Computational Statistics Lab4

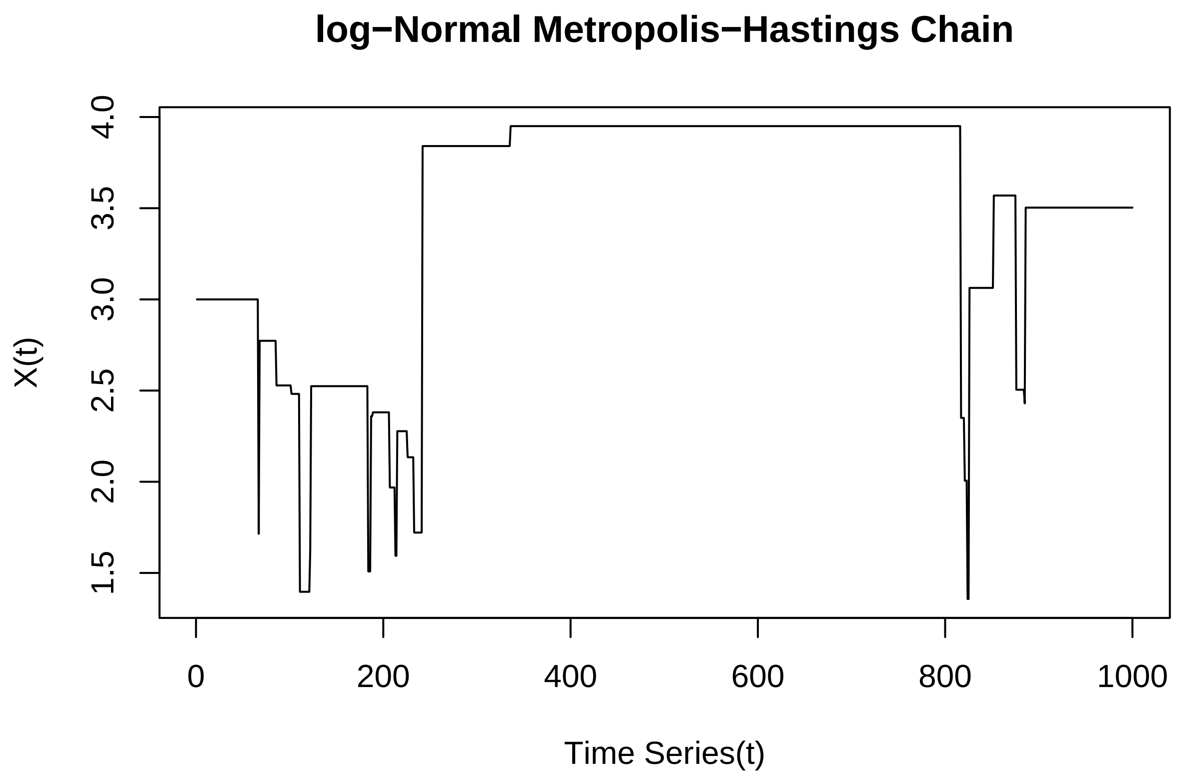
Vasileia Kampouraki, Weng Hang Wong, Jooyoung Lee

2020 2 20

Question 1: Computations with Metropolis-Hastings

1. Use Metropolis-Hastings algorithm to generate samples from this distribution by using proposal distribution as log-normal LN(Xt,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 data points will be sampled from the target distribution $\pi(x) \propto x^5 e^{-x}$, by using log-Normal $LN(X_t, 1)$ as proposal distribution.



As the figure above, the algorithm was run for 1000 times. The plot of chain obtained are represented as a time series plot.

$X_0 = 3$ was set to be the arbitrary starting point. The basic idea of the algorithm is to simulate a Markov Chain with stationary distribution, the "target"-distribution ($\pi(x)$).

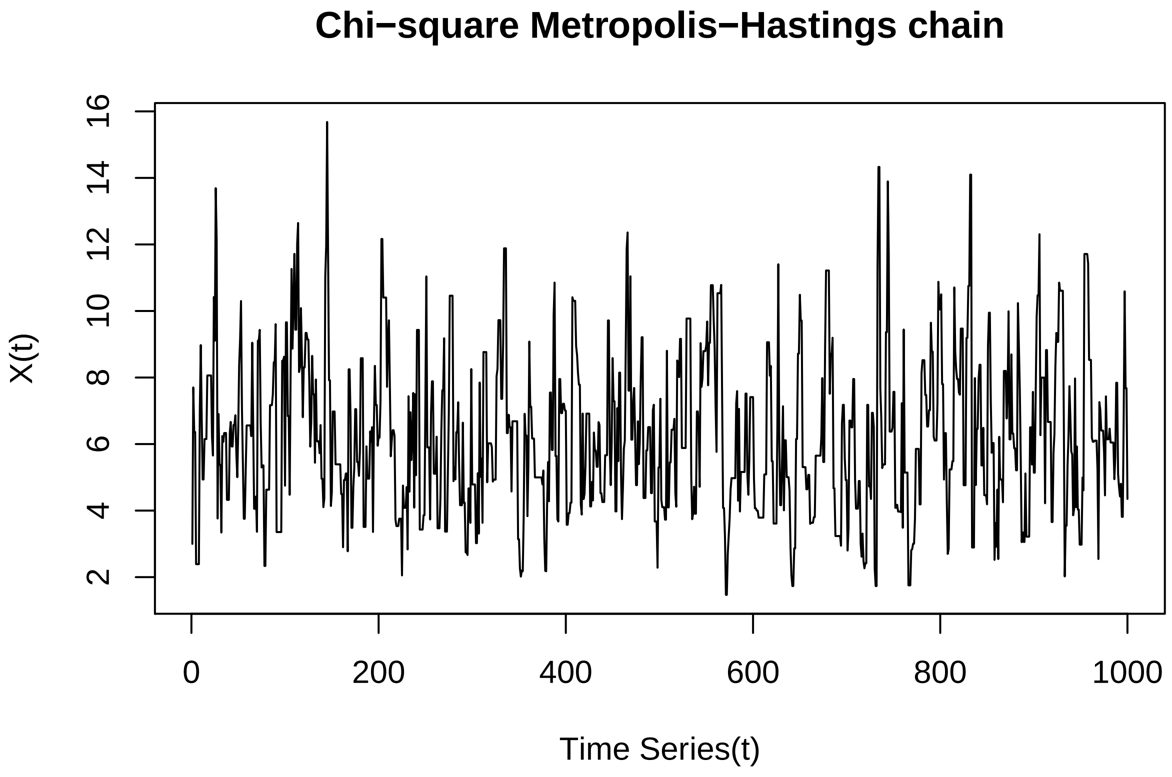
However, since X_0 is chosen arbitrarily how quick the chain is going to be stabilized is unknown. Such period that takes until the simulated data become stabilized is called burn-in period of the chain. The samples from this period can be discarded to obtain stabilized result.

In above case, looking at the time series plot of the chain it would be plausible to say that at the beginning the plot have a good image of the chain and the burn-in period seems almost insignificant. It would be possible to say that there is no need to reject part of the sample at the beginning. However, it is observed that some periods where the algorithm constantly rejects values of y and the chain remains stable at the same point, as it happens for example at around t=400 to t=800.

In the end the chain, it can be seen that the result does not converge to the target distribution ($\pi(x) \propto x^5 e^{-x}$) at t=1000

2

Repeat the Step 1 by suing the chi-square distribution $\chi^2([X_t + 1])$ as proposal distribution.



Now the proposal distribution is changed from $LN(X_t, 1)$ to $\chi^2(\lfloor X_t + 1 \rfloor)$. After plotting the chain of the target distribution it can be concluded that the result is actually very good. The burn-in period is very small, which can be considered insignificant, and the chain stabilizes quickly and stay stabilized throughout the whole process. Quite high oscillations are observed in the state space but the algorithm seems to accept generally the values of Y as we cannot observe periods where the chain remained stable at the same distribution (i.e. being flat for a while).

Finally, the chain seems to converge to the stationary distribution π.

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

From the above result, it can be said the result seems better when the chi-square $\chi^2(\lfloor X_t + 1 \rfloor$ is utilized as a proposal distribution. Since the chain is more likely to be converged and more stable, and it does not has some value stucked in the chain as the log-normal distribution one.

4. Use the Gelman–Rubin method to analyze convergence of these sequences.

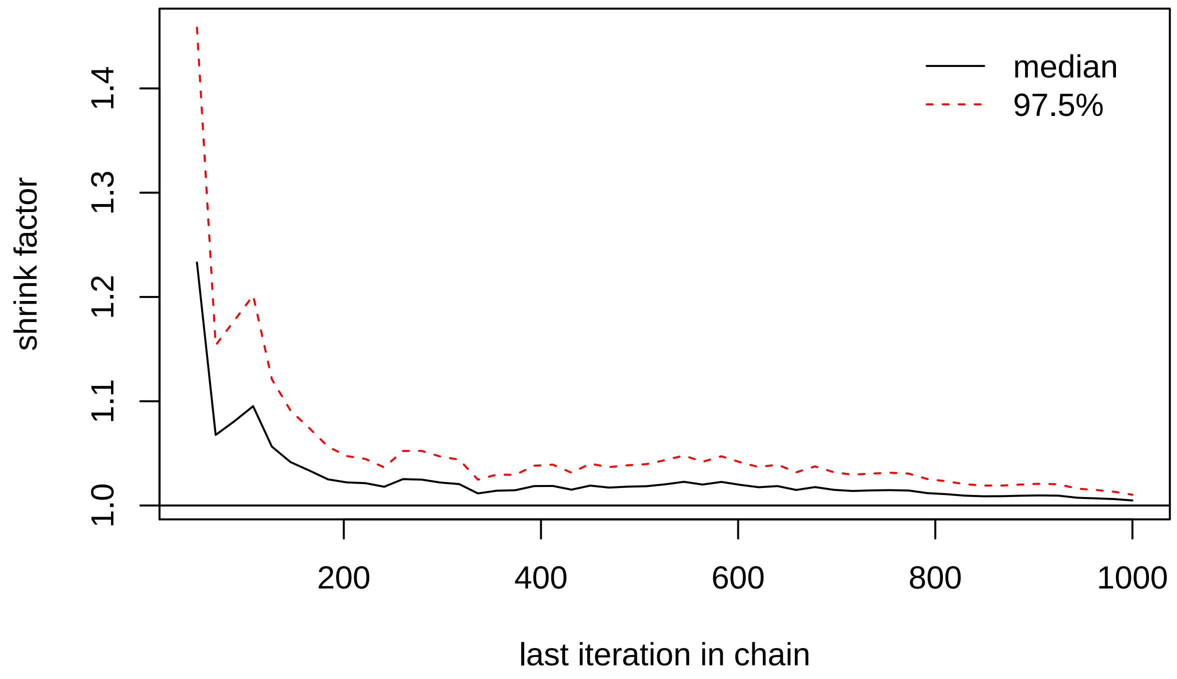
10 MCMC sequences are generated using the $\chi^2(\lfloor X_t+1 \rfloor$ as proposal distribution from above. Then Gelman-Rubin method is used to analyze the convergence of the chain.

`gelman.diag()` function compares the variability within the chains to the variability between the chains. If the Upper C.I is very close to 1 then it can be said that with 95% confidence that the chains have converged, otherwise values much larger than 1 is an indication of non-convergence.

In this case, the upper C.I has the value of 1.02 and it is very close to 1. Therefore, with using Chi-square as proposal distribution has a good convergence result that it can be concluded that the simulation reached to the stationary distribution.

```
## The Gelman-Rubin converage anaylsis:

## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1       1.01
```



5

To estimate $\int_0^\infty xf(x)dx$, the mean of the samples generated from the step1 and step2 can be used.

```
## The samples mean from Step 1 (Log-likelihood)
## [1] 3.272259
```

```
## The samples mean from Step 2 (chi-sqaure)
```

```
## [1] 5.950549
```

6.

The probability distribution function of a Gamma(a,b) is

$$f_x(x) = \frac{\beta^a}{\Gamma(a)} x^{a-1} e^{-\beta x}$$

for x>0 and $\alpha, \beta > 0$.

So in this case, the given distribution is Gamma(6,1) and thus the integral value that is required to be computed is

$$\frac{1}{\Gamma(6)} \int_0^\infty x^6 e^{-x} dx$$

.

```
## The real value of the integral is:
```

```
## 6 with absolute error < 3.9e-05
```

Compare the value obtained from the above step 1 and step 2, we can say that the sample mean we estimated from the chi-sqaure is very close to the real value from the Gamma(6,1) distribution, which is equal to 5.950549.

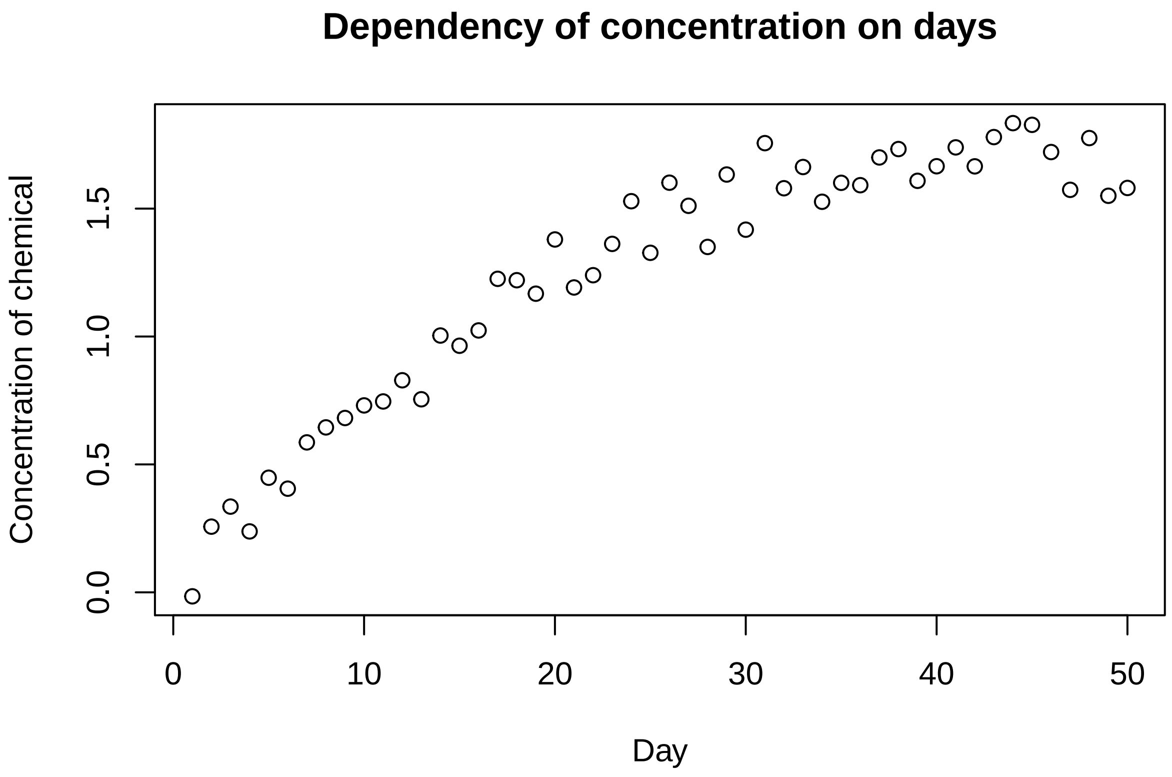
Question 2: Gibbs sampling

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 the way the data are on the plot it can be assumed that a logarithmic model could fit well those data.

2. A researcher has decided to use the following (random-walk) Bayesian model (n=number of observations,$\vec{\mu} = \mu_1, \mu_2, ..., \mu_n$ are unknown parameters):

Yi $\sim N(\mu_i, variance = 0.2)$ **i = 1,. . . ,n**

where the prior is

$$p(\mu_1) = 1$$

$$p(\mu_{i+1}|\mu_i) = N(\mu_i, 0.2)$$

Present the formulae showing the likelihood $P(\vec{Y}|\vec{\mu})$ and the prior $p(\vec{\mu})$. Hint: a chain rule can be used here $p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)...p(\mu_n|\mu_{n_1})$.

Yi $\sim N(\mu_i, variance = 0.2)$, so the likelihood will be:

Likelihood:

$$P(\vec{Y}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi 0.2}} e^{-\frac{1}{2\cdot 0.2}(y-\mu)^2} \propto e^{-\frac{1}{0.4}\sum_{i=1}^{n}(y_i-\mu_i)^2}$$

For the **prior** probability $p(\vec{\mu})$, the chain rule will be used as below.

$$p(\mu_1) = 1$$

$$p(\mu_2|\mu_1) \sim N(\mu_1, 0.2) = \frac{1}{\sqrt{2\pi 0.2}} e^{-\frac{1}{2\cdot 0.2}(\mu_2-\mu_1)^2}$$

$$p(\mu_3|\mu_2) \sim N(\mu_2, 0.2) = \frac{1}{\sqrt{2\pi 0.2}} e^{-\frac{1}{2\cdot 0.2}(\mu_2-\mu_3)^2} \vdots$$

$$p(\mu_n|\mu_{n_1}) \sim N(\mu_{n_1}, 0.2) = \frac{1}{\sqrt{2\pi 0.2}} e^{-\frac{1}{2\cdot 0.2}(\mu_{n_1}-\mu_n)^2}$$

Thus, using the chain rule and combining all those equations, the prior is:

$$p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)...p(\mu_n|\mu_{n_1}) \propto e^{-\frac{\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2}{0.4}}$$

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

Based on the Bayes' Theorem **Posterior probability $\propto$ Likelihood x Prior probability**

From the previous question the Likelihood $P(\vec{Y}|\vec{\mu})$ and the prior probability p($\vec{\mu}$) are already computed.

The given task is to find the distributions of $(\mu_i|\vec{\mu_{-i}}, \vec{Y})$ where $\vec{\mu_{-i}}$ is a vector containing all μ values except of μ_i.

Therefore, to find the distributions, the posterior probabilities have to be found first. Then such formulae will be investated to see from which distribution they come from.

Posterior probability

$$P(\mu_i|\vec{\mu_{-i}}, \vec{Y}) \propto P(\vec{Y}|\vec{\mu})P(\vec{\mu})$$

For μ_1**:**

$$P(\mu_1|\vec{\mu_{-1}}, \vec{Y}) \propto e^{-\frac{1}{2\cdot 0.2}\{(y_1-\mu_1)^2+(\mu_2-\mu_1)^2\}} = e^{-\frac{1}{\frac{0.4}{2}}\{\mu_1-\frac{y_1+\mu_2}{2}\}^2}$$

Thus,

$$\mu_1|\vec{\mu_{-1}} \sim N(\frac{y_1+\mu_2}{2}, 0.1)$$

For μ_i ,i=2,. . . ,n-1:

$$P(\mu_i|\vec{\mu_{-i}}, \vec{Y}) \propto e^{-\frac{1}{2\cdot 0.2}\{(y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2\}} = e^{-\frac{1}{\frac{0.4}{3}}\{\mu_i-\frac{y_i+\mu_{i-1}+\mu_{i+1}}{3}\}^2}$$

Thus,

$$\mu_i|\vec{\mu_{-i}} \sim N(\frac{y_i + \mu_{i-1} + \mu_{i+1}}{3}, 0.06)$$

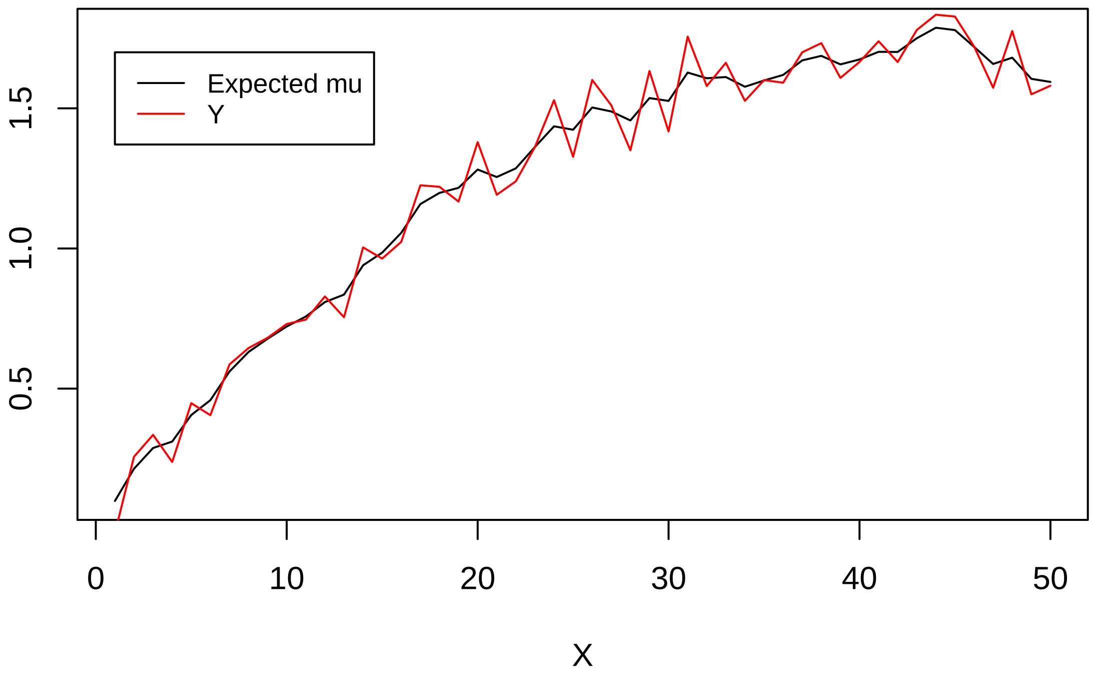
For μ_n:

$$P(\mu_n|\vec{\mu_{-n}}, \vec{Y}) \propto e^{-\frac{1}{2\cdot 0.2}\{(y_n-\mu_n)^2+(\mu_n-\mu_{n-1})^2\}} = e^{-\frac{1}{\frac{0.4}{2}}\{\mu_n - \frac{y_n+\mu_{n-1}}{2}\}^2}$$

Thus,

$$\mu_n|\vec{\mu_{-n}} \sim N(\frac{y_n + \mu_{n-1}}{2}, 0.1)$$

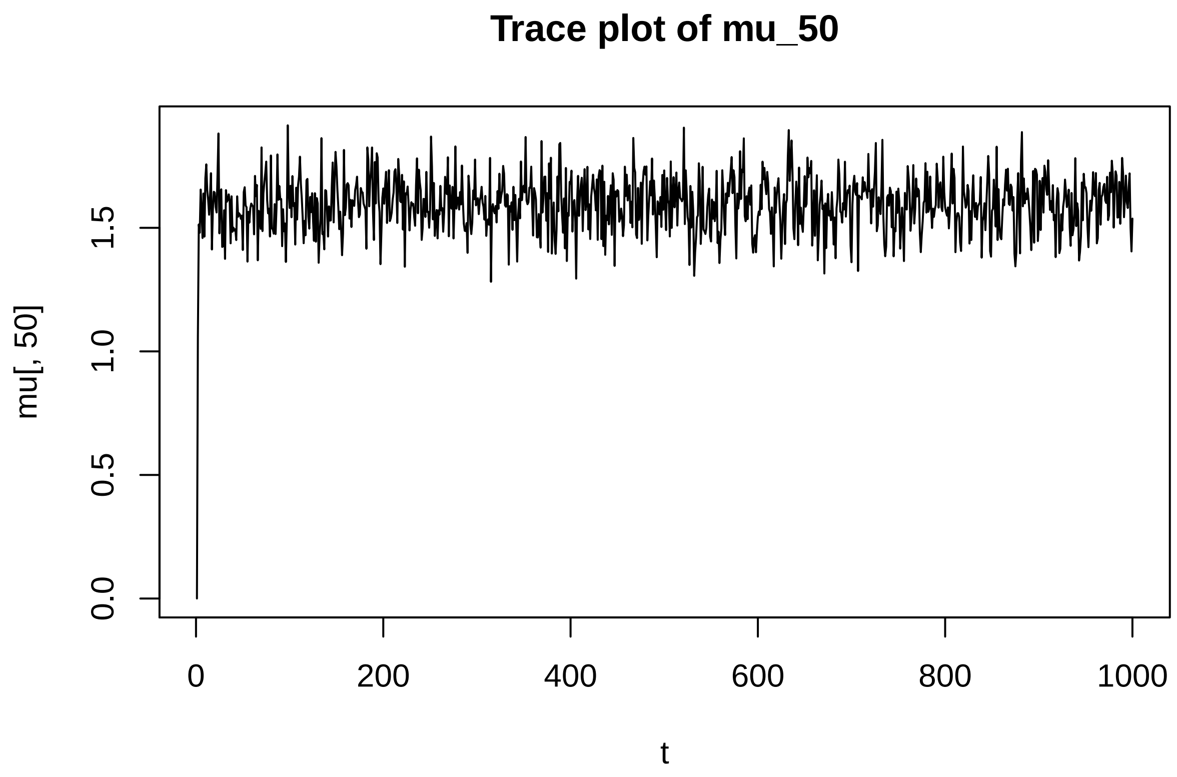
4. Use the distributions derived in Step 3 to implement a Gibbs sampler that uses $\vec{\mu}^0 =$ (0,...,0) as a starting point. Run the Gibbs sampler to obtain 1000 values of $\vec{\mu}$ and then compute the expected value of $\vec{\mu}$ by using a Monte Carlo approach. Plot the expected value of $\vec{\mu}$ 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?



Observing the plot, it can be said the noise has been removed as the black line is quite smooth, at least compare to the red line which represents Y.

Also, from the form of the line of expected μ it can be said that the expected value of μ can catch the true underlying dependence between Y and X as it follows the red line well.

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



It is possible to observe that the burn-in beriod of μ_{50} is very small, basically it takes around 10 or less iterations until it is stabilized. Therefore it can be concluded that this simulation has a small burn-in period.

Finally, it seems that that the chain has converged to the true posterior.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
#question1
set.seed(12345)
MH_normal <- function(tmax, X0){
  t=1
  X=rep(X0,tmax)
  while(t < tmax){
    Y <- rlnorm(1, X[t], 1)
    U <- runif(1,0,1)

    alpha_fun <- function(X,Y){
      pi_Y = (Y^5)*exp(-Y)
      pi_Xt = (X^5)*exp(-X)
      q_Xt = dlnorm(X, Y,1)
      q_Y = dlnorm(Y,X,1)
      out = min(1,pi_Y*q_Xt/(pi_Xt*q_Y))
      return(out)
    }
      if(U <= alpha_fun(X[t],Y)){
        X[t+1] = Y
      }else{
        X[t+1] = X[t]
        }
    t=t+1
  }
  return(X)
}
tmax=1000
X0=3
plot(1:tmax, MH_normal(tmax,X0),"l",xlab="Time Series(t)", ylab="X(t)", main="log-Normal Metropolis-Has
# 2
MH_chi <- function(tmax, X0){
  t=1
  X=rep(X0,tmax)
  while(t < tmax){
    Y <- rchisq(1, floor(X[t]+1))
    U <- runif(1,0,1)

    alpha_fun <- function(X,Y){
      pi_Y = (Y^5)*exp(-Y)
      pi_Xt = (X^5)*exp(-X)
      q_Xt = dchisq(X, floor(Y+1))
      q_Y = dchisq(Y,floor(X+1))
      out = min(1,pi_Y*q_Xt/(pi_Xt*q_Y))
      return(out)
    }

    if(U <= alpha_fun(X[t],Y)){
      X[t+1] = Y
    }else{
      X[t+1] = X[t]
    }
    t=t+1
  }
  return(X)
}
tmax=1000
X0 = 3
plot(1:tmax, MH_chi(tmax,X0),"l",  xlab="Time Series(t)", ylab="X(t)", main="Chi-square Metropolis-Hast
library(coda)
f = mcmc.list()
for(X0 in 1:10){
  f[[X0]] = as.mcmc(MH_chi(tmax,X0))
}
cat("The Gelman-Rubin converage anaylsis:\n")
print(gelman.diag(f))
gelman.plot(f)
cat("The samples mean from Step 1 (Log-likelihood)\n")
```

```r
mean(MH_normal(1000,1))
cat("The samples mean from Step 2 (chi-sqaure)\n")
mean(MH_chi(1000,1))
f<-function(x){ 1/gamma(6)*(x^6)*exp(-x)}
gammadist <- integrate(f ,0, Inf )
cat("The real value of the integral is:","\n")
gammadist
data <- load("~/Desktop/732A90_VT2020_Materials/chemical.RData")
df<- data.frame("day"=X,"concentration"=Y)
plot(df$day,df$concentration,xlab="Day",ylab="Concentration of chemical",main="Dependency of concentrati
#http://www.mit.edu/~ilkery/papers/GibbsSampling.pdf
#we have 50 observations and want to simulate 1000 values of mu
mu=matrix(0,nrow=1000,ncol=50)
for (i in 2:1000){ #first row will remain zero, the initialized
  for (j in 2:49){
    mu[i,1]<- rnorm(1,mean=(Y[1]+mu[i-1,2])/2,sd=0.1)  #mu_1

    mu[i,j] <- rnorm(1,mean=(Y[j] + mu[i,j-1] + mu[i-1,j+1])/3,sd=0.06)  #mu_ij

    mu[i,50] <- rnorm(1,mean=(mu[i,49]+Y[50])/2,sd=0.1)  #mu_50

  }
}
expected_mu <- colMeans(mu)
plot(X,expected_mu,type="l",ylab="")
lines(X,Y,col="red")
legend(1,1.7, legend=c("Expected mu", "Y"),
       col=c("black", "red"), lty=1, cex=0.8)
#trace plot for mu_50
plot(mu[,50],type = "l",main="Trace plot of mu_50",xlab="t")
```