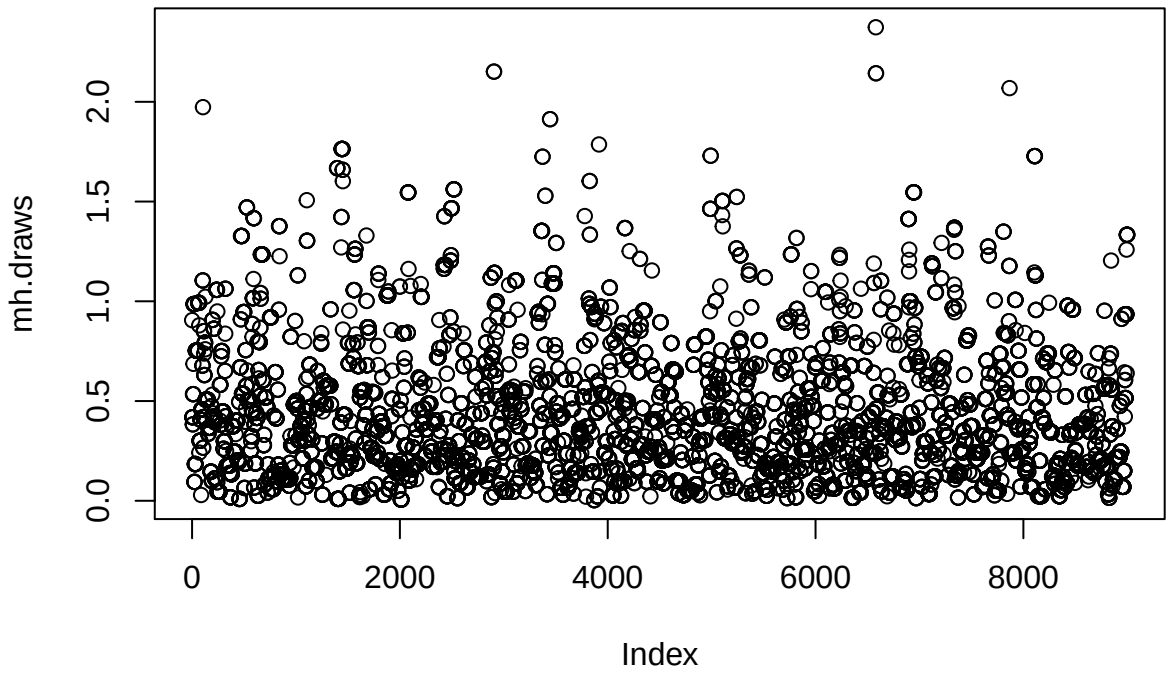
STAT37810_HW2

Seungah Ha

2018 10 17

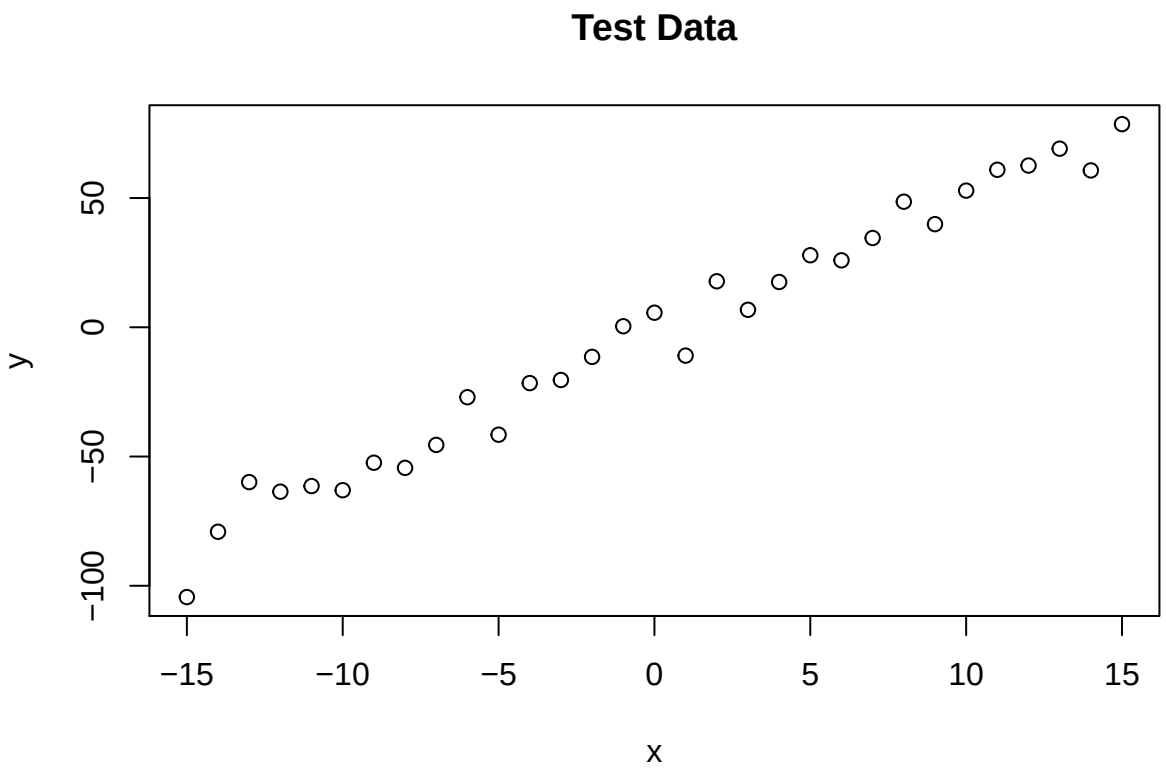
```
metroh<-read.csv('http://uchicago-stat-comp-37810.github.io/webpage/lectures/lecture5_data.csv')

mh.gamma <- function(n.sims, start, burnin, cand.sd, shape, rate) {
  theta.cur <- start
  draws <- c()
  theta.update <- function(theta.cur, shape, rate) {
    theta.can <- rnorm(1, mean = theta.cur, sd = cand.sd)
    accept.prob <- dgamma(theta.can, shape = shape, rate = rate)/dgamma(theta.cur,shape = shape, rate =
    if (runif(1) <= accept.prob) theta.can else {theta.cur}
  }
  for (i in 1:n.sims) {
    draws[i] <- theta.cur <- theta.update(theta.cur, shape = shape,
                                          rate = rate)
  }
  return(draws[(burnin + 1):n.sims])
}
mh.draws <- mh.gamma(10000, start = 1, burnin = 1000, cand.sd = 2,shape = 1.7, rate = 4.4)
plot(mh.draws)
```



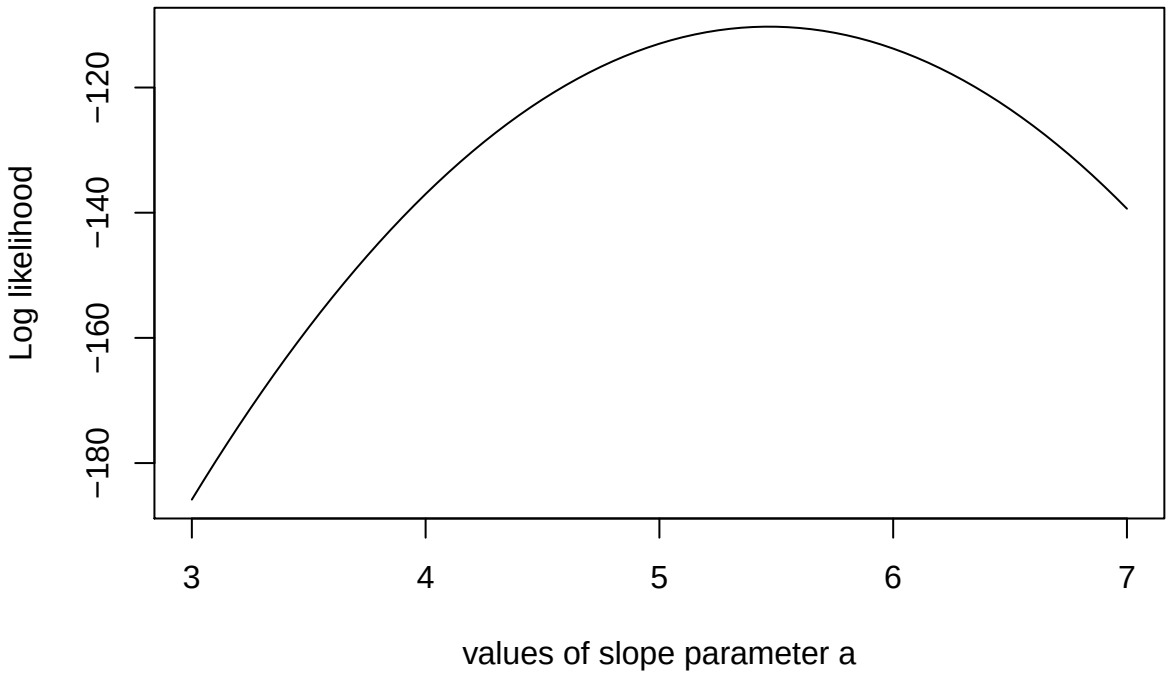
1.

Here is the code in a blog post by Florian Hartig:

```r
trueA <- 5
trueB <- 0
trueSd <- 10
samplesize <- 31

# create independent x-values
x <- (-(samplesize-1)/2):((samplesize-1)/2)
# create dependent values according to ax+b+N(0, sd)
y <- trueA*x+trueB+rnorm(n=samplesize, mean=0, sd=trueSd)

plot(x,y, main="Test Data")
```

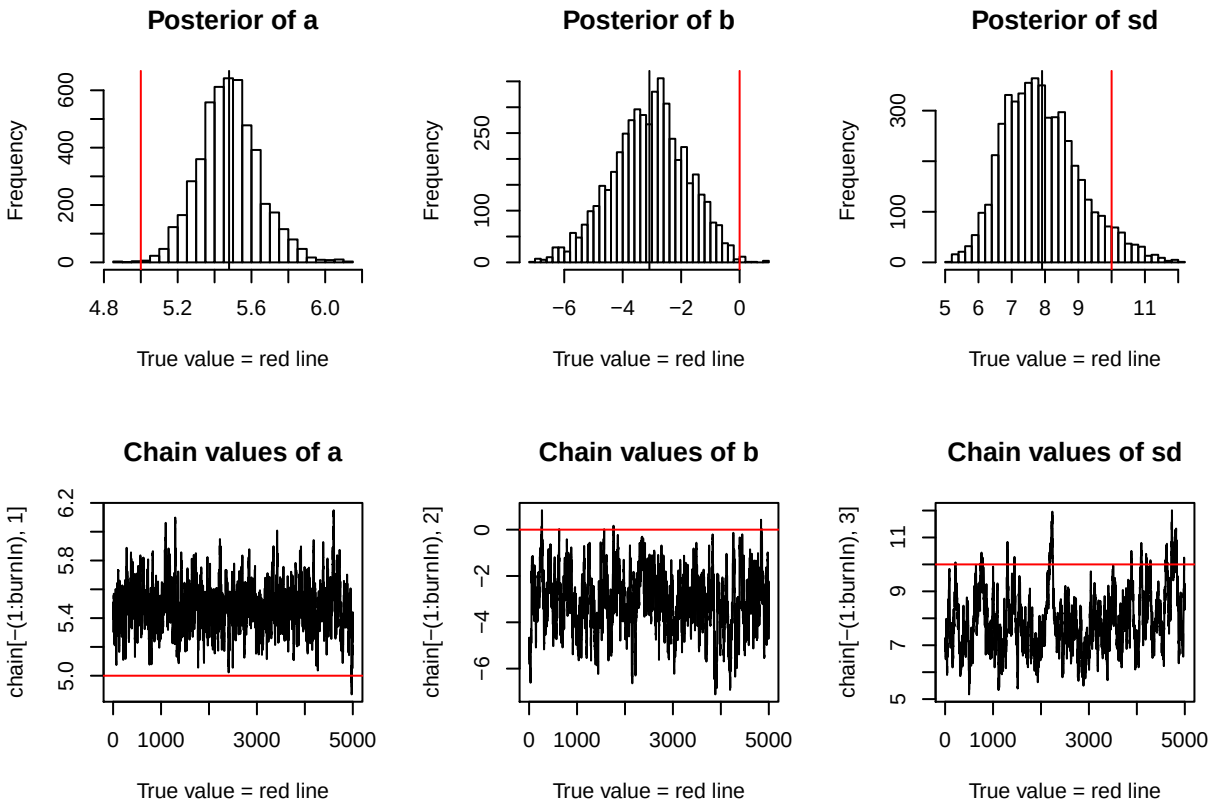



```r
likelihood <- function(param){
    a=param[1]
    b=param[2]
    sd=param[3]

    pred=a*x+b
    singlelikelihoods = dnorm(y, mean = pred, sd=sd, log = T)
    sumll = sum(singlelikelihoods)
    return(sumll)
}

# Example: plot the likelihood profile of the slope a
slopevalues <- function(x){return(likelihood(c(x, trueB, trueSd)))}
slopelikelihoods <- lapply(seq(3,7,by=.05), slopevalues)
plot(seq(3,7,by=.05), slopelikelihoods, type="l", xlab="values of slope parameter a", ylab = "Log likeli
```




```
# Prior distribution
prior <- function(param){
    a = param[1]
    b = param[2]
    sd = param[3]
    aprior = dunif(a, min=0, max=10, log = T)
    bprior = dnorm(b, sd = 5, log = T)
    sdprior = dunif(sd, min=0, max=30, log = T)
    return(aprior+bprior+sdprior)
}

posterior <- function(param){
  return(likelihood(param)+prior(param))
}

# Metropolis algorithm

proposalfunction <- function(param){
  return(rnorm(3, mean = param, sd=c(0.1, 0.5, 0.3)))
}

run_metropolis_MCMC <- function(startvalue, iterations){
    chain = array(dim = c(iterations+1,3))
    chain[1,] = startvalue
    for (i in 1:iterations){
        proposal = proposalfunction(chain[i,])

        probab = exp(posterior(proposal) - posterior(chain[i,]))
        if (runif(1) < probab){
            chain[i+1,] = proposal
        }else{
            chain[i+1,] = chain[i,]
```

```
        }
    }
    return(chain)
}

startvalue = c(4,0,10)
chain=run_metropolis_MCMC(startvalue, 10000)
burnIn=5000
acceptance=1-mean(duplicated(chain[-(1:burnIn),]))

# Summary

par(mfrow = c(2,3))
hist(chain[-(1:burnIn),1], nclass=30, main="Posterior of a", xlab="True value = red line" )
abline(v = mean(chain[-(1:burnIn),1]))
abline(v = trueA, col="red" )
hist(chain[-(1:burnIn),2],nclass=30, main="Posterior of b", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),2]))
abline(v = trueB, col="red" )
hist(chain[-(1:burnIn),3],nclass=30, main="Posterior of sd", xlab="True value = red line")
abline(v = mean(chain[-(1:burnIn),3]) )
abline(v = trueSd, col="red" )
plot(chain[-(1:burnIn),1], type = "l", xlab="True value = red line" , main = "Chain values of a")
abline(h = trueA, col="red" )
plot(chain[-(1:burnIn),2], type = "l", xlab="True value = red line" , main = "Chain values of b")
abline(h = trueB, col="red" )
plot(chain[-(1:burnIn),3], type = "l", xlab="True value = red line" , main = "Chain values of sd")
abline(h = trueSd, col="red" )
```




```
# for comparison:
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.890  -4.329   0.409   4.105  14.652
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.4540     1.3799  -2.503   0.0182 *
## x             5.4685     0.1543  35.445   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.683 on 29 degrees of freedom
## Multiple R-squared:  0.9774, Adjusted R-squared:  0.9767
## F-statistic:  1256 on 1 and 29 DF,  p-value: < 2.2e-16
```