

732A90: Computational Statistics

Computer lab4 - Group11

Sofie Jörgensen, Oriol Garrobé Guilera, David Hrabovszki

19 February 2020

Question 1: Computations with Metropolis-Hastings

In this question we define the density function $f(x)$ as

$$f(x) \propto x^5 e^{-x}, \quad x > 0.$$

This density function is known up to some constant of proportionality, which is 120.

1.

In the first step we use the Metropolis-Hastings algorithm to generate samples from the posterior distribution $f(x)$ by using the log-normal $LN(X_t, 1)$ as the proposal distribution. We take 1 as a starting point, because we expect this value to be within the range of possible values of the distribution defined above.

The time series plot of the obtained chain of samples can be observed in Figure 1. The plot does not show that the chain converges, because it often takes the same value for many iterations and then jumps to a very different one. This implies that the sample is not close to the posterior. There does not seem to be a burn-in period, because samples from many iterations are close to the starting point (which was 1 in this case).

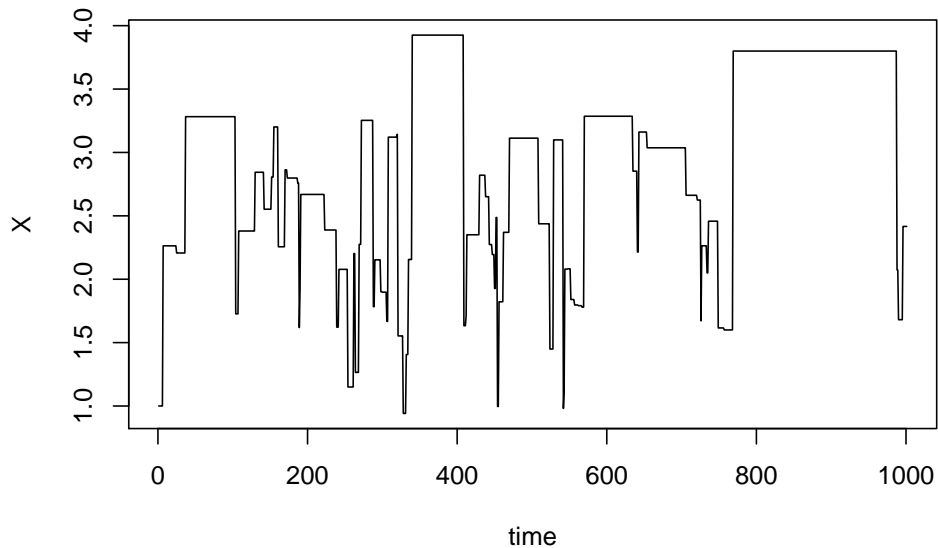


Figure 1: Metropolis-Hastings sampler with log-normal proposal

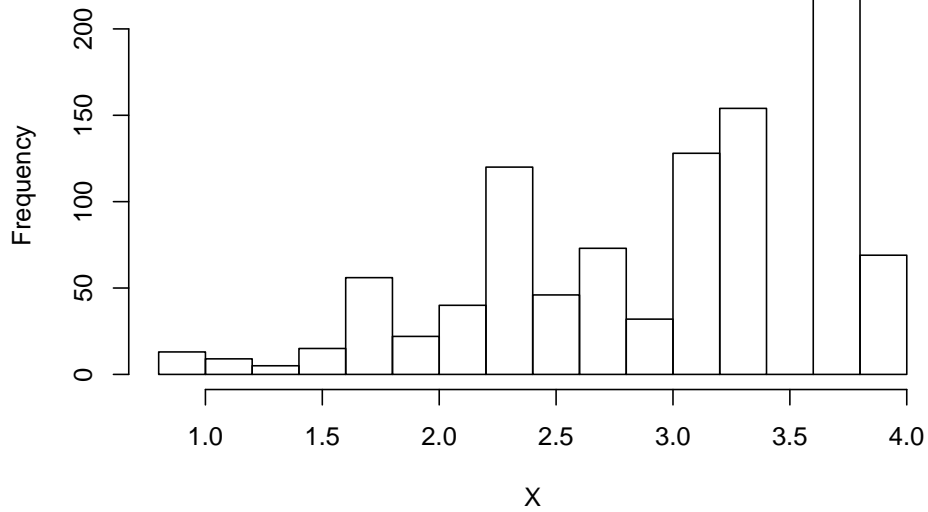


Figure 2: Histogram of generated samples with log-normal proposal (MH sampler)

2.

Now we perform Step 1 using the chi-square distribution $\chi^2(\lfloor X_t + 1 \rfloor)$ as the proposal distribution, where $\lfloor x \rfloor$ is the `floor()` function.

The time series plot is shown in Figure 3, and it can be seen that the chain converges much more than in the previous case. Again there does not seem to be a burn-in period, the starting value of 1 seems to be a probable value of the distribution, so we do not discard any samples from the beginning of the chain.

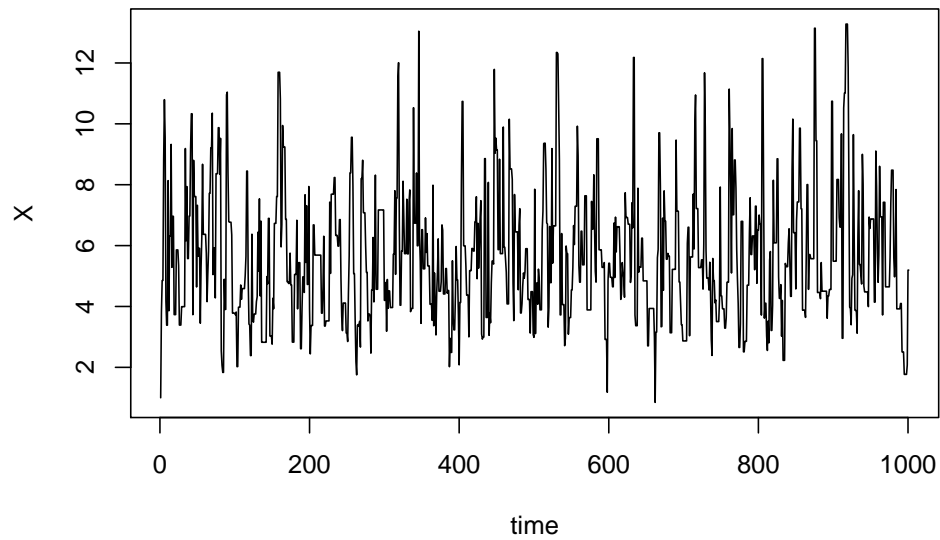


Figure 3: Metropolis-Hastings sampler with chi-square proposal

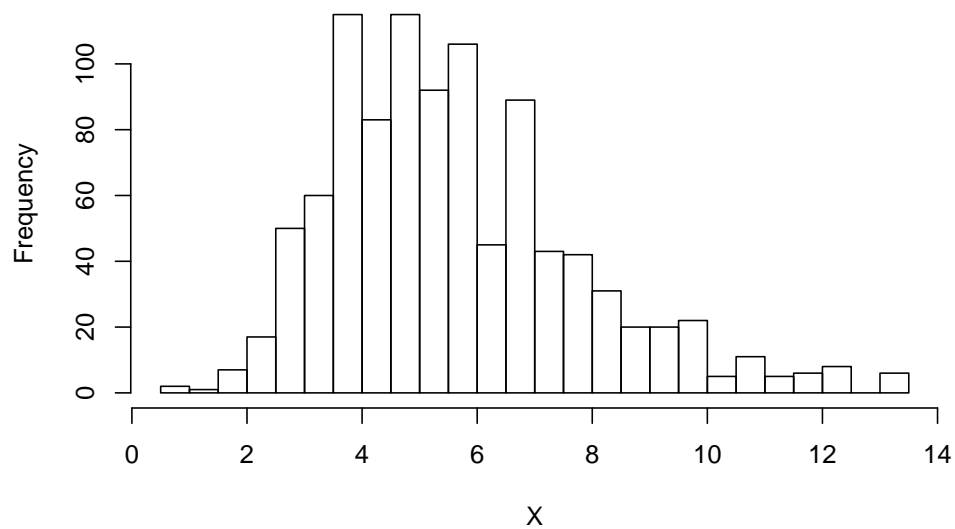


Figure 4: Histogram of generated samples with Chi-square proposal (MH sampler)

3.

We recognise that the probability distribution function $f(x)$ belongs to a certain gamma distribution, therefore we know that $f(x)$ must have a positive skew. This attribute is also true for the distribution of samples from

Step 2 (Figure 4), but untrue for the distribution of samples from Step 1 (Figure 2). Also, the chain with the log-normal proposal does not converge, but the chain with the Chi-square proposal does, so we conclude that the samples generated from Step 2 are a better approximation of samples from the posterior distribution.

4.

Now we generate 10 MCMC sequences using the sample generator from Step 2, and analyze their convergence using the Gelman-Rubin method. The starting points for the 10 sequences are 1, 2, ..., 10.

The output of the Gelman-Rubin's convergence diagnostic for 1000 iterations gives the potential scale reduction factors with the corresponding point estimate and the upper confidence limit. The potential scale reduction factor is based on the comparison of within-chain and between-chain variances. For 1000 iterations, its point estimate is 1.01 and its upper confidence limit is 1.02, which is very close to 1. So according to this diagnostics, approximate convergence has been achieved at 1000 iterations. Even for only 100 iterations, both the point estimate and the upper limit are under 1.1.

5.

In this step we estimate

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

using the samples generated from Steps 1 and 2.

Based on the slides from Lecture 4 we can use MCMC samples from $p(\theta|D)$ to obtain a point estimator by calculating the sample average:

$$\theta^* = \int \theta p(\theta|D) \approx \frac{1}{n} \sum_{i=1}^n \theta_i$$

The sample average of the chain generated in Step 1 is **2.95** and in Step 2 is **5.63**.

If we do not set the seed before the sample generation, then we get highly variable averages from Step 1, but from Step 2 it is almost always between 5.5 and 6.5.

6.

According to Definition 4.5 in Mathematical Statistics with Applications by Wackerly et al., the integral in Step 5 is equal to the expected value of random variable X , where X can take values between 0 and ∞ .

The generated distribution with probability density function $f(x)$ is a gamma distribution, so we can express $f(x)$ in its general form as

$$f(x) = \left[\frac{1}{\Gamma(\alpha)\beta^\alpha} \right] x^{\alpha-1} e^{-x/\beta},$$

where $0 < x < \infty$. Thus $E(X) = \alpha\beta$. In our case $\alpha = 6, \beta = 1$, so $E(X) = 6$.

This is the actual value of the integral in Step 5, so we conclude that our estimation using the samples generated in Step 2 with a Chi-square proposal distribution seems accurate, whereas the estimation from the samples of Step 1 with the log-normal proposal does not seem accurate at all.

Question 2: Gibbs sampling

1.

The aim of this task is to restore the expected concentration values, given the data `chemical.RData`. We import the data into R, containing the concentration of a certain chemical in a water sample, having the following variables:

- **X**: day of the measurment,
- **Y**: measured concentration of the chemical.

The code can be found in the Appendix.

Figure 5 shows the the dependence of **Y** on **X**, where a linear relationship between **Y** and **X** can be observed. Therefore a linear regression with normally distributed residuals seems like a good approach.

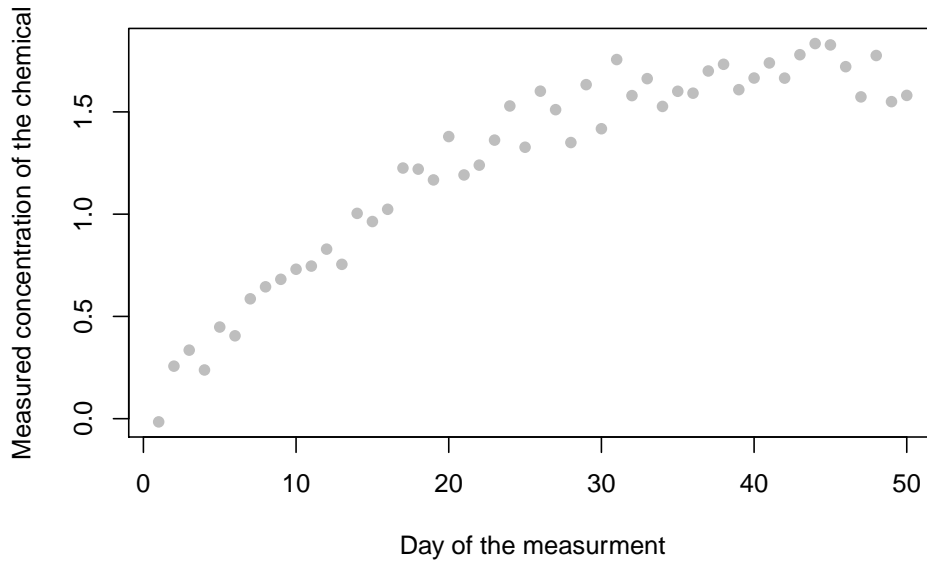


Figure 5: Dependence of measured concentration of the chemical on day of the measurment

2.

Using the random-walk Bayesian model $Y_i \sim N(\mu_i, \sigma^2 = 0.2)$ for $i = 1, \dots, n$, with the prior $p(\mu_1) = 1$, $p(\mu_{i+1}|\mu_i) = N(\mu_i, \sigma^2 = 0.2)$, for $i = 1, \dots, n - 1$. We denote the the number of observations with n and we consider $\vec{\mu} = (\mu_1, \dots, \mu_n)$ as unknown parameters.

Given this information we are able to compute the likelihood $p(\vec{Y}|\vec{\mu})$. Under the assumption that Y_i are iid, the likelihood is given by

$$\begin{aligned}
p(\vec{Y}|\vec{\mu}) &= \prod_{i=1}^n f(Y_i|\mu_i, \sigma^2) \\
&= \prod_{i=1}^n \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(Y_i - \mu_i)^2}{2\sigma^2}\right] \\
&= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left[-\frac{\sum_{i=1}^n (Y_i - \mu_i)^2}{2\sigma^2}\right].
\end{aligned}$$

Further, we will compute the prior by using the chain rule $p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)\dots p(\mu_n|\mu_{n-1})$. Then the prior is given by

$$\begin{aligned}
p(\vec{\mu}) &= p(\mu_1) \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i) \\
&= \prod_{i=2}^n \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(\mu_i - \mu_{i-1})^2}{2\sigma^2}\right] \\
&= \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left[-\frac{\sum_{i=2}^n (\mu_i - \mu_{i-1})^2}{2\sigma^2}\right].
\end{aligned}$$

3.

In order to get the posterior, up to a constant proportionality, we use the Bayes' Theorem. Therefore,

$$\begin{aligned}
p(\vec{\mu}|\vec{Y}) &\propto p(\vec{Y}|\vec{\mu})p(\vec{\mu}) \\
&\propto \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left[-\frac{\sum_{i=1}^n (Y_i - \mu_i)^2}{2\sigma^2}\right] \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^n \exp\left[-\frac{\sum_{i=2}^n (\mu_i - \mu_{i-1})^2}{2\sigma^2}\right] \\
&\propto \exp\left[-\frac{\sum_{i=1}^n (Y_i - \mu_i)^2}{2\sigma^2}\right] \exp\left[-\frac{\sum_{i=2}^n (\mu_i - \mu_{i-1})^2}{2\sigma^2}\right] \\
&\propto \exp\left[-\frac{\sum_{i=1}^n (Y_i - \mu_i)^2 + \sum_{i=2}^n (\mu_i - \mu_{i-1})^2}{2\sigma^2}\right].
\end{aligned}$$

And finally, once we have the posterior, we can compute the the probability distribution $p(\mu_i|\vec{\mu}_{-i}, \vec{Y})$, where $\vec{\mu}_{-i}$ is a vector containing all μ values except for μ_i . We also know that we can discard all the elements that do not contain $\vec{\mu}_i$, as we are looking for a posterior up to a constant proportionality. Given Hint A, we will consider separate formulas for $p(\mu_1|\vec{\mu}_{-1}, \vec{Y})$ and for $p(\mu_n|\vec{\mu}_{-n}, \vec{Y})$. By using Hint B, we obtain

$$p(\mu_1|\vec{\mu}_{-1}, \vec{Y}) \propto \exp\left[-\frac{(\mu_1 - (Y_1 + \mu_2)/2)^2}{\sigma^2}\right], \quad \text{for } i = 1,$$

and

$$p(\mu_n|\vec{\mu}_{-n}, \vec{Y}) \propto \exp\left[-\frac{(\mu_n - (Y_n + \mu_{n-1})/2)^2}{\sigma^2}\right], \quad \text{for } i = n.$$

By using Hint C, we get to the result

$$p(\mu_i | \vec{\mu}_{-i}, \vec{Y}) \propto \exp \left[-\frac{(\mu_i - (Y_i + \mu_{i-1} + \mu_{i+1})/3)^2}{2\sigma^2/3} \right], \quad \text{for } i = 2, \dots, n-1.$$

4.

We implement the Gibbs samples using the distributions found and initial value $\vec{\mu}^0 = (0, \dots, 0)$. This code snippet (see Appendix) computes n values of $\vec{\mu}$ and computes the expected value of $\vec{\mu}$ using a Monte Carlo approach. In this particular case $n = 1000$.

The expected value of $\vec{\mu}$ is:

In 6 there is a plot of the dependence of Y on X and the comparison with the expected values of mu.

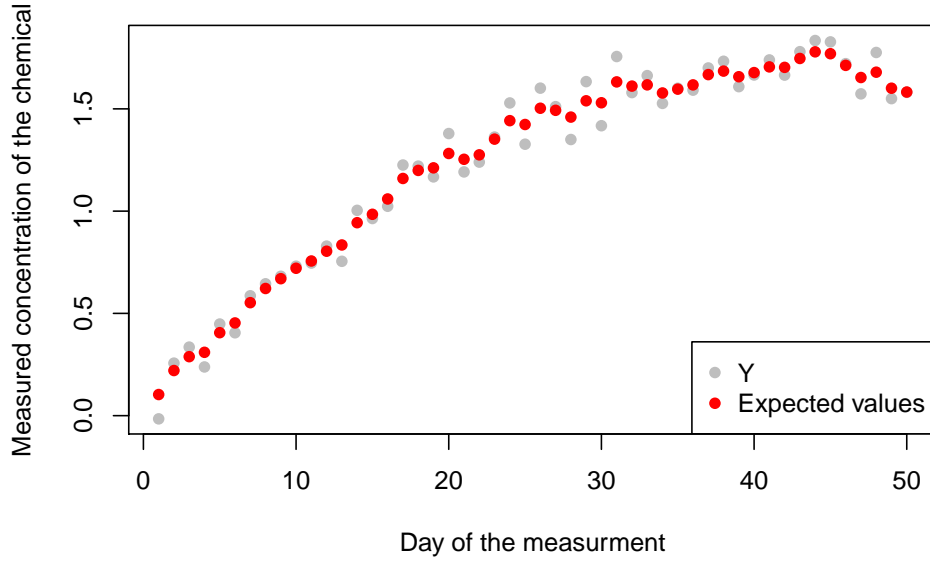


Figure 6: Comparison between Y and expected value of mu

The result is considered as positive as observing the plot it is possible to see that the expected values of μ follow the same pattern that Y in terms of dependence on X. Also, the expected values seem to have removed the noise produced by the accuracy of the instrumentat used to measure, as the expected values of mu are less spread than those of Y.

5.

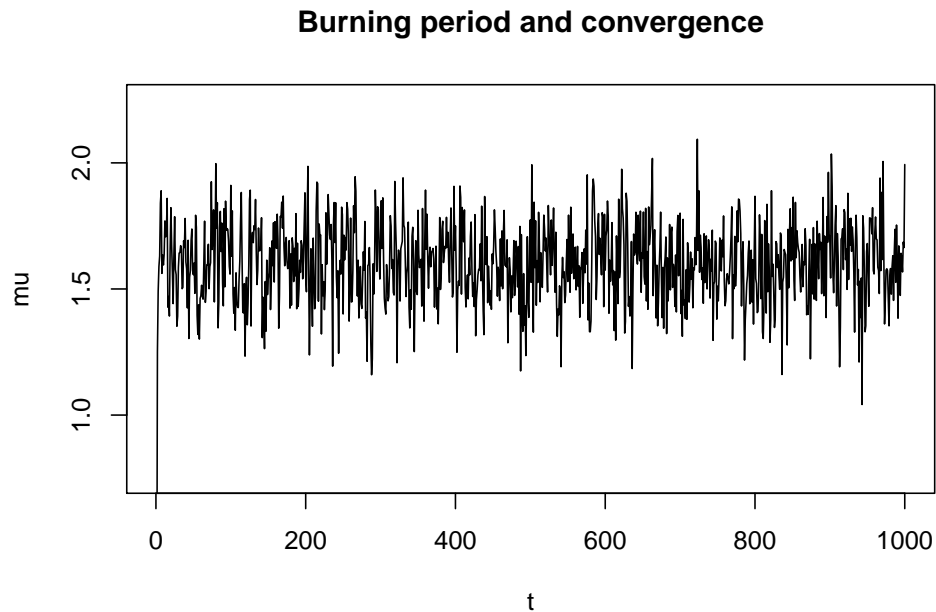


Figure 7: Burn in period and convergence

In order to verify that the MCMC explores the parameter space efficiently, we want to assure that the chain does not stay in the same state for too long or has too many consecutive steps in one direction. In this case, the traceplot behaves accordingly.

Burning in is a method that erases iterations at the beginning of a Markov Chain Monte Carlo (MCMC) run, it is a way to select a better starting point than just one by random. It can be seen from the plot that there is a burn in period of 1 iteration, after that one the value of μ moves between 1.25 and 1.75.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
# R version
RNGversion('3.5.1')
#libraries
library(coda)

# 1.1.
posterior <- function(x){
  x^5 * exp(-x)
}

# Seed
set.seed(1234567890)

posterior_sampler_ln <- function(tmax=1000, starting_point = 1) {
```



```

X=rep(starting_point,tmax)
for (t in 1:tmax) {
  Y = rlnorm(1,meanlog = X[t],sdlog = 1)
  U = runif(1,min = 0, max = 1)
  alpha = min(1,
              (posterior(Y) * dlnorm(X[t],meanlog = Y, sdlog = 1)) /
              (posterior(X[t]) * dlnorm(Y, meanlog = X[t], sdlog = 1)))
  if (U < alpha) {
    X[t+1] = Y
  } else {
    X[t+1] = X[t]
  }
  t = t+1
}
return(X)
}
tmax=1000
# Seed
set.seed(1234567890)
ln_sample = posterior_sampler_ln(tmax = tmax, starting_point = 1)
plot(1:(tmax+1),ln_sample, type = 'l', xlab = "time", ylab = "X")

hist(ln_sample, breaks = 20, main = "", xlab = "X")

# 1.2
#Set seed
set.seed(1234567890)

posterior_sampler_chisq <- function(tmax=1000, starting_point = 1) {
  X=rep(starting_point,tmax)
  for (t in 1:tmax) {
    Y = rchisq(1,df = floor(X[t]+1))
    U = runif(1,min = 0, max = 1)
    alpha = min(1,
                (posterior(Y) * dchisq(X[t],df = floor(Y+1))) /
                (posterior(X[t]) * dchisq(Y, df = floor(X[t]+1))))
    if (U < alpha) {
      X[t+1] = Y
    } else {
      X[t+1] = X[t]
    }
    t = t+1
  }
  return(X)
}
tmax=1000
# Seed
set.seed(1234567890)
chi_sample = posterior_sampler_chisq(tmax = tmax, starting_point = 1)
plot(1:(tmax+1), chi_sample, type = 'l', xlab = "time", ylab = "X")
hist(chi_sample, breaks = 20, main = "", xlab = "X")

#1.4

```

```

starting_points = seq(1,10)
MCMC_seqs = mcmc.list()
tmax=1000
# Seed
set.seed(1234567890)
for (k in starting_points) {
  MCMC_seqs[[k]] = as.mcmc(posterior_sampler_chisq(tmax = tmax, starting_point = k))
}

print(gelman.diag(MCMC_seqs))

# 1.5
# Seed
set.seed(1234567890)
#using Step1
#mean(ln_sample) #large variance; 2.949404

#using Step2
#mean(chi_sample) #small variance; 5.636084

##### Question 2: Gibbs sampling #####

load(file = "chemical.RData")
# 2.1

plot(X,Y, xlab = "Day of the measurment"
      , ylab = "Measured concentration of the chemical", col = "grey", pch = 16)
# 2.4

gibbs_sampler <- function(Tmax) {
  d <- length(Y)
  sigma <- 0.2
  mu <- matrix(0, nrow = Tmax, ncol = d)
  t <- 1
  while (t<Tmax) {
    for (i in 1:d) {
      if (i==1) {
        mu[t+1,i] <- rnorm(1, (Y[i]+mu[t,i+1])/2, sigma/sqrt(2))
      }
      else if (i==d) {
        mu[t+1,i] <- rnorm(1,(Y[i] + mu[t+1,i-1])/2, sigma/sqrt(2))
      }
      else {
        mu[t+1,i] <- rnorm(1, (Y[i]+mu[t+1,i-1]+mu[t,i+1])/3, sigma/sqrt(3))
      }
    }
    t=t+1
  }
  mu
}

Y_gibbs <- colSums(gibbs_sampler(1000))/1000

```

```

plot(X,Y, xlab = "Day of the measurment"
      , ylab = "Measured concentration of the chemical", col = "grey", pch = 16)
points(Y_gibbs, col = "red", pch = 16)
legend("bottomright", legend=c("Y", "Expected values"), col=c("grey", "red"), pch = 16)

mu_n <- gibbs_sampler(1000)[,length(Y)]
plot(mu_n, type="l", ylim = c(0.75, 2.25), main="Burning period and convergence",
      ylab="mu", xlab="t")

```