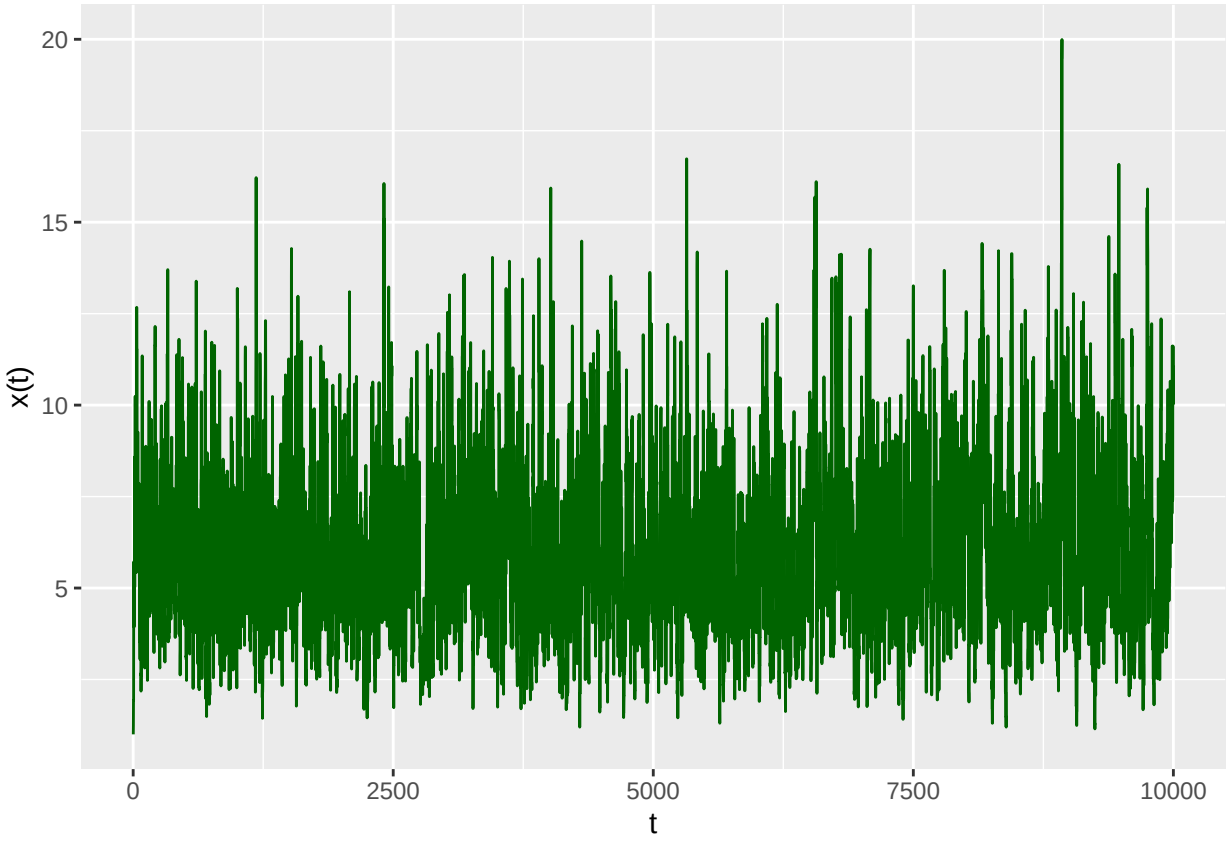
Computational Statistics: Lab 4

Nicolas Taba & Yuki Washio

01/12/2020

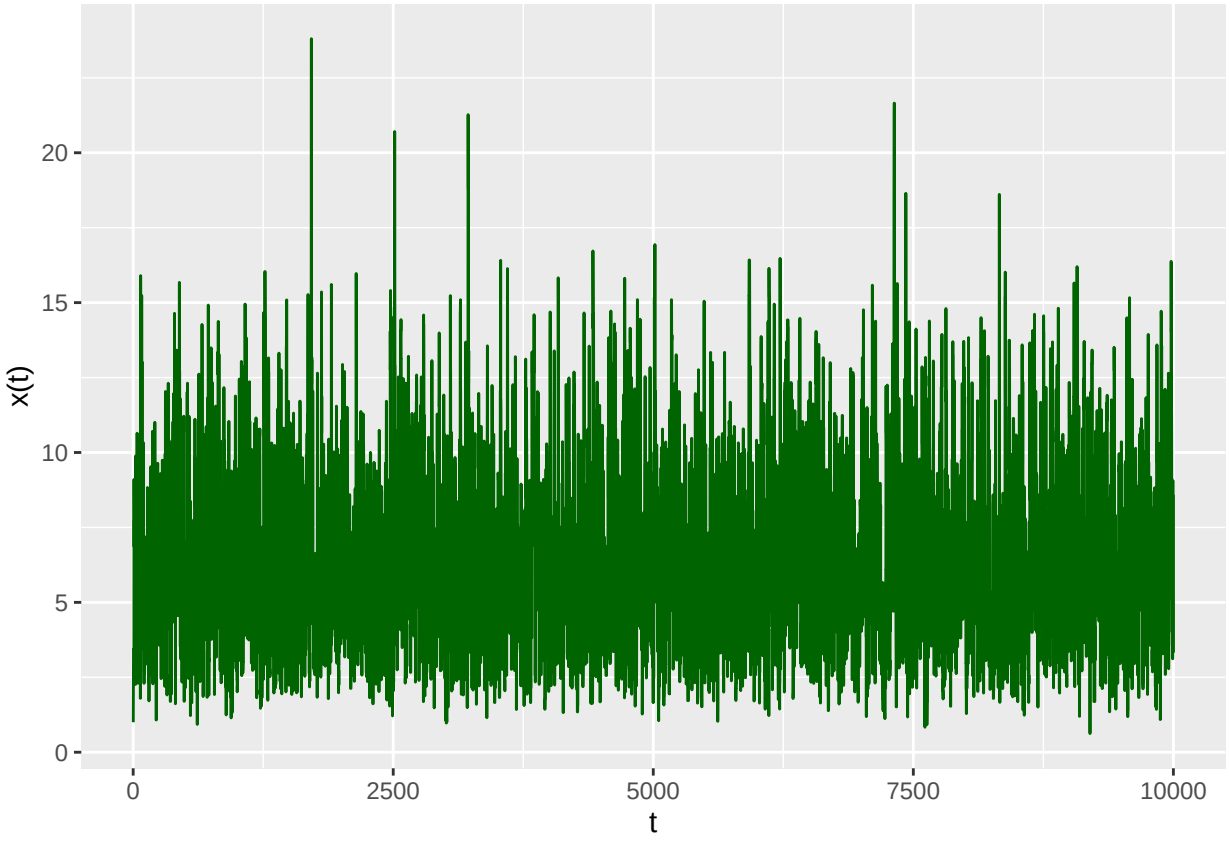
Computations with Metropolis-Hastings

1

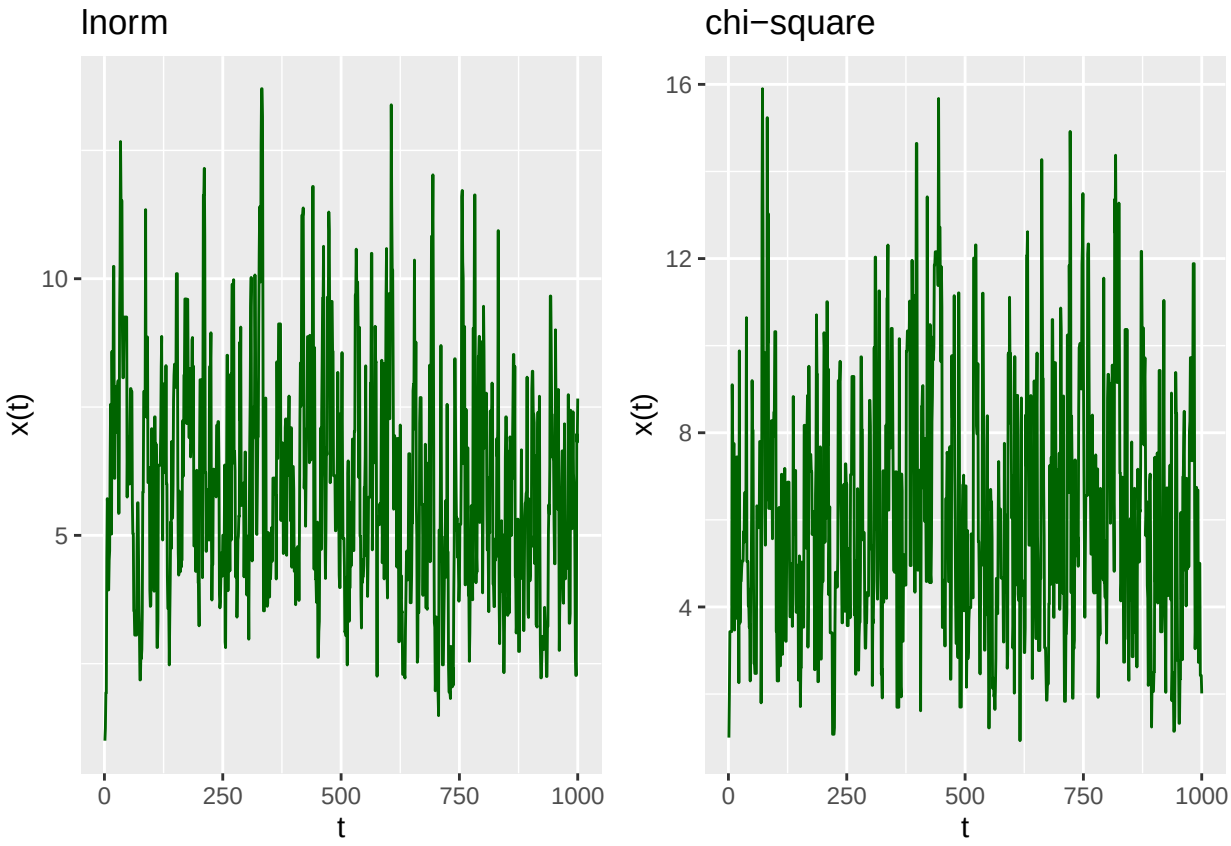


The chain displays a convergence behavior and the burn-in period is very short (around 50 time steps).

2



3



Both of the chains converge. Their burn-in periods differ but they are both very short. The Chi-square chain presents less outlier behavior and seems to be more stable than the log-normal chain.

4

```
## Warning: package 'coda' was built under R version 4.0.3
```

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

The Gelman-Rubin factor is equal to 1. This means that the chi-square chains converge.

5

```
## The estimate of the integral using the log-normal method yields an estimate of:  6.083172
```

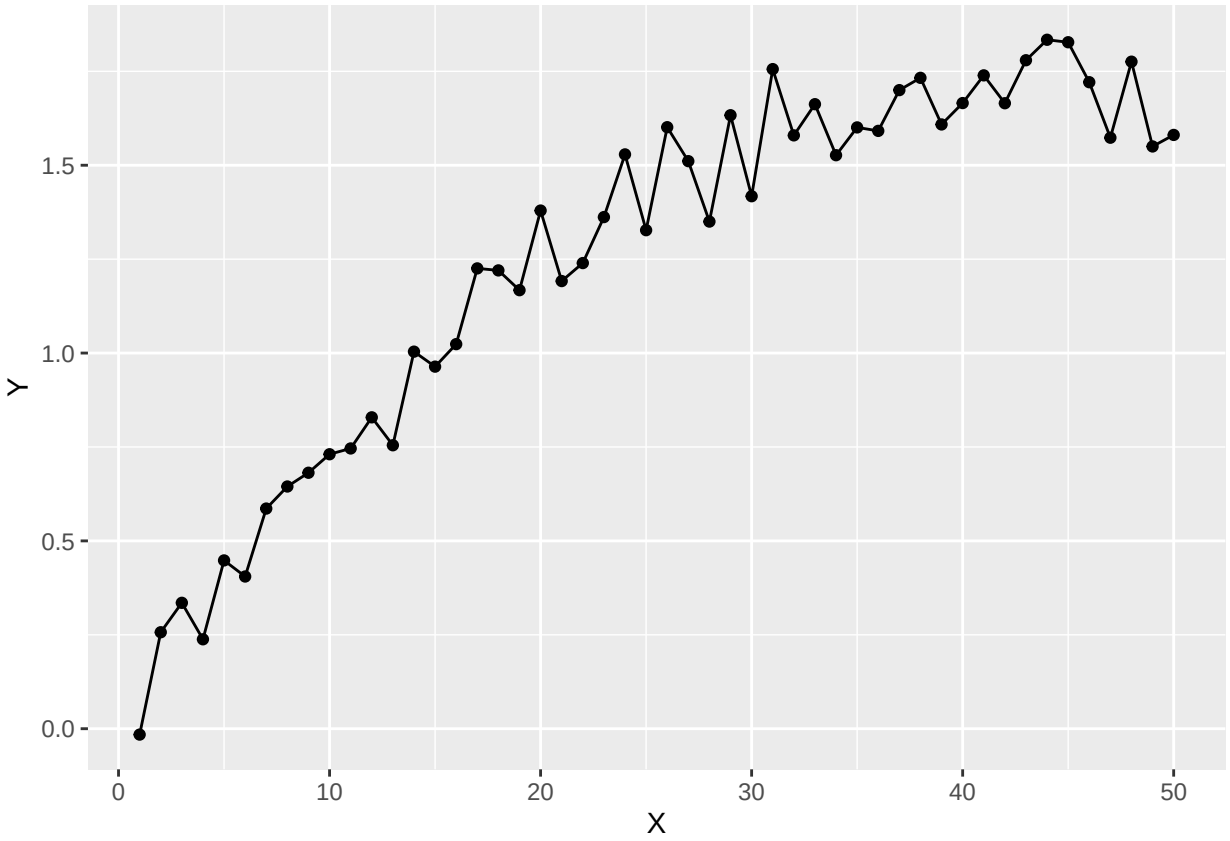
```
##
## The estimate of the integral using the Chi-square method yields an estimate of:  6.324324
```

6

This integral represents the calculation of the expected value of a gamma distribution with parameters $\alpha = 6$ and $\beta = 1$ up to a constant. We know that the expectation value of the distribution over the whole domain is equal to its mean. The mean of a gamma distribution is $\alpha\beta$, therefore the integral can be solved exactly and is equal to 6.

Gibbs sampling

1



This curve looks quadratic with a negative value to its quadratic term and a positive value for its other terms.

2

The probability distribution function of a gaussian distribution is:

$$N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i-\mu_i)^2}{2\sigma^2}}$$

We calculate the Likelihood as the product of successive probability distributions:

$$p(Y|\mu) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \cdot exp\left(-\frac{(y_i-\mu_i)^2}{2\sigma^2} \right) = \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right)^n \cdot exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i-\mu_i)^2 \right)$$

For the calculation of the prior, the first element of the product of probabilities is just equal to 1, we therefore compute the product of the normal distributions from 1 to n-1. The prior is thus:

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

3

The posterior is:

$$p(\vec{\mu}|\vec{Y}) \propto P(\vec{Y}|\vec{\mu})P(\vec{\mu})$$

We compute the product and associate terms under the exponential. We then isolate the last term of the sum of the likelihood and perform the sum from 1 to n-1:

$$p(\vec{\mu}|\vec{Y}) \propto \exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i-\mu_i)^2\right]\exp\left[-\frac{1}{2\sigma^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\right]$$

$$p(\vec{\mu}|\vec{Y}) \propto exp\left[-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n}(y_i-\mu_i)^2+\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2\right)\right] \propto exp\left[-\frac{1}{2\sigma^2}\left(\sum_{i=1}^{n-1}\left[(\mu_i-\mu_{i+1})^2+(\mu_i-y_i)^2\right]+(\mu_n-y_n)^2\right)\right]$$

From here, we can compute the posterior probability at each step. For the first and last step, the calculation is straightforward making use of hint B given in the exercise sheet. However, for middle steps, we encounter an issue in the generalization of the calculation of the prior in the middle of our sequence. the i-th prior's depends on the previous value and the one that follows.

FOr the first step, we have:

$$p(\mu_1|\vec{\mu}_{-1},\vec{Y}) \propto exp\left[-\frac{1}{2\sigma^2}\left[(\mu_1-\mu_2)^2+(\mu_1-y_1)^2\right]\right]$$

Frr the last step we have:

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto exp\left[-\frac{1}{2\sigma^2}\left[(\mu_{n-1}-\mu_n)^2+(\mu_n-y_n)^2\right]\right]$$

We can use hint B to transform both these expressions respectively into:

$$p(\mu_1|\vec{\mu}_{-1},\vec{Y}) \propto exp\left[-\frac{1}{\sigma^2}\left[\mu_1-\frac{\mu_2+y_1}{2}\right]^2\right] \sim N\left(\frac{\mu_2+y_1}{2},\frac{\sigma^2}{2}\right)$$

and

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto exp\left[-\frac{1}{\sigma^2}\left[\mu_n-\frac{\mu_{n-1}+y_n}{2}\right]^2\right] \sim N\left(\frac{\mu_{n-1}+y_n}{2},\frac{\sigma^2}{2}\right)$$

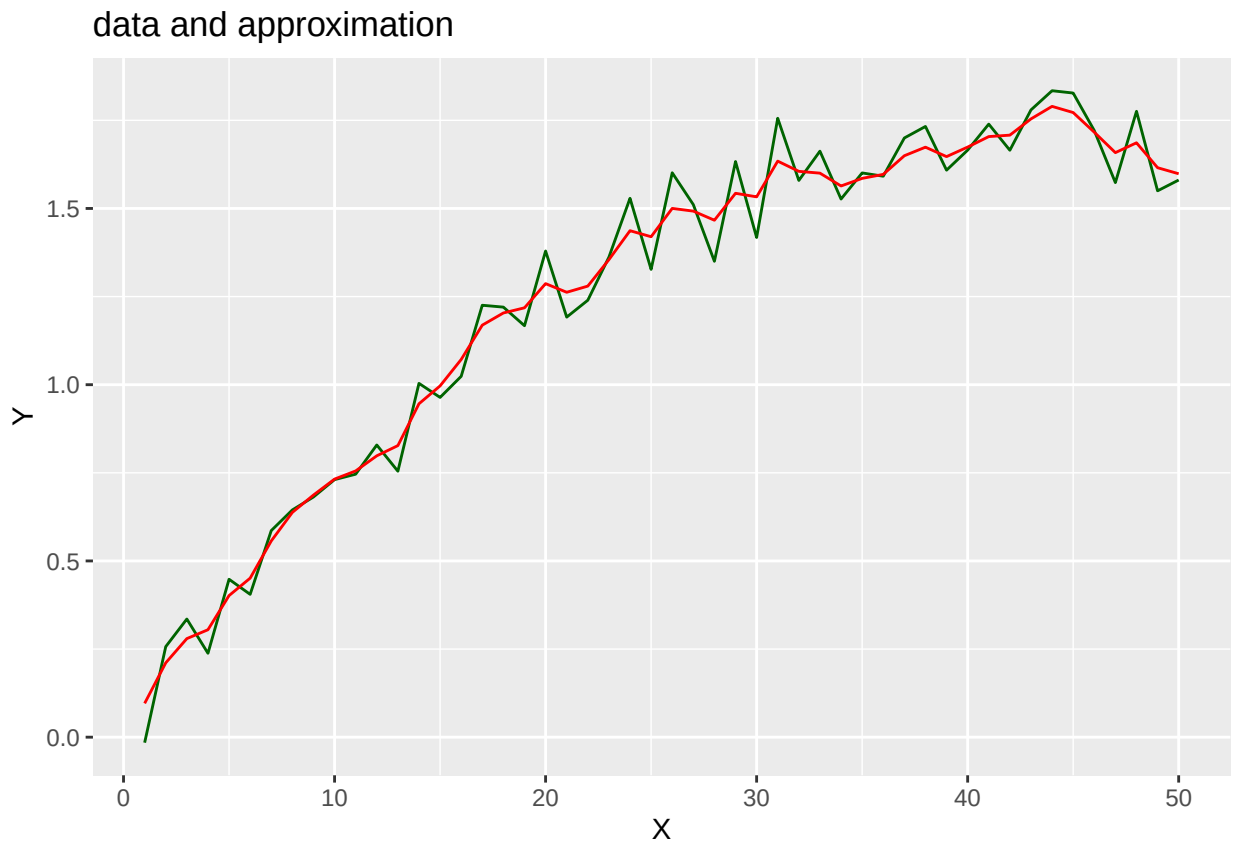
For the middle step we have terms that show dependency on the previous and the next step in the calculation of the prior-term:

$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto exp\left[-\frac{1}{2\sigma^2}\left[(\mu_{i-1}-\mu_i)^2+(\mu_i-\mu_{i+1})^2+(\mu_i-y_i)^2\right]\right]$$

We now use hint C given in the exercise sheet and obtain the following:

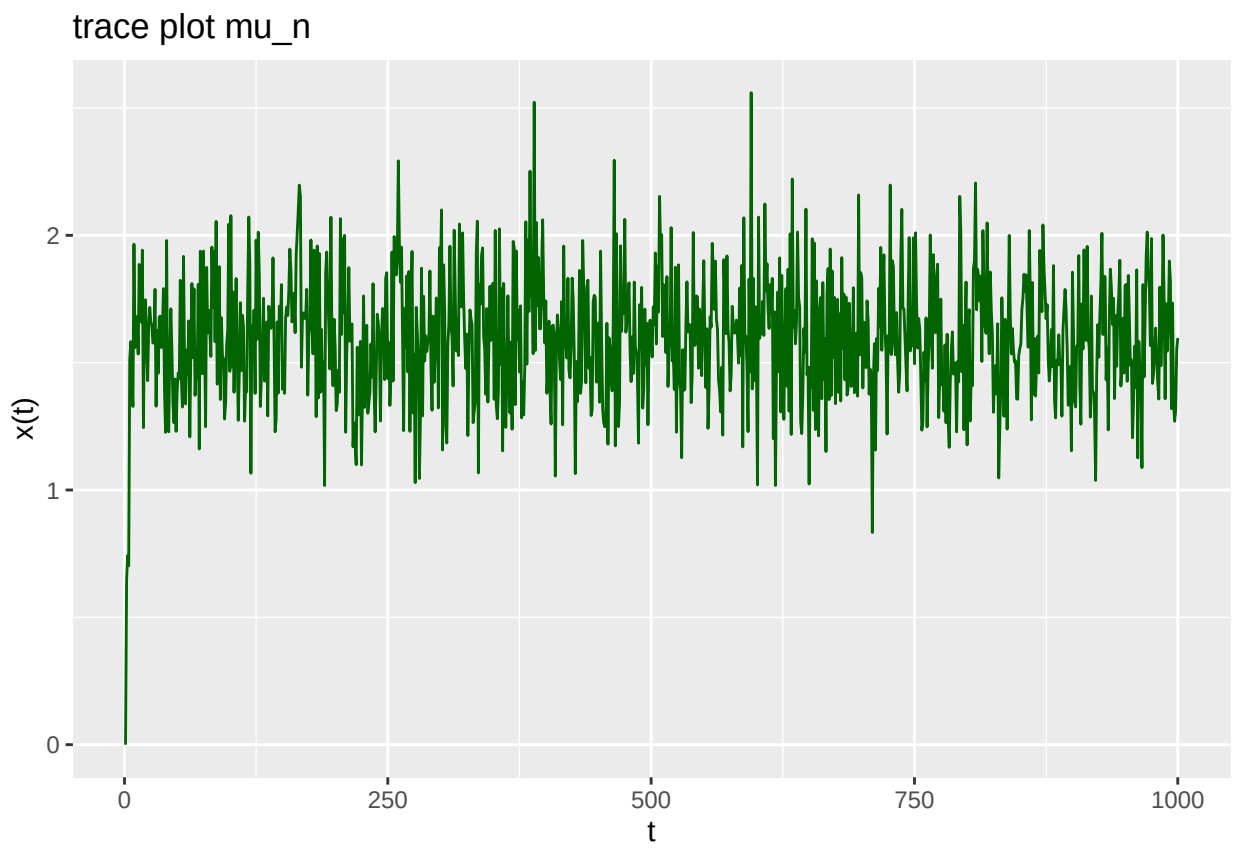
$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto exp\left[-\frac{3}{2\sigma^2}\left[\mu_i-\frac{\mu_{i-1}+\mu_{i+1}+y_i}{3}\right]^2\right] \sim N\left(\frac{\mu_{i-1}+\mu_{i+1}+y_i}{3},\frac{\sigma^2}{3}\right)$$

4



The approximated line follows the general trend of the data but has less variance. Thus the noise was removed and smoothed.

5



The burn-in period is short (<50) and the chain converges.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

# rm(list = ls())
library(ggplot2)

# target probability density function
f_target  =  function(x){
  return((x^5)*exp(-x))
}

f.MCMC.MH.lnorm = function(nstep, X0, props){
  vN = 1:nstep
  vX = rep(X0, nstep);
  for (i in 2:nstep){
    X = vX[i-1]
    Y = rlnorm(1, log(X), sdlog=props)
    u = runif(1)
    a = min(c(1, (f_target(Y)*dlnorm(X, meanlog=log(Y), sdlog=props)) / (f_target(X)*dlnorm(Y, meanlog=l
    if (u <= a) {
      vX[i] = Y
    } else {
```

```
      vX[i] = X
    }
  }
  return(vX)
}

set.seed(12345)
lnorm_vals = f.MCMC.MH.lnorm(nstep = 10000, X0 = 1, props = f_target(1))

lnorm_plot = ggplot(data = data.frame(lnorm_vals), aes(x = 1:length(lnorm_vals), y = lnorm_vals)) +
  geom_line(color="darkgreen") +
  xlab("t") + ylab("x(t)")

lnorm_plot
f.MCMC.MH.chisq = function(nstep, X0){
  vN = 1:nstep
  vX = rep(X0,nstep)
  for (i in 2:nstep){
    X = vX[i-1]
    Y = rchisq(1, df=floor(X+1))
    u = runif(1)
    a = min(c(1, (f_target(Y)*dchisq(X, df=floor(X+1))) / (f_target(X)*dchisq(Y, df=floor(X+1)))))
    if (u <= a){
      vX[i] = Y
    } else {
      vX[i] = X
    }
  }
  return(vX)
}

set.seed(12345)
chisq_vals = f.MCMC.MH.chisq(nstep = 10000, X0 = 1)


chisq_plot = ggplot(data = data.frame(chisq_vals), aes(x = 1:length(chisq_vals), y = chisq_vals)) +
  geom_line(color="darkgreen") +
  xlab("t") + ylab("x(t)")

chisq_plot
library(gridExtra)

# Plotting only first 1000 values for a better view on burn-in period.
lnorm_vals_burn_in = lnorm_vals[1:1000]
lnorm_burn_in = ggplot(data = data.frame(lnorm_vals_burn_in), aes(x = 1:length(lnorm_vals_burn_in), y =
  geom_line(color="darkgreen") +
  xlab("t") + ylab("x(t)") + ggtitle("lnorm")

chisq_vals_burn_in = chisq_vals[1:1000]
chisq_burn_in = ggplot(data = data.frame(chisq_vals_burn_in), aes(x = 1:length(chisq_vals_burn_in), y =
  geom_line(color="darkgreen") +
  xlab("t") + ylab("x(t)") + ggtitle("chi-square")
```

```r
grid.arrange(lnorm_burn_in, chisq_burn_in, ncol=2)

library(coda)

last_start_value = 10
mcmc_list = mcmc.list()

for (i in 1:last_start_value) {
  mcmc_list[[i]] = as.mcmc(f.MCMC.MH.chisq(nstep = 10000, X0 = i))
}

gelman.diag(mcmc_list)
# We get the estimate by calculating the mean of our samples from 1 and 2
# lnorm
lnorm_estimate = mean(lnorm_vals[-(1:1000)])

# chisquare
chisq_estimate = mean(chisq_vals[-(1:1000)])

cat("The estimate of the integral using the log-normal method yields an estimate of: ", lnorm_estimate)
cat("\nThe estimate of the integral using the Chi-square method yields an estimate of: ", chisq_estimat

load("chemical.RData")
data = data.frame("X" = X,"Y" = Y)

plot1 <- ggplot(data, aes(x = X, y = Y))+
  geom_point()+
  geom_line()
plot1
num_values = 1000

f.MCMC.Gibbs<-function(nstep, X0, Y, variance){
  vN<-1:nstep
  mX<-matrix(0,nrow=nstep,ncol=length(Y))
  mX[1,]<-X0
  length_y = length(Y)
  for (i in 2:nstep){
    prev_sample<-mX[i-1,]
    curr_sample<-rep(0,length(Y))
    curr_sample[1]<-rnorm(1, mean=((prev_sample[2]+Y[1])/2), sqrt(variance)/2)
    for (j in 2:(length_y-1)){
      curr_sample[j]<-rnorm(1, mean=(Y[j] + prev_sample[j-1] + prev_sample[j+1])/3, sqrt(variance)/3)
    }
    curr_sample[length_y]<-rnorm(1, mean=((Y[length_y]+prev_sample[length_y-1])/2), sqrt(variance)/2)
    mX[i,]<-curr_sample
  }
  mX
}

gibbs_vals = f.MCMC.Gibbs(num_values, rep(0, length(Y)), Y, variance=0.2)
column_means = colMeans(gibbs_vals)

data_extend = cbind(data, column_means)
```

```
plot1  =  ggplot(data_extend) +
  geom_line(aes(x=X, y=Y), color="darkgreen") +
  geom_line(aes(x=X, y=column_means), color="red") + ggtitle("data and approximation")
plot1
mu_n = gibbs_vals[,50]

mu_n_trace = ggplot(data = data.frame(mu_n), aes(x = 1:length(mu_n), y = mu_n)) +
  geom_line(color="darkgreen") +
  xlab("t") + ylab("x(t)") + ggtitle("trace plot mu_n")
mu_n_trace
```