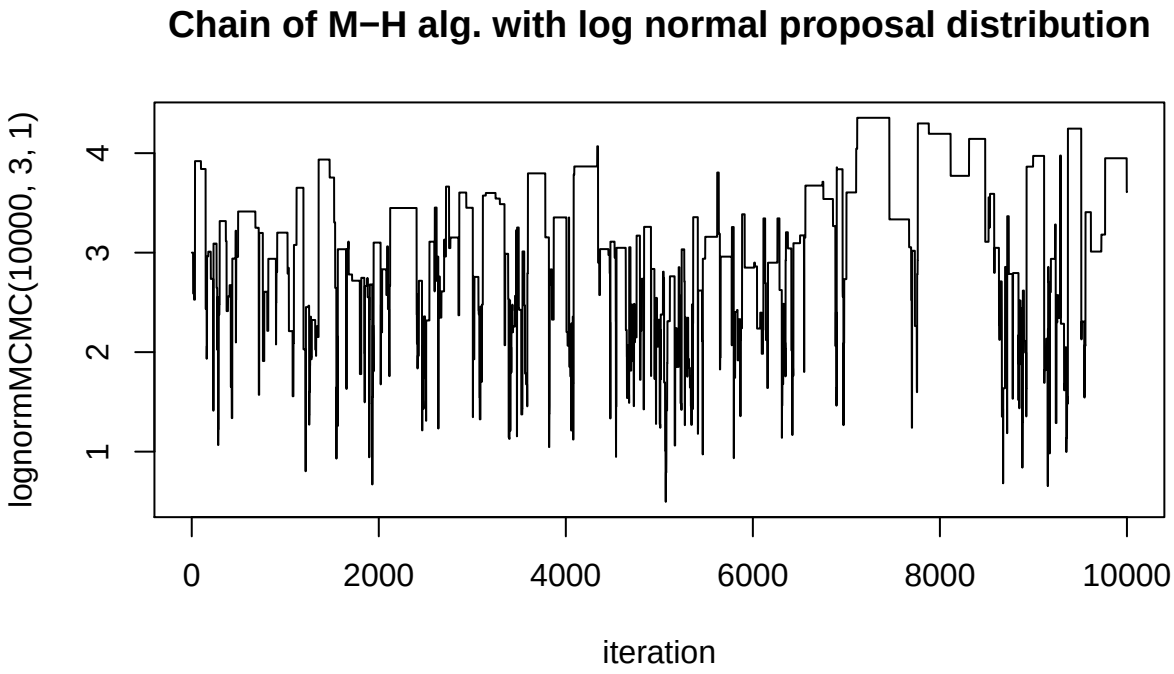
Computer Lab 4

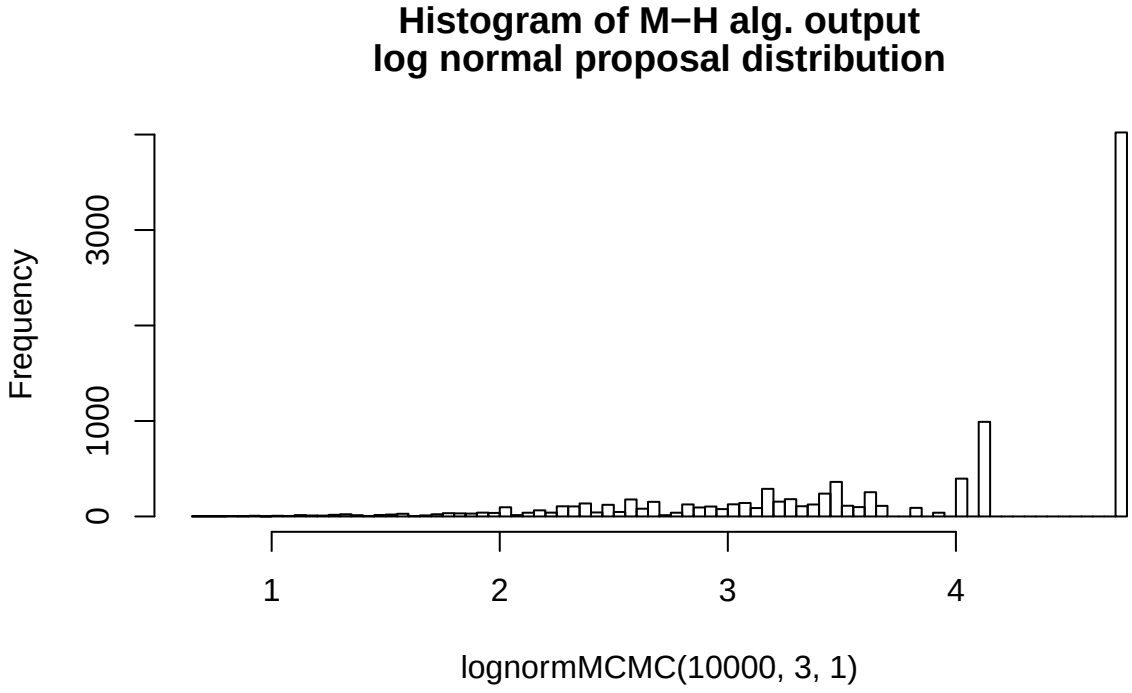
Thomas Zhang

2016 M02 1

Assignment 1

We start by running the Metropolis-Hastings algorithm in order to generate samples from the probability distribution $f(x)$ proportional to $x^5 \exp(-x)$ for $x > 0$ with a log-normal proposal distribution with mean X_t and standard deviation 1. We plot the traceplot and the histogram of outcomes.



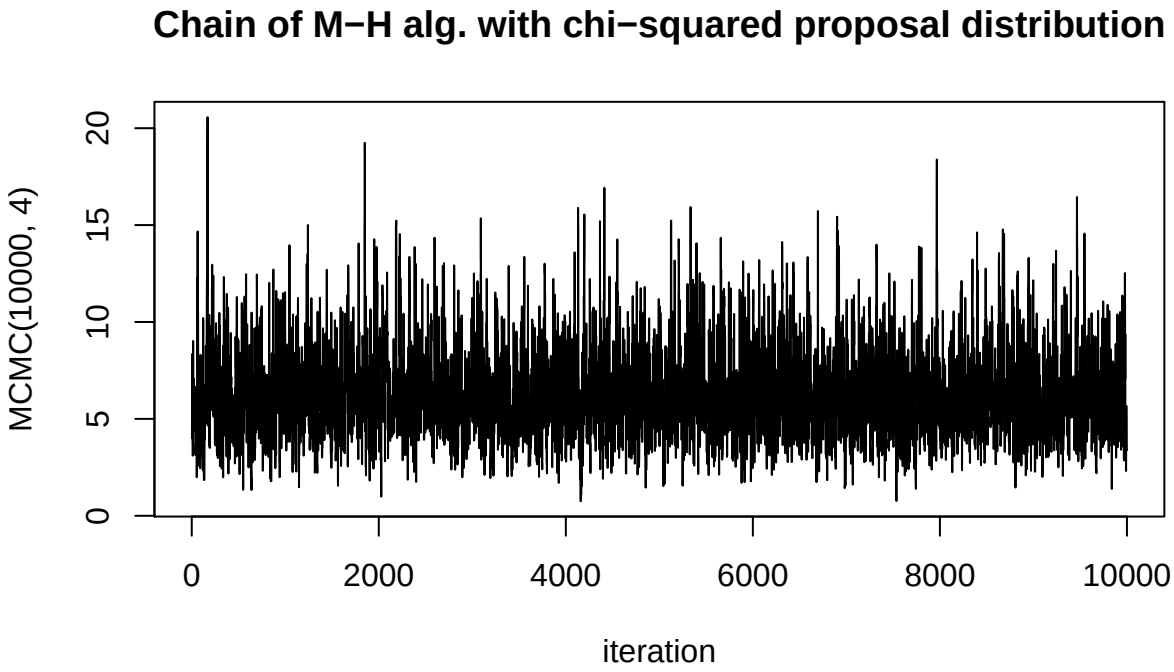


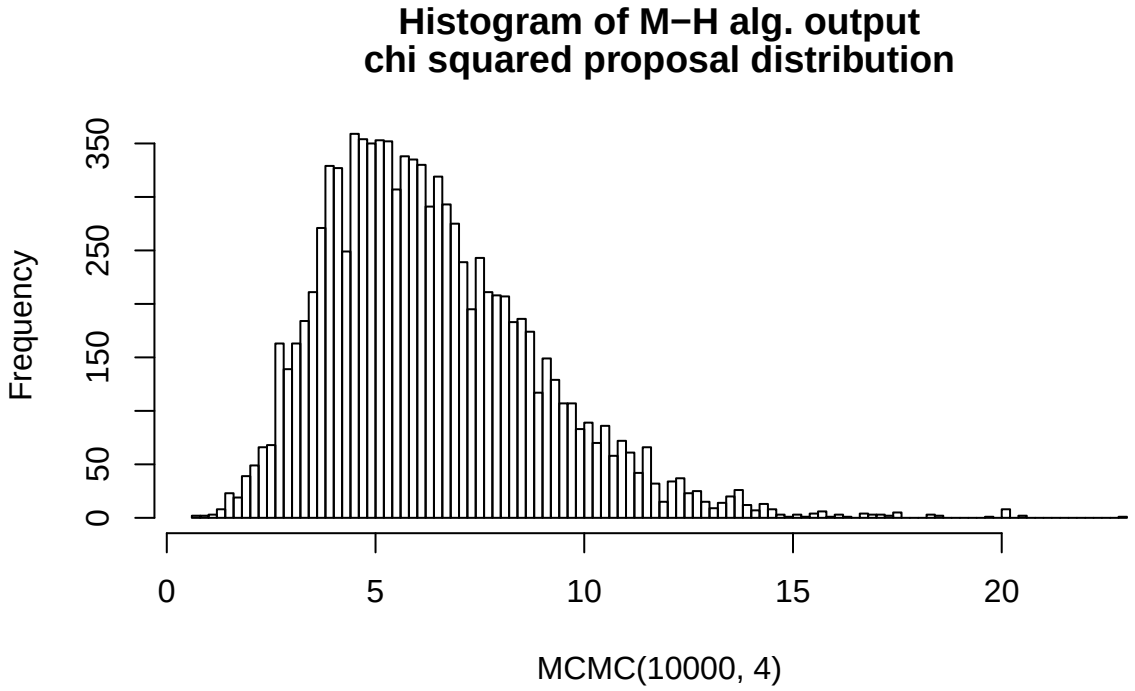

```
## [1] "Mean value of lognormMCMC chain: 3.579"
```

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.

Now let us repeat the above process, but replacing the log-normal proposal distribution with a chi-squared proposal distribution with parameter (degrees of freedom) equal to the integer part of $X_t + 1$.





```
## [1] "Mean value of MCMC chain: 6.337"
```

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. We seem to have convergence to a value around 6, according to the histogram. I Believe it is safe to say 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.

Now we use the Gelman-Rubin method to analyze convergence of chains in the case of chi-squared proposal distribution. We use ten chains with integer starting points ranging from one to ten. Length of chains is set to 500.

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.01       1.03
```

We see that for chain length 500, the upper confidence limit of the potential scale reduction factor is already approaching one. (We could also make it go closer to one by increasing chain length). Thus we have convergence of the MCMC chains produced.

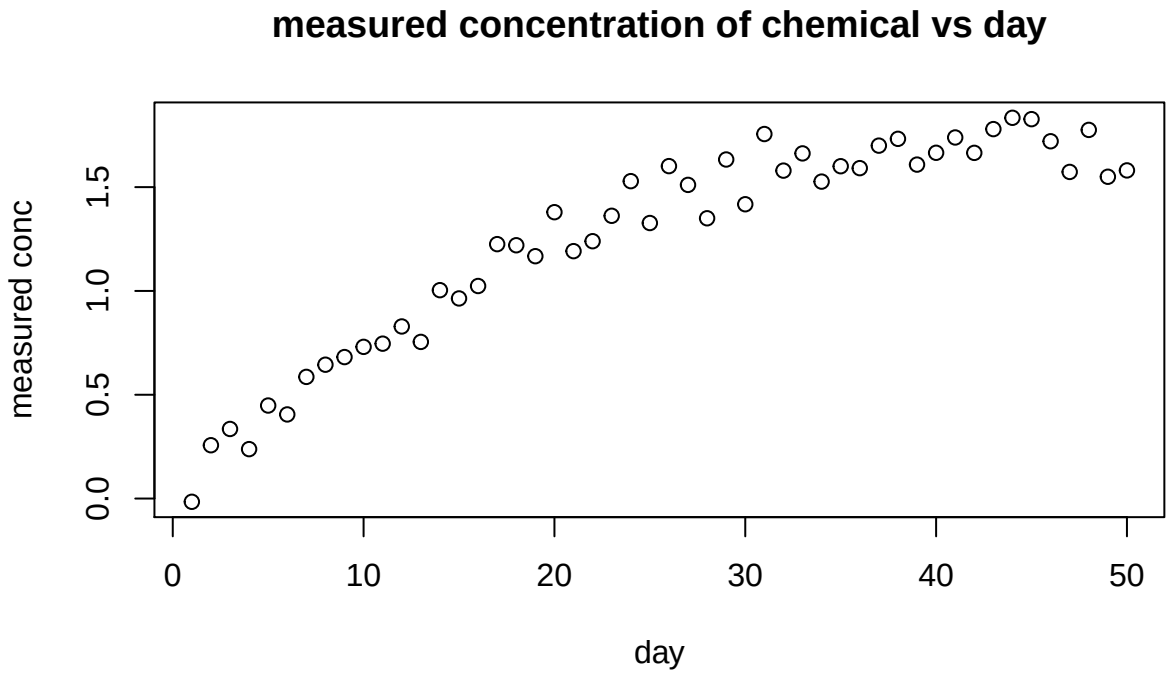
Now to estimate the integral $\int_0^\infty xf(x)dx$, which is the expression for the expected value of the probability distribution $f(x)$, we simply take the average of the output produced by the M-H algorithms. We see above that for log normal proposal distribution it is not close to six while for chi-squared proposal distribution it is close to six. According to literature for the gamma distribution:

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

the expected value should be precisely $k\theta$, or six in our case.The chi-squared proposal distribution gives a better estimate.

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.

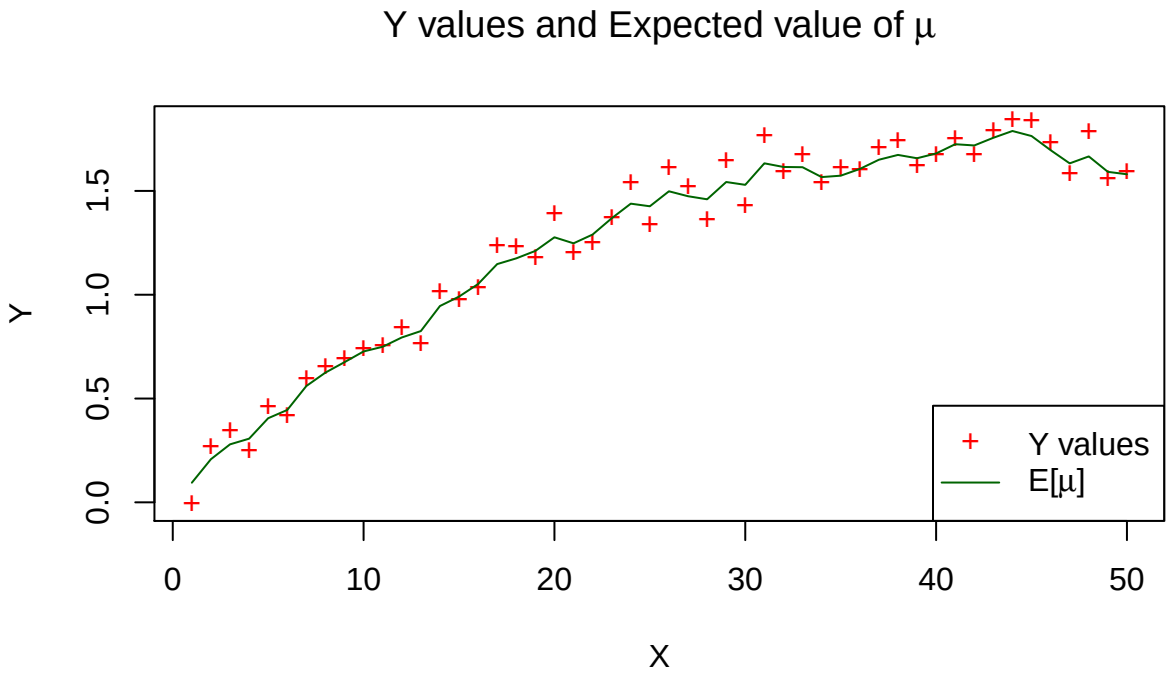
To find the distribution of $\mu_i|\boldsymbol{\mu_{-i}}, \boldsymbol{Y}$ where $\boldsymbol{\mu_{-i}}$ is a vector containing all μ values except for μ_i, we calculate the posterior distribution $P(\mu_i|\boldsymbol{\mu_{-i}}, \boldsymbol{Y})$. We discover there are three different cases and we derive 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: $\mu_k \sim N(\frac{Y_k+\mu_{k-1}+\mu_{k+1}}{3}, \frac{\sigma^2}{3})$

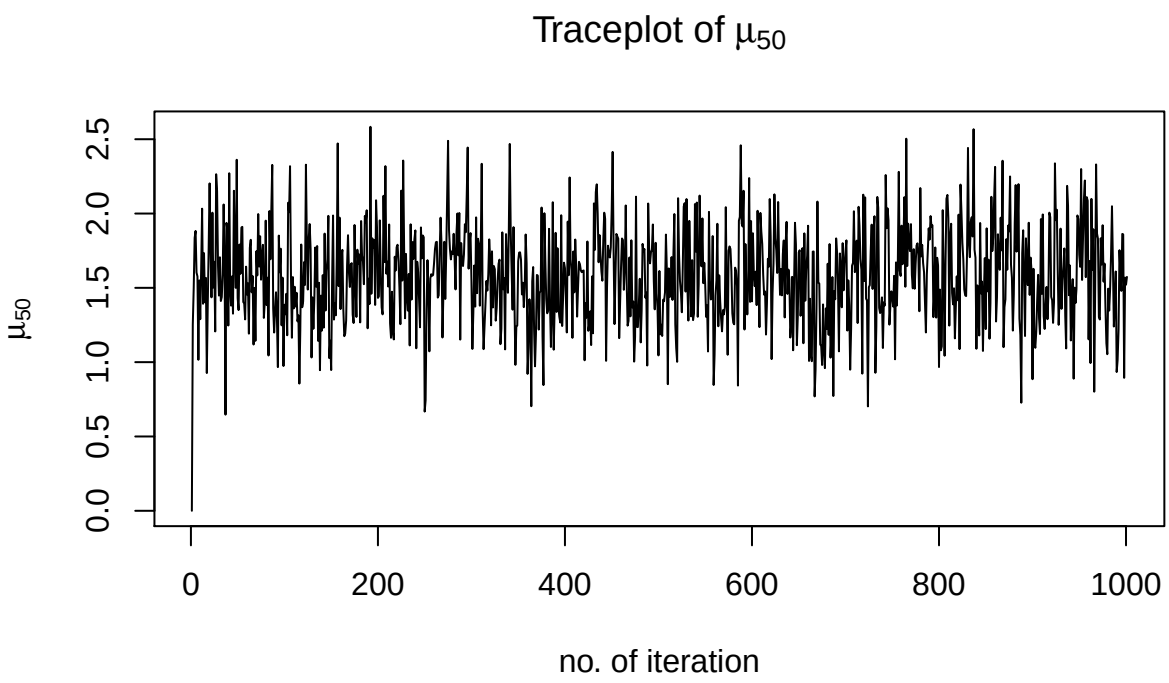
3. Last observation: $\mu_{50} \sim N(\frac{Y_{50}+\mu_{49}}{2}, \frac{\sigma^2}{2})$

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.



We see that the result is a good fit for the data and that the measurement noise has been taken away, to some degree.

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.

Appendix

R code

```
library(coda)
phi <- function(x){
  result <- x^5 * exp(-x)
  return(result)
}
#xes <- seq(from=0, to=100,by=0.01)
#plot(xes,1/100 * phi(xes),xlim=c(-1,30),type="l")
#lines(dlnorm(xes,),col=2)
#lines(dchisq(xes, df= 3),col=6)

lognormMCMC <- function(length,start,stddev){
  X <- c(start)
  for(i in 1:(length-1)){
    Y <- rlnorm(1,meanlog = X[i],sdlog = stddev)
    U <- runif(1)
    ratio <- phi(Y)/phi(X[i]) *
      dlnorm(X[i], meanlog = Y)/dlnorm(Y, meanlog = X[i])
    alpha <- min(1,ratio)
    if(U <= alpha){
      X[i + 1] <- Y
```

```
    }else{
      X[i + 1] <- X[i]
    }
  }
  return(X)
}

plot(1:10000,lognormMCMC(10000,3,1),type="l",
     main ="Chain of M-H alg. with log normal proposal distribution",
     xlab = "iteration")
# You can pick any start as long as its not too large above 5
hist(lognormMCMC(10000,3,1),breaks=100,
     main=c("Histogram of M-H alg. output","log normal proposal distribution"))
paste("Mean value of lognormMCMC chain:",round(mean(lognormMCMC(10000,3,1)),3)) # not good.
MCMC <- function(length,start){
  X <- c(start)
  for(i in 1:(length-1)){
    Y <- rchisq(1,floor(X[i] + 1))
    U <- runif(1)
    ratio <- phi(Y)/phi(X[i]) *
      dchisq(X[i], df = Y)/dchisq(Y, df = floor(X[i] + 1))
    alpha <- min(1,ratio)
    if(U <= alpha){
      X[i + 1] <- Y
    }else{
      X[i + 1] <- X[i]
    }
  }
  return(X)
}

plot(1:10000,MCMC(10000,4),type="l",
     main ="Chain of M-H alg. with chi-squared proposal distribution",
     xlab ="iteration") # You can pick any start as long as its not too large above 5
#Burn-in around 30 iterations
#well exploring the dchisq density
hist(MCMC(10000,4),breaks=100,main=c("Histogram of M-H alg. output","chi squared proposal distribution"))
paste("Mean value of MCMC chain:",round(mean(MCMC(10000,4)),3)) # This should be 6. it is close.

frame <- data.frame(MCMC(500,4))
for(i in 1:10){
  frame[,i] <- MCMC(500,i)
}
f = mcmc.list()
for(i in 1:10) f[[i]] = as.mcmc(frame[,i])
gelman.diag(f)
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
```