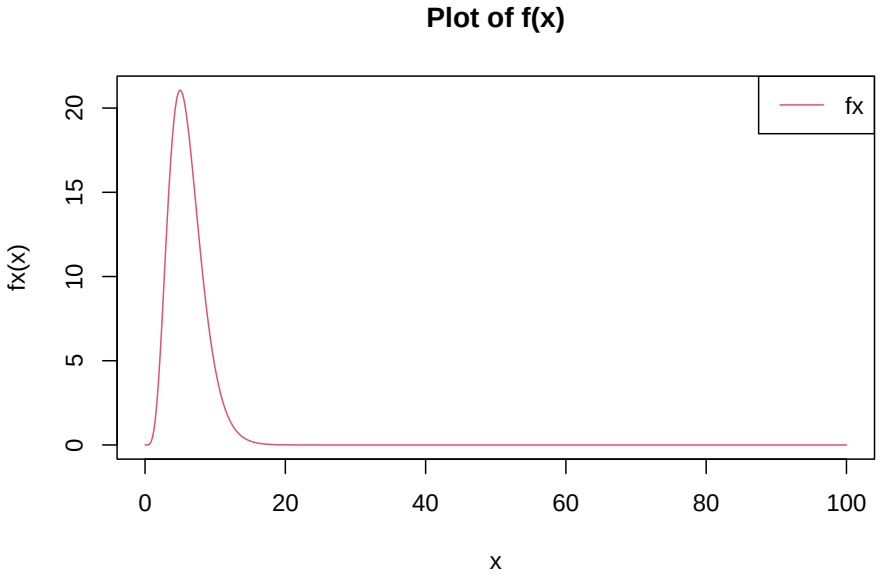
Computational Statistics Lab 4 Report

Jonathan Dorairaj,YiHung Chen

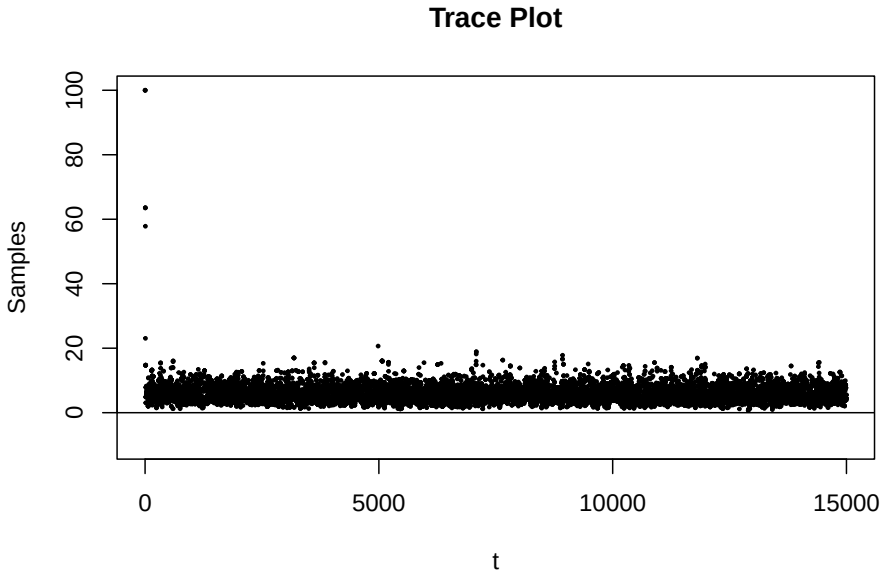
2022-12-05

Task 1

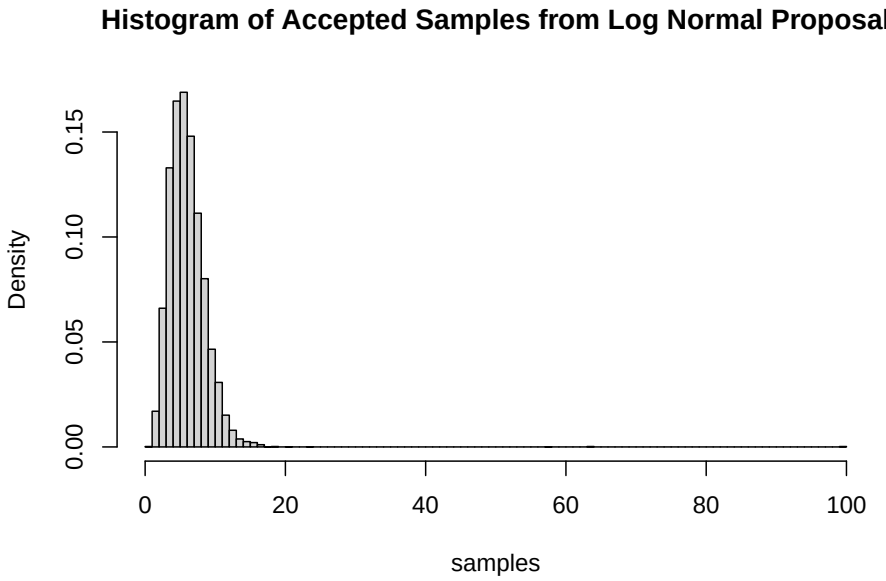
Q: Use Metropolis–Hastings algorithm to generate samples from this distribution by using proposal distribution as log–normal LN(Xt,1), take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn–in period, what can be the size of this period?



We plot the function `fx()` to get an initial look at the distribution. Using the Metropolis-Hastings algorithm shown below with proposal as the log-normal function LN(Xt,1).


Trace Plot
Samples
0
20
40
60
80
100
0
5000
10000
15000
t

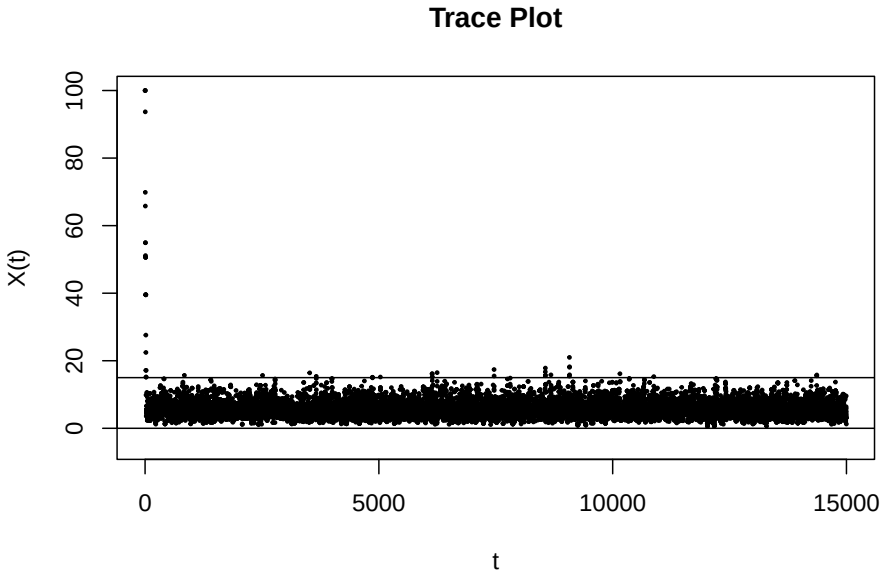

Trace plot for the first 200 iterations
Samples
0
20
40
60
80
100
0
50
100
150
200
t



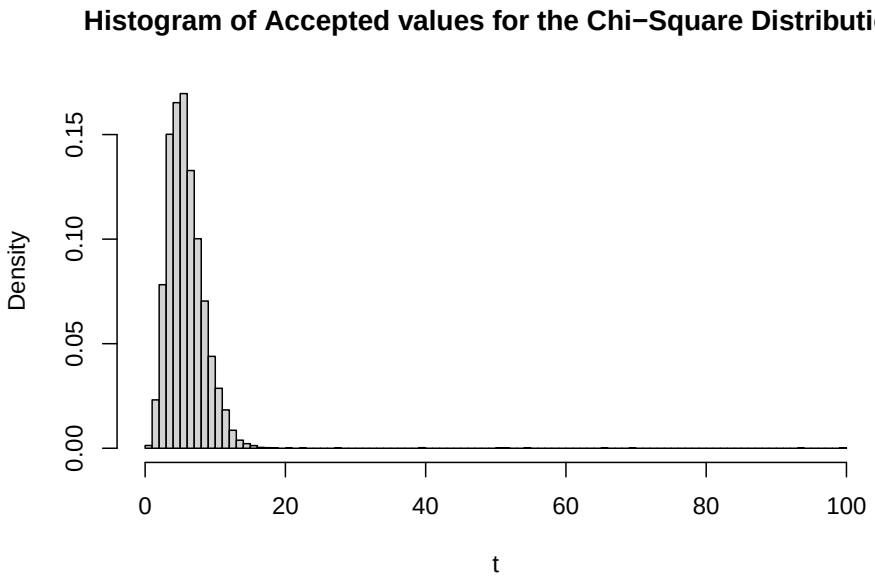
Looking at the trace plot for the first 200 iterations, we see that the burn-in period is very short, around a 100 iterations. For a starting point of `100` and `nstep=15000`, we observe from the plot that chain has a very small burn-in period. We see the chain converge almost immediately as it continues to samples values from 0 to 20 for the remainder of the iterations. Comparing the histogram of accepted values to the plot of `fx` in Figure 1, we conclude that the sampler does a good job of accepting values for the region of `fx`.

Q: Perform Step 1 by using the chi–square distribution X2(Xt + 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

```
t <- MetHas_chi(15000,100,flag = T)
```



```
hist(t,breaks = 100,freq = FALSE, main = "Histogram of Accepted values for the Chi-Square Distribution")
```



1.3 : Compare the results of Steps 1 and 2 and make conclusions.

The samples start from the specified starting points and after a short burn in period, converge quickly and have the majority of sample values between 0 and 20. Comparing the histogram of the samples to the plot of the function `fx`, we conclude that sampling from the chi-square distribution is similarly good as the samples generally cover the region that `fx` encapsulates.

1.4 : 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.

We create a `mcmc.list` of markov chains with different starting values as specified in the loop.

```
seq_list <-  mcmc.list()
GR_has <- function(k,nstep,flag = FALSE){

  for (i in 1:k) {
    y <- MetHas_chi(nstep = nstep,i,flag)
    seq_list[[i]] <- as.mcmc(y)
     }

print(gelman.diag(seq_list))

}
```

```
GR_has(10,15000,FALSE)
```

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

The Gelman - Rubin factor is 1 for both chains, indicating that the chains converge.

1.5

Estimate

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

using the samples from Steps 1 and 2.

$$E[x] = \int_0^\infty xf(x)dx$$

Given that the samples we obtain from the Metropolis Hastings algorithms are accepted values for the PDF of the target function,$x \approx p(x)$, where $\int_0^\infty p(x)dx = 1$. Therefore,we get

$$E[x] = \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \mu$$

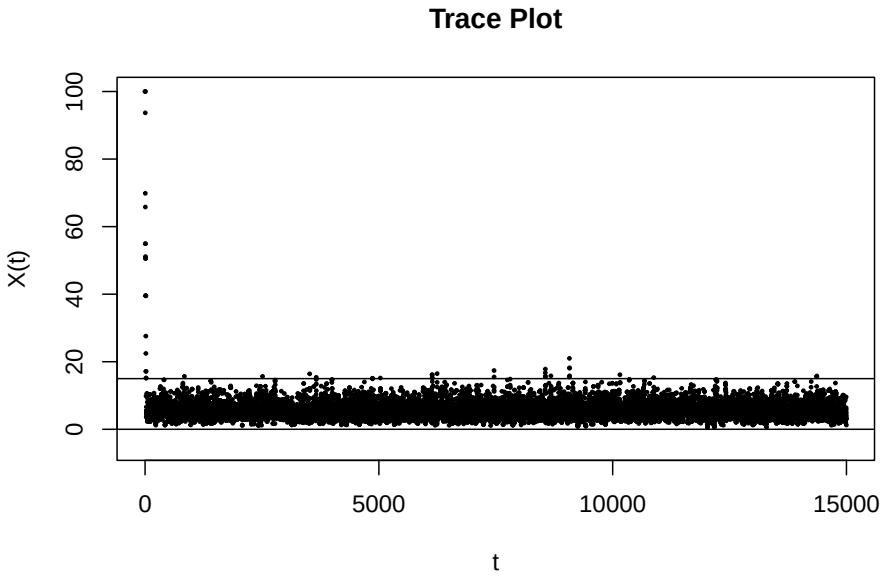
,where μ is the mean.

Therefore, we calculate the mean of the samples from **Q1.1** (log-normal proposal) and **Q1.2** (chi-square proposal).

```
samples1 <- MetHas(15000,1)
paste("The mean of the accepted values are:",mean(samples1))
```

```
## [1] "The mean of the accepted values are: 5.97913290357686"
```

```
t <- MetHas_chi(15000,100,flag = T)
```




```
paste("The mean of the accepted values are:",mean(t))
```

```
## [1] "The mean of the accepted values are: 5.87904077373727"
```

1.6

Q: 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 PDF for the gamma distribution:

$$\int \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{\frac{-x}{\beta}} dx$$

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

The PDF for our target distribution is:

$$\int \frac{1}{120} x^{5} e^{-x} dx$$

Comparing the two PDFs, we arrive at $\alpha = 6$ and $\beta = 1$. The value of the PDF is given by expected value which is the mean.

The mean of the gamma distribution if given by : α . β = 6*1 = 6.

Table 1: Comparision of values

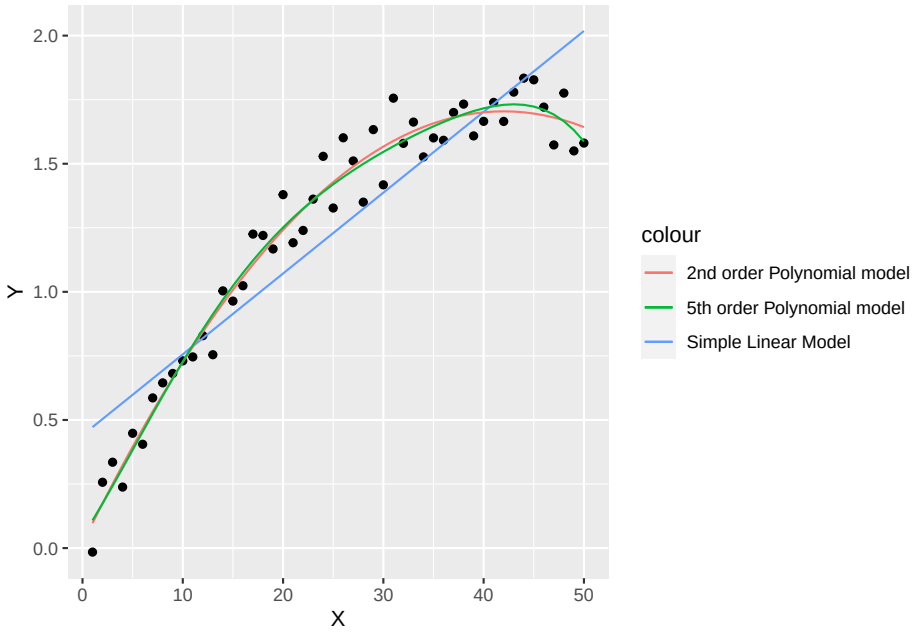
	Actual	LogNormal	ChiSquare
Mean	6	5.979133	5.879041

Comparing it with the values obtained from the log normal and chi-square distribution in Table 1, we see that the log-normal as proposal produces samples that are slightly better (and closer) than the chi-square distribution.

Question 2: Gibbs sampling

Q.1 Import Data

```
load("chemical.RData")
df <-  data.frame("X" = X,"Y" = Y)
```



Above is the scatter plot of the dependence of Y on X. Three regression model are also presented (Simple Linear model and two Polynomial model). By observing the graph, the Polynomial models fit the data better than the simple linear model, with both 2nd and 5th order polynomial both fit it reasonably. In short, Polynomial model is suitable for this data set.

Q.2 Likelihood and Prior

Given the distribution of Y is $Y_i \sim N(\mu_i, varinace = 0.2), i = 1, ..., n$. Calculating the likelihood $p(\vec{Y}|\vec{\mu})$ as following.

The pdf of Normal distribution has the form.

$$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2\right)$$

The likelihood is calculate as below

$$p(\vec{Y}|\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_i-\mu_i}{\sigma}\right)^2\right) = (\frac{1}{\sigma\sqrt{2\pi}})^n \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right)$$

The prior $p(\vec{\mu})$ is calculate as below
Given

$$p(\mu_1) = 1 p(\mu_{i+1} \mid \mu_i) = \mathcal{N}(\mu_i, 0.2) \quad i = 1, ..., n_1 in\ this\ case\ 0.2 = \sigma^2$$

Using Chain Rule provide by the Lab guide,

$$\begin{aligned} p(\vec{\mu}) &= p(\mu_1) \cdot p(\mu_2|\mu_1) \cdot p(\mu_3|\mu_2) \cdot \cdots \cdot p(\mu_n|\mu_{n-1}) \\ &= 1 * \Big(\frac{1}{\sqrt{2\pi\sigma^2}}\Big)^{n-1} \exp(-\frac{\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2}{2\sigma^2}) \end{aligned}$$

Q.3 Postierior

The postierior is calculated as follow.

$$\begin{aligned} p(\vec{\mu}, \vec{Y}) = \;& p(\vec{Y}|\vec{\mu}) \cdot p(\vec{\mu}) \\ \propto \;& \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right)\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\right) \\ \propto \;& \exp\left[-\frac{1}{2\sigma^2}\Big(\sum_{i=1}^{n}(y_i-\mu_i)^2 + \sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\Big)\right] \\ \propto \;& \exp\left[-\frac{1}{2\sigma^2}\Big(\sum_{i=1}^{n-1}[(\mu_{i+1}-\mu_i)^2 + (\mu_i-y_i)^2] + (\mu_n-y_n)^2\Big)\right] \end{aligned}$$

Also, since

$$p(\vec{\mu}, \vec{Y}) = p(\mu_i|\vec{\mu}_{-i}, \vec{Y}) * p(\vec{\mu}_{-i}, \vec{Y})$$

$$p(\mu_i|\vec{\mu}_{-i}, \vec{Y}) = \frac{p(\vec{\mu}, \vec{Y})}{p(\vec{\mu}_{-i}, \vec{Y})}$$

With the following step, Using hints provide by the lab guide to get the distribution of $(\mu_1|\vec{\mu}_{-1}, \vec{Y})$, $(\mu_n|\vec{\mu}_{-n}, \vec{Y})$ *and* $(\mu_i|\vec{\mu}_{-i}, \vec{Y})$

$$\begin{aligned} p(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \propto \;& \exp\Big(-\frac{(y_1-\mu_1)^2 + (\mu_2-\mu_1)^2}{2\sigma^2}\Big) \\ \propto \;& \exp(-\frac{(\mu_1-(y_1+\mu_2)/2)^2}{2\sigma^2/2}) \\ \sim \;& N(\frac{\mu_2+y_1}{2}, \frac{\sigma^2}{2}) \end{aligned}$$

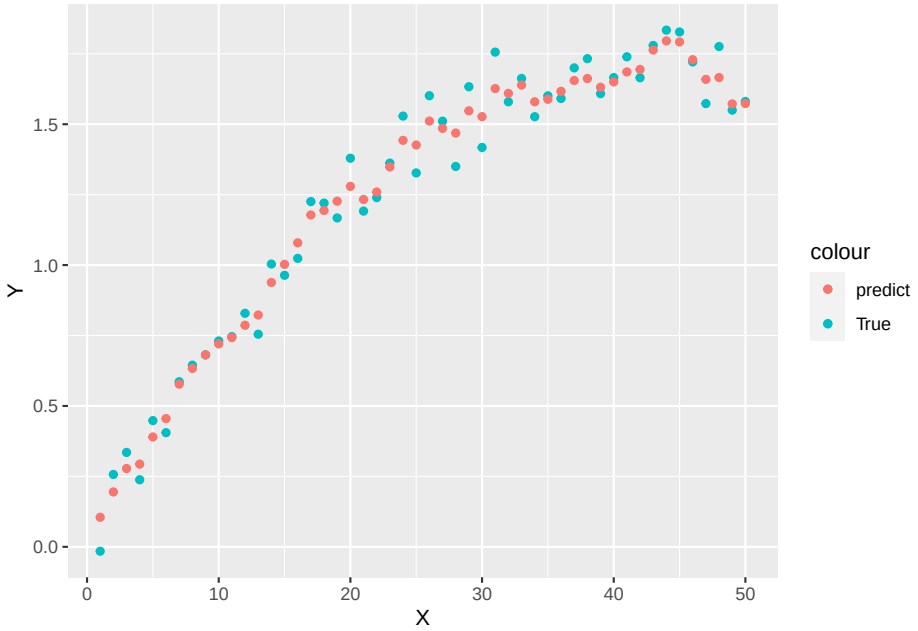
and

$$\begin{aligned} p(\mu_n|\vec{\mu}_{-n}, \vec{Y}) \propto \;& \exp\Big(-\frac{(y_n-\mu_n)^2 + (\mu_n-\mu_{n-1})^2}{2\sigma^2}\Big) \\ \propto \;& \exp(-\frac{(\mu_n-(y_n+\mu_{n-1})/2)^2}{2\sigma^2/2}) \\ \sim \;& N(\frac{\mu_{n-1}+y_n}{2}, \frac{\sigma^2}{2}) \end{aligned}$$

and

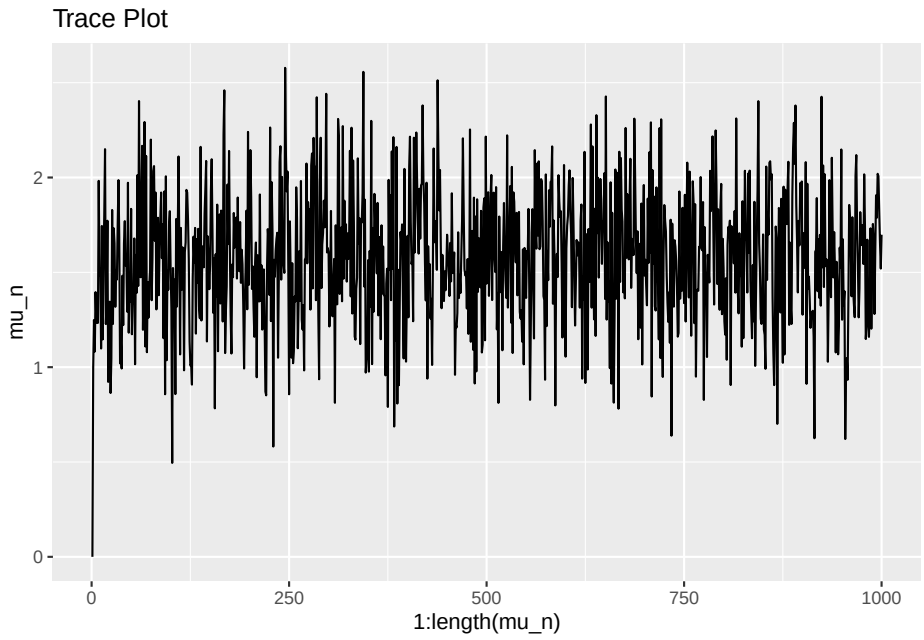
$$\begin{aligned} p(\mu_i|\vec{\mu}_{-i}, \vec{Y}) \propto \;& \exp\Big(-\frac{(\mu_{i+1}-\mu_i)^2 + (\mu_i-\mu_{i-1})^2 + (y_i-\mu_i)^2}{2\sigma^2}\Big) \\ \propto \;& \exp(-\frac{(\mu_i-(y_i+\mu_{i-1}+\mu_{i+1})/3)^2}{2\sigma^2/3}) \\ \sim \;& N(\frac{\mu_{i-1}+\mu_{i+1}+y_i}{3}, \frac{\sigma^2}{3}) \end{aligned}$$

Q.4 Implement a Gibbs sampler

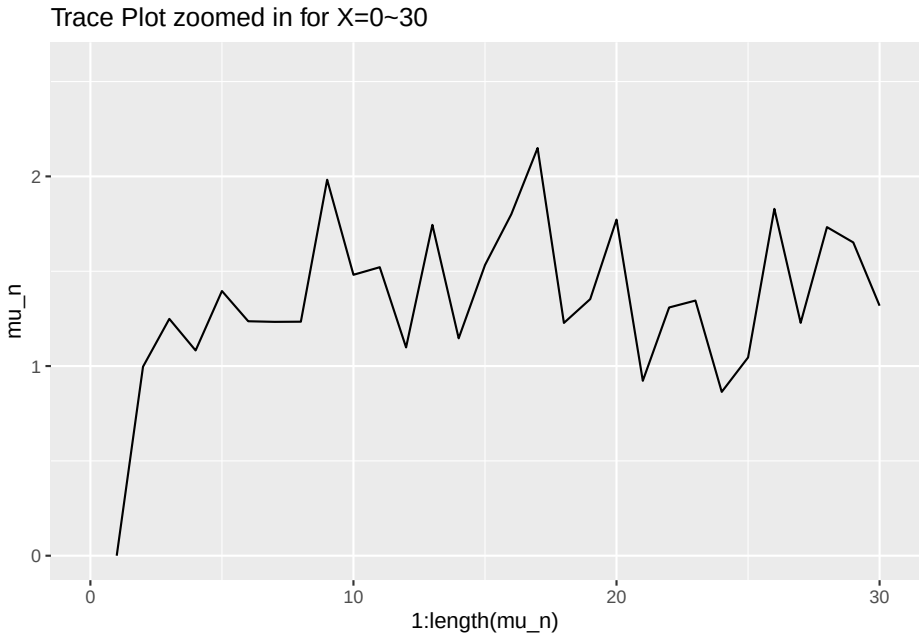


By observing the plot,this approach has manage to remove the noise as the variance become smaller (originally 0.2516844 and predicted 0.240714). Also, the expected value of $\vec{\mu}$ can catch the dependence of X and Y, with no predictions outside the true data points visually.

Q.5 Burn-in Period and Convergence



According to the trace plot above, the chain converges after a short period of burn-in.
Another plot is also presented to illustrate how the burn-in period behave. It can be conclude that the burn-in period is short in this case.


Trace Plot zoomed in for X=0~30
mu_n
2
1
0
0
10
20
30
1:length(mu_n)

Appendix

```
library(ggplot2)
#=====Task1=====

fx <- function(x){
  result <- 1*(x^5)*exp(-x)
  return(result)
}

x <- seq(0.01,100,0.01)

plot(y = fx(x), x = x, type = "l", col = 2,main = "Plot of f(x)"  )
legend(legend = c("fx"), x = "topright", lty = 1,col = 2)

MetHas <-  function(nstep,x0,flag = FALSE){
  set.seed(12345)
  vN<-1:nstep
  vX<-rep(x0,nstep)
  for (i in 2:nstep) {
    Xt<-vX[i-1]
    # random sample from log normal distribution
    y <- rlnorm(1,meanlog = log(Xt),sd = 1)
    # random sample from uniform distribution
    u <- runif(1)
    a<-min(c(1,
             (fx(y)*dlnorm(Xt,meanlog=log(y),sd =1))/(fx(Xt)*dlnorm(y,
                                                                   meanlog = log(Xt),
                                                                   sd =1))))

    if(u <= a){
      vX[i]<-y
    }else{
      vX[i]<-Xt
    }
  }
  if(isTRUE(flag)){
  plot(vN,vX,pch=19,cex=0.3,col="black",xlab="t",ylab="Samples",
       main="Trace Plot",
       ylim=c(min(x0-0.5,-10),max(x0,20)))
  abline(h=0)

  }
  return(vX)
}

  t <- MetHas(15000,100, flag = TRUE)
  plot(1:200,t[1:200],pch=19,cex=0.3,col="black",xlab="t",ylab="Samples",
       main="Trace plot for the first 200 iterations",
       ylim=c(min(100-0.5,-10),max(100,20)))
  abline(h=0)

  hist(t,breaks = 100,freq = F,
```

```
       main = "Histogram of Accepted Samples from Log Normal Proposal",
       xlab = "samples")

  MetHas_chi <-  function(nstep,x0,flag = FALSE){
  set.seed(12345)
  vN<-1:nstep
  vX<-rep(x0,nstep)
  for (i in 2:nstep) {
    Xt<-vX[i-1]
    # sample from chi-square dist
    y <- rchisq(1,df = floor(Xt+1))
    #sample from uniform
    u <- runif(1)
    a<-min(c(1,(fx(y)*dchisq(Xt,df = floor(Xt+1) ))/(fx(Xt)*dchisq(y,
                                                                   df = floor(y+1)))))
    if(u <= a){
      vX[i]<-y
    }else{
      vX[i]<-Xt
    }
  }
  if(isTRUE(flag)){
  plot(vN,vX,pch=19,cex=0.3,col="black",xlab="t",ylab="X(t)",
       main="Trace Plot",ylim=c(-5,100))
  abline(h=0)
  abline(h=15)
  }

  return(vX)
  }

  t <- MetHas_chi(15000,100,flag = T)
  hist(t,breaks = 100,freq = FALSE,
       main = "Histogram of Accepted values for the Chi-Square Distribution")


  seq_list <-  mcmc.list()
GR_has <- function(k,nstep,flag = FALSE){

  for (i in 1:k) {
    y <- MetHas_chi(nstep = nstep,i,flag)
    seq_list[[i]] <- as.mcmc(y)
     }

print(gelman.diag(seq_list))

}

   GR_has(10,15000,FALSE)


  samples1 <- MetHas(15000,1)
  paste("The mean of the accepted values are:",mean(samples1))
```

```
  t <- MetHas_chi(15000,100,flag = T)
  paste("The mean of the accepted values are:",mean(t))

  summary_df <- data.frame(Actual = numeric(),LogNormal = numeric(),
                           ChiSquare = numeric())
summary_df[1,] <- c(6,mean(samples1),mean(t))
row.names(summary_df) <- c("Mean")

summary_df


### Question 2

#=====Task1=====
load("chemical.RData")
df <-  data.frame("X" = X,"Y" = Y)

model_linear <- lm(Y~X, data=df)
model_poly_2 <- lm(Y ~ poly(X,2), data=df)
model_ploy_5 <-  lm(Y ~ poly(X,5), data=df)

ggplot(df,aes(x=X,y=Y))+
  geom_point()+
  geom_line(aes(x=X,y=predict(model_linear),colour="Simple Linear Model"))+
  geom_line(aes(x=X,y=predict(model_poly_2),colour="2nd order Polynomial model"))+
  geom_line(aes(x=X,y=predict(model_ploy_5),colour="5th order Polynomial model"))

#=====Task4=====
gibbs_sampler<-function(nstep,Y,var){
  d<-length(Y)
  mX<-matrix(0,nrow=nstep,ncol=d)
  mX[1,]<-rep(0, length(Y))

  for (i in 2:nstep){
    previous_sample<-mX[i-1,]
    current_sample<-rep(0,d)

    current_sample[1]<-rnorm(1,
                             mean=((previous_sample[2]+Y[1])/2),
                             sqrt((var)/2)
                             )


    for (j in 2:(d-1)){
      current_sample[j]<-rnorm(1,
                               mean=(Y[j] + previous_sample[j-1] + previous_sample[j+1])/3,
                               sqrt((var)/3)
                              )

    }

    current_sample[d]<-rnorm(1,
                                mean=((Y[d]+previous_sample[d-1])/2),
```

```
                                sqrt((var)/2)
                              )
    mX[i,]<-current_sample
  }
  return(mX)
}

output_samples <- gibbs_sampler(1000,Y,var=0.2)
predict_value <- c(colMeans(output_samples))
df$prediction <- predict_value
original_var <- var(Y)
predict_var <- var(predict_value)

ggplot(df)+geom_point(aes(x=X,y=Y,col="True"))+
  geom_point(aes(x=X,y=predict_value,col="predict"))

#=====Task5=====

mu_n = output_samples[,50]
trace = ggplot(data = data.frame(mu_n), aes(x = 1:length(mu_n), y = mu_n)) +
  geom_line()+ ggtitle("Trace Plot")
trace

#Zoomed
ggplot(data = data.frame(mu_n), aes(x = 1:length(mu_n), y = mu_n)) +
  geom_line() +xlim(0, 30)+ ggtitle("Trace Plot zoomed in for X=0~30")
```