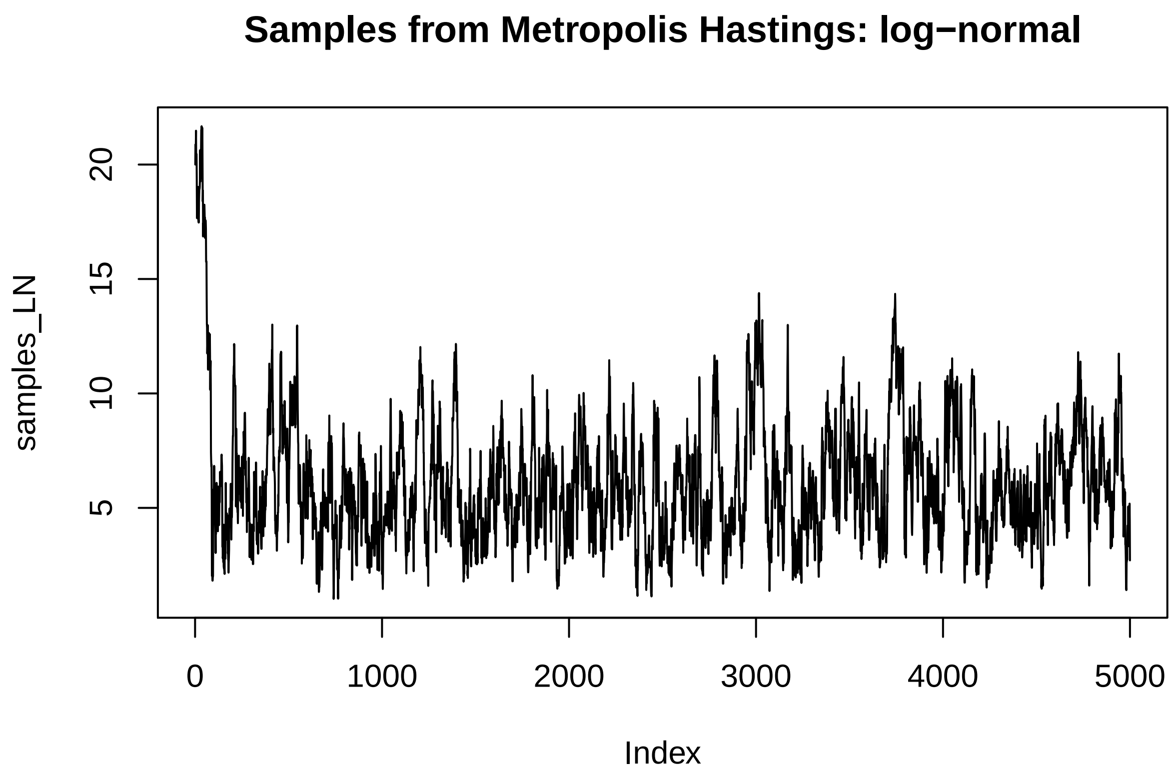
732A90: Lab 4

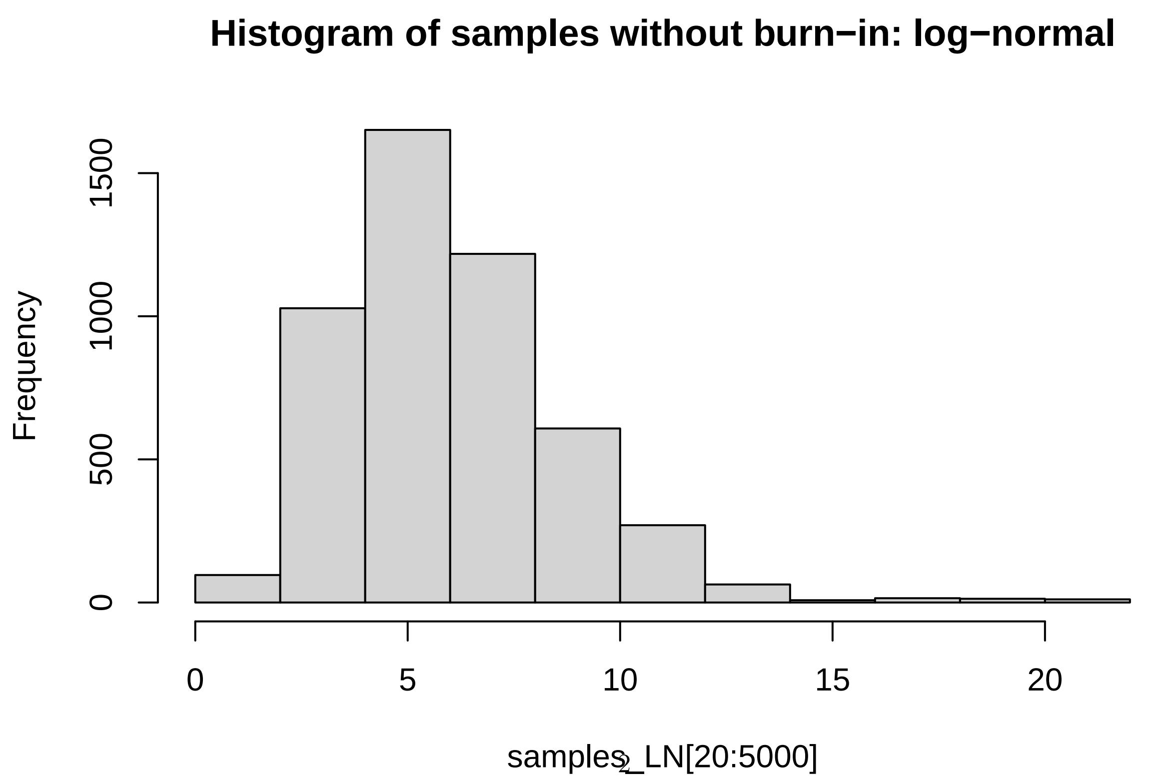
David Björelind, davbj395

11/25/2020

Question 1: Computations with Metropolis–Hastings

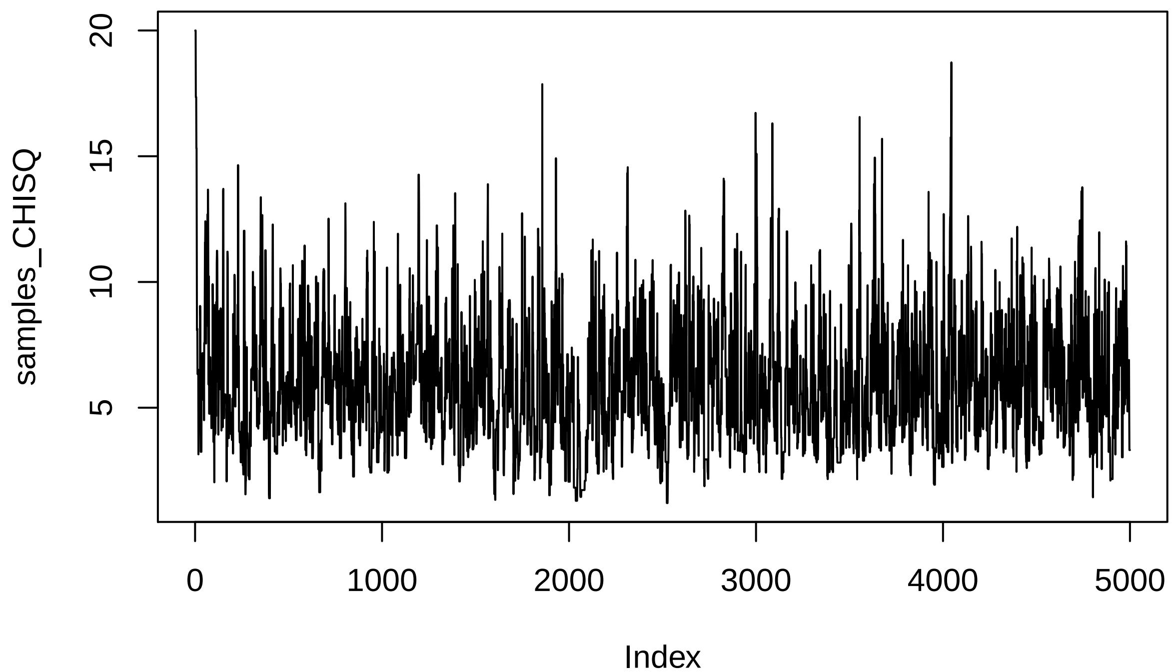
1.



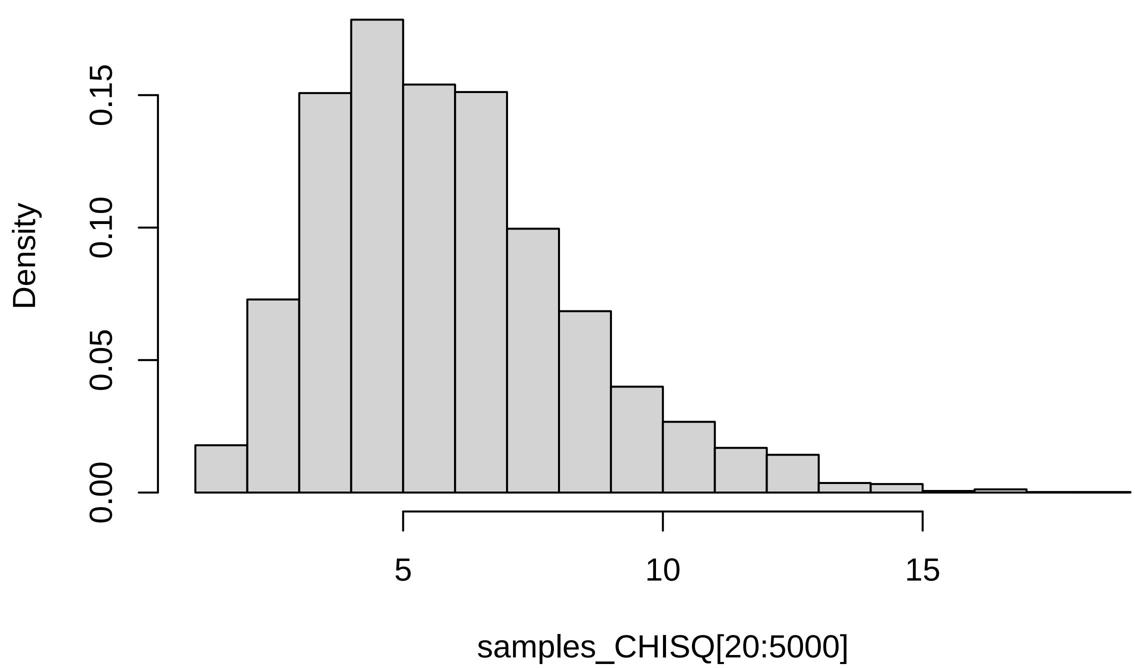


2.

Samples from Metropolis Hastings: chi−2 distribution



Histogram of samples without burn−in: chi−2 distribution



3.

There is a short burn-in period of about 20 iterations for the log-normal proposal distributions. The chi-2 distribution has a shorter burn-in of less than 5. Both are very short, however. The produces samples are varying from about 3 to 10 for both distributions. The mean and variance of the samples are also the same. Looking on the plots however, we can see some differences. Samples from the log-normal distribution is more smooth, and the samples from chi-2 distribution have higher (and lower) extreme values.

Looking at the histograms produced, the distribution looks very similar and one might guess that it is some kind of skewed normal distribution.

4.

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

5.

```
## Estimated integral from LN draws:  5.915351
```

```
##
## Estimated integral from CHISQ draws:  5.836959
```

6.

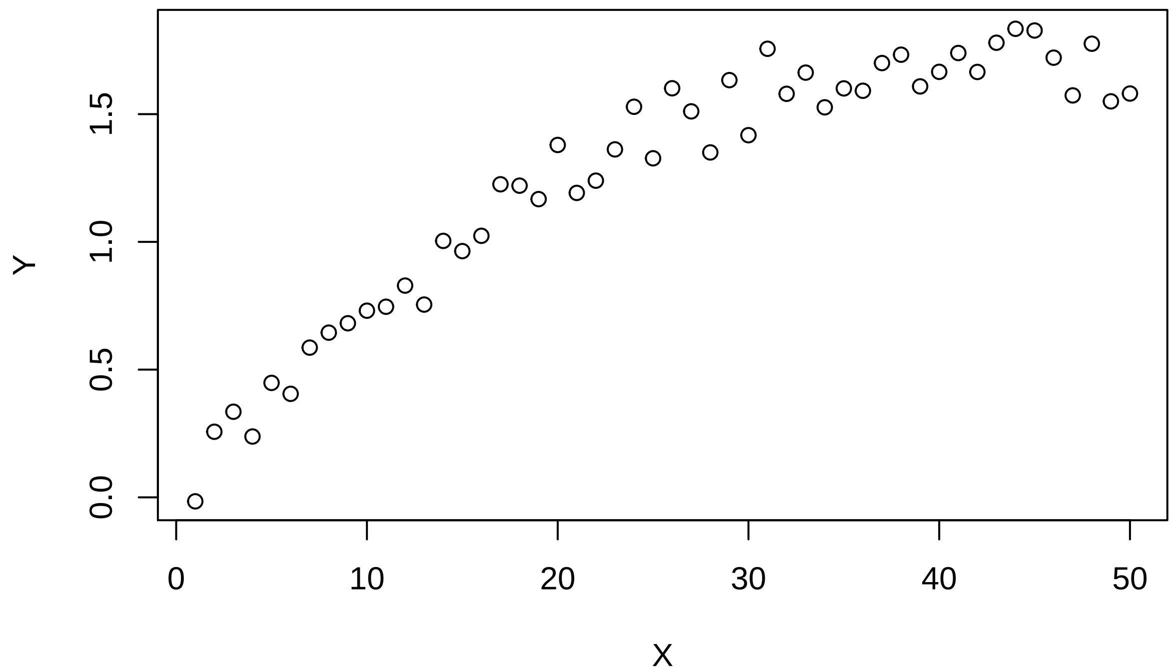
Gamma distribution with alpha = 6 and beta = 1. The integral actually calculated the expected value of f(x), E[f(x)]. Because we now know that it is a gamma distribution with known parameters, the mean is calculated

$$\frac{\alpha}{\beta}$$

. The actual mean is then $\frac{6}{1} = 6$. The stimations of the integrals produced is very similar to the real value!

Question 2: Gibbs sampling

1.



Exponential model?? Markov-chain model??

2.

The Prior:

Using the chain rule, the following distribution is achieved:

$$p(\vec{\mu}) = \prod_{i=1}^{n} \frac{1}{0.2\sqrt{2\pi}} * exp\left(-\frac{(\mu_{i+1} - \mu_i)^2}{2 * 0.2^2}\right)$$

which calculates to:

$$p(\vec{\mu}) = \left(\frac{1}{\sqrt{0.08\pi}}\right)^n * exp\left(-\frac{\sum_{i=1}^{n}(\mu_{i+1} - \mu_i)^2}{0.08}\right)$$

The likelihood:

The likelihood is calculated similar to the prior:

$$p(\vec{Y} \mid \vec{\mu}) = \prod_{i=1}^{n} \frac{1}{0.2\sqrt{2\pi}} * exp\left(-\frac{(y_i - \mu_i)^2}{2 * 0.2^2}\right)$$

which calculates to:

$$p(\vec{Y} \mid \vec{\mu}) = \left(\frac{1}{\sqrt{0.08\pi}}\right)^n * exp\left(-\frac{\sum_{i=1}^n (y_i - \mu_i)^2}{0.08}\right)$$

3.

Given that

$$Posterior \propto Prior * Likelihood$$

an expression for the posterior is:

$$(\mu_i \mid \vec{\mu}_{-i}, \vec{Y}) = (\mu_i) * (\vec{Y} \mid \mu_i)$$

For i=1 this expression is:

$$(\mu_1 \mid \vec{\mu}_{-1}, \vec{Y}) = (\mu_1) * (\vec{Y} \mid \mu_{-1}) = 1 * \left(\frac{1}{\sqrt{0.08\pi}}\right)^{n-1} * exp\left(-\frac{\sum_{i=2}^n (y_i - \mu_i)^2}{0.08}\right) \propto exp\left(-\frac{\sum_{i=2}^n (y_i - \mu_i)^2}{0.08}\right)$$

For i=n this expression is:

$$(\mu_n \mid \vec{\mu}_{-n}, \vec{Y}) = (\mu_n) * (\vec{Y} \mid \mu_{-n}) = \left(\frac{1}{\sqrt{0.08\pi}}\right)^{n-1} * exp\left(-\frac{\sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2}{0.08}\right) * \left(\frac{1}{\sqrt{0.08\pi}}\right)^{n-1} * exp\left(-\frac{\sum_{i=1}^{n-1} (y_i - \mu_i)^2}{0.08}\right) \propto$$

$$\propto exp\left(-\frac{\sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2 (y_i - \mu_i)^2}{0.08}\right)$$

Lastly, for 1<i<50 the expression is:

$$(\mu_i \mid \vec{\mu}_{-i}, \vec{Y}) = (\mu_i) * (\vec{Y} \mid \mu_{-n}) = \left(\frac{1}{\sqrt{0.08\pi}}\right)^{n-1} * exp\left(-\frac{\sum_{k=1}^{i-1} (\mu_{k+1} - \mu_k)^2 + \sum_{k=i+1}^{n} (\mu_{k+1} - \mu_k)^2}{0.08}\right) * \left(\frac{1}{\sqrt{0.08\pi}}\right)^{n-1} *$$

$$* exp\left(-\frac{\sum_{k=1}^{i-1} (y_k - \mu_k)^2 + \sum_{k=i+1}^{n} (y_k - \mu_k)^2}{0.08}\right) \propto$$

$$\propto$$

4.

Include all code for this report

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)
# Include packages here

pdf = function(x){
  return(exp(-x)*x^5)
}

metro_hastings = function(start, niter){
  res = array(0, niter)
  res[1] = start
  for(i in 2:niter){
    y = rnorm(1, mean=res[i-1] , sd=1)#, log=TRUE)
    u = runif(1, 0, 1)
```

```
    alpha = min(1, (pdf(y)*dnorm(res[i-1], mean=y , sd=1, log=TRUE))/(pdf(res[i-1])*dnorm(y, mean=res[i
    if(u< alpha){
      res[i] = y
    } else {
      #print('alpha > u')
      res[i] = res[i-1]
    }
  }

  return(res)
}

samples_LN = metro_hastings(20, 5000)
plot(samples_LN, type='l', main='Samples from Metropolis Hastings: log-normal')
hist(samples_LN[20:5000], freq=TRUE, main='Histogram of samples without burn-in: log-normal')
metro_hastings = function(start, niter){
  res = array(0, niter)
  res[1] = start
  for(i in 2:niter){
    y = rchisq(1, df=floor(res[i-1]))
    u = runif(1, 0, 1)
    alpha = min(1, (pdf(y)*dchisq(res[i-1], df=floor(y)))/(pdf(res[i-1])*dchisq(y, df=floor(res[i-1])))
    if(u< alpha){
      res[i] = y
    } else {
      #print('alpha > u')
      res[i] = res[i-1]
    }
  }

  return(res)
}

samples_CHISQ = metro_hastings(20, 5000)
plot(samples_CHISQ, type='l', main='Samples from Metropolis Hastings: chi-2 distribution')
hist(samples_CHISQ[20:5000], freq=FALSE, main='Histogram of samples without burn-in: chi-2 distribution

# Gelman - Rubin
library(coda)
start_point = seq(1, 10, length.out = 10)
niter = 2000
samples = matrix(0, nrow=length(start_point), ncol=niter)
for (i in 1:length(start_point)){
  samples[i,] = metro_hastings(start_point[i], niter)
}

burnin = 20
mcmclist = mcmc.list(
  as.mcmc(samples[1,burnin:niter]),
  as.mcmc(samples[2,burnin:niter]),
  as.mcmc(samples[3,burnin:niter]),
  as.mcmc(samples[4,burnin:niter]),
  as.mcmc(samples[5,burnin:niter]),
```

```r
  as.mcmc(samples[6,burnin:niter]),
  as.mcmc(samples[7,burnin:niter]),
  as.mcmc(samples[8,burnin:niter]),
  as.mcmc(samples[9,burnin:niter]),
  as.mcmc(samples[10,burnin:niter])
  )
gelman.diag(mcmclist)
#gelman.plot(mcmclist)
draws_LN = samples_LN[500:5000]
draws_CHISQ = samples_CHISQ[500:5000]
cat('Estimated integral from LN draws: ', mean(draws_LN))
cat('\nEstimated integral from CHISQ draws: ', mean(draws_CHISQ))

#gammafunc = function(x){
#  res = exp(-x)*(x^6)
#  return(res)
#}
#grid = seq(0,20, length.out = 1000)
#val = gammafunc(grid)
#plot(x=grid, y=val, type='l')
# Reading data
load("chemical.RData")
plot(x=X, y=Y)


# Gibbs sampler
gibbs = function(start, niter){
  d = length(start)
  res = matrix(0, nrow=niter, ncol=d)
  res[1,] = start

  for(i in 2:(niter)){
    res[i,1] = 123123123
    for(j in 2:(d-1)){
      res[i,j] = 123123123
    }
    res[i,d] = 123123123
  }

  return(res)
}

start_val = rep(0, 50)
niter = 1000
samples = gibbs(start_val, niter)
```