linear regression can not be used to describe the model. In the plot we have shown a second degree polynomial regression and log(x). Best on the plot, the second degree polynomial is the better choice.

2

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

$$Y_i = \mathcal{N}(\mu, \sigma = 0.2), i = 1, ..., n$$

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 $p(Y \ \mu)$ and the prior $p(\mu)$. Hint: a chain rule can be used here $p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)...p(\mu_n|\mu_{n-1})$. the PDF of Gaussian Distribution is :

$$\mathcal{N}(\mu,\, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\vec{y_i} - \vec{\mu_i})^2}{2\sigma^2}}$$

and $\sigma^2 = 0.2$

to obtain a formulae for likehood we should take product of this PDF:

$$P(\vec{Y}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi 0.2^2}} exp(-\frac{(y_i - \mu_i)^2}{2 * 0.2^2}) = \Big(\frac{1}{\sqrt{0.08\pi}}\Big)^n exp\Big(-\frac{\sum_{i=1}^{n}(y_i - \mu_i)^2}{0.08}\Big)$$

Prior formulae:

$$p(\vec{\mu}) = \prod_{i=1}^{n-1} \frac{1}{\sqrt{2\pi * 0.2^2}} exp(-\frac{(\mu_{i+1} - \mu_i)^2}{2 * 0.2^2})$$

$$P(\vec{\mu}) = \left(\frac{1}{\sqrt{2\pi * 0.2^2}}\right)^n \exp\left[-\frac{1}{2 * 0.2^2}\sum_{i=1}^{n-1}(\vec{\mu_{i+1}} - \vec{\mu_i})^2\right]$$

3:

2.3:

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

From Bayes' Theorem we know that:

$$Posterior \propto likelihood * prior$$

hence,

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

if $n = 1$ from Hint A:

$$P(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \propto \exp\left[-\frac{2}{2 * 0.2^2}\left[\mu_1 - \frac{y_1 + \mu_2}{2}\right]^2\right] \sim \mathcal{N}(\frac{\mu_2 + y_1}{2}, \frac{0.2^2}{2})$$

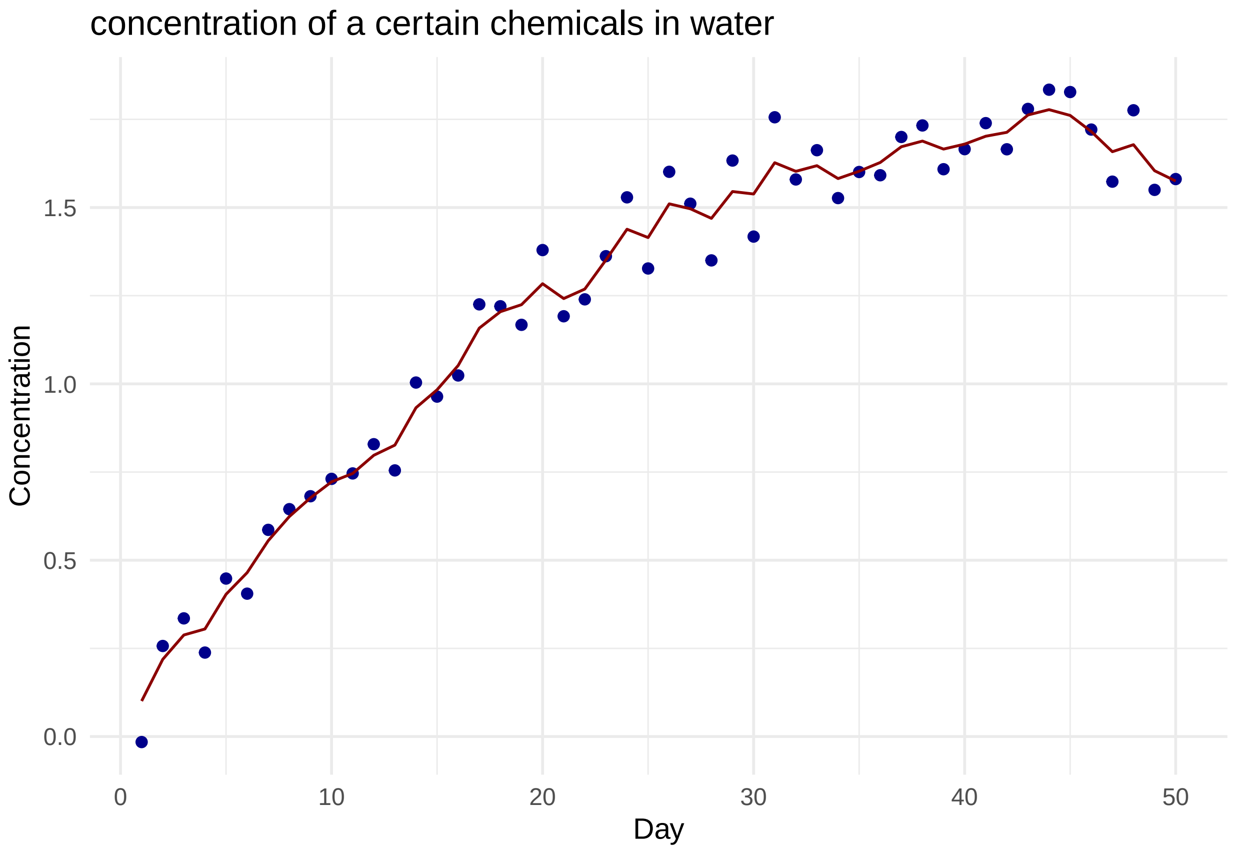
if $i \neq 1, n$ from Hint c:

$$P(\mu_i|\vec{\mu}_{-i}, \vec{Y}) \propto \exp\left[-\frac{3}{0.08}\left[\mu_i - \frac{y_i + \mu_{i-1} + \mu_{i+1}}{3}\right]^2\right] \sim \mathcal{N}(\frac{\mu_{i-1} + \mu_{i+1} + y_i}{3}, \frac{0.2^2}{3})$$

if $i = n$ from Hint B:

$$P(\mu_n|\vec{\mu}_{-n}, \vec{Y}) \propto \exp\left[-\frac{1}{0.04}\left[\mu_n - \frac{y_n + \mu_{n-1}}{2}\right]^2\right]) \sim \mathcal{N}(\frac{\mu_{n-1} + y_n}{2}, \frac{0.2^2}{2})$$

4



By using the formulae we calculated in 2.3 , we implemented a Gibbs Sampler. The algorithm ran for 1000 times. In 1000 time the Gibbs Sampler seems to capture the original data and is a better model for our data than quadratic.

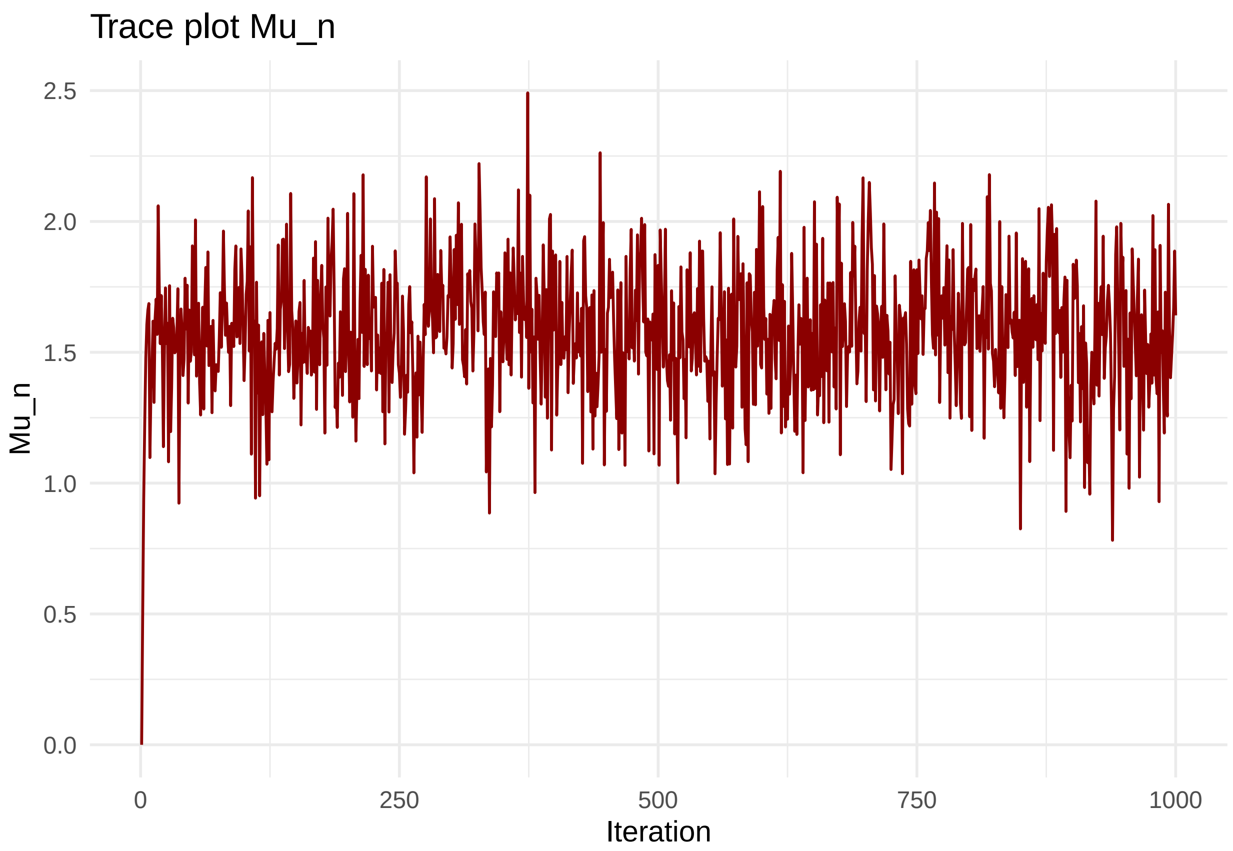
5:

trace plot of μ_n over its 1000 iterations. The trace plot, shows the sampled values of a parameter over time. It exhibits rapid up-and-down variation with almost no long-term trends or drifts.The burn-in period of the chain slows down near convergence (almost after 450 iterations).

```
mu_n = as.data.frame(mu)

gg4 = ggplot(data = mu_n, aes(x = 1:nrow(mu_n), y = mu_n[,50])) +
  geom_line(color="darkred") +  theme_minimal()+ggtitle("Trace plot Mu_n")+theme_minimal()+xlab("Iterat

gg4
```


Trace plot Mu_n
Mu_n
2.5
2.0
1.5
1.0
0.5
0.0
0
250
500
750
1000
Iteration

Appendix

```r
knitr::opts_chunk$set(echo = TRUE)
f = function(x){
  res = x^5 * exp(-x)
  return(res)
}
propose_lognorm_MCMC<-function(nstep,X0){
  vN<-1:nstep
  vX<-rep(X0,nstep);
  for (i in 2:nstep){
    X<-vX[i-1]
    Y<-rlnorm(1,mean=X,sd=1)
    u<-runif(1)
    a<-min(c(1,(f(Y)*dlnorm(X,meanlog = Y,sdlog = 1))/(f(X)*dlnorm(Y,meanlog = X,sdlog = 1))))
    if(u<=a){vX[i] = Y} else {vX[i] = X}
  }
  return(vX)
}
set.seed(12345)
lognorm = propose_lognorm_MCMC(40000,1)

plot(1:40000,lognorm,pch=19,cex=0.3,type = "l",col="black",xlab="t",ylab="X(t)",main="")
propose_chi_MCMC<-function(nstep,X0){
  vN<-1:nstep
  vX<-rep(X0,nstep);
  for (i in 2:nstep){
    X<-vX[i-1]
    Y<-rchisq(1,df=floor(X+1))
    u<-runif(1)
    a<-min(c(1,(f(Y)*dchisq(X,floor(Y+1)))/(f(X)*dchisq(Y,floor(X+1)))))
    if(u<=a){vX[i] = Y} else {vX[i] = X}
  }
  return(vX)
}
set.seed(12345)
chi = propose_chi_MCMC(10000,1)

plot(1:10000,chi,pch=19,cex=0.3,type = "l",col="black",xlab="t",ylab="X(t)",main="",ylim=c(0,20))
# reference : http://ugrad.stat.ubc.ca/R/library/coda/html/gelman.diag.html
library(coda)
set.seed(12345)
mcmc_data = mcmc.list()
for(i in 1:10){
  mcmc_data[[i]] = as.mcmc(propose_chi_MCMC(10000,i))
}

gelman.plot(mcmc_data)

gelman.diag(mcmc_data)
mean(lognorm)
mean(chi)
#--------------------------------------------------------------
```

```r
#A2
#---------------------------------------------------------------
load("chemical.RData")
require(ggplot2)
data = data.frame("X" = X,
                  "Y" = Y)
x.2 <- lm(Y ~  poly(poly(X, 2)) ,
          data = data)
lg.x = lm(Y ~ log(X),
          data = data)

gg <- ggplot(data, aes(x = X, y = Y)) +
  geom_point(color="purple")+
  ggtitle("concentration of a certain chemicals in water")+
  geom_line(aes(x = X, y = predict(x.2), colour = "x^2"))  +
  geom_line(aes(x = X, y = predict(lg.x), colour = "log (x)"))+  theme_minimal() +
  xlab("Day") + ylab("Concentration")

gg
## the model is noisy

#source : 732A90_ComputationalStatisticsHT2020_Lecture04codeSlide18.R

f.MCMC.Gibbs<-function(nstep, X0, Y, mVar){
  d <- length(Y)
  X0 <- rep(0, d)
  vN<-1:nstep
  mX<-matrix(0,nrow=nstep,ncol=length(Y))
  mX[1,]<-X0

  for (i in 2:nstep){
    y_old<-mX[i-1,]
    y_new<-rep(0,length(Y))
    y_new[1]<-rnorm(1, mean=((y_old[2]+Y[1])/2), sqrt(mVar)/2)

   # for i's between 1 to 50

    for (j in 2:(d-1)){
      y_new[j]<-rnorm(1, mean=(Y[j] + y_old[j-1] + y_old[j+1])/3, sqrt(mVar)/3)
    }

    #i=50,updating y

    y_new[d]<-rnorm(1, mean=((Y[d]+y_old[d-1])/2), sqrt(mVar)/2)
    mX[i,]<-y_new
  }
  mX
}
set.seed(12345)
mu = f.MCMC.Gibbs(nstep=1000,X0, Y, mVar=0.2)
final_predicted_mu  = colMeans(mu)

data = cbind(data, final_predicted_mu)
```

```r
gg3  <-  ggplot(data) +
  geom_point(aes(x=X, y=Y), color="darkblue") +
  geom_line(aes(x=X, y=final_predicted_mu), color="darkred") +
  theme_minimal() + xlab("Day") +
  ylab("Concentration") +
  ggtitle("concentration of a certain chemicals in water") + theme(legend.position="right")

gg3

mu_n = as.data.frame(mu)

gg4 = ggplot(data = mu_n, aes(x = 1:nrow(mu_n), y = mu_n[,50])) +
  geom_line(color="darkred") +  theme_minimal()+ggtitle("Trace plot Mu_n")+theme_minimal()+xlab("Iterat

gg4
```