Lab 4 Report

Andrea Bruzzone

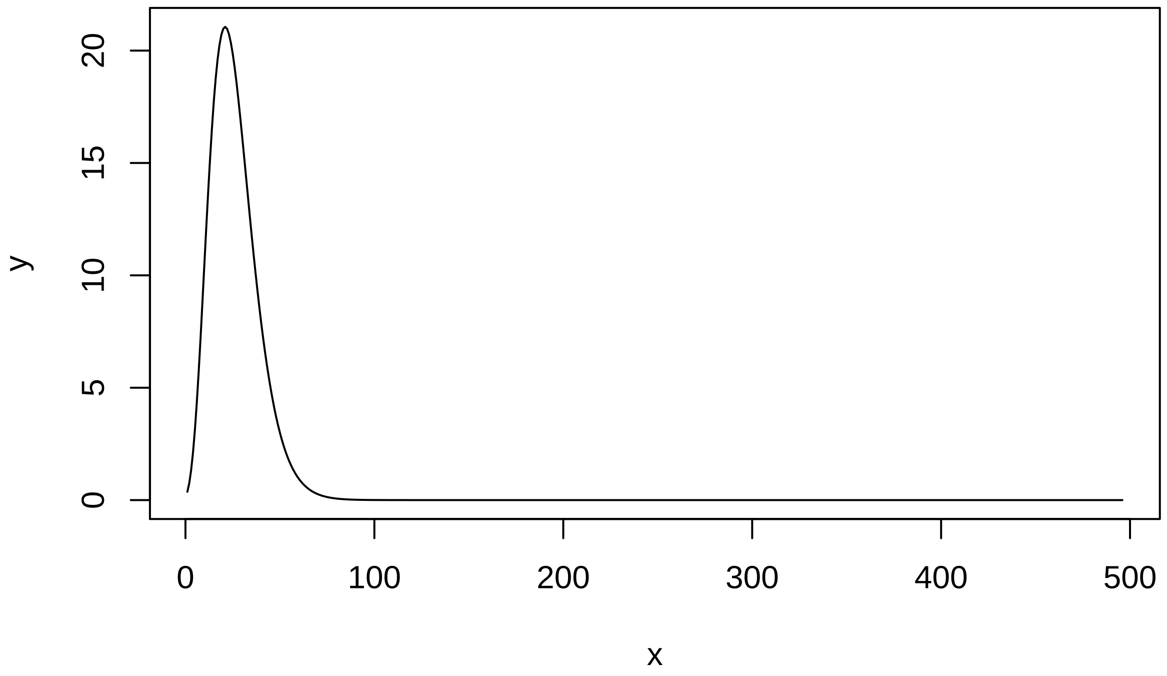
February 24, 2016

Assignment 1

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

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

This distribution looks like:

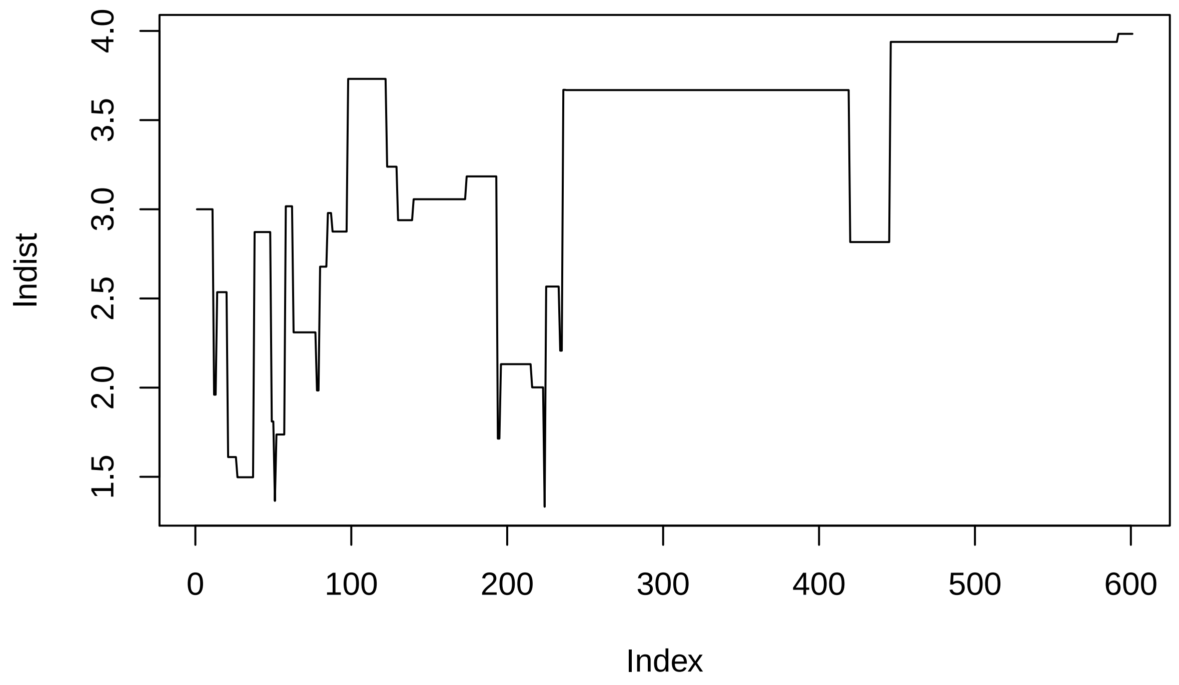


- 1.1

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:

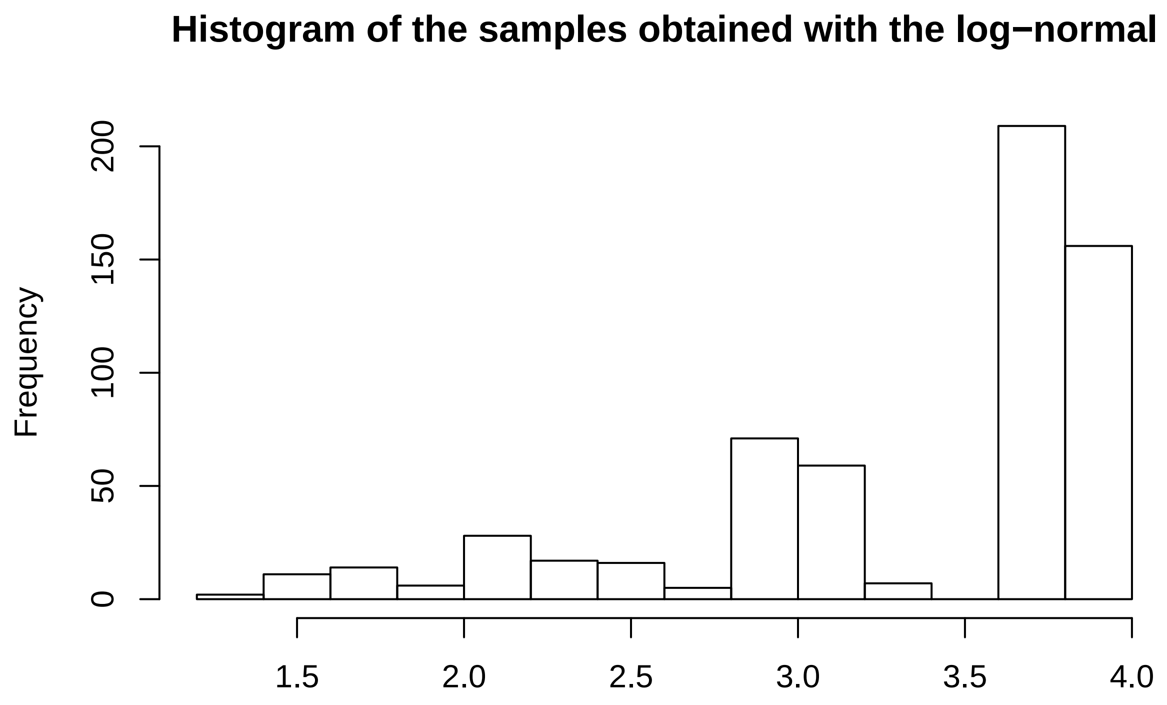
$$q(.|x_t) = \frac{e^{-\frac{(\log x - \mu)^2}{2}}}{x\sqrt{2\pi}}$$

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



The chain does not really seem to converge, for this reason there is not a burn-in period.

The histogram of the sample obtained:

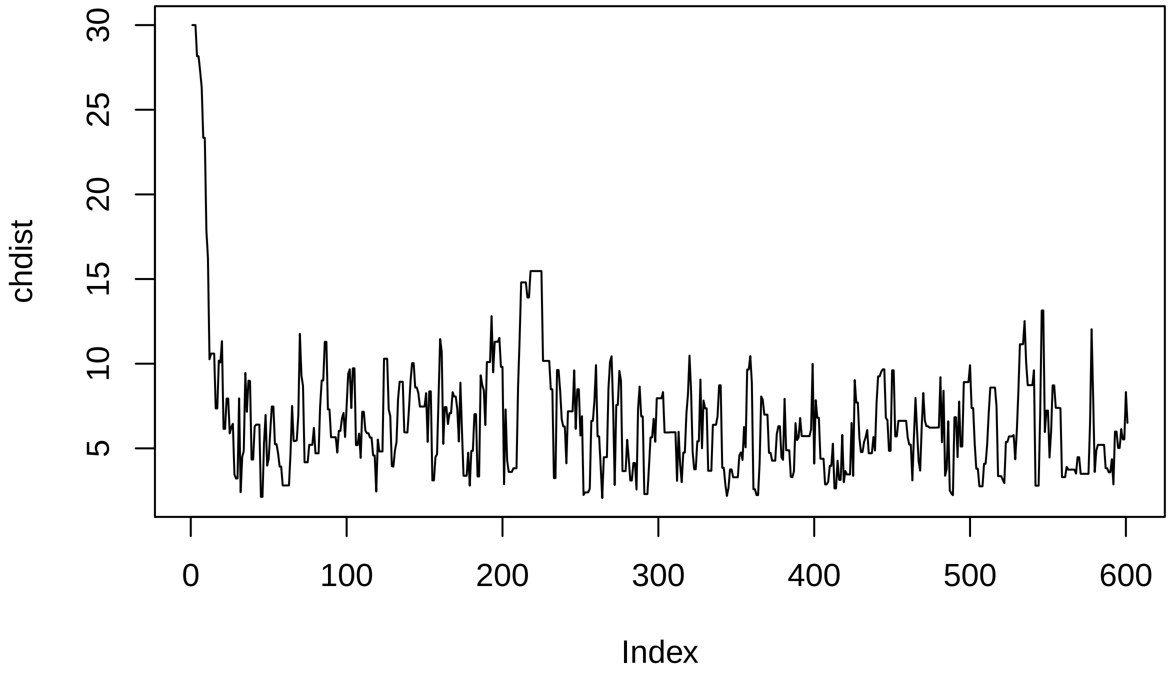


It can be seen that the histogram is very different from the pattern of the target distribution.

- 1.2

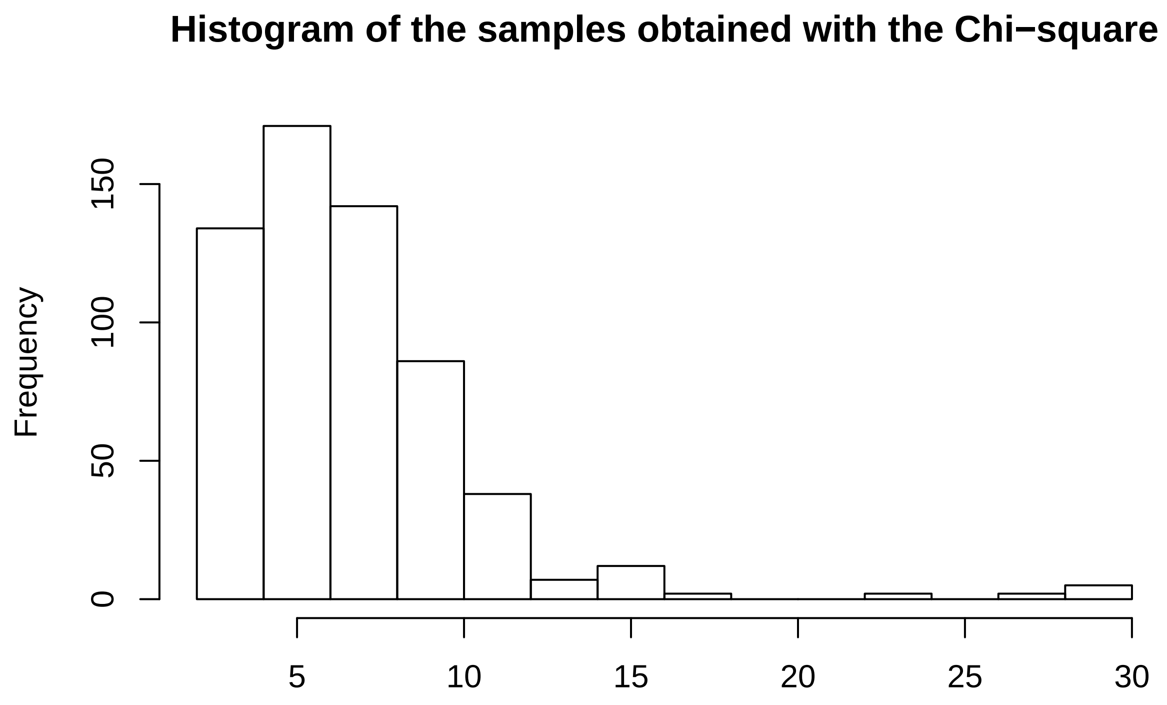
The same algorithm as before is used, this time with proposal distribution $\chi^2(\text{floor}(X_t + 1))$. The starting point is choosen to be 30 and the number of samples is 600.

The obtained chain is:



The chain seems to converge quite fast, the size of the burn-in period seems to be equal to the value of the starting point. This is due to the fact that the starting value is quite an unusual value for such this distribution.

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.

- 1.3

Looking at the chains and at the histograms of the samples it can be said that the best proposal distribution for this case is the Chi-Square, the log-normal is not a good distribution in this case.

- 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.

```
## Warning: package 'coda' was built under R version 3.2.3
```

```
## 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 factor (Upper C.I in the output) is close to 1. This is our case so it can be said that the convergence is achieved.

- 1.5

We want to estimate $\int_0^\infty xf(x)dx$ using samples from steps 1 and 2. Using MC for inference, to estimate an integral like $\int_D f(x)dx$ we have to:

1. Decompose the function $\theta = f(x) = g(x)p(x)$
2. Simulate sample $x_1, .., x_m$ from $p(x)$
3. Estimate the integral as: $\hat{\theta} = \frac{\sum g(x_i)}{m}$

In this case $f(x)$ is already divided as: $g(x) = x$ and $p(x) = f(x)$, so we have just to do the mean of the samples found in steps 1 and 2.

With samples from the log-normal:

```
## [1] 3.327059
```

With samples from the Chi-Square:

```
## [1] 6.690233
```

- 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 6.

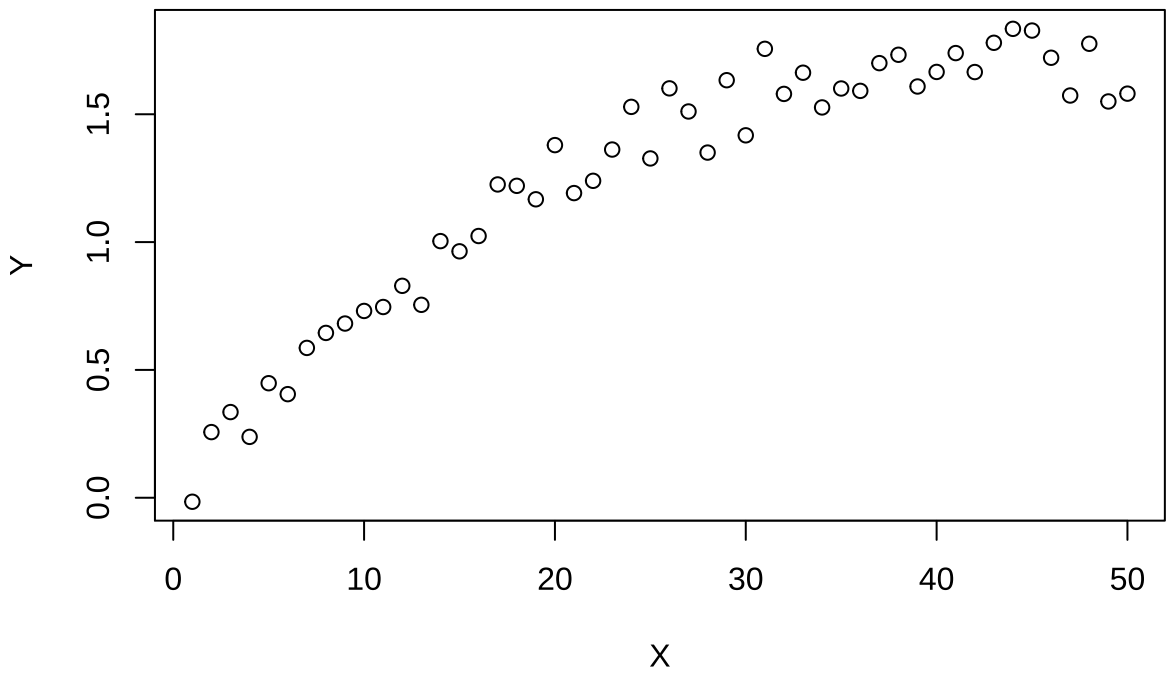
Again the Chi-Square is better then the log-normal distribution, since with this distribution the value is always close to 6, instead for the log-normal it varies every time and usually it is not close to the real value.

Assignment 2

The data set contains two variables: X is day of measurement done in a water sample and Y is the measured concentration of the chemical.

- 2.1

We plot the dependence of Y on X:



It seems that the quadratic model can be a reasonable model to fit the data.

- 2.2

We have this 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 :

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

and the prior is:

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

- 2.3

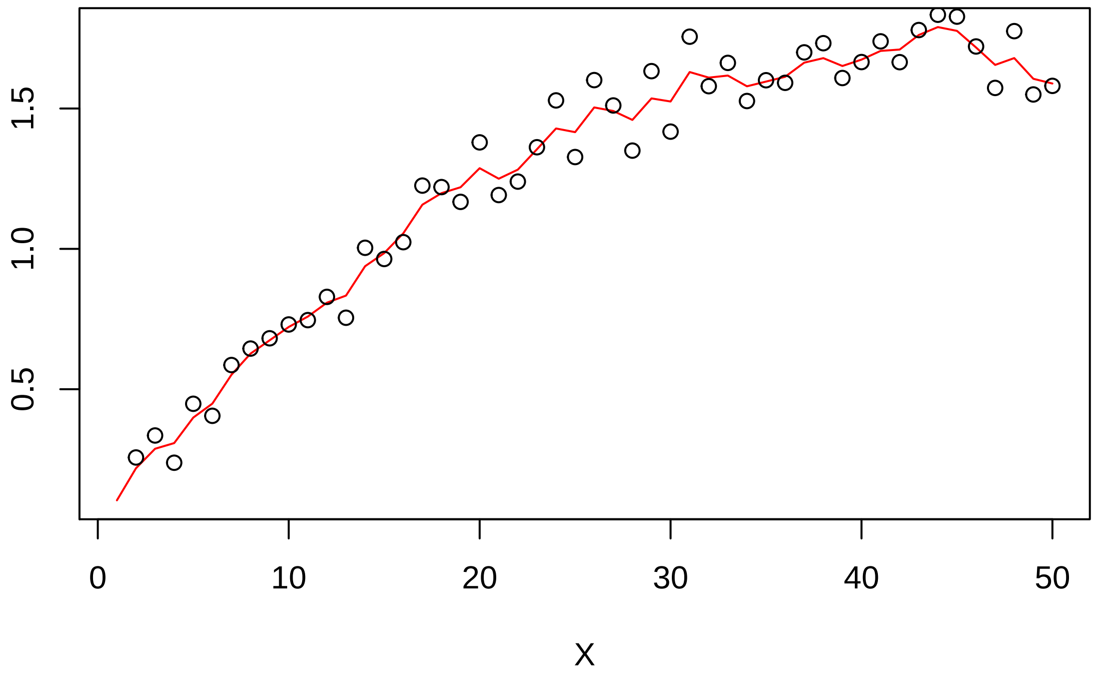
For the Bayes theorem, the product between the likelihood and the prior gives the posterior up to a costant of proportionality.

To find the distribution of $\mu_i|\mu_{-i}, Y$ where μ_{-i} is a vector containing all μ values except for μ_i, we use the posterior find before and we differentiate three different case and derive the distributions doing some calculus that involved just the μ choosen, we get:

1. Firs 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: $\mu_{50} \sim N(\frac{Y_{50}+\mu_{49}}{2}, \frac{\sigma^2}{2})$

- 2.4

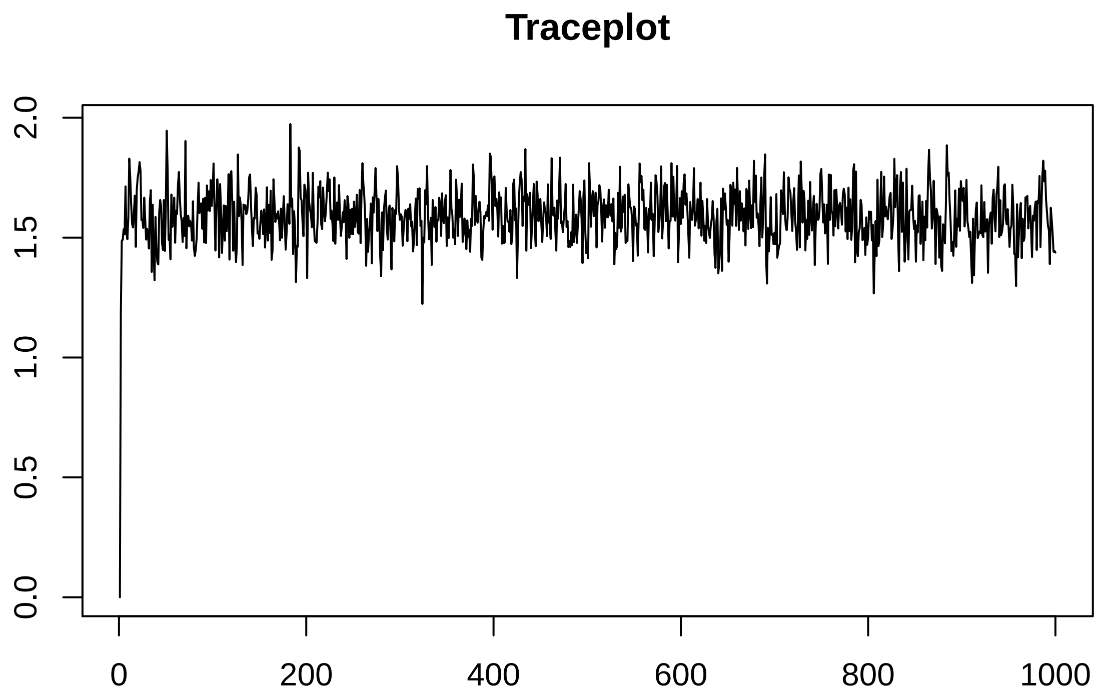
Using the distribution found before and as starting points $\mu^0 = (0, 0, .., 0)$ we obtain 1000 values of μ. Then, we compute the expected value for μ by using Monte Carlo approach and plot it versus X(in red) together with the plot of Y versus X:



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.

- 2.5

The traceplot for μ_{50} is:



The plot shows that we have convergence and the burn-in period is very small, looking at the traceplot it can be said to be around 10.

Code

```
##Assignment 1
#1.1
#probability density function of target
target <- function(x){
  result <- x^5 * exp(-x)
  return(result)
}

# q <- function(x, mu){
# out <- exp(-(log(x) - mu)^2 / 2) / x*sqrt(2*pi)
# return(out)
# }

g <- seq(1, 100, 0.2)
plot(target(g), type = "l", ylab = "y", xlab = "x")
```

```r
x <- rep(0, 600)

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

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 = "")


#1.2
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)
```

```r
plot(chdist, type = "l")
hist(chdist, main = "Histogram of the samples obtained with the Chi-square", xlab = "")


#1.4
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)

#1.5
sum(lndist) / length(lndist)

sum(chdist) / length(chdist)


##Assignment 2
#2.1
data <- load("chemical.RData")
plot(X, Y)

#2.4
gibbs <- function(s, n){
  u <- matrix(s, ncol = length(s), nrow = n)
  for (i in 2:n){
    for(j in 1:ncol(u)){
    if(j == 1){
      u[i,j] <- rnorm(1, ((Y[j] + u[i-1, j + 1]) / 2), (0.2 /2))
    }else if(j > 1 & j < 50){
      u[i,j] <- rnorm(1, ((Y[j] + u[i, j - 1] + u[i - 1, j + 1]) / 3), (0.2 / 3))
    }else{
      u[i,j] <- rnorm(1, ((Y[j] + u[i, j - 1]) / 2), (0.2 / 2))
    }
    }
  }
  return(u)
}

mu0 <- rep(0, 50)
gibsample <- gibbs(mu0, 1000)

MCmean <- c()
for(i in 1:ncol(gibsample)){
  MCmean[i] <- mean(gibsample[ ,i])
}
```

```
plot(X, MCmean, type="l", col = "red", ylab ="")
points(X, Y)



#2.5
plot(gibsample[ ,50], type = "l", xlab="", ylab="", main = "Traceplot")
```