Lab4

Assignment1

We start out by finding the constant for this function. If we realize this is a Gamma function finding the constant is quite easy. Looking at $x^5e^{-x}, x > 0$. If we have a look at the pdf for a Gamma PDF we have $f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x}$. Looking at this we realise that for $\beta = 1$ and $\alpha = 6$ we get x^5e^{-x} and we now just need to find the constant multiplier. Since we now know β and α we can use the constant multiplier formula for a gamma function which we know to be

$$\frac{\beta^\alpha}{\Gamma(\alpha)}$$

, by plugging in our values we get: $\frac{1^6}{\Gamma(6)} = \frac{1}{5!}$

This gives $f(x) = \frac{1}{5!}x^5e^{-x}$

1.1

In this assignment we will be creating a Metropolis-Hastings algorithm. It works as follows.

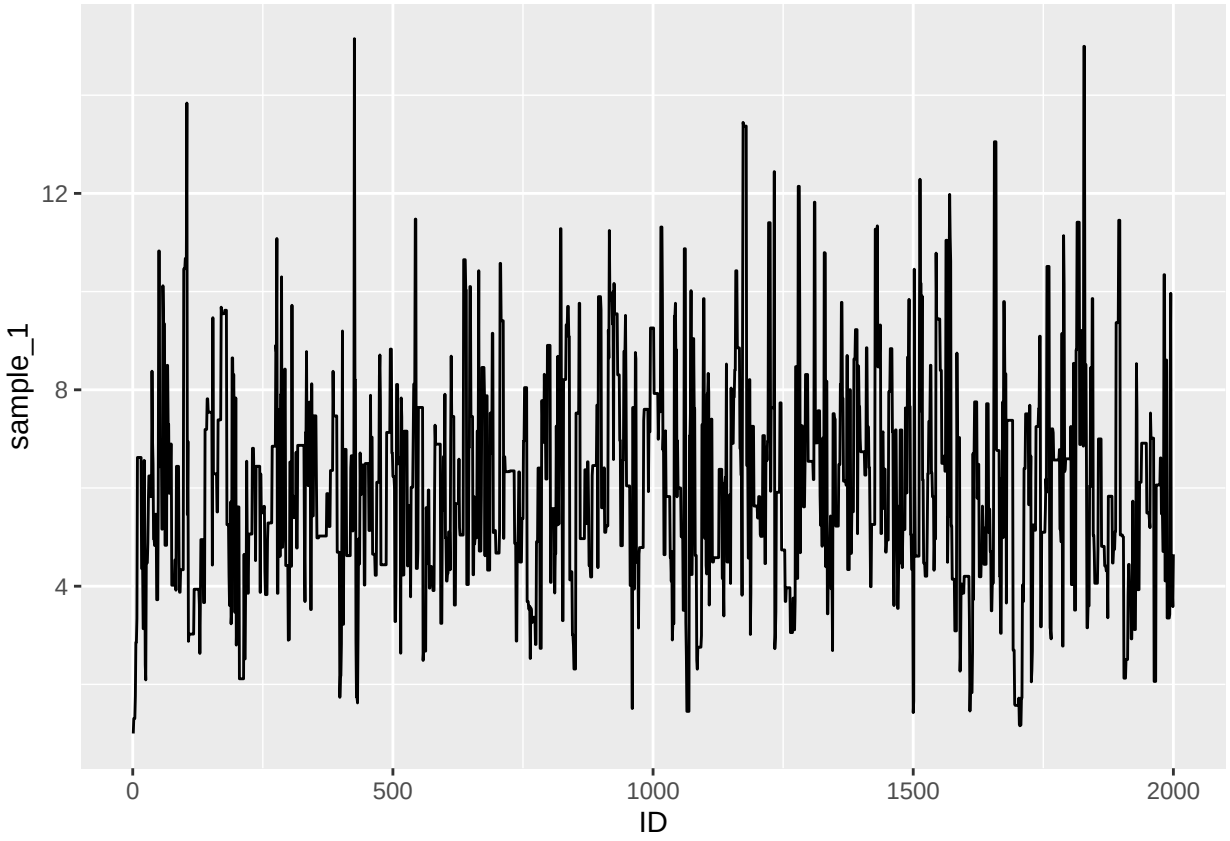
First we select a initial θ_0 value and then we do the following. For $i = 1, ..., n$ we draw a candiate value θ^*~$g(\theta^*|\theta_{i-1})$. After this we calulate our α value.
$\alpha = \frac{g(\theta^*)/g(\theta^*|\theta_{i-1})}{g(\theta_{i-1})/g(\theta_{i-1}|\theta^*)} = \frac{g(\theta^*)g(\theta_{i-1}|\theta^*)}{g(\theta_{i-1})g(\theta^*|\theta_{i-1})}$
If $\alpha \geq 1$ we accept the θ^* and set that to our current θ_i. For $0 < \alpha < 1$ we accept the θ and assign $\theta_i = \theta^*$ with the probability α.If non of these happens we reject θ^* and we assign $\theta_i = \theta^*$ with the probability $1 - \alpha$. This is what our code below does.
Bellow our code can be seen. We first create a function that contains our targeted distrubution given in the assignment. Then we create the main function which takes three arguments , the numbers of steps , a starting point and probability P.The rest of the code does the things stated above.

```
library(ggplot2)
library(coda)
# Target distrubution function
pdf_function <- function(x) x^5*exp(-x)

# Initiate Metropolis-Hastings algorithm
Metro_hast <- function(steps ,starting_point, p ){
  #variable to keep track of our current point.
  current_point <- starting_point

  #Loop to do our calulations
  for(i in 2:steps){
    currentX <- current_point[i-1]
    X_to_evaluate <- rlnorm(1, meanlog = log(currentX) , p)
    Uniform_var <- runif(1)

    ratio_numerator <- (pdf_function(X_to_evaluate)* dlnorm(currentX, meanlog = log(X_to_evaluate), p)
    denominator <- (pdf_function(currentX)*dlnorm(X_to_evaluate,meanlog=log(X_to_evaluate), p))
#comment: denominator<-(pdf_function(currentX)*dlnorm(X_to_evaluate,meanlog=log(currentX), p))
```

```
    alpha <- min(1 , ratio_numerator / denominator)

    if(Uniform_var <= alpha)current_point[i] <- X_to_evaluate

    else current_point[i] <- currentX
  }
 # Return the points
  current_point

}
```

Now we want to visualize the data generated by our function.

```
sample_1<-Metro_hast(2000,1,1)
df_1 <- as.data.frame(sample_1)
df_1$ID <- 1:2000

ggplot(df_1 , aes(x = ID , y = sample_1))+geom_line()
```

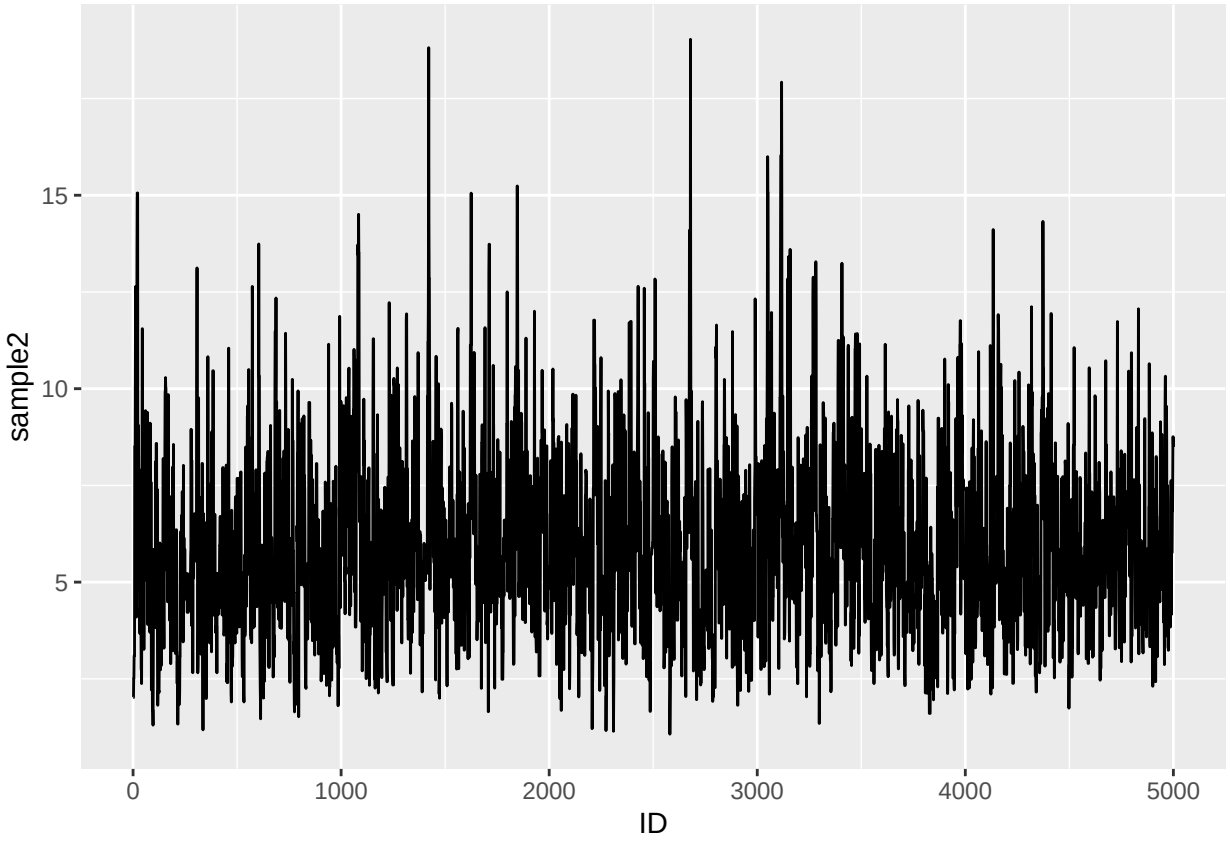


We set the numbers of steps to 2000, the starting point to one and the probability to one and get the plot above. Looking at the plot it does look like the mcmc is converging.

##1.2

```
Metro_hast_chi <- function(steps ,starting_point ){
  #create our vectors
  current_point <- rep(starting_point, steps)
```

```
  for(i in 2:steps) {

    currentX <- current_point[i-1]

    X_to_evaluate <- rchisq(1, df=floor(currentX +1))

    Uniform_var <- runif(1)

    ratio_numerator <- pdf_function(X_to_evaluate)*dchisq(currentX, df=floor(X_to_evaluate+1))

    denominator <- pdf_function(currentX)*dchisq(X_to_evaluate, df=floor(currentX+1))

    alpha <- min(1 , ratio_numerator / denominator)

    if(Uniform_var <= alpha)current_point[i] <- X_to_evaluate
    else current_point[i] <- currentX



  }
  current_point


}

sample2<-Metro_hast_chi(5000,rchisq(1,1))

df_2 <- as.data.frame(sample2)
df_2$ID <- 1:5000

ggplot(df_2 , aes(x = ID , y = sample2))+geom_line()
```

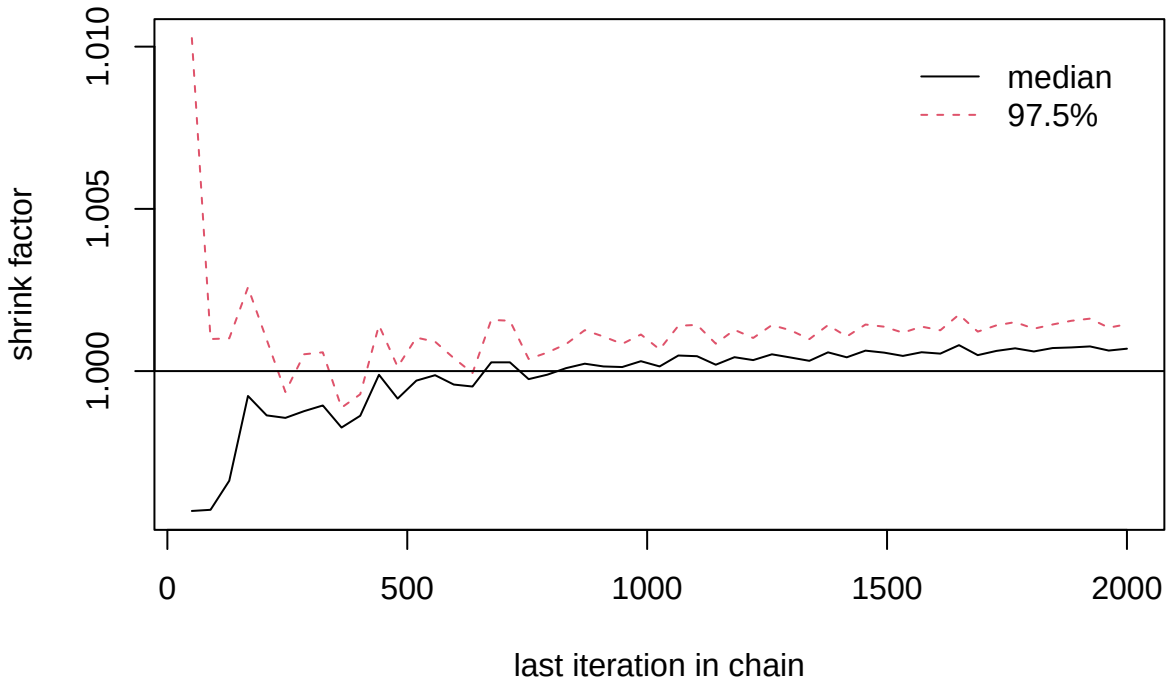


Above we create another Metropolis-Hastings algorithm that uses $\chi^2([X_t + 1])$ as the proposal distrubution. The code works in the same way as our function in 1.1. Looking at the plot it does seem like this mcm also converges. ### 1.3

```
one_to_ten <- 1:10

Matrix_1<-matrix(,10,2000)
set.seed(1234)
for(i in 1:10){
  Matrix_1[one_to_ten,] <- Metro_hast_chi(2000, i )
}

Gelman_list <- list()

for (i in 1:10) {

  Gelman_list[[i]]<- as.mcmc(Matrix_1[i,])

}
gelman.diag(Gelman_list, confidence = 0.95, transform=FALSE, autoburnin=TRUE,
            multivariate=TRUE)
```

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

```
gelman.plot(Gelman_list, bin.width = 10, max.bins = 50,
            confidence = 0.95)
```



Above we see the Gelman–Rubin method and get a value of one which should be considered very good. This means that the variance within chains and between chains variance are equal. Looking at the plot showing a Gelman plot we can also see that the chains seems to converge after around 300 steps. After this point there is a few upticks but it seems to mostly converge.

1.4

We now want to investigate the following integral. $\int_0^\infty xf(x)dx$ where $f(x) = \frac{1}{5!}x^5e^{-x}$. We can immediately realise that $\int_0^\infty f(x)dx = 1$ since integrating over a Probability density function gives us the cumulative distribution function which always equals one. We now set $g(x) = x$ and obtain the following expression. $\int_0^\infty g(x)f(x)dx = E[g(X)] = \frac{1}{n}\sum_{i=1}^n E[X_i] = \frac{1}{n}\sum_{i=1}^n x_i$
We now calculate the mean from the data generated by our two functions.

```
set.seed(1234)
draw_1 <-Metro_hast(100000,rlnorm(1,0,1),1)
mean(draw_1)
```

```
## [1] 6.004613
```

```
set.seed(1234)
draw_2 <- Metro_hast_chi(100000,rchisq(1,floor(1)))
mean(draw_2)
```

```
## [1] 6.001918
```

We can see here that both functions has a mean ≈ 6 for a big sample. If we increase the size of the samples we should get even closer to 6.

Comment: Even if chain is converging, still consider dropping some samples to account for burn-in

1.5 The Gamma pdf is as follows: $f_x(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$ To find the expected value we do the following $E[X] = \int_0^\infty x f(x) dx = \frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^\infty x^\alpha e^{-\beta x} dx$. We now set $t = \beta x$ giving us a easier expression. $\frac{\beta^\alpha}{\Gamma(\alpha)} \int_0^\infty \left(\frac{t}{\beta}\right)^\alpha e^{-t} \frac{dt}{\beta} = \frac{\beta^\alpha}{\beta^{\alpha+1}\Gamma(\alpha)} \int_0^\infty t^\alpha e^{-t} dt = \frac{\Gamma(\alpha+1)}{\beta\Gamma(\alpha)}$ Here we now use the definition of the Gamma function to get: $\frac{\alpha\Gamma(\alpha)}{\beta\Gamma(\alpha)} = \frac{\alpha}{\beta}$ $\square$
Now we plug in your values for α and β and get $\frac{6}{1} = 6$. This is in line with what we got in 1.4

Assignment2

We are given the following:

$$n = number\ of\ observations$$

$$\vec{\mu} = (\mu_1, ..., \mu_n)\ are\ unknown\ parameters$$

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

$$Prior:\ p(\mu_1) = 1\ ;\ p(\mu_{i+1}|\mu_i) = N(\mu_i, 0.2),\ i = 1, ..., (n-1)$$

We'll be interested in deriving the posterior i.e. P(μ|Y) using the Bayes' theorem as follows:

$$posterior = likelihood\ *\ prior$$
$$P(\vec{\mu}|Y) = \frac{P(Y|\vec{\mu}) * P(\vec{\mu})}{\int_\mu P(Y|\vec{\mu})P(\vec{\mu})d\mu}$$

Since the denominator is constant w.r.t μ, we can drop it in favour of proportionality

$$posterior \propto likelihood\ *\ prior$$
$$P(\vec{\mu}|Y) \propto P(Y|\vec{\mu}) * P(\vec{\mu})$$

First we define the likelihood:

$$L(Y_i|\mu_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} exp\Big(-\frac{1}{2\sigma^2}(Y_i - \mu_i)^2 \Big)$$

Since, we are only interested in the terms dependent on the parameter μ we can drop the term at the start of the expression and the likelihood then becomes:

$$L(Y_i|\mu_i) \propto exp\Big(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu_i)^2 \Big)$$

Similarly, now for the prior using chain rule,

$$P(\vec{\mu}) = P(\mu_1).P(\mu_2|\mu_1).P(\mu_3|\mu_2)...P(\mu_n|\mu_{n-1}) \qquad ...(a)$$
$$= \prod_{i=1}^{n-1} P(\mu_{i+1}|\mu_i)\ ;\ P(x) = \ N(x|\mu_i, \sigma^2_{\mu_i})$$
$$= \prod_{i=1}^{n-1} \frac{1}{\sqrt{2\pi\sigma^2}} exp\Big(-\frac{1}{2\sigma^2_\mu}(\mu_{i+1} - \mu_i)^2 \Big)$$
$$\propto exp\Big(-\frac{1}{2\sigma^2_\mu} \sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2 \Big)$$

Hence, we can now write our posterior as the following:

$$P(\vec{\mu}|Y) \propto exp\Big(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu_i)^2\Big) \ * \ exp\Big(-\frac{1}{2\sigma_\mu^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\Big)$$

$$\propto exp\Big[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu_i)^2 - \frac{1}{2\sigma_\mu^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\Big] \quad ...(b)$$

Since, σ is given to be same everywhere in the problem we can use common notation across the expression.

Moving to the next part to find conditional probability $(\mu_k|\mu_{-k})$, we know that by the definition of conditional probability, the joint probability of the prior, $P(\vec{\mu})$, can be expanded into the conditional components of each individual μ_k and can be expressed as

$$P(\vec{\mu}) \propto P(\mu_k|\vec{\mu}_{-k}) * P(\mu_1, ..., \mu_{k-1}, \mu_{k+1}, ..., \mu_n)$$

Or in a more condensed form as,

$$P(\mu_i|\vec{\mu}_{-i}, \vec{Y}) = \frac{P(\vec{\mu}|\vec{Y})}{P(\vec{\mu}_{-i}|\vec{Y})} \qquad ...(c)$$

$$= \frac{P(\vec{\mu}|\vec{Y})}{\int P(\vec{\mu}|\vec{Y})d\mu}$$

We now solve for μ_1

$$P(\mu_1|\vec{\mu}_{-1}, \vec{Y}) = \frac{P(\vec{\mu}, \vec{Y})}{\int P(\vec{\mu}, \vec{Y})d\mu_1}$$

$$= \frac{\prod_{i=2}^{n} P(Y_i|\mu_i)\prod_{i=3}^{n} P(\mu_i|\mu_{i-1})P(\mu_2|\mu_1)P(Y_1|\mu_1)P(\mu_1)}{\prod_{i=2}^{n} P(Y_i|\mu_i)\prod_{i=3}^{n} P(\mu_i|\mu_{i-1})\int P(\mu_2|\mu_1)P(Y_1|\mu_1)P(\mu_1)d\mu_1}$$

Common product terms cancel out in numerator and denominator, and the integral is constant w.r.t. μ_1

$$P(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \propto P(\mu_2|\mu_1)P(Y_1|\mu_1)P(\mu_1)$$

$$\propto exp\Big(-\frac{1}{2\sigma^2}(\mu_2-\mu_1)^2\Big) \ * \ exp\Big(-\frac{1}{2\sigma^2}(Y_1-\mu_1)^2\Big) \ * \ 1$$

$$\propto exp\Big(-\Big(\frac{1}{2\sigma^2}\Big)\Big((\mu_2-\mu_1)^2 + (Y_1-\mu_1)^2\Big)\Big)$$

using *Hint B*

$$P(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \propto exp\Big(-\frac{(\mu_1-(Y_1+\mu_2)/2)^2}{\sigma^2}\Big) \ \sim \ N\Big((\frac{Y_1+\mu_2}{2}), \frac{\sigma^2}{2}\Big)$$

Similarly, we now check the case for μ_n. Here we use the posterior we defined in (b) and the relationship we established in (c)

$$P(\mu_n|\vec{\mu}_{-n}, \vec{Y}) = \frac{P(\vec{\mu}|\vec{Y})}{P(\vec{\mu}_{-n}|\vec{Y})}$$

$$\propto \frac{exp\Big(-\sum_{i=1}^{n-1}\frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n}\frac{(Y_i-\mu_i)^2}{2\sigma^2}\Big)}{exp\Big(-\sum_{i=1}^{n-2}\frac{(\mu_{i+1}-\mu_i)^2}{2\sigma^2} - \sum_{i=1}^{n-1}\frac{(Y_i-\mu_i)^2}{2\sigma^2}\Big)}$$

$$\propto exp\Big(-\Big(\frac{1}{2\sigma^2}\Big)\Big((\mu_n-\mu_{n-1})^2 + (Y_n-\mu_n)^2\Big)\Big)$$

again using *hint B*

$$P(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto exp\Big(-\frac{(\mu_n-(Y_n+\mu_{n-1})/2)^2}{\sigma^2}\Big) \sim N\Big((\frac{Y_n+\mu_{n-1}}{2}),\frac{\sigma^2}{2}\Big)$$

Now for the tedious case of μ_i where is i $\notin$ (1,n)

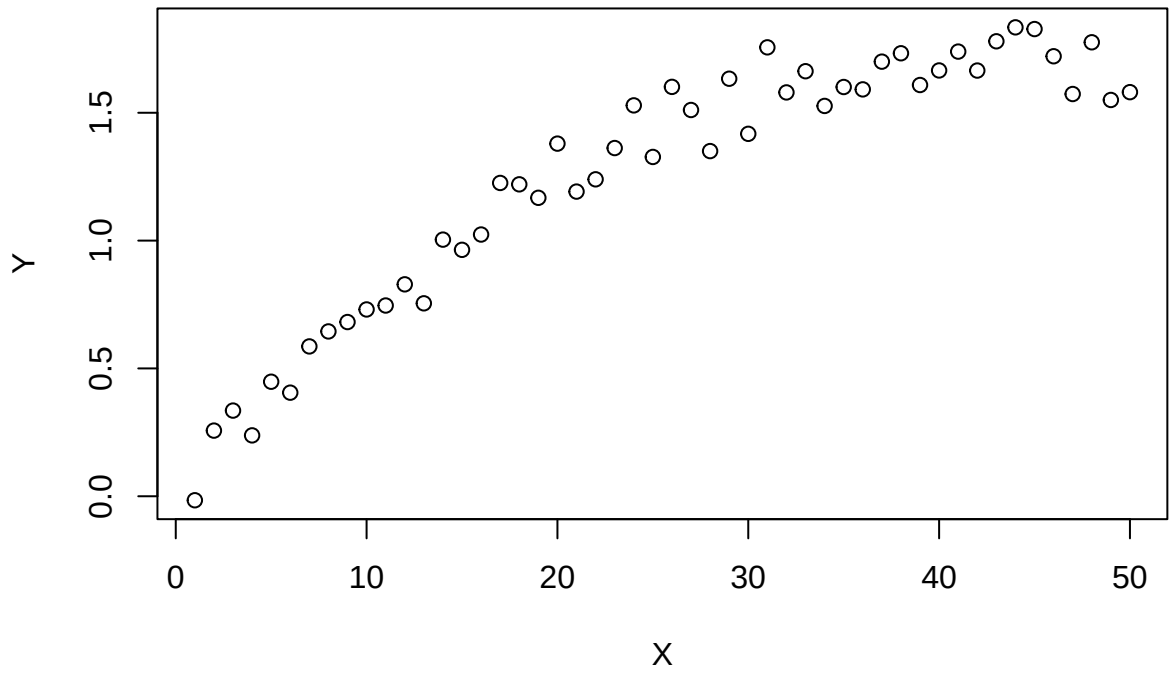
$$P(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto P(\mu_{i+1}|\mu_i)P(Y_i|\mu_i)P(\mu_i|\mu_{i-1}) \propto exp\Big[-\Big(\frac{1}{2\sigma^2}\Big)\Big((\mu_{i+1}-\mu_i)^2+(Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2\Big)\Big]$$

Using *hint C*

$$P(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto exp\Big[-\frac{(\mu_i-(Y_i+\mu_{i-1}+\mu_{i+1})/3)^2}{2\sigma^2/3}\Big] \sim N\Big((\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3}),\frac{\sigma^2}{3}\Big)$$

We now use these derivations to design a Gibbs sampler which will draw 1000 samples from the posterior of μ

```
#load the data
load("chemical.RData")
#converting data into a DF
chemical_df <- data.frame(X, Y)
#initial plot of data to understand distribution
plot(X, Y)
```

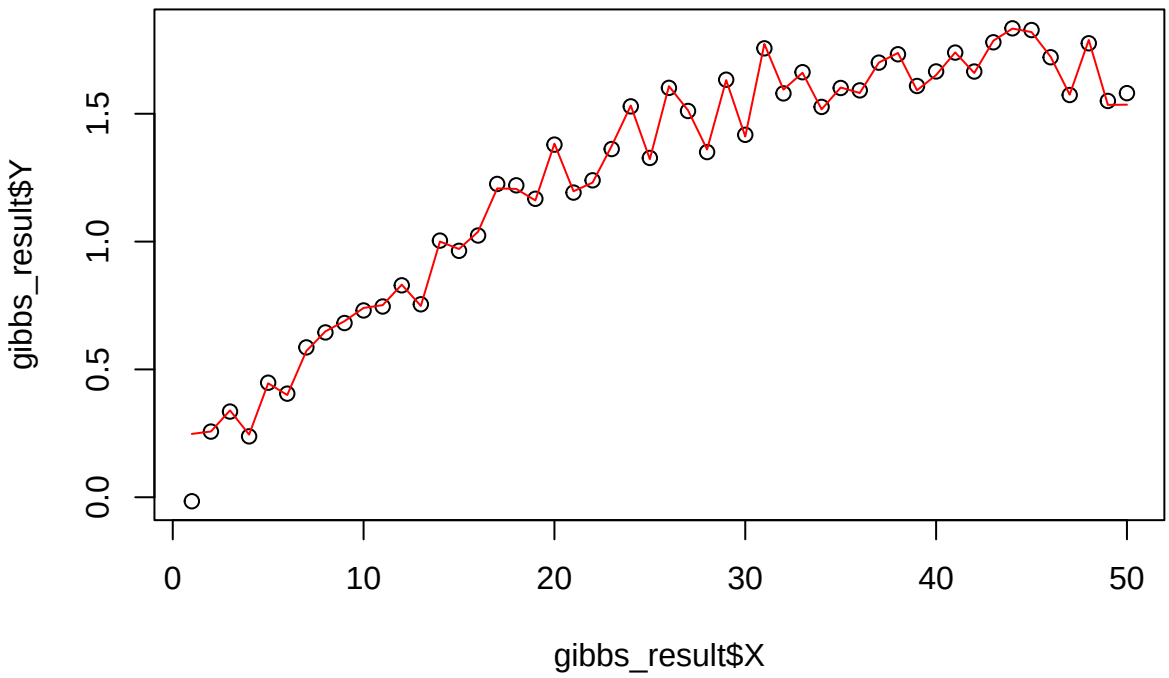



```
#3 inputs to sampler: # of time steps, Y to be estimated, known sigma
mc_gibbs <- function(step, Y, sig) {
  mu_i <- 0
```

```
  #initialize matrix to store results of sampler in each iteration
  result_matrix <- matrix(0, nrow = 50, ncol = step)
  for (tstep in 1:(step-1)) {
    for (i in 1:50) {
      #Case for mu_1
      if (i==1) {
        mu_i <- mean(result_matrix[i+1,tstep],Y[i])
        result_matrix[i,tstep+1] <- rnorm(1, mean = mu_i, sd = sig/sqrt(2))
      }
      #Case for mu_n
      else if (i == 50) {
        mu_i <- mean(result_matrix[i-1,tstep+1],Y[i])
        result_matrix[i,tstep+1] <- rnorm(1, mean = mu_i, sd = sig/sqrt(2))
      }
      #Case for mu_i
      else {
        mu_i <- mean(Y[i],result_matrix[i-1,tstep+1], result_matrix[i+1,tstep])
        result_matrix[i,tstep+1] <- rnorm(1, mean = mu_i, sd = sig/sqrt(3))
      }
    }
  }
  return(result_matrix)
}
```

Plotting the results of the sampler:

```
#function call with 1000 steps and variance 0.2
gibbs_output <- mc_gibbs(1000, chemical_df$Y, sqrt(0.2))
#append expected values of mu to original DF
gibbs_result <- cbind(chemical_df, rowMeans(gibbs_output))
names(gibbs_result) <- c("X", "Y", "expected_mu")
#Plot original points
plot(gibbs_result$X, gibbs_result$Y)
lines(gibbs_result$expected_mu, col = "red")
```

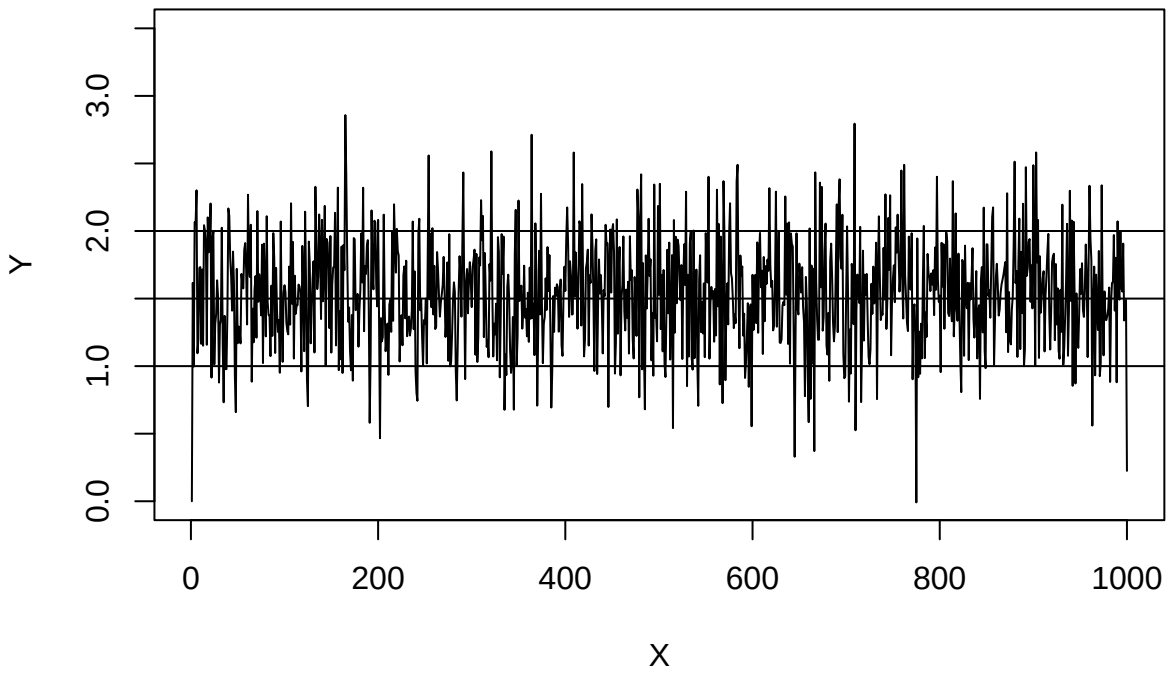



```
#Expected values as a line
```

The result shows that our expected values have followed a similar result as the original data. Implying that not much improvement has happened in terms of reducing the measurement noise of the concentration. But the noise reduction at the start and end is quite pronounced, especially at the start, thus, suggesting that our burning period may have been very short.

Trace Plot:

```
vN <- 1:ncol(gibbs_output)
vX <- gibbs_output
plot(vN,vX[50,],pch=19,cex=0.3,col="black",xlab="X",ylab="Y", ylim=c(0,3.5), type = 'l')
abline(h=1)
abline(h=2)
abline(h=1.5)
```



Our above conclusion is supported by the trace plot, where we see that the chain doesnt converge *(comment: Don't think I agree with this, it seems to converge to some stationary distribution. Yes, this distribution is probably not what we are expecting the posterior to be, but it seems to have converged nonetheless)* and keeps producing noisy values, but the burning period is evidently quite small. But this can also be because the data set that we are working with is quite small (only 50 observations) *(comment: Note that we would only get better results if we have multiple observations per time step. If we have a dataset of 100 observations at 100 different times, we would have 100 parameters μ to estimate. So this would not necessarily give better results -> we doubled the data but also made the problem twice as hard)* . To expect a good reduction in noise, we would ideally want a larger data set.