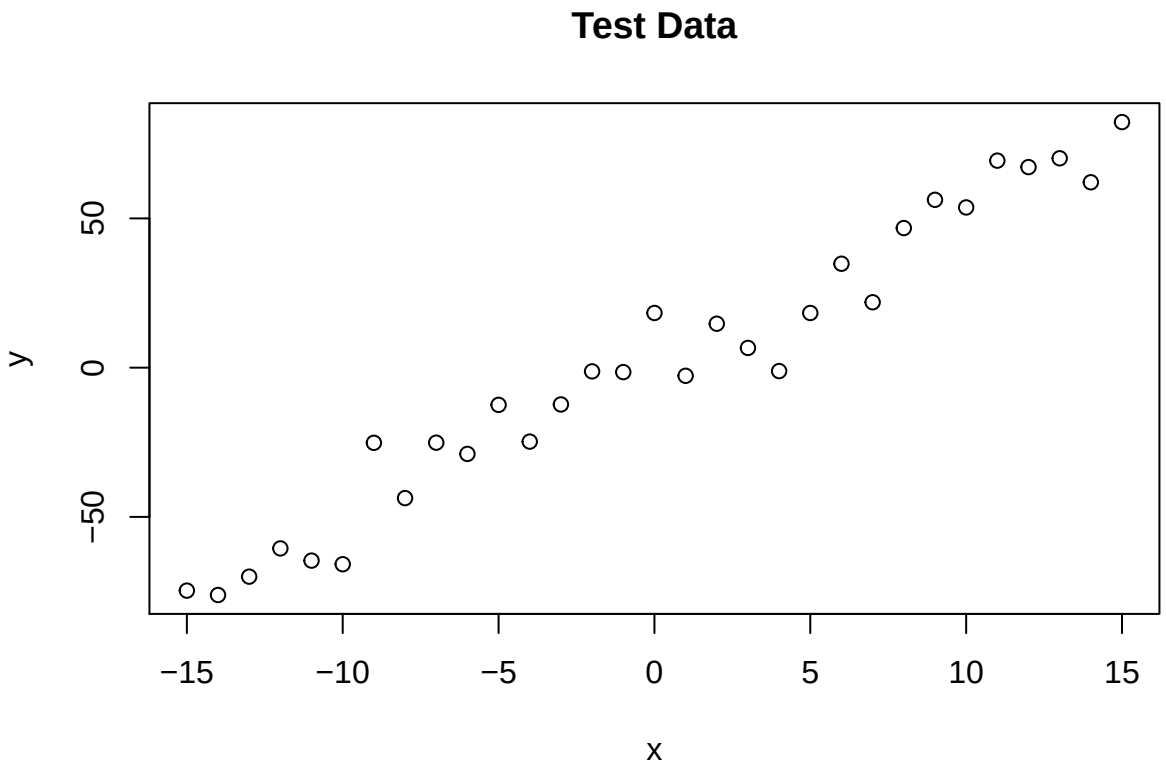
Assignment2

Suchan Park

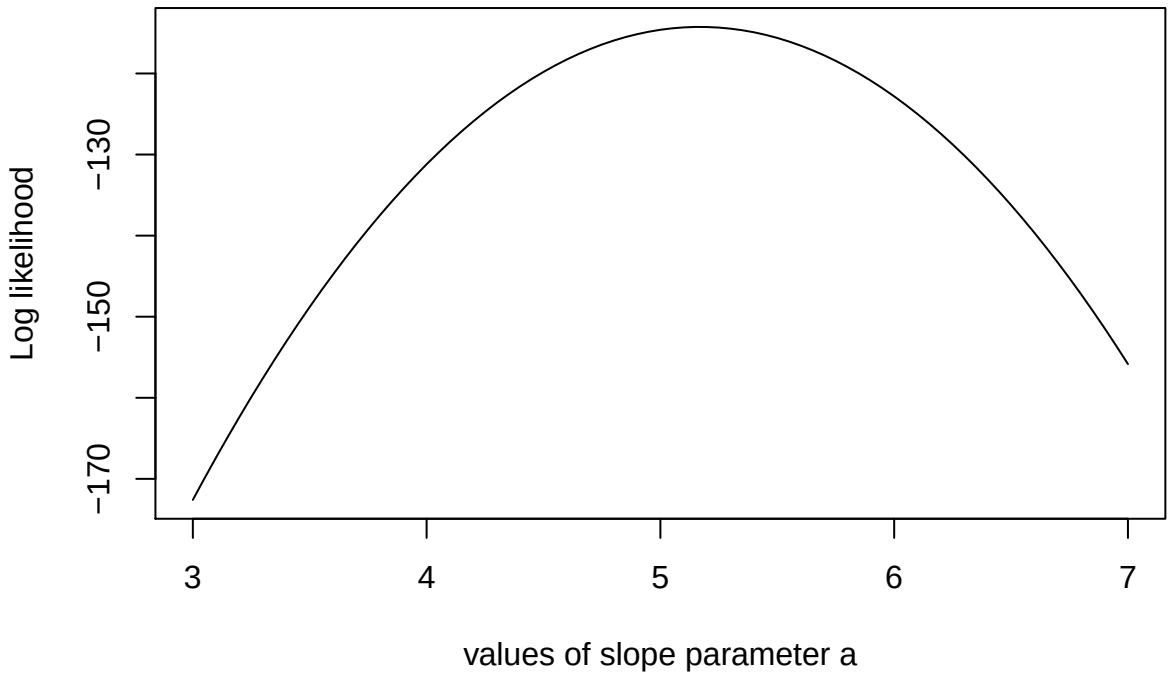
2017 10 14

Main Source

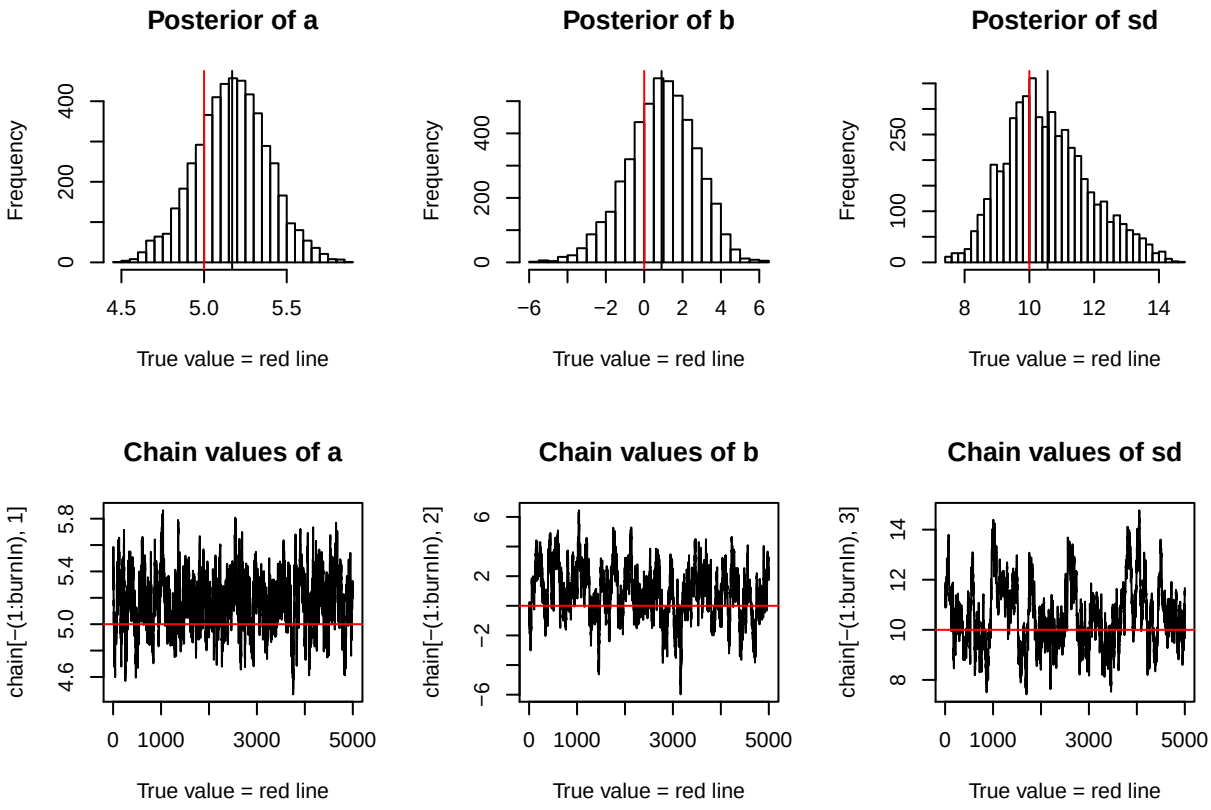
```
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/likelihood.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/slopevalues.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/prior.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/proposalfunction.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/posterior.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/run_mh_mcmc.R")
source("~/Desktop/2017-Autumn/Github/assignment-2-sucpark/Summary.R")

#  Data structure
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)

par(mfrow=c(1,1))
plot(x,y, main="Test Data")
```



```r
# Example: plot the likelihood profile of the slope a

#slopevalues function works for each element in the prior vector.
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 likelihood")
```




```r
# The MCMC : Metropolis algorithm

startvalue = c(4,0,10)
chain = run_metropolis_MCMC(startvalue, 10000)

burnIn = 5000 # Discard 5000 data when computing acceptance.
#Compute the rejection rate of proposal function
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))

# Summary:
Summary(chain, burnIn,trueA,trueB,trueSd)
```




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

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.832  -6.830   1.825   4.280  20.327
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.0133     1.7793   0.569    0.573
## x             5.1689     0.1989  25.982   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.907 on 29 degrees of freedom
## Multiple R-squared:  0.9588, Adjusted R-squared:  0.9574
## F-statistic: 675.1 on 1 and 29 DF,  p-value: < 2.2e-16
```

```
#Make a new function, compare_outcomes that takes as input an iteration number.
#Should loop 10 times.
#Each time, it should initialize the MCMC chain with randomly selected startvalue
#After each loop, the function should compute mean std for a, and print.
```

```r
compare_outcomes<-function(n)
{
  mean_a_v<-c()
  sd_a_v<-c()
  for (i in c(1:10))
  {
    a<-rnorm(1,mean=4,sd=2) #Initialize startvalue for MCMC
    b<-rnorm(1,mean=0,sd=2)
    sd<-rnorm(1,mean=10,sd=2)

    startvalue1<-c(a,b,sd) #set a new start value
    chain1=run_metropolis_MCMC(startvalue1, n)
    mean_a<-mean(chain1[,1]) #Compute the mean of a in chain1
    sd_a<-sd(chain1[,1]) #Compute the sd of a in chain1
    mean_a_v<-c(mean_a_v,mean_a)
    sd_a_v<-c(sd_a_v,sd_a)
    print(paste("Mean a : ", mean_a,", Sd of a : ", sd_a))
    #print mean(a) and sd(a)
  }
  print(paste("The Mean of Mean_a     : ",mean(mean_a_v)))
  print(paste("The Variance of Mean_a : ",var(mean_a_v)))
  print(paste("The Variance of Sd_a   : ",var(sd_a_v)))
}
```

```r
compare_outcomes(1000) #Computes in 1000 iterations
```

```
## [1] "Mean a :  5.1142337215701 , Sd of a :  0.206212283884563"
## [1] "Mean a :  5.18879928292781 , Sd of a :  0.194592275254575"
## [1] "Mean a :  4.97844035941668 , Sd of a :  0.696944184745483"
## [1] "Mean a :  5.15179739850818 , Sd of a :  0.207193560048858"
## [1] "Mean a :  5.1756755079504 , Sd of a :  0.224354148192256"
## [1] "Mean a :  5.2390953579361 , Sd of a :  0.284704462923551"
## [1] "Mean a :  5.2585887004263 , Sd of a :  0.284950658914714"
## [1] "Mean a :  5.16864365474187 , Sd of a :  0.598464193001839"
## [1] "Mean a :  5.10319606885626 , Sd of a :  0.301979171828145"
## [1] "Mean a :  5.15275514172892 , Sd of a :  0.174297178116643"
## [1] "The Mean of Mean_a     :  5.15312251940626"
## [1] "The Variance of Mean_a :  0.00611750618729138"
## [1] "The Variance of Sd_a   :  0.0326818302393566"
```

```r
compare_outcomes(10000) #Computes in 10000 iterations
```

```
## [1] "Mean a :  5.16300577909472 , Sd of a :  0.213941353816693"
## [1] "Mean a :  5.18198062629495 , Sd of a :  0.206148799886696"
## [1] "Mean a :  5.18427203271265 , Sd of a :  0.220348396562094"
## [1] "Mean a :  5.14424053240589 , Sd of a :  0.209378125501507"
## [1] "Mean a :  5.16437967164219 , Sd of a :  0.215237485612593"
## [1] "Mean a :  5.14898452603919 , Sd of a :  0.263360743797942"
## [1] "Mean a :  5.16396676913064 , Sd of a :  0.305871408895399"
## [1] "Mean a :  5.14131192764973 , Sd of a :  0.263722434764213"
## [1] "Mean a :  5.15282188117791 , Sd of a :  0.285409064834342"
## [1] "Mean a :  5.17548774090501 , Sd of a :  0.221573272790498"
## [1] "The Mean of Mean_a     :  5.16204514870529"
## [1] "The Variance of Mean_a :  0.000231638411423817"
## [1] "The Variance of Sd_a   :  0.0012898423051625"
```

```r
compare_outcomes(100000) #Computes in 100000 iterations
```

```
## [1] "Mean a :  5.16290468702906 , Sd of a :  0.240751992114248"
## [1] "Mean a :  5.16475918500601 , Sd of a :  0.214045950003683"
## [1] "Mean a :  5.1637686303961 , Sd of a :  0.215040575241378"
## [1] "Mean a :  5.16468211222543 , Sd of a :  0.216643906681719"
## [1] "Mean a :  5.16682430229392 , Sd of a :  0.20967086943597"
## [1] "Mean a :  5.16748268731315 , Sd of a :  0.215232706038693"
## [1] "Mean a :  5.17040553503665 , Sd of a :  0.217969024267466"
## [1] "Mean a :  5.17435050084769 , Sd of a :  0.209311374090999"
## [1] "Mean a :  5.17358911456262 , Sd of a :  0.209245929944395"
## [1] "Mean a :  5.16562267708319 , Sd of a :  0.207749003677912"
## [1] "The Mean of Mean_a     :  5.16743894317938"
## [1] "The Variance of Mean_a :  1.63205910814359e-05"
## [1] "The Variance of Sd_a   :  9.09877713016561e-05"
```

From the result, we can find that mean value of "a"" in chain is similar but as an interation number increases, the variance of "a"" in chain value is decreased. Therefore, it is pretty safe to say that more iteration number can make a stable and accurate outcome.