Computational statistics Lab 04 Report

Karthikeyan Devarajan - Karde799

Question 1: Computations with Metropolis - Hastings

Given Function in the lab :- $f(x) \propto x^5 e^{-x}$

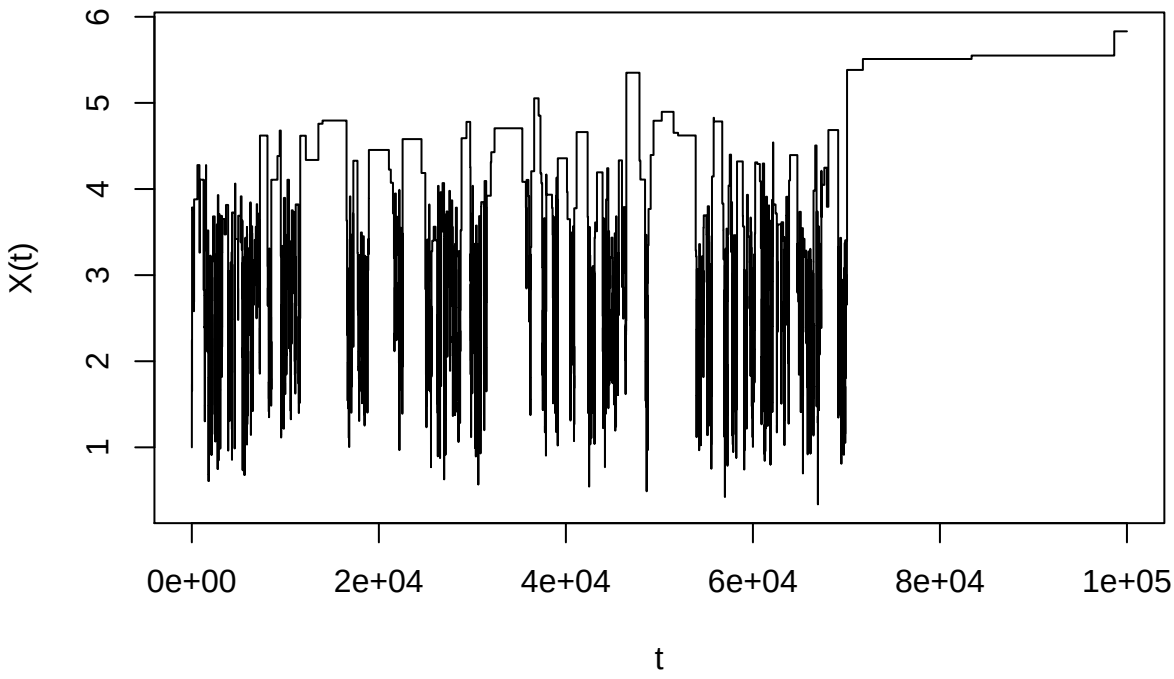
We could check the constant by applying integration by parts.

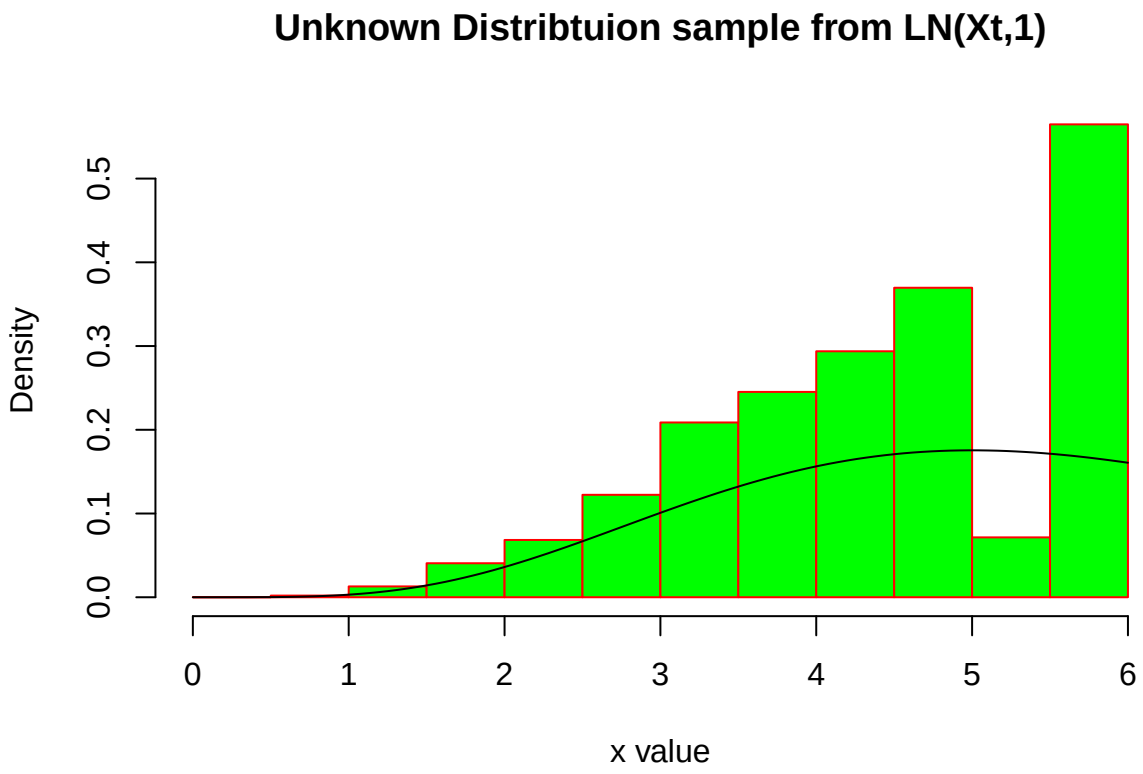
```
## 120 with absolute error < 7.3e-05
```

```
## Function Constant is  120
```

Part 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?

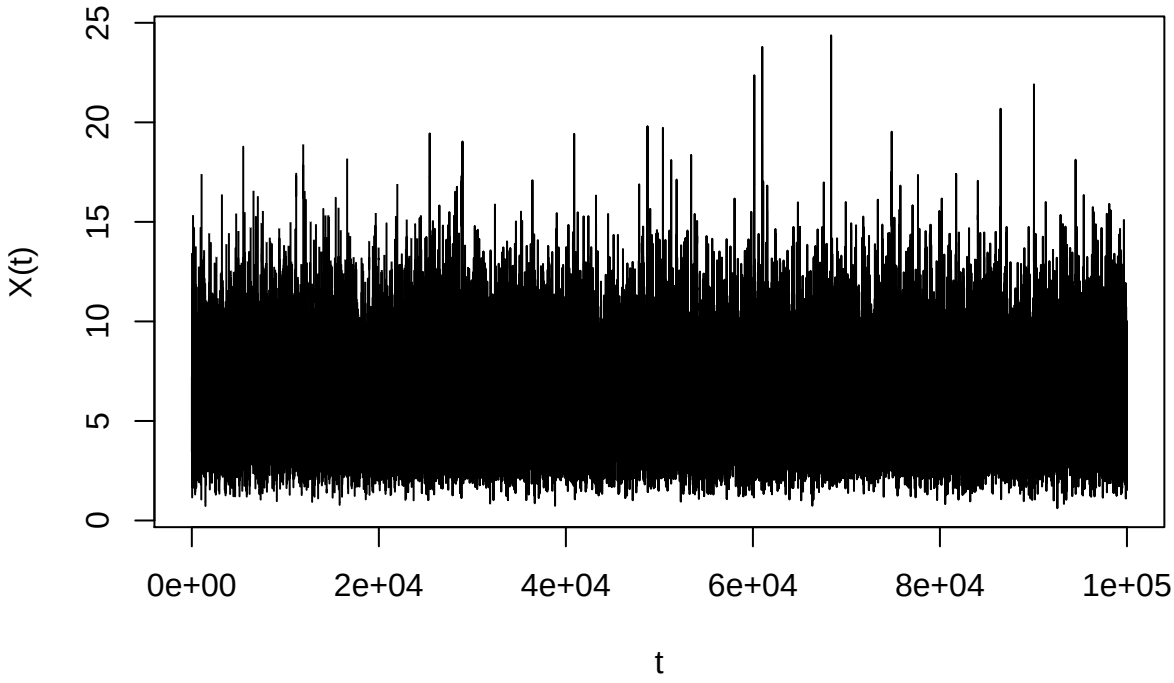


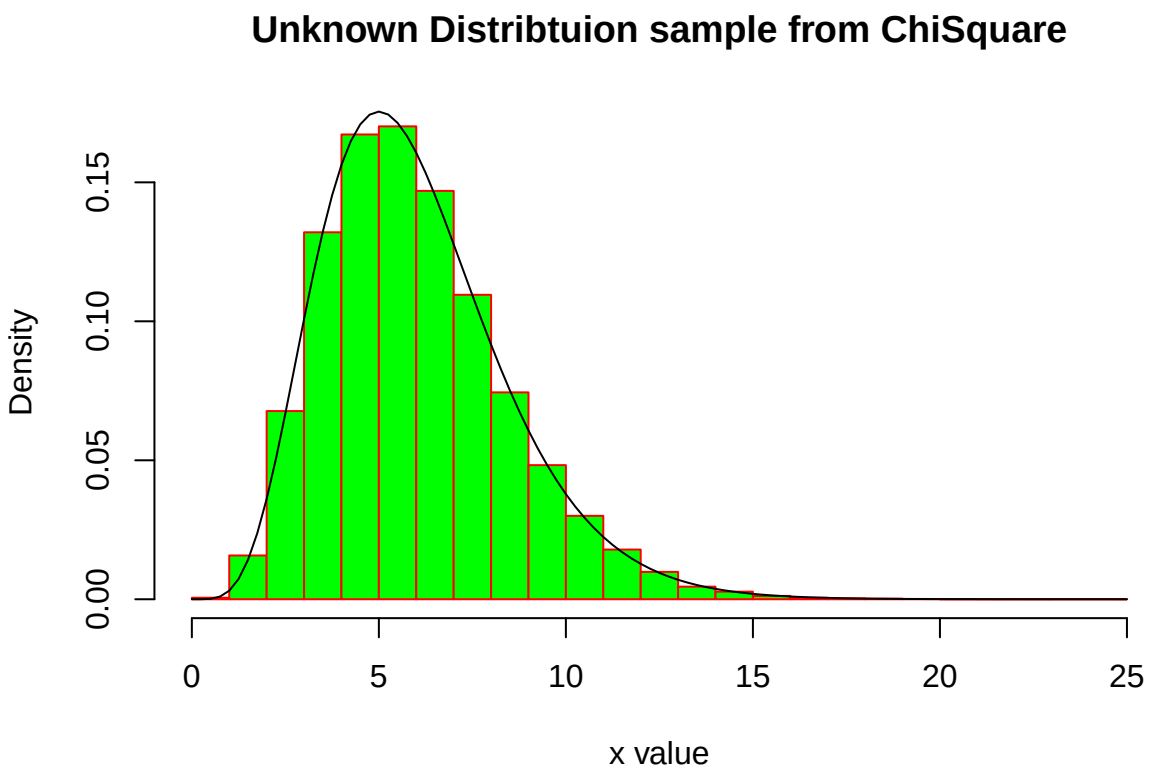


Burn - in period is the time taken for Markov chain to reach the equilibium. There is no burn in period and 100000 would be a good number for finding the shape of convergence. Since the log-normal will try to reduce the dimensionality and the rate of slope changing.

Part 2

Perform Step 1 by using the chi - square distribution $Y\ ^2(X_t + 1)$ *as a proposal distribution, where* x *is the floor function, meaning the integer part of* x *for positive* x*, i.e.* $2.95 = 2$


X(t)
0
5
10
15
20
25
0e+00
2e+04
4e+04
6e+04
8e+04
1e+05
t



There is no convergence pattern in this graph but it is normally distributed, when we look at the histogram graph. The graph is too densely plotted and no burn in period.

Part 3

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

Both the log-normal distribution and chi square distribution there is no burn in period. Log normal for this equation is getting staurated at one point rapidly and the density curve is negatively skewed and has low kurtosis. whereas the chi square distribution for the function has a density curve with better skweness and kurtosis. So chi square distribution is giving out good sample points.

Part 4

Generate 10 MCMC sequences using the generator from Step 2 and starting points 1, 2, . . . ,10. Use the Gelman - Rubin method to analyze convergence of these sequences.

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

Gelman - rubin is used for finding the nominal initial value and the reduction factor for the make the distribution converge.

Part 5

Estimate $\mu = \int_{-\infty}^{+\infty} xf(x)dx$ *using sample 1 and sample 2.*

$\theta = \int_0^{\infty} xf(x)dx$

Where $\int_0^{\infty} f(x)dx = 1$

So let's take $f(x) = \frac{x^5 e^{-x}}{120}$

then $\int_0^{\infty} f(x)dx = 1$

$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} x_i$

```
## theta_hat_step1  4.1222
```

```
## theta_hat_step2  6.007717
```

Part 6

The distribution generated is in fact a gamma distribution and look in literature and define the actual value of the integral. Compare it with the one you obtained.

$\int_0^{\infty} xf(x)dx = \int_0^{\infty} \frac{x^6 e^{-x}}{120}dx$

This Equation can be Integrated by using R

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

```
## Distribution Actual Integral is  6
```

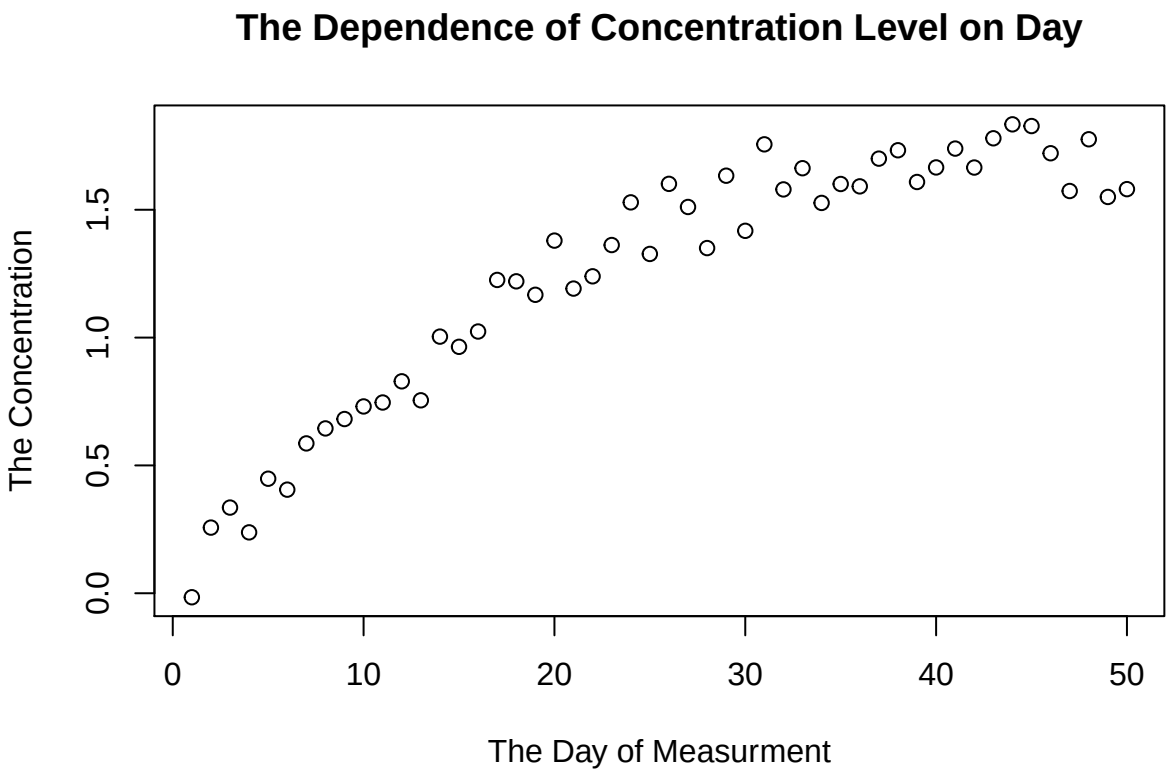
The closest value to the actual distribution value is $\hat{\theta}$ of step 2 which points is produced by chi-square distribution.

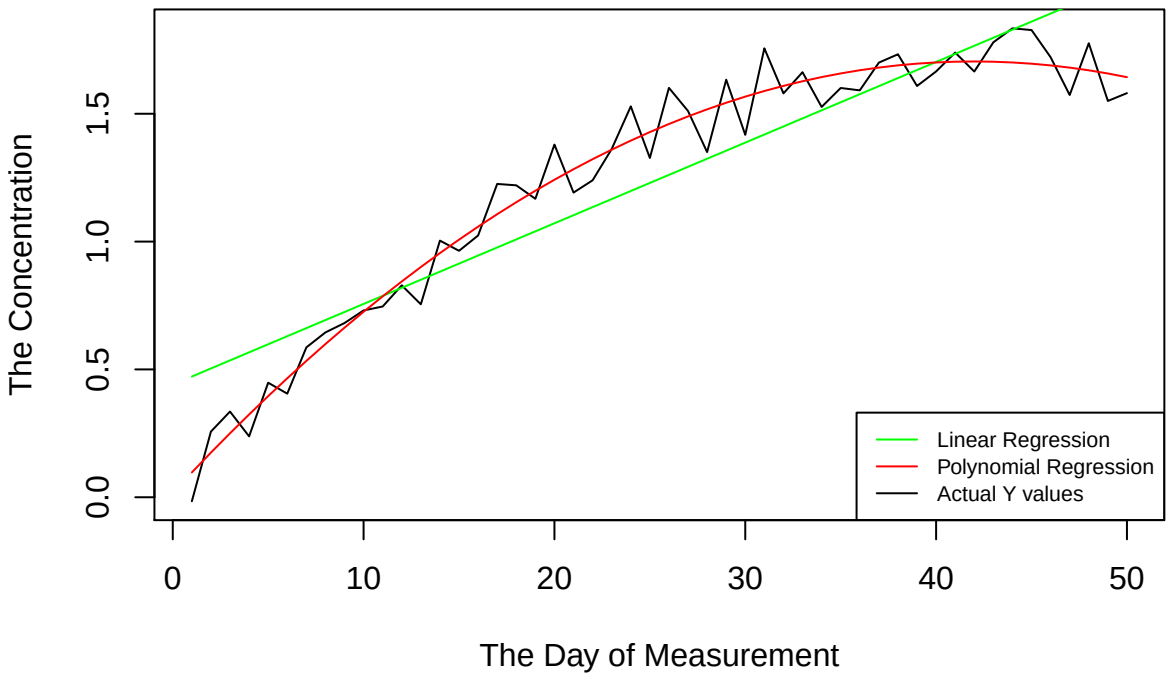
Question 2: Gibbs sampling

Part 1:

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.

Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?


The Dependence of Concentration Level on Day
The Concentration
0.0
0.5
1.0
1.5
0
10
20
30
40
50
The Day of Measurment



It can be determined that the polynomial equation with order 2 is found to be reasonable model since the predicted value of the polynomial equation is similar to Y.

Part 2:

b) A researcher has decided to use the following (random walk) Bayesian model (n=numberof observations,$\vec{\mu} = (\mu_1, ..., \mu_n)$ *are unknown parameters):*
$Y_i \sim \mathcal{N}(\mu_i, variance = 0.2), i = 1, ..., n$
where the prior is
$p(\mu_1) = 1$
$p(\mu_{i+1}/\mu_i) \sim \mathcal{N}(\mu_i, 0.2), i = 1, ..., n_1$
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* $p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)...p(\mu_n|\mu_{n1})$*.*

The likelihood is the product of our function, which is a normal distribution function in our case:

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

The normal distribution function is:

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

$$\mathrm{p}(\mu_1 = 1)$$

Likelihood $= P(\vec{Y}|\vec{\mu})$

$$= \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi\sigma^2)}}\ e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}$$

$$= (\frac{1}{\sqrt{(2\pi\sigma^2)}})^n \prod_{i=1}^{n}\ e^{-\frac{(y_i-\mu)^2}{2\sigma^2}}$$

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

Secondly we calculate the prior from:

$$\mathrm{p}(\mu_1) = 1$$

$$p(\vec{\mu}) = p(\mu_1)\ p(\mu_2|\mu_1)\ p(\mu_3|\mu_2)\ ...\ p(\mu_n|\mu_{n-1})$$

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

The general forumula for the prior would be:

$$\mathrm{p}(\mu_i|\mu_{i-1})$$

Prior $= \mathrm{p}(\vec{\mu}) = \prod_{i=1}^{n} \mathrm{p}(\mu_i|\mu_{i-1})$

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

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

And since: $\mathrm{p}(\mu_1) = 1$

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

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

And this is the formula for our prior.

Part 3:

Use Bayesian 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} is a vector containing all μ values except of μ_i.

$$posterior = likelihood\ *\ prior$$

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

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

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

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

$$= (\frac{1}{\sqrt{(}2\pi\sigma^2)})^{2n} \ e^{-\frac{1}{2\sigma^2}(\ (y_1-\mu_1)^2+\sum_{i=2}^{n}(y_i-\mu_i)^2+\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2)}$$

$$= (\frac{1}{\sqrt{(}2\pi\sigma^2)})^{2n} \ e^{-\frac{1}{2\sigma^2}(\ (y_1-\mu_1)^2+\sum_{i=2}^{n}((y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2))}$$

And this is our posterior formula.

and since $(x-\mu)^2 = (\mu - x)^2$ we can rewrite our formula in this way:

$$= (\frac{1}{\sqrt{(}2\pi\sigma^2)})^{2n} \ e^{-\frac{1}{2\sigma^2}(\ (\mu_1-y_1)^2+\sum_{i=2}^{n}((\mu_i-y_i)^2+(\mu_i-\mu_{i-1})^2))}$$

To consider three forumulas for μ, and write them up to a conistant we have to drop all the terms that don't include the value that we are looking for and considering proportionality:

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

This equation is proportional to the formula:

$$\exp(\frac{1}{d}((x-a)^2+(x-b)^2))\ \alpha\ \exp(-\frac{(x-(a+b)/2)}{d/2})$$

Thus the formula becomes:

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

Thus

$$p(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \sim \mathcal{N}(\frac{y_1+\mu_2}{2}, \frac{\sigma^2}{2})$$

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

This formula also follows the same formula:

$$\exp(\frac{1}{d}((x-a)^2+(x-b)^2))\ \alpha\ \exp(-\frac{(x-(a+b)/2)}{d/2})$$

Then it becomes:

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

And it follows:

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \sim \mathcal{N}(\frac{y_n+\mu_{n-1}}{2},\frac{\sigma^2}{2})$$

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

For this formula, it follows:

$$\exp(\frac{1}{d}((x-a)^2+(x-b)^2+(x-c)^2))\ \alpha\ \exp(-\frac{(x-(a+b+c)/3)}{d/3})$$

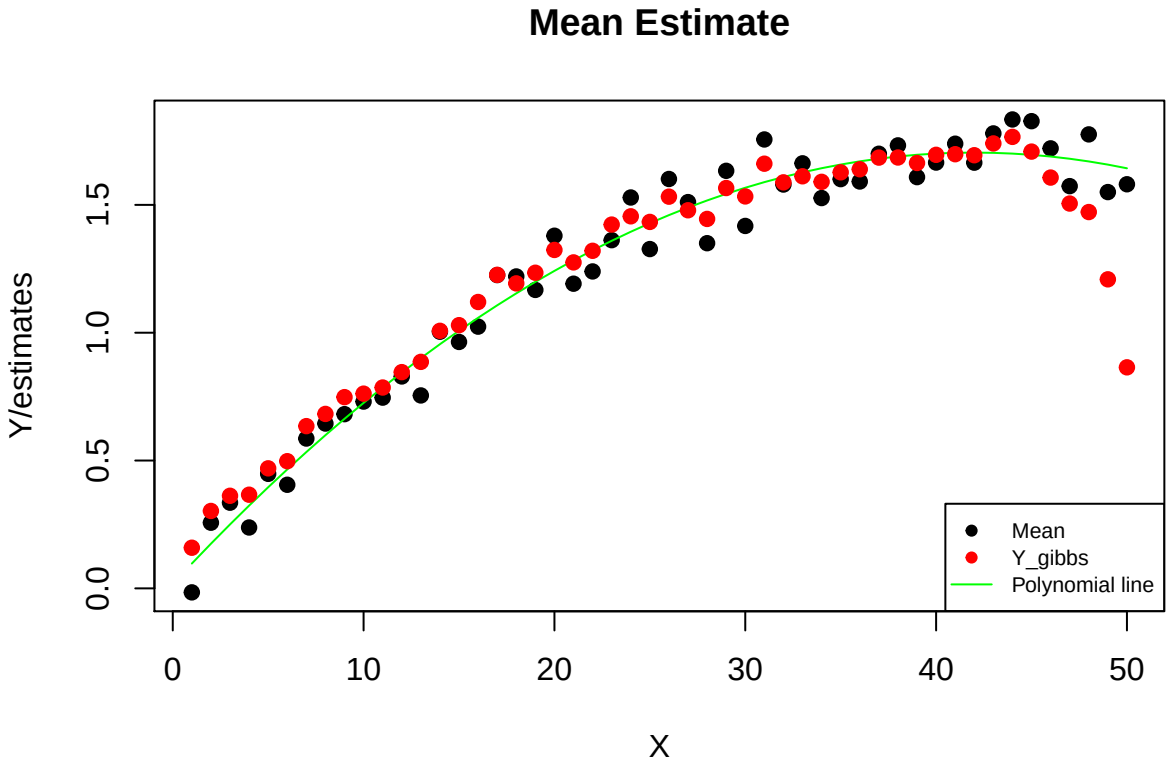
Then the formula becomes:

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

And the distribution of this would be:

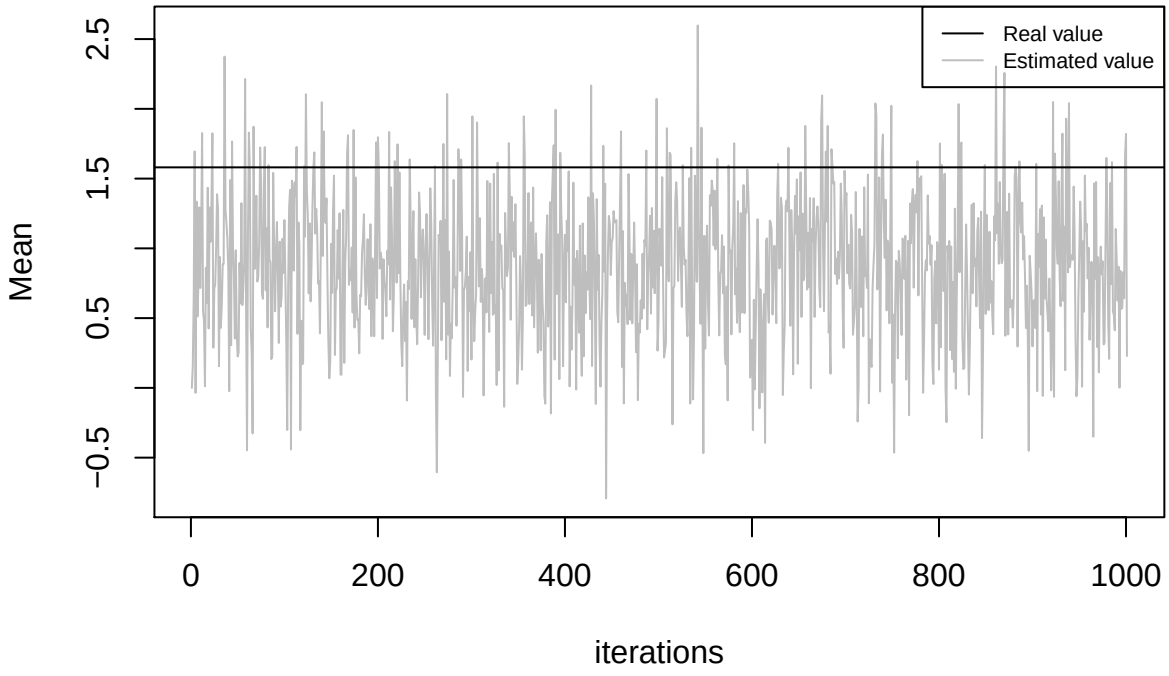
$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \sim \mathcal{N}(\frac{y_i+\mu_{n-1}+\mu_{i+1}}{3},\frac{\sigma^2}{3})$$

d) Use the distributions derived in Step 3 to implement a Gibbs sampler that uses $\mu_o = (0,...,0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of μ and then compute the expected value of μ by using a Monte Carlo approach. Plot the expected value of μ 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 μ can catch the true underlying dependence between Y and X?



From the graph, the noisy data is reduced but not completely. Towards the end the gibbs estimated

value awya from the actual data, but from intial to last few points the noise is reducing. it is proving the dependancy between X and Y. *e) Make a trace plot for μ_n and comment on the burn - in period and convergence*



The burning period is very small. The convergence is rate of change of slope and it has a very large value.

References

https://www.youtube.com/watch?v=U561HGMWjcw

https://www.reddit.com/r/rstats/comments/884rlc/how_to_calculate_gelmanrubin_convergence_from/

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
RNGversion(min(as.character(getRversion()),"3.6.2"))
set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")
targetFunc = function(x) {
  return(ifelse(x > 0, x**5 * exp(-x), 0))
}

targetFuncPDF = function(x) {
  return(ifelse(x > 0, (x**5 * exp(-x)) / 120, 0))
}

integrate(targetFunc, 0, Inf)
cat('Function Constant is ', 120)
MCMCSample = function(nstep,X0,props,getData = FALSE) {
```

```
  vN = 1:nstep
  vX = rep(X0,nstep)

  for (i in 2:nstep) {
    X = vX[i-1]
    Y = rlnorm(1, meanlog = X,sdlog = props) # Proposal Distribution

    u = runif(1)
    numVal = targetFunc(Y) * dlnorm(X, meanlog = Y,sdlog = props)
    denVal = targetFunc(X) * dlnorm(Y, meanlog = X,sdlog = props)
    a = min(c(1, numVal/denVal))

    if (u <= a) {
      vX[i] = Y
    } else {
      vX[i] = X
    }
  }

  if (getData) {
    returnData = data.frame('vN' = vN,
                            'vX' = vX)
    return(returnData)
  } else {
    plot(x = vN,
         y = vX,
         pch = 19,
         cex = 0.3,
         type = "l",
         col = "black",
         xlab = "t",
         ylab = "X(t)",
         main = ""
    )

    hist(vX,
         main = "Unknown Distribtuion sample from LN(Xt,1)",
         xlab = "x value",
         border = "red",
         col = "green",
         prob = TRUE
    )
    curve(targetFuncPDF(x), add = TRUE)
    # abline(h=0)
    # abline(h=1.96)
    # abline(h=-1.96)
  }

}


MCMCSample(100000,1,1)
MCMCSampleChi = function(nstep,X0,getData = FALSE) {
```

```
  vN = 1:nstep
  vX = rep(X0,nstep)

  for (i in 2:nstep) {
    X = vX[i-1]
    Y = rchisq(1, df = abs(floor(X + 1))) # Proposal Distribution

    u = runif(1)
    numVal = targetFunc(Y) * dchisq(X, df = abs(floor(Y + 1)))
    denVal = targetFunc(X) * dchisq(Y, df = abs(floor(X + 1)))
    a = min(c(1, numVal/denVal))

    if (u <= a) {
      vX[i] = Y
    } else {
      vX[i] = X
    }
  }

  if (getData) {
    returnData = data.frame('vN' = vN,
                            'vX' = vX)
    return(returnData)
  } else {
    plot(x = vN,
         y = vX,
         pch = 19,
         cex = 0.3,
         type = "l",
         col = "black",
         xlab = "t",
         ylab = "X(t)",
         main = ""
    )

    hist(vX,
         main = "Unknown Distribtuion sample from ChiSquare",
         xlab = "x value",
         border = "red",
         col = "green",
         prob = TRUE
    )
    curve(targetFuncPDF(x), add = TRUE)
    # abline(h=0)
    # abline(h=1.96)
    # abline(h=-1.96)
  }
}

MCMCSampleChi(100000,2.95)
library(coda)

data_list = mcmc.list()
```

```
for (index in 1:10) {
  mcmcItem = mcmc(MCMCSampleChi(100000,index,getData = TRUE)$vX)
  data_list[[index]] = mcmcItem
}

gelman.diag(data_list)
theta_hat_logNormal = (1 / 100000) * sum(MCMCSample(100000,1,1,getData = TRUE)$vX)
theta_hat_chiSquare = (1 / 100000) * sum(MCMCSampleChi(100000,1,getData = TRUE)$vX)

#print(theta_hat_logNormal)
#print(theta_hat_chiSquare)

cat('\n')
cat('theta_hat_step1 ', theta_hat_logNormal, '\n')
cat('theta_hat_step2 ', theta_hat_chiSquare, '\n')
sampleGammaFunc = function(x) {
  # x f(x)
  return(ifelse(x > 0, (x**6 * exp(-x)) / 120, 0))
}

integrate(sampleGammaFunc, 0, Inf)

cat('Distribution Actual Integral is ', 6, '\n')
load(file.choose())
plot(X,Y, xlab = "The Day of Measurment", ylab = "The Concentration", main = "The Dependence of Concenti
Predict_lin <- predict(lm(Y~X),X=X)
Predict_pol <- predict(lm(Y~poly(X,2)),X=X)
plot(Y ~ X,xlab = "The Day of Measurement",ylab = "The Concentration",pch = 20,type = "l")
lines(Predict_lin,col="green")
lines(Predict_pol,col="red")
legend("bottomright",c("Linear Regression","Polynomial Regression","Actual Y values"),col=c("green","red
Gibbs <- function(N=1000, Y, mu) {
  k <- 50
  res <- matrix(nrow=N+1, ncol=k)
  res[1,] = mu
  for(i in 2:(N+1)) {
    res[i,1] <- -rnorm(1, -(Y[1]+res[i-1,2])/2, sqrt(0.2))

    for(d in 2:(k-1)) {
      res[i,d] <- -rnorm(1, -(Y[d]+res[i-1,d+1])/2, sqrt(0.2))
    }
    res[i,k] <- -rnorm(1, -(Y[k]+res[i-1,1])/2, sqrt(0.2))
  }
  result <- list(Steps=res)
}

Y_gibbs <- Gibbs(1000, Y=Y, mu=rep(0,50))
output_est <- colMeans(Y_gibbs$Steps[-1*1:200,])
plot(Y~X, type="p", main="Mean Estimate",xlab = "X",ylab = "Y/estimates",pch = 19)
lines(Predict_pol,col = "green")
lines(output_est, col="red",type = "p",pch = 19)
legend("bottomright", legend=c("Mean","Y_gibbs","Polynomial line"), col=c("black","red","green"), lty=c
i <- 50
```

```r
iterations <- 1:1001
plot(Y_gibbs$Steps[,i] ~ iterations, col=8, type="l",ylab="Mean")
abline(h=Y[i])
legend("topright", c("Real value","Estimated value"),col=c(1,8),cex=.7,lty=1)
```