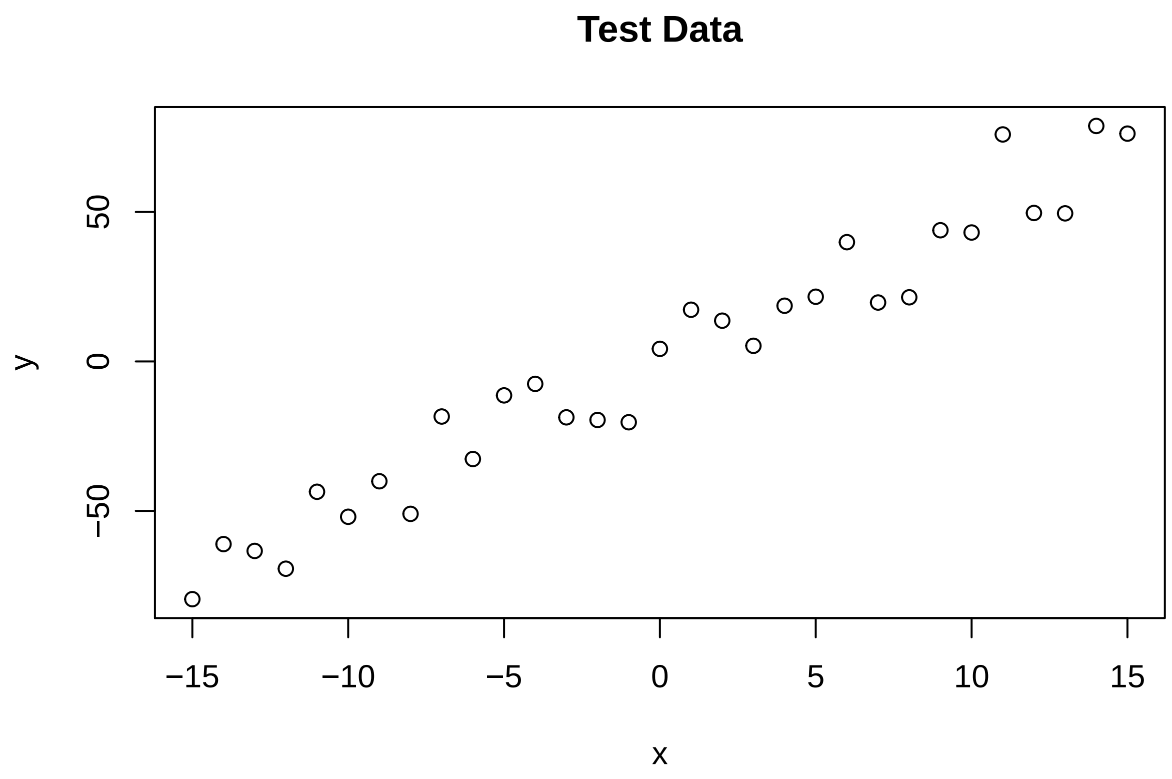
Rscript for MCMC_commented

Kim Ting Li

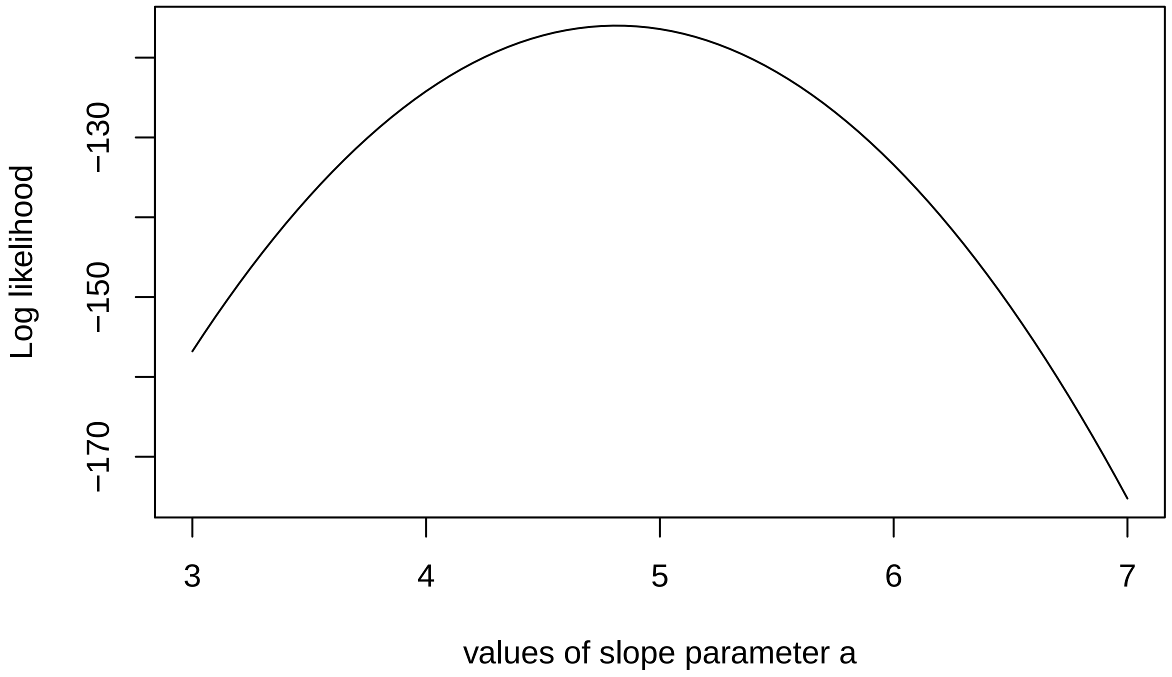
October 13, 2017

```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")
```



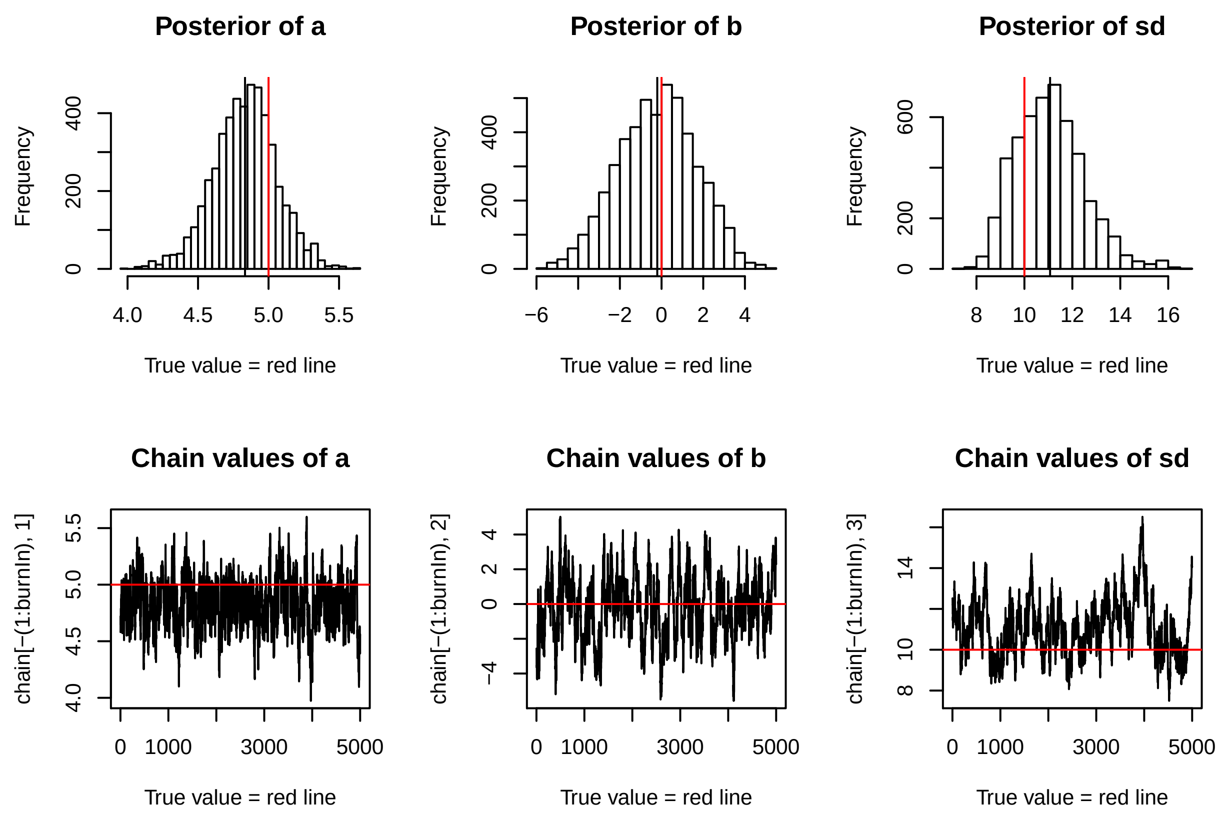
```r
likelihood <- function(param){
a = param[1]
b = param[2]
sd = param[3]
pred = a*x + b
singlelikelihoods = dnorm(y, mean = pred, sd = sd, log = T)
sumll = sum(singlelikelihoods)
```

```r
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 l
```



```r
# 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){
```

```r
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" )
```



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

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.7408  -8.3466  -0.4893   8.1353  23.2944
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.3145     1.8930  -0.166    0.869
## x             4.8141     0.2116  22.746   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.54 on 29 degrees of freedom
## Multiple R-squared:  0.9469, Adjusted R-squared:  0.9451
## F-statistic: 517.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

the algorithm gets an estimate pretty close to the true A (mean close to 5). However, increasing the number of iterations do not seem to improve accuracy. See results below.

```
compare_outcomes<-function(iteration){

  for (j in 1:10){

    startvalue <- rnorm(3,mean = c(5,0,10), sd= c(0.1,0.5,0.3))
    chain = run_metropolis_MCMC(startvalue, iteration)
    #print(chain)
    burnIn = iteration/2
    #instead of putting a number here, the burnIn will be adjusted according to the iteration number

    posta=chain[-(1:burnIn),1]
    #return(posta)
    amean = mean(posta)
    asd = sd(posta)
    print(amean) #print mean first
    print(asd) #print sd second
  }
}
onethou.it<-compare_outcomes(1000)
```

```
## [1] 4.73678
## [1] 0.236293
## [1] 4.787871
## [1] 0.1942404
## [1] 4.790595
## [1] 0.2424005
## [1] 4.804148
## [1] 0.1944941
## [1] 4.865813
## [1] 0.1982235
## [1] 4.850765
## [1] 0.2010755
## [1] 4.810729
## [1] 0.2742639
## [1] 4.813833
## [1] 0.1912593
## [1] 4.783717
## [1] 0.2546529
## [1] 4.823685
## [1] 0.2592132
```

```
tenthou.it<-compare_outcomes(10000)
```

```
## [1] 4.813733
## [1] 0.2092885
## [1] 4.82797
## [1] 0.2227358
## [1] 4.818309
## [1] 0.2234341
## [1] 4.818303
## [1] 0.2247459
## [1] 4.827782
## [1] 0.2416846
## [1] 4.793683
## [1] 0.221718
## [1] 4.822759
## [1] 0.2136332
## [1] 4.81626
## [1] 0.2184882
## [1] 4.809155
## [1] 0.2102208
## [1] 4.797442
## [1] 0.2182705
```

```
hunthou.it<-compare_outcomes(100000)
```

```
## [1] 4.825467
## [1] 0.2187737
## [1] 4.800889
## [1] 0.2243745
## [1] 4.82095
## [1] 0.2207796
## [1] 4.811575
## [1] 0.2166723
## [1] 4.815966
## [1] 0.2230877
## [1] 4.82236
## [1] 0.2185986
## [1] 4.816553
## [1] 0.22563
## [1] 4.816624
## [1] 0.2265592
## [1] 4.803637
## [1] 0.2279667
## [1] 4.821023
## [1] 0.2195719
```