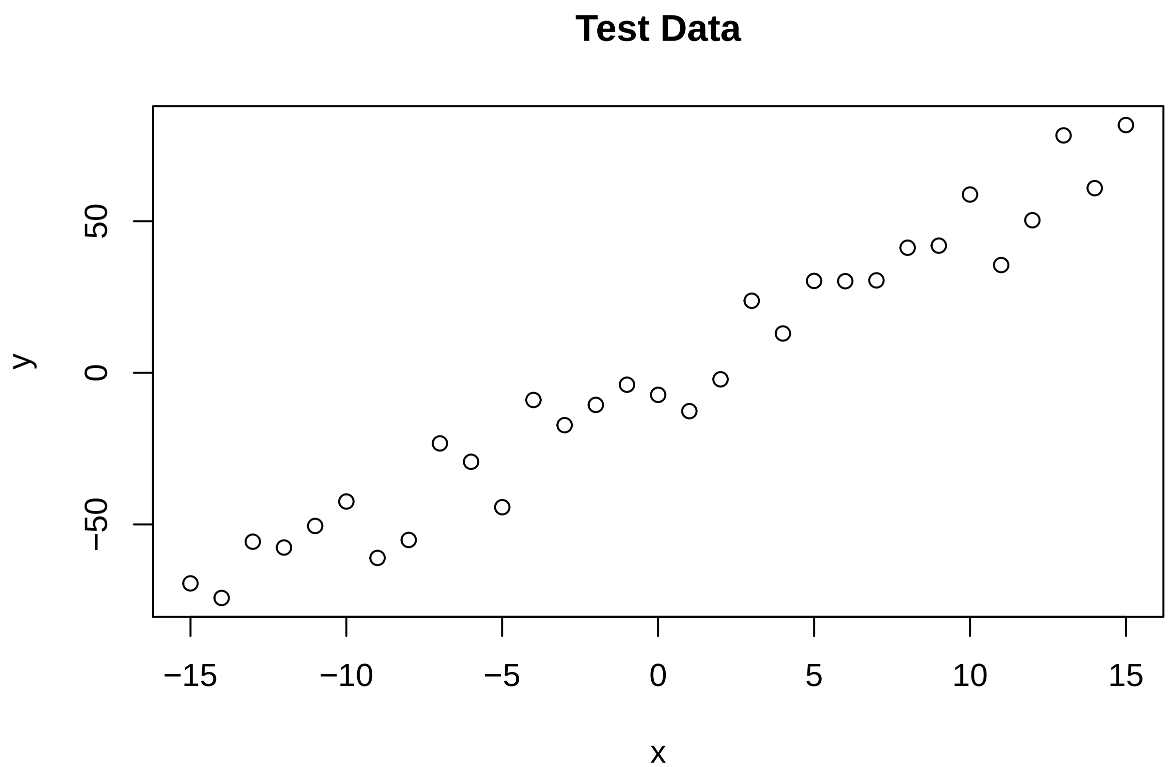
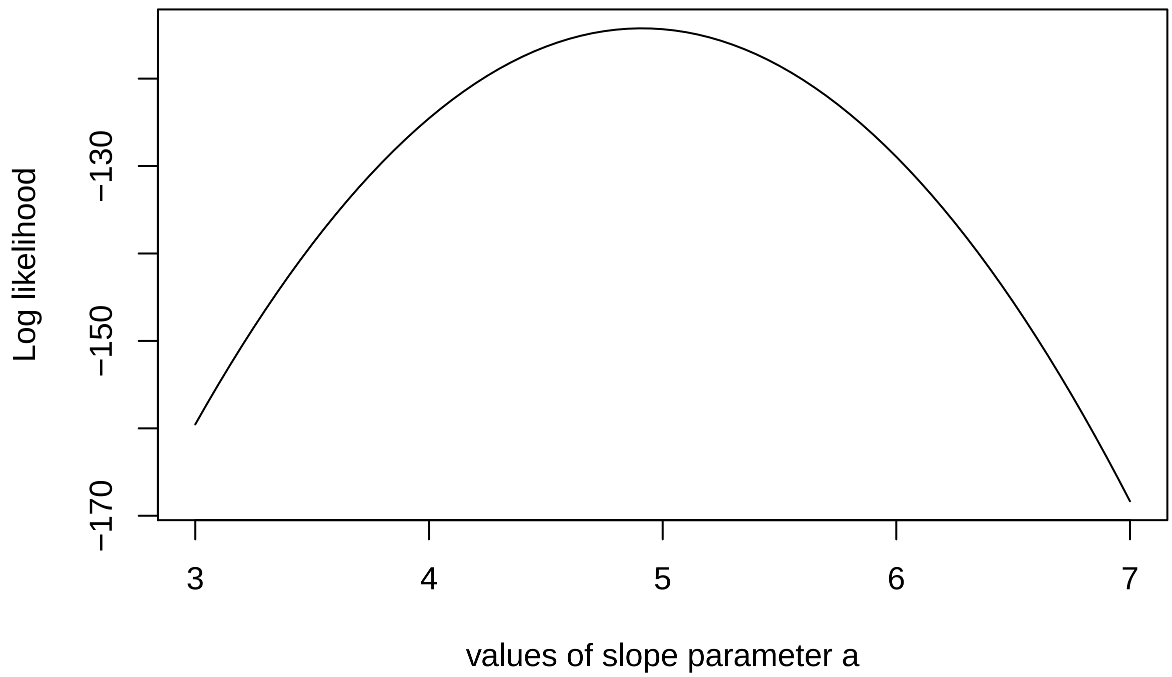
Accuracy Report

Wonbin Song

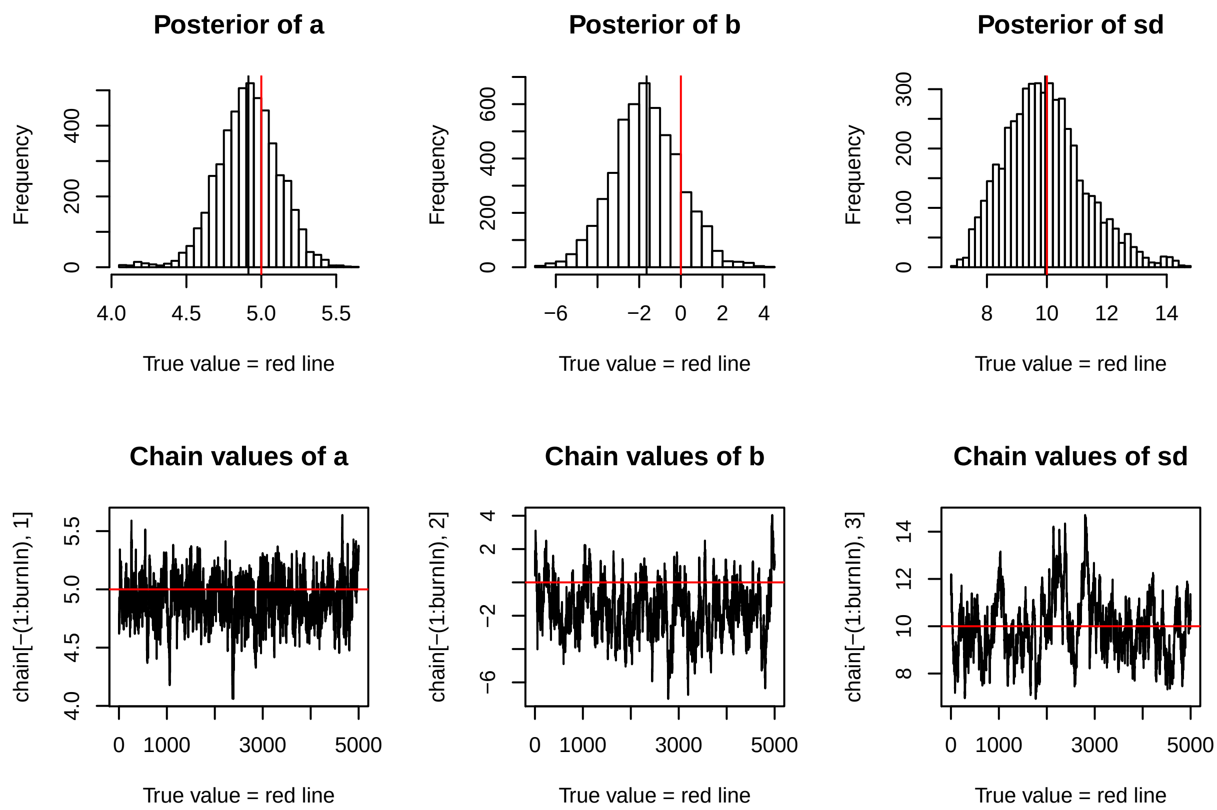
```r
source('separate_file.R')

# Creating test data

trueA <- 5 # slope
trueB <- 0   # intercept
trueSd <- 10  # standard deviation of error
sampleSize <- 31  # sample size

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

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




```r
slopelikelihoods <- lapply(seq(3, 7, by=.05), slopevalues )  # apply the function slopevalues to the co
plot (seq(3, 7, by=.05), slopelikelihoods , type="l", xlab = "values of slope parameter a", ylab = "Log
```




```
startvalue = c(4,0,10)  # startvalue = (4,0,10)
chain = run_metropolis_MCMC(startvalue, 10000)  # compute the value "chain" from the funtion "run_metro


burnIn = 5000  # burnIn = 5000
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))  # compute acceptance rates

# plot the results
summary.plot(chain, burnIn,trueA,trueB,trueSd)
```




```r
# for comparison:
summary(lm(y~x))  # summaries of the results of model fitting
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.173  -5.966   1.716   7.799  16.047
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.5882     1.7641    -0.9    0.375
## x             4.9114     0.1972    24.9   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.822 on 29 degrees of freedom
## Multiple R-squared:  0.9553, Adjusted R-squared:  0.9538
## F-statistic: 620.1 on 1 and 29 DF,  p-value: < 2.2e-16
```

```r
# new function

compare_outcomes <- function(iterations){  # create a function with "iterations" as an input
```

```r
  a.matrix <- array(dim=c(10,2))  # creat an empty matrix
  colnames(a.matrix) <- c('mean', 'std')  # column names
  for(i in 1:10){  # for loop starting from 1 to 10
    startvalue <- c(runif(1,0,10), rnorm(1,0,5), runif(1,0,30))  # set a starting value
    chain <-  run_metropolis_MCMC(startvalue, iterations)  # use the function "run_metropolis_MCMC" to
    a.matrix[i,1] <- mean(chain[,1])  # compute the mean of the values in the chain for a
    a.matrix[i,2] <- sd(chain[,1])  # compute the standard deviation of the values in the chain for a
  }
  return(a.matrix)  # return the computed mean and standard deviation
}

# Test the function for 1,000, 10,000, and 100,000 iterations
compare_outcomes(1000)
```

```
##           mean       std
##  [1,] 4.789681 0.3461592
##  [2,] 4.718021 0.7076853
##  [3,] 5.331610 1.0409435
##  [4,] 4.747596 0.5963721
##  [5,] 5.495010 1.3035896
##  [6,] 4.673074 0.8734947
##  [7,] 4.837919 0.4190807
##  [8,] 4.831589 0.1966257
##  [9,] 4.719721 0.4989302
## [10,] 4.396092 1.2077150
```

```r
compare_outcomes(10000)
```

```
##           mean       std
##  [1,] 4.931565 0.2238603
##  [2,] 4.876886 0.3656853
##  [3,] 4.908385 0.2416311
##  [4,] 4.904625 0.2476450
##  [5,] 4.907536 0.2291882
##  [6,] 4.907507 0.2297831
##  [7,] 4.891732 0.2786114
##  [8,] 4.912359 0.2057280
##  [9,] 4.947299 0.2964820
## [10,] 4.863630 0.3968799
```

```r
compare_outcomes(100000)
```

```
##           mean       std
##  [1,] 4.907988 0.2141141
##  [2,] 4.910638 0.2131626
##  [3,] 4.913832 0.2069193
##  [4,] 4.915440 0.2319948
##  [5,] 4.910926 0.2097763
##  [6,] 4.909756 0.2113765
##  [7,] 4.906740 0.2426225
##  [8,] 4.918152 0.2090453
##  [9,] 4.905929 0.2086539
## [10,] 4.911696 0.2089489
```

These results are the mean and the standard deviation of the values in the chain for a with 1000,10000, and 100000 iterations. As the number of iteration increases, the mean of the values in the chain for a converge to trueA, which is 5, and the standard deviation of the values in the chain for a become smaller. In other words, the accuracy of this algorithm increases as the number of iteration increases.