Computational Statistics Lab4 Group 29

Jin Yan (jinya425), Yaning Wang (yanwa579)

Question 1: Computations with Metropolis–Hastings

TASK1

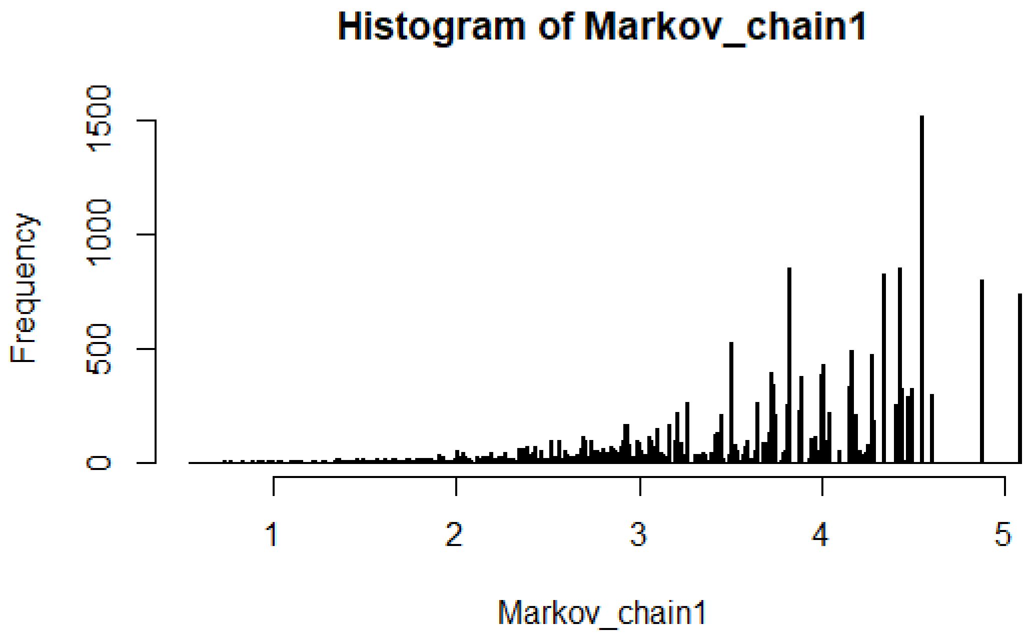
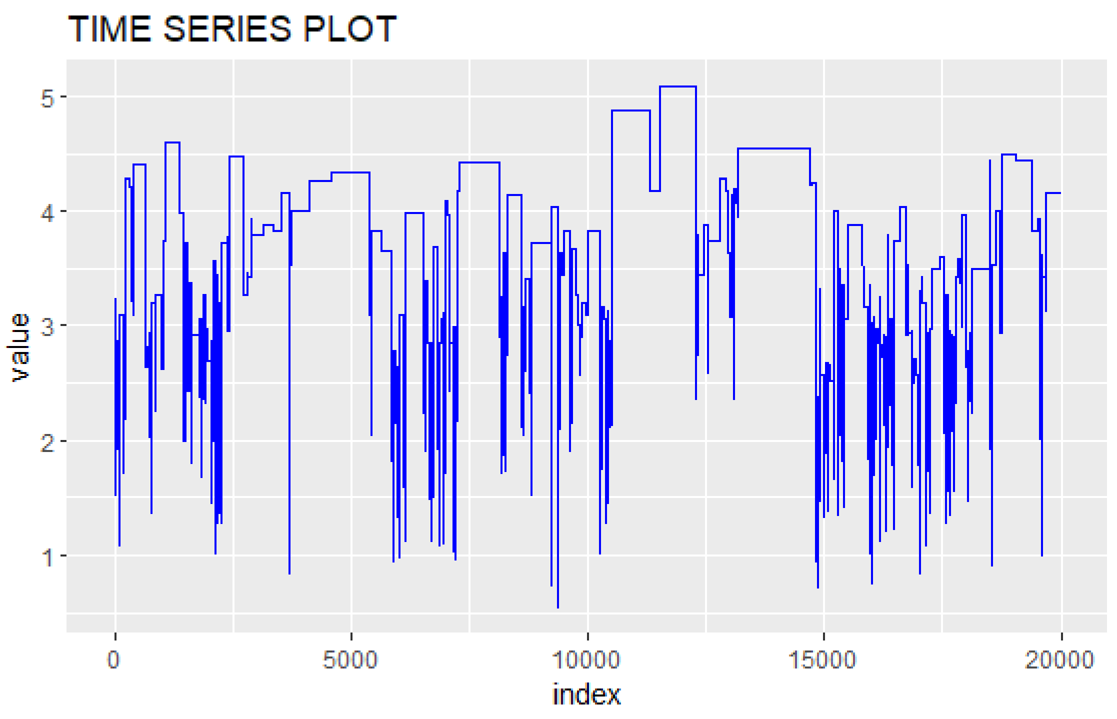


Figure 1: histogram of chain one



*

According to the histogram and plot the Markov chain has not converged so far, it will eventually converge but need a long time I think.

TASK2

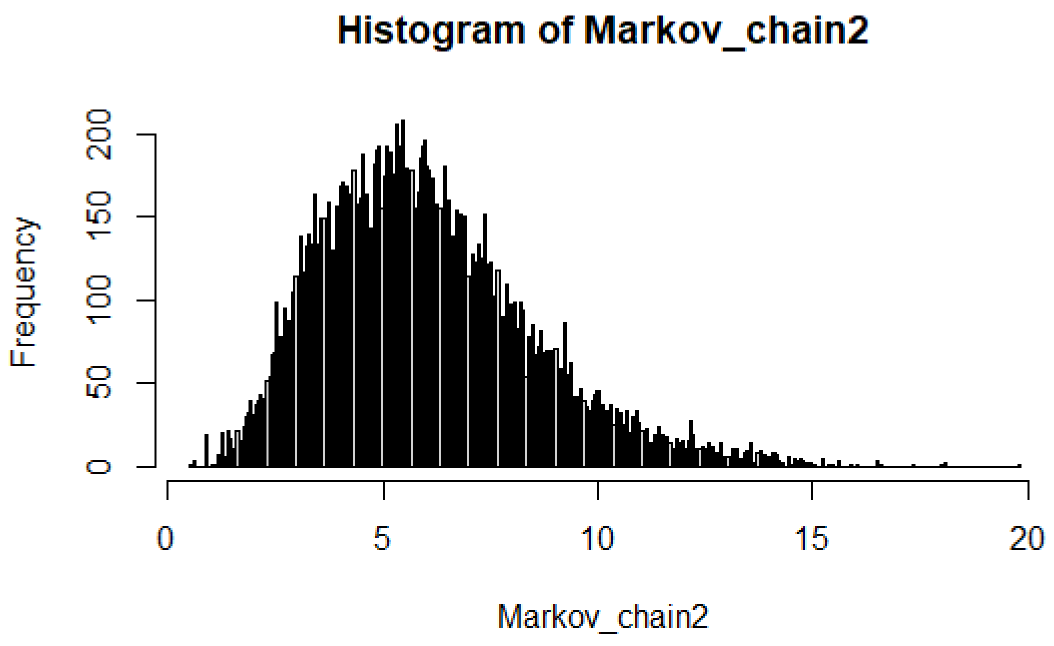
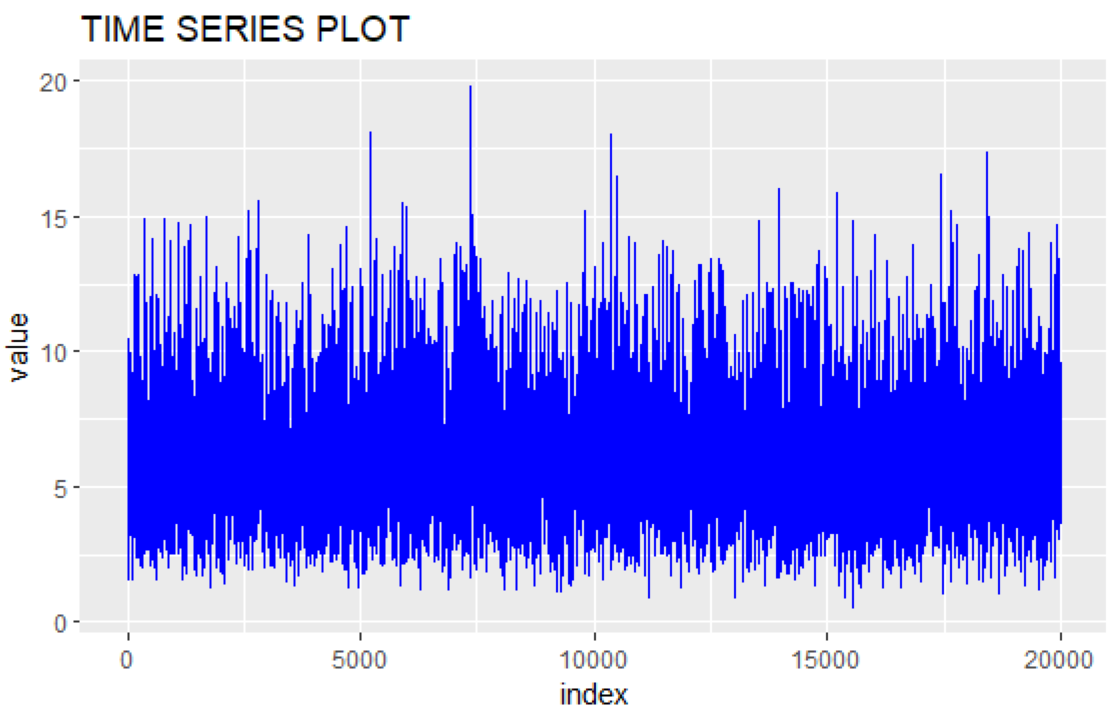


Figure 2: histogram of chain two



*

TASK3

Compared to situation in the first question, the Markov chain in the second question converge more easily.

TASK4

Since the upper C.I. is 1.01 and is close to 1, so these sequences converged.

TASK5

According to our formula, if we want to calculate the integral of this, we can just calculate the mean of values of Markov chain. So far, we have already gotten these chains. So the only step left is to calculate the mean values.

After calculation we find the result_1 is around 3.99. The result_2 is about 6.01.

TASK6

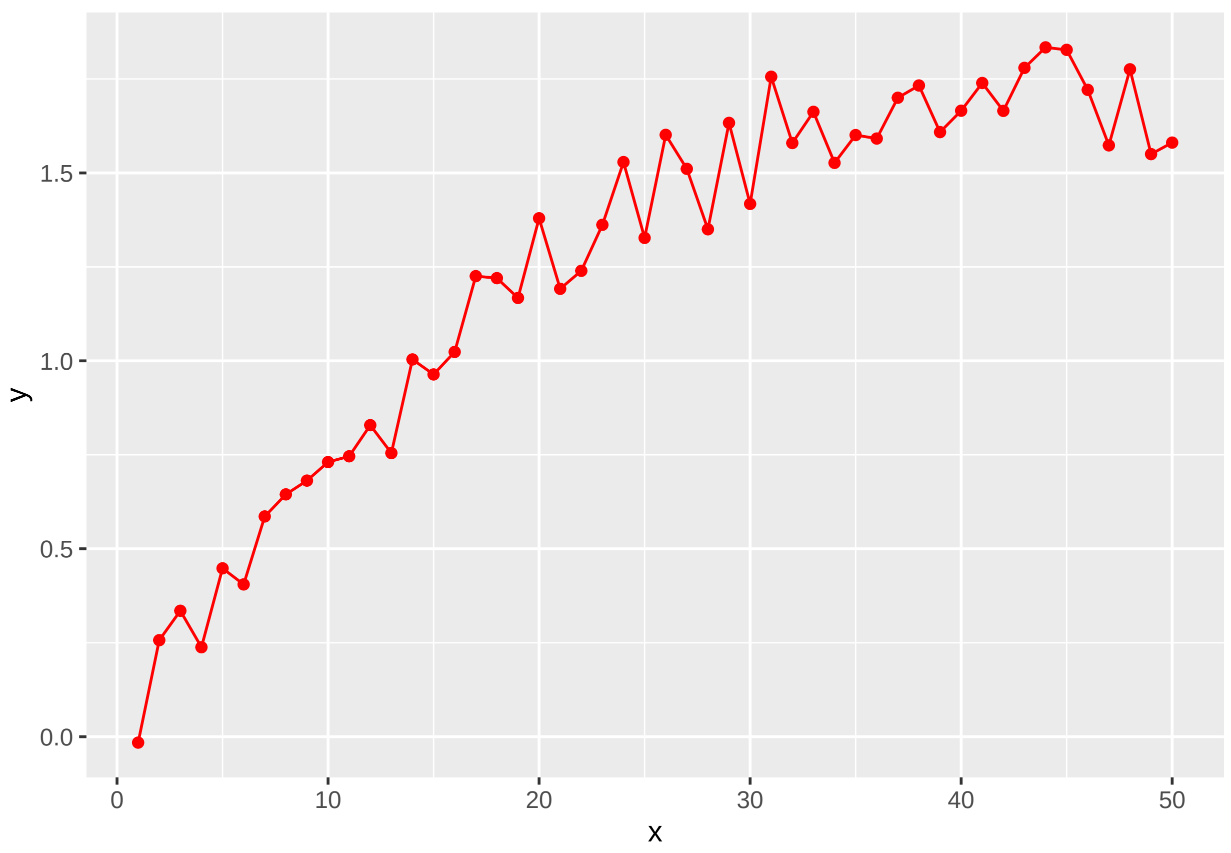
According to the literature, I found that the alpha is 6, the beta is 1. The mean of the distribution 6. Combined the above results, we will find MCMC is indeed an effective way to sample from an unknown distribution.

Question 2: Gibbs sampling

task1

Import the data

```
library(ggplot2)
load("chemical.RData")
df<-data.frame(x=X,y=Y)
ggplot(df,aes(x=x,y=y))+
  geom_point(colour="red")+
  geom_line(colour="red")
```



Q: What kind of model is reasonable to use here?

It seems a linear regression model is reasonable to use.

task2

Show the likelihood and prior

We know the normal density function $X \sim N(\mu, \sigma^2)$is:

$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$

From this formula, we get the density function of $Yi \sim N(\mu_i, \sigma^2 = 0.2)$

$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}(\frac{y_i-\mu_i}{\sigma})^2}$

The likelihood $p(\vec{Y}|\vec{\mu})$ is:

$\prod_{i=1}^{n} p(\vec{Y_i}|\vec{\mu_i})$

$=\frac{1}{(\sigma\sqrt{2\pi})^n}e^{-\sum_{i=1}^{n}\frac{(yi-\mu_i)^2}{2\sigma^2}}$

The prior $p(\vec{\mu})$ is :

$p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)...p(\mu_n|\mu_{n-1})$ $=p(\mu_1)\prod_{i=2}^{n} p(\mu_i|\mu_{i-1})$ $=\frac{1}{(\sigma\sqrt{2\pi})^{n-1}}e^{-\frac{1}{2}\sum_{i=2}^{n}\frac{(\mu_i-\mu_{i-1})^2}{\sigma^2}}$

task3

Get the posterior and find out the distributions

Use Bayes' Theorem to get the posterior:

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

We put the $p(\vec{Y}|\vec{\mu})$ and $p(\vec{\mu})$ in the formula:

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

For i is equal to 1, we can get:

$p(\mu_1|\vec{\mu_{-1}},\vec{Y}) = e^{-\frac{(y_1-\mu_1)^2-(\mu_2-\mu_1)^2}{2\sigma^2}}$

Based on hint B, we can get:

$p(\mu_1|\vec{\mu_{-1}},\vec{Y}) = e^{-\frac{(\mu_1-(\mu_2+y_1)/2)^2}{2\sigma^2/2}}$

For i is equal to n, we use the same way and get: $p(\mu_n|\vec{\mu_{-n}},\vec{Y})$:

$p(\mu_n|\vec{\mu_{-n}},\vec{Y}) = e^{-\frac{(\mu_n-(\mu_{n-1}+y_n)/2)^2}{2\sigma^2/2}}$

For i between 1 and n(except 1 and n), we can follow the same approach as above and based on hint C:

$p(\mu_i|\vec{\mu_{-i}},\vec{Y}) = e^{-\frac{(\mu_i-(\mu_{i-1}+\mu_{i+1}+y_i)/3)^2}{2\sigma^2/3}}$

From the three formulas above,we can write them in the form of a normal distribution:

i=1,the distribution is N($\frac{\mu_2+y_1}{2}$,0.1)

i∈(1,n),the distribution is N($\frac{\mu_{i-1}+\mu_{i+1}+y_i}{3}$,$\frac{0.2}{3}$)

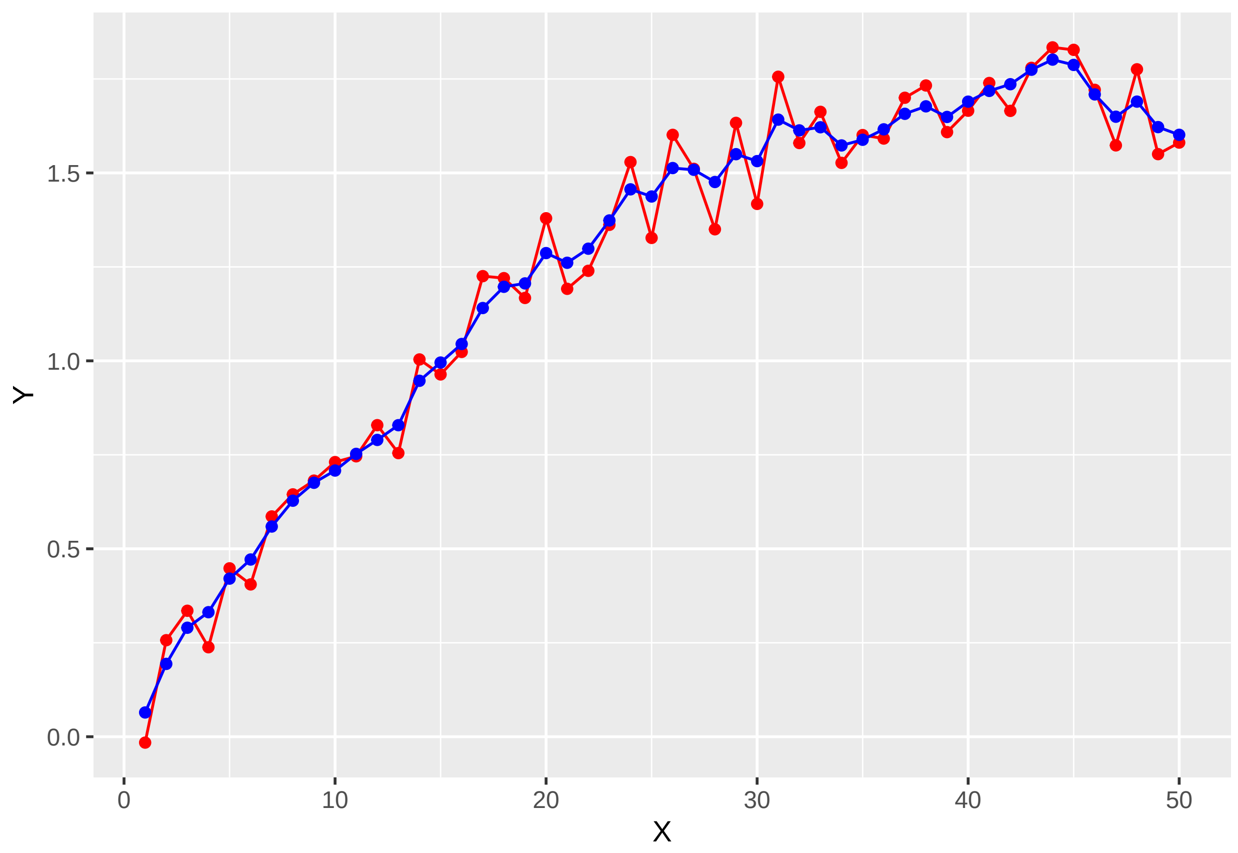
i=n, the distribution is N($\frac{\mu_{n-1}+y_n}{2}$,0.1)

task4

Implement a Gibbs sampler and plot a graph

```r
nstep<-1000
 d<-length(Y)
 mu<-matrix(0,nrow=nstep,ncol=d)
 for (i in 2:nstep){
   mu[i,1]<-rnorm(1,mean=(mu[i-1,2]+Y[1])/2,sd=sqrt(0.1))
   for (j in 2:(d-1)){
     mu[i,j]<-rnorm(1,mean=(mu[i,j-1]+mu[i-1,j+1]+Y[j])/3,sd=sqrt(0.2/3))
   }
   mu[i,d]<-rnorm(1,mean=(mu[i,d-1]+Y[d])/2,sd=sqrt(0.1))
 }
 expect_mu<-colMeans(mu)

 df_new<-data.frame(X,Y,expect_mu)
 ggplot(df)+
   geom_point(aes(x=X,y=Y),colour="red")+
   geom_line(aes(x=X,y=Y),colour="red")+
   geom_point(aes(x=X,y=expect_mu),colour="blue")+
   geom_line(aes(x=X,y=expect_mu),colour="blue")
```



Q: 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?

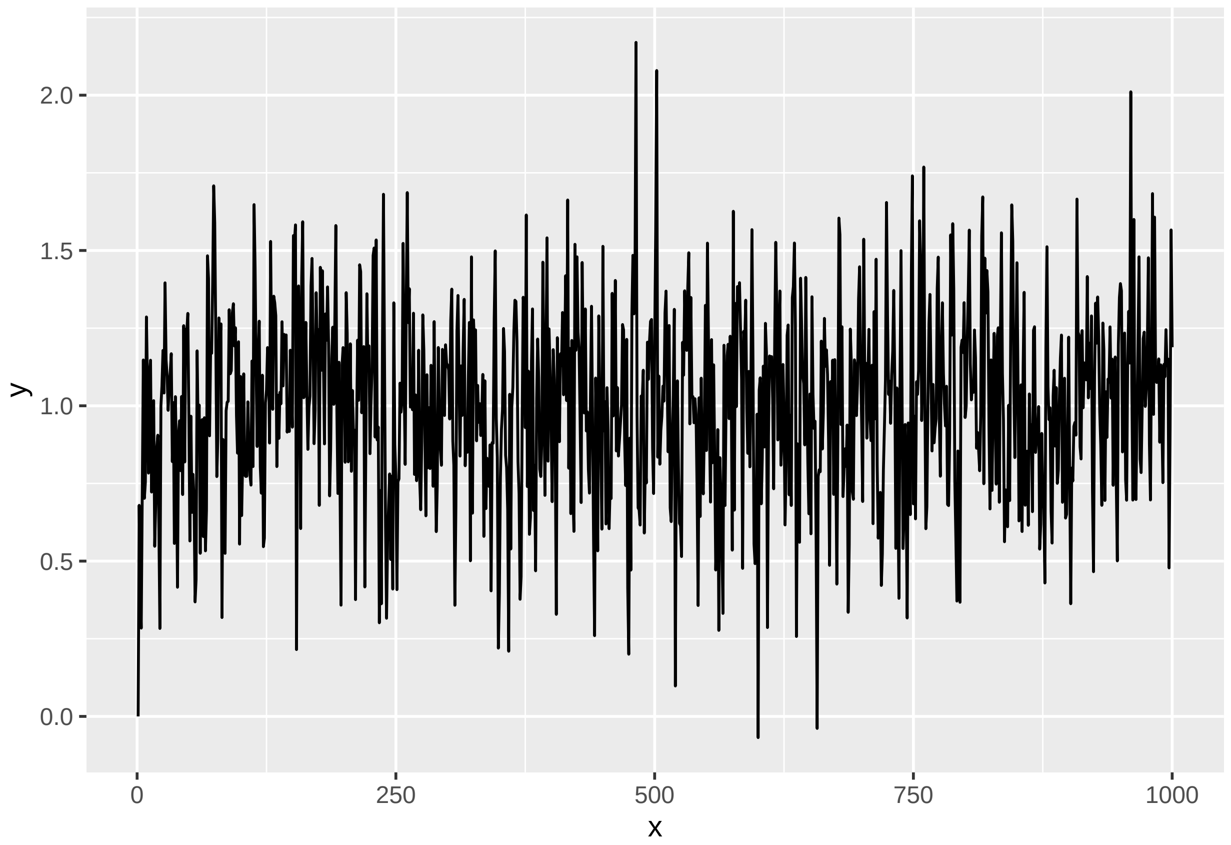
The blue line looks smoother and has the same trend as the red line. So we have removed the noise and $\vec{\mu}$ can catch the true underlying dependence between Y and X.

task4

Make a trace plot

We use μ_i to plot the graph.

```
df<-data.frame(x=c(1:nstep),y=mu[,15])
  ggplot(df,aes(x=x,y=y))+
    geom_line()
```



Q: Comment on the burn-in period and convergence.

From the graph, we can see it convergences. And the burn-in period is short.It quickly reached convergence.

Appendix

Question 1: Computations with Metropolis–Hastings

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

MCMC <- function(num,X0){
  X <- c()
  # determine the value of X1
  Xnew <- rlnorm(1, meanlog = X0)
  alpha <- min(1,(fx(Xnew) * dlnorm(X0, meanlog = Xnew) / (fx(X0) * dlnorm(Xnew,
                                                              meanlog = X0))))
  u <- runif(1)
  if(u >= alpha){X[1] <- X0}
  else{X[1] <- Xnew}

  # determine the values of following samples
  for(i in 1:(num-1)){
    Xnew <- rlnorm(1, meanlog = X[i])
    alpha <- min(1,(fx(Xnew) * dlnorm(X[i], meanlog = Xnew) / (fx(X[i]) *
                                              dlnorm(Xnew, meanlog = X[i]))))
```

```r
    u <- runif(1)
    if(u >= alpha){X[i + 1] <- X[i]}
    else{X[i + 1] <- Xnew}
  }
  return(X)
}

# plot the Markov chain


Markov_chain1 <- MCMC(20000,1.5)
hist(Markov_chain1,breaks = 400)

data_1 <- data.frame(index = 1:20000, value = Markov_chain1)
ggplot2::ggplot(data = data_1, ggplot2::aes(x = index, y = value)) +
  ggplot2::geom_line(color = "blue") + ggplot2::ggtitle("TIME SERIES PLOT")


# According to the histogram and plot the Markov chain has not converged so far,
# it will eventually converge but need a long time I think.


################################################################################



MCMC_2 <- function(num,X0){
  X <- c()
  # determine the value of X1
  Xnew  <- rchisq(1, df = floor(X0 + 1))
  alpha <- min(1,(fx(Xnew) * dchisq(X0, df = floor(Xnew + 1)) / (fx(X0) *
                                            dchisq(Xnew, df = floor(X0 + 1)))))
  u <- runif(1)
  if(u >= alpha){X[1] <- X0}
  else{X[1] <- Xnew}

  # determine the values of following samples
  for(i in 1:(num-1)){
    Xnew <-rchisq(1,  df = floor(X[i] + 1))
    alpha <- min(1,(fx(Xnew) * dchisq(X[i], df = floor(Xnew + 1)) / (fx(X[i]) *
                                          dchisq(Xnew, df = floor(X[i] + 1)))))
    u <- runif(1)
    if(u >= alpha){X[i + 1] <- X[i]}
    else{X[i + 1] <- Xnew}
  }
  return(X)
}

# plot the Markov chain


Markov_chain2 <- MCMC_2(20000,1.5)
hist(Markov_chain2,breaks = 400)
```

```r
data_1 <- data.frame(index = 1:20000, value = Markov_chain2)
ggplot2::ggplot(data = data_1, ggplot2::aes(x = index, y = value)) +
  ggplot2::geom_line(color = "blue") + ggplot2::ggtitle("TIME SERIES PLOT")
# Compared to situation in the first question, the Markov chain converge more
# easily.



###############################################################################
###############################################################################

# Compared to situation in the first question, the Markov chain converge more
# easily.
###############################################################################
###############################################################################
###############################################################################






library(coda)
MCMC_list <- list()

for (i in 1:10 ) {
  chain <- MCMC_2(2000,i)
  MCMC_list[[i]] <- as.mcmc(chain)

}

gelman.diag(MCMC_list)



# Since the upper C.I. is 1.01 and is close to 1, so these sequences converged.

###############################################################################
###############################################################################
###############################################################################
###############################################################################

# According to our formula, if we want to calculate the integral of this, we can
# just calculate the mean of values of Markov chain. So far, we have already
# gotten these chains. So the only step left is to calculate the mean values.



result_1 <- mean(Markov_chain1)
result_2 <- mean(Markov_chain2)

# After calculation we find the result_1 is around 3.99. The result_2 is about
# 6.01.
```

```
################################################################################
################################################################################
################################################################################
################################################################################
################################################################################


# According to the literature, I found that the alpha is 6, the beta is 1. The
# mean of the distribution 6. Combined the above results, we will find MCMC is
# indeed an effective way to sample from an unknown distribution.
```

Question 2: Gibbs sampling

```
#task1
library(ggplot2)
load("chemical.RData")
df<-data.frame(x=X,y=Y)
ggplot(df,aes(x=x,y=y))+
  geom_point(colour="red")+
  geom_line(colour="red")

#task2
#Please see text above

#task3
#Please see text above

#task4
nstep<-1000
  d<-length(Y)
  mu<-matrix(0,nrow=nstep,ncol=d)
  for (i in 2:nstep){
    mu[i,1]<-rnorm(1,mean=(mu[i-1,2]+Y[1])/2,sd=sqrt(0.1))
    for (j in 2:(d-1)){
      mu[i,j]<-rnorm(1,mean=(mu[i,j-1]+mu[i-1,j+1]+Y[j])/3,sd=sqrt(0.2/3))
    }
    mu[i,d]<-rnorm(1,mean=(mu[i,d-1]+Y[d])/2,sd=sqrt(0.1))
  }
  expect_mu<-colMeans(mu)

  df_new<-data.frame(X,Y,expect_mu)
  ggplot(df)+
    geom_point(aes(x=X,y=Y),colour="red")+
    geom_line(aes(x=X,y=Y),colour="red")+
    geom_point(aes(x=X,y=expect_mu),colour="blue")+
    geom_line(aes(x=X,y=expect_mu),colour="blue")

#task5
  df<-data.frame(x=c(1:nstep),y=mu[,15])
  ggplot(df,aes(x=x,y=y))+
    geom_line()
```