Computer Lab 4

Thomas Zhang,Andrea Bruzzone

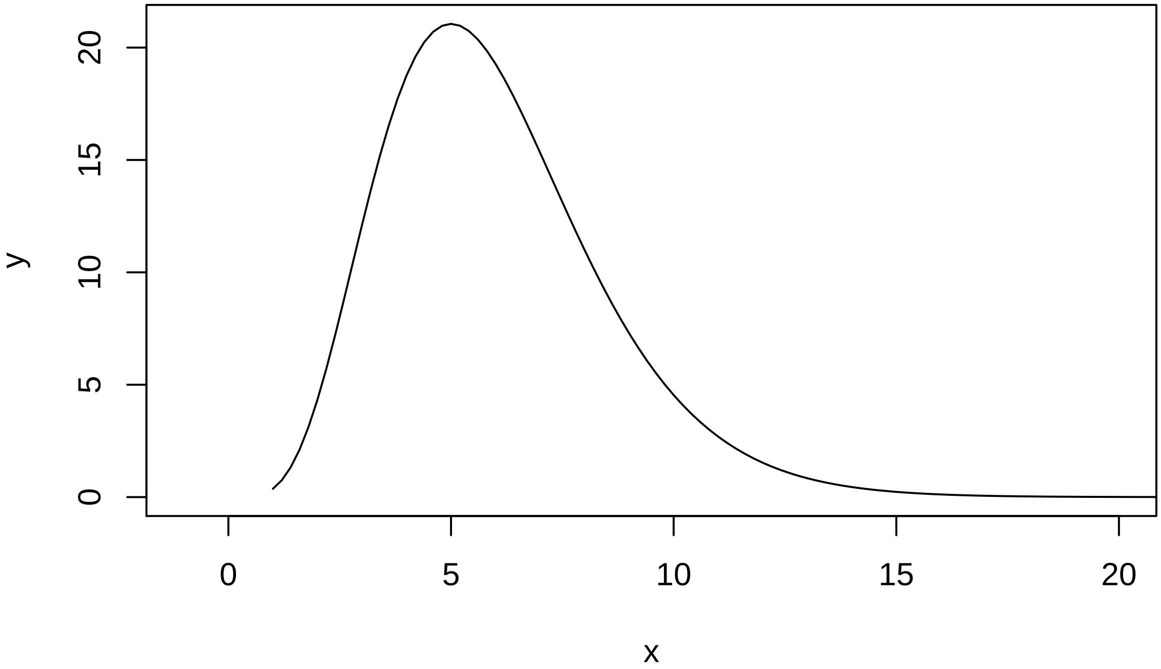
2016 M02 29

Assignment 1

We have this probability distribution from which we want to generate from:

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

This distribution looks like:

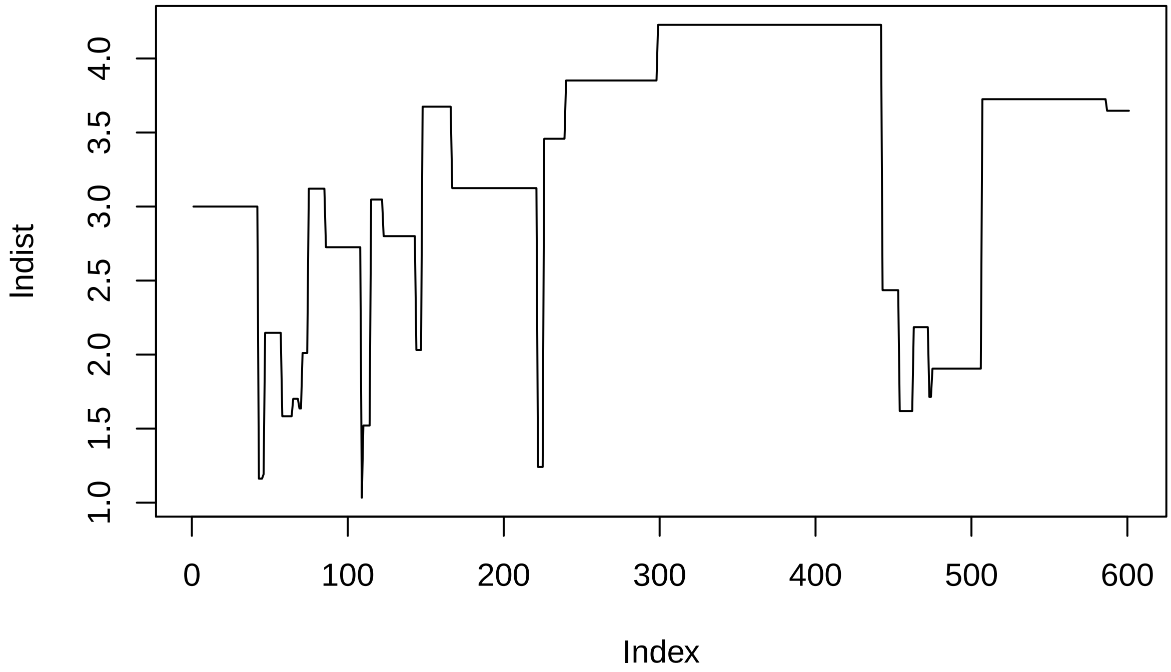


- 1.1

Using Metropolis-hastings algorithm we try to generate 600 samples from the previous distribution by using proposal distribution as log-normal LN$(X_t, 1)$, that is:

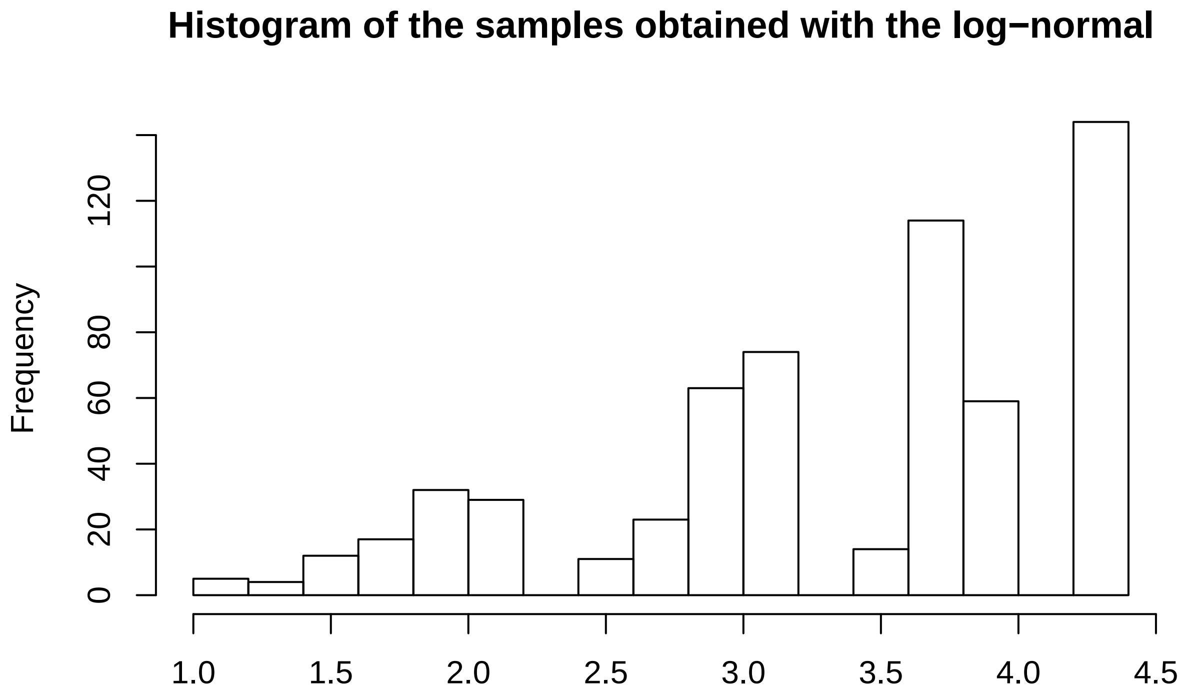
$$q(.|x_t) = \frac{e^{-\frac{(\log x-\mu)^2}{2}}}{x\sqrt{2\pi}}$$

We run the algorithm using as starting point 3 and plot the chain obtained.



We see that the traceplot show that the algorithm is reluctant to jump to a new value whenever X_t is a high value, suggesting that the value of α is then too low. Our start point here was $X_0 = 3$, but if a higher start value is chosen the algorithm never jumps. I do not think we can talk of burn-in period or convergence in this case, since the trace plot, when it occasionally jumps, jumps in a too random fashion.

The histogram of the sample obtained:

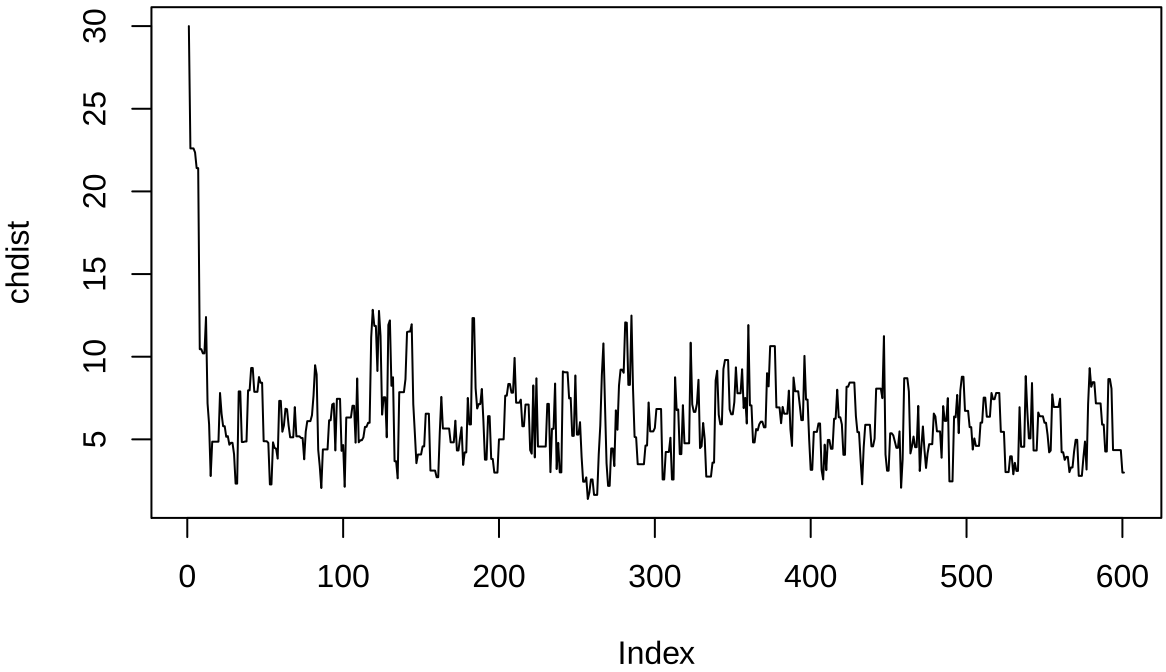


It can be seen that the histogram is very different from the pattern of the target distribution.

- 1.2

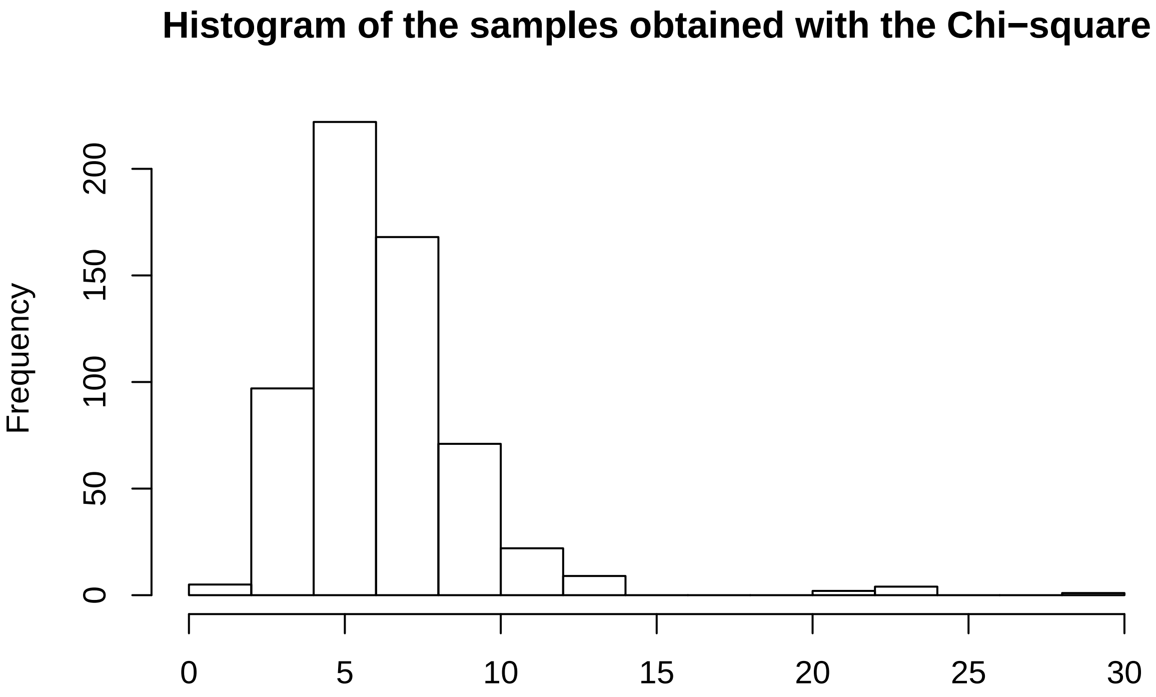
The same algorithm as before is used, this time with proposal distribution $\chi^2(\text{floor}(X_t + 1))$. The starting point is choosen to be 30 and the number of samples is 600.

The obtained chain is:



We see that in this case the traceplot never gets stuck anywhere for long, and furthermore that the algorithm sometimes spikes into high values. This is consistent with exploring all the values allowed by the chi-squared distribution. While it cannot be seen in the plot above, the burn-in period is around 30 iterations regardless of starting value.

The histogram of the sample obtained:



It can be seen that the histogram has a pattern really close to the one of our target function, and we seem to have convergence to a value around 6, according to the histogram.

- 1.3

Looking at the chains and at the histograms of the samples it can be said that the chi-squared proposal distribution is the more suitable one for us to use in the M-H algorithm, mainly because it does not get stuck at high values of X_t.

- 1.4

Using the generator done in step 2, we generate 10 MCMC sequences with starting points 1,2,..,10 and then we use Gelman-Rubin method to analyze convergence of these sequences.

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.01       1.02
```

It is known that, we can assume the convergence achieved, if the upper confidence limit of the potential scale reduction factor (Upper C.I in the output) is close to 1. This is our case where number of samples is 600 in each sequence, so it can be said that the convergence is achieved.(We could also make it go closer to one by increasing number of samples in each sequence).

- 1.5

Now to estimate the integral $\int_0^\infty xf(x)dx$, which is the expression for the expected value of the probability distribution $f(x)$, we simply take the average of the output produced by the M-H algorithm with lognormal proposal distribution and chi-squared proposal distribution.

With samples from the log-normal:

```
## [1] 3.314
```

With samples from the Chi-Square:

```
## [1] 6.191
```

- 1.6

The distribution generated is a gamma distribution:

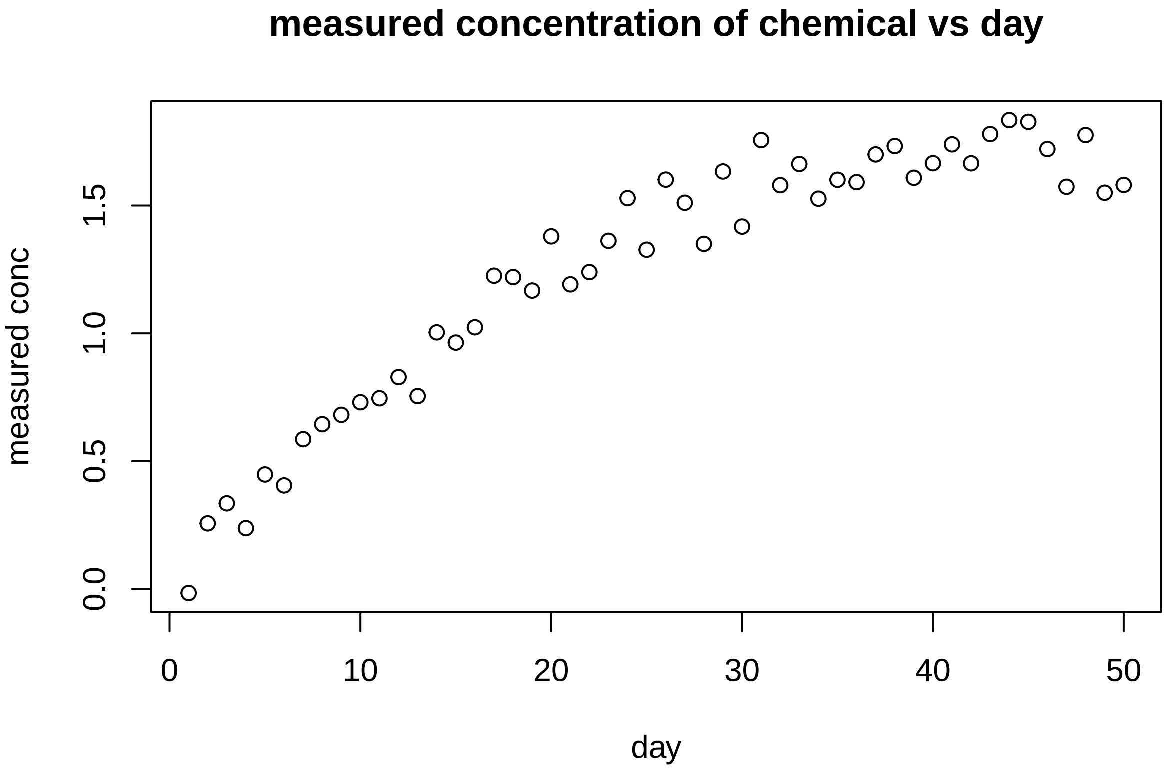
$$\frac{1}{\Gamma(k)\theta^k}x^{k-1}e^{-\frac{x}{\theta}}$$

In particular with the integral we are computing the mean, the mean for a gamma is equal to $k\theta$, in this case $\theta = 1$ and $k = 6$, so the actual value of the integral is six.

Again the Chi-Square is better then the log-normal distribution, since with this distribution the value is always around six, instead for the log-normal it varies every time and usually it is not close to the real value.

Assignment 2

We plot the chemical concentration data in `chemical.RData`.



It can be seen that the overall development of the chemical concentration is increasing over time and then reaches some equilibrium level. Maybe some logistic function would be a good model here.

We are given a Bayesian model:

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

where the prior is

$$p(\mu_1) = 1$$

$$\mu_{i+1} \sim N(\mu_i, 0.2) \text{ i= 1,...,n-1}$$

Now, for the vectors $\boldsymbol{Y}$ and $\boldsymbol{\mu}$ we find that

The likelihood is :

$$P(\boldsymbol{Y}|\boldsymbol{\mu}) = (2\pi \times 0.2)^{-\frac{n}{2}} e^{(-\frac{\sum_{i=1}^{n}(y_i-\mu_i)^2}{2\times 0.2})}$$

and the prior is:

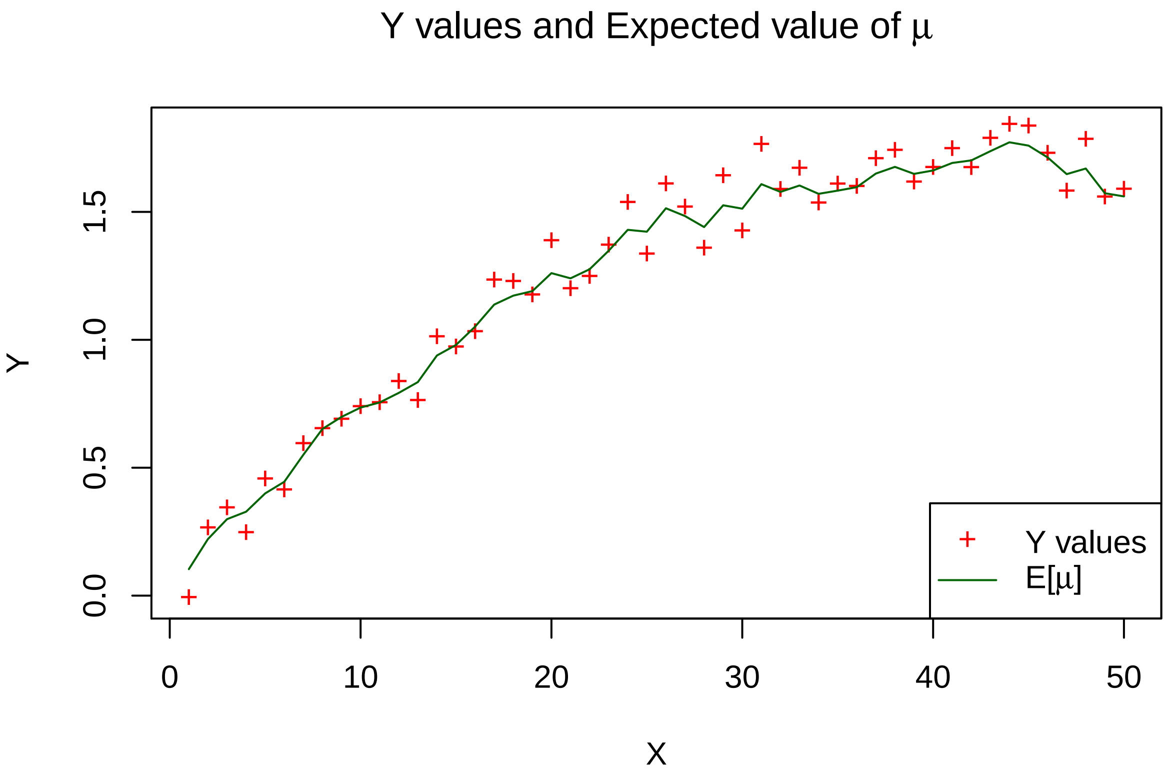
$$P(\boldsymbol{\mu}) = \prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi \times 0.2)}} e^{-\frac{(\mu_{i+1}-\mu_i)}{2\times 0.2}}$$

According to the Bayes theorem, the product between the likelihood and the prior gives the posterior up to a constant of proportionality.

To find the distribution of $\mu_i|\boldsymbol{\mu_{-i}}, \boldsymbol{Y}$ where $\boldsymbol{\mu_{-i}}$ is a vector containing all μ values except for μ_i, we calculate the posterior distribution $P(\mu_i|\boldsymbol{\mu_{-i}}, \boldsymbol{Y})$. We discover there are three different cases and we derive their (as it turns out) normal distributions.

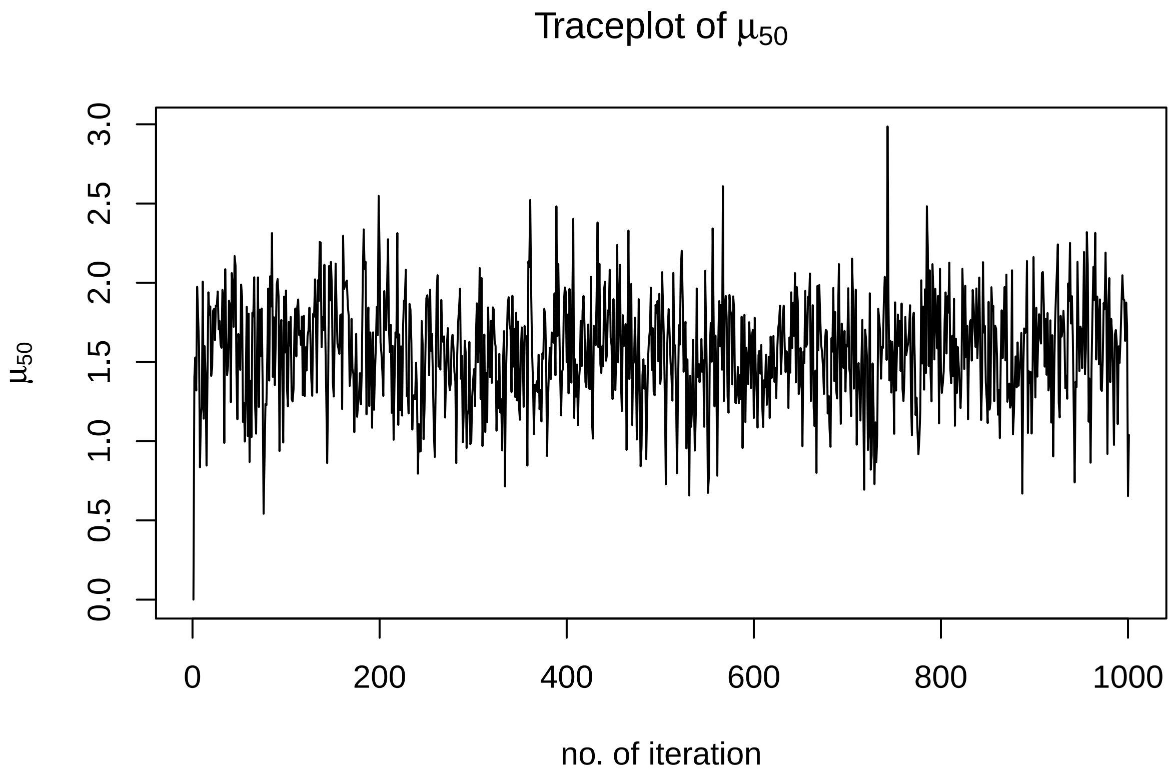
1. First observation: $\mu_1 \sim N(\frac{Y_i+\mu_2}{2}, \frac{\sigma^2}{2})$
2. Observations 2 to 49: $\mu_k \sim N(\frac{Y_k+\mu_{k-1}+\mu_{k+1}}{3}, \frac{\sigma^2}{3})$
3. Last observation: $\mu_{50} \sim N(\frac{Y_{50}+\mu_{49}}{2}, \frac{\sigma^2}{2})$

We now run a gibbs sampler to find 1000 values of vector $\boldsymbol{\mu}$, take the expected value of $\boldsymbol{\mu}$ to be the average value in every point and plot the resulting line together with the original data.



It seems that the expected value can catch the true underlying dependence between Y and X. Moreover, it seems that we have managed to remove a little bit the noise.

Finally we plot the traceplot of μ_{50} and observe its burn-in period and convergence properties.



It appears the burn-in is very fast (almost instant) and the value stays within a radius of $|\mu_{50} - 1.5| < 1$ for the most part, so we have relatively good convergence.

Contributions

Both parties contributed to this lab report. The code for assignment 1 comes from Andrea while the code for assignment 2 comes from Thomas. We both discussed the issues and corrected each other and wrote different segments of the text.

Appendix

R code

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

g <- seq(1, 100, 0.2)
plot(g,target(g), type = "l", ylab = "y", xlab = "x",xlim=c(-1,20))
library(coda)
x <- rep(0, 600)
```

```
mhlnor <- function(start){
  x[1] <- start
for(i in 1:length(x)){
  Y <- rlnorm(1, meanlog = x[i], sdlog = 1)
  u <- runif(1,0,1)
  c <- (target(Y)*dlnorm(x[i], Y, 1)) / (target(x[i])*dlnorm(Y, x[i], 1))
  alpha <- min(1, c)
  if(u <= alpha){
    x[i + 1] <- Y
  }else{
    x[i + 1] <- x[i]
  }
}
  return(x)
}

lndist <- mhlnor(3)
plot(lndist, type = "l")
hist(lndist, main = "Histogram of the samples obtained with the log-normal", xlab = "",breaks=20)
mhchi <- function(start){
  x[1] <- start
for(i in 1:length(x)){
  Y <- rchisq(1, floor(x[i] + 1))
  u <- runif(1,0,1)
  c <- (target(Y)*dchisq(x[i], floor(Y + 1))) / (target(x[i])*dchisq(Y, floor(x[i] + 1)))
  alpha <- min(1, c)
  if(u <= alpha){
    x[i + 1] <- Y
  }else{
    x[i + 1] <- x[i]
  }
}
  return(x)
}

chdist <- mhchi(30)
plot(chdist, type = "l")
hist(chdist, main = "Histogram of the samples obtained with the Chi-square", xlab = "")
startpoints <- seq(1, 10, 1)
X <- data.frame(mhchi(3))
for(i in 1:10){
  X[,i] <- mhchi(startpoints[i])
}


library(coda)
f=mcmc.list()
for (i in 1:10) f[[i]]=as.mcmc(X[,i])
gelman.diag(f)
round(sum(lndist) / length(lndist),3)
round(sum(chdist) / length(chdist),3)
load("chemical.RData")
plot(X,Y,main="measured concentration of chemical vs day",xlab="day",ylab="measured conc")
```

```
mu0 <- rep(0,length(Y))

gibbs_sampler <- function(start,length = 1000){
  n <- length(start)
  result <- matrix(data = 0, nrow = (length + 1), ncol = n)
  result[1,] <- start
  turn <- 2
  repeat{
    if(turn > (length + 1)){
      break
    }
    result[turn,1] <- rnorm(1, mean = (Y[1] + result[(turn - 1),2]) / 2, sd = sqrt(0.2 / 2))
    for(j in 2:(n - 1)){
      result[turn,j] <- rnorm(1,
          mean = ( Y[j] + result[turn,(j - 1)] + result[(turn - 1),(j + 1)]) / 3,
          sd = sqrt(0.2 / 3))
    }
    result[turn,n] <- rnorm(1, mean = (Y[n] + result[turn,(n - 1)]) / 2, sd = sqrt(0.2 / 2))
    turn <- turn + 1
  }
  return(result)
}
expctedvalu <- function(matr){
  output <- rep(0,ncol(matr))
  for(j in 1:ncol(matr)){
    output[j] <- mean(matr[,j])
  }
  return(output)
}
res <- gibbs_sampler(start = mu0)

res2 <- expctedvalu(res)
plot(X,Y,pch="+",col = "red",main= expression(paste("Y values and Expected value of ", mu)))
lines(X,res2,col="darkgreen")
legend("bottomright",c("Y values",expression(paste("E[",mu,"]"))),pch = c("+",""),lty = c(NA,1),
       col = c("red","darkgreen"))
plot(1:1001,res[,50],type="l",main=expression(paste("Traceplot of ",mu[50])),
     xlab = "no. of iteration",ylab = expression(mu[50]))
## NA
```