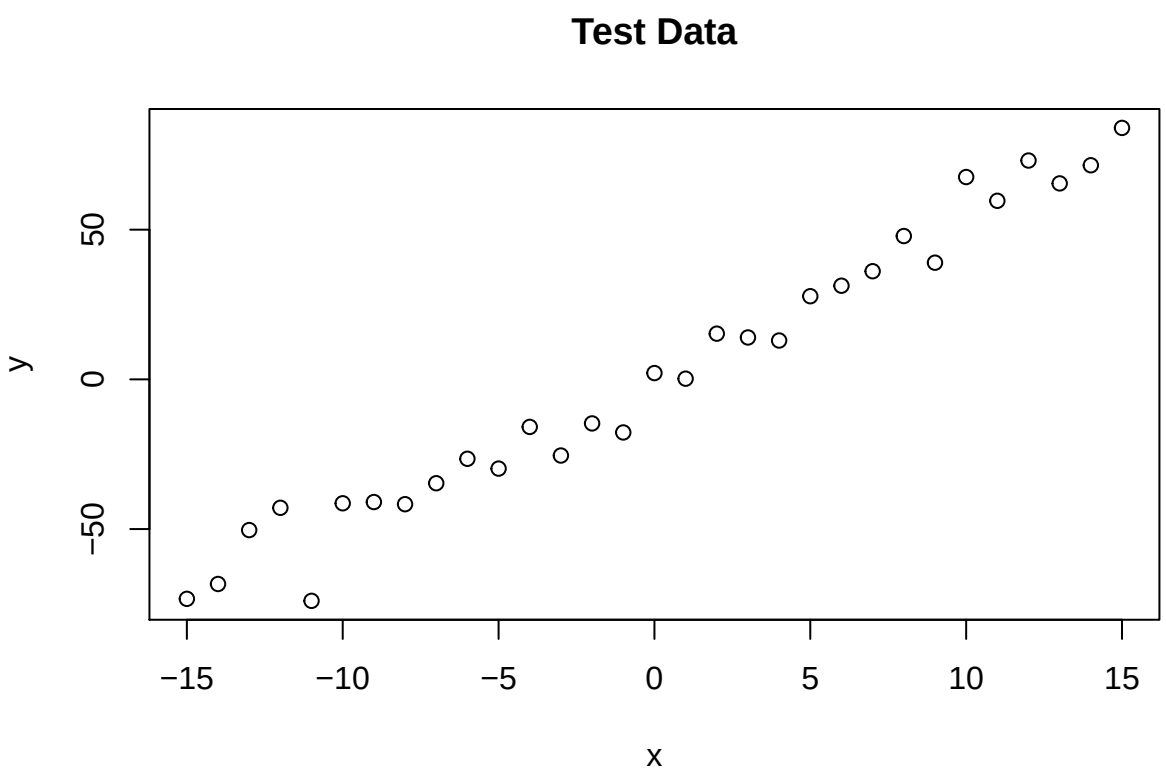
Assignment2 37810

Yanfei Zhou

21/10/2018

```
########## Creating test data ###############
#set true values to parameter A
trueA <- 5
#set true values to parameter B
trueB <- 0
#set true values to parameter sd
trueSd <- 10
#set sample size
sampleSize <- 31
# create independent x-values, in this case from -15 to 15
x <- (-(sampleSize-1)/2):((sampleSize-1)/2)
# create dependent values according to ax + b + N(0,sd), with parameters all equal to their true values
y <-  trueA * x + trueB + rnorm(n=sampleSize,mean=0,sd=trueSd)
#plot y and x, we can see a clear linear relationship
plot(x,y, main="Test Data")
```




```
source("functions.R")


# Example: plot the likelihood profile of the slope a
# define a function which inputs a slope parameter x, calculate the likelihood for x, with real paramet
#slopevalues <- function(x){return(likelihood(c(x, trueB, trueSd)))}
# apply the above function to a range of x values (from 3 to 7 by 0.05), calculate the likelihood at ea
```

```
#slopelikelihoods <- lapply(seq(3, 7, by=.05), slopevalues )
# plot the loglikelihood for x, we see the plot peaks at around 5, which is equal to the true value we
#plot (seq(3, 7, by=.05), slopelikelihoods , type="l", xlab = "values of slope parameter", ylab = "Log
```

```
#example
#startvalue = c(4,0,10)
#chain = run_metropolis_MCMC(startvalue, 10000)
# set a burnIn value as the first steps of the algorithm may be biased by initial value, thus we discar
#burnIn = 5000
#calculate the acceptance rate by 1-rejection rate
#acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
```

```
compare_outcomes<-function(num_iterations){
  #set the number of first iterations to be discarded
  burnIn=0.5*num_iterations
  for (i in 1:10){
    #randomly generated start values for a, b, sd
    startvalue = c(runif(1)*5,runif(1,-1,1),runif(1)*10)
    chain = run_metropolis_MCMC(startvalue = startvalue,iterations = num_iterations)
    mean = mean(chain[-(1:burnIn),1])
    sd = sd(chain[-(1:burnIn),1])
    print(c("the mean is: ",mean,"sd is: ",sd))
  }
}
```

```
set.seed(2)
#iteration number = 1000
compare_outcomes(1000)
```

```
## [1] "the mean is: "     "5.13943247855915"  "sd is: "
## [4] "0.180626289796792"
## [1] "the mean is: "     "5.07192844370662"  "sd is: "
## [4] "0.168440894913973"
## [1] "the mean is: "     "5.10958969016211"  "sd is: "
## [4] "0.147461779711624"
## [1] "the mean is: "     "5.11359615378143"  "sd is: "
## [4] "0.174852015709628"
## [1] "the mean is: "     "5.11272832954324"  "sd is: "
## [4] "0.157069556320639"
## [1] "the mean is: "     "5.10610316541612"  "sd is: "
## [4] "0.165633981546869"
## [1] "the mean is: "     "5.08538973823158"  "sd is: "
## [4] "0.180878125972246"
## [1] "the mean is: "     "5.06212962877053"  "sd is: "
## [4] "0.146053444307803"
## [1] "the mean is: "     "5.11320837394197"  "sd is: "
## [4] "0.177416133054218"
## [1] "the mean is: "     "5.12175679441018"  "sd is: "
```

```
## [4] "0.149457066138375"
```

```
set.seed(2)
#iteration number = 10000
compare_outcomes(10000)
```

```
## [1] "the mean is: "     "5.08980112142055"  "sd is: "
## [4] "0.177525457452382"
## [1] "the mean is: "     "5.10892096961104"  "sd is: "
## [4] "0.180911429381916"
## [1] "the mean is: "    "5.11034529644288" "sd is: "
## [4] "0.19415561139517"
## [1] "the mean is: "     "5.11445158487826"  "sd is: "
## [4] "0.170835635355596"
## [1] "the mean is: "     "5.09573928744216"  "sd is: "
## [4] "0.186119494934382"
## [1] "the mean is: "     "5.0969690205367"   "sd is: "
## [4] "0.170232851068362"
## [1] "the mean is: "     "5.0865336297718"   "sd is: "
## [4] "0.180457567001956"
## [1] "the mean is: "     "5.11083601890564"  "sd is: "
## [4] "0.166531537253901"
## [1] "the mean is: "     "5.11538613470281"  "sd is: "
## [4] "0.173190131425111"
## [1] "the mean is: "     "5.10990615511627"  "sd is: "
## [4] "0.199233692188262"
```

```
set.seed(2)
#iteration number = 100000
compare_outcomes(100000)
```

```
## [1] "the mean is: "     "5.10485617944128"  "sd is: "
## [4] "0.173616536765679"
## [1] "the mean is: "     "5.09787386697708"  "sd is: "
## [4] "0.174270535270696"
## [1] "the mean is: "     "5.10530496628017"  "sd is: "
## [4] "0.180303267945173"
## [1] "the mean is: "     "5.10128727463537"  "sd is: "
## [4] "0.180599681230759"
## [1] "the mean is: "     "5.09689430914041"  "sd is: "
## [4] "0.174333075884306"
## [1] "the mean is: "     "5.10701526594322"  "sd is: "
## [4] "0.172917616850751"
## [1] "the mean is: "    "5.10531046941457" "sd is: "
## [4] "0.17303197866535"
## [1] "the mean is: "     "5.10525000955901"  "sd is: "
## [4] "0.175050690872141"
## [1] "the mean is: "     "5.10370713470615"  "sd is: "
## [4] "0.175176055999182"
## [1] "the mean is: "     "5.10284264324904"  "sd is: "
## [4] "0.180493411453126"
```

We can see that the accuracy of this algorithm in finding a increases as we increase the number of iteration, but the standard deviation does not change much.