Report on the accuracy of metropolis-hasting algorithm

When finding the value of a in linear model using metropolis-hasting algorithm, different number of iterations would lead to different accuracy on the result. Here, we define a function called "compare outcomes" to evaluate the difference.

Original settings

We assume a linear relationship between x and y, with noises having a normal distribution. Thus, the test data could be created as follow.

```
trueA <- 5
trueB <- 0
trueSd <- 10
sampleSize <- 31
x <- (-(sampleSize - 1) / 2): ((sampleSize - 1) / 2)
y <-  trueA * x + trueB + rnorm(n = sampleSize, mean = 0, sd = trueSd)
```

Define the function

First, we show the codes for the function. By using this function, we could get the mean and standard deviation of a in the chain.

```
source("/Users/liyuhan/Documents/GitHub/assignment-2-liyuhan529/Functions.R")
compare_outcomes=function(iteration){
  result_a=matrix(rep(0,20),ncol=2)
  #create the matrix that would save the result later

  for(i in 1:10){
    startvalue=c()
    startvalue[1]=runif(1,0,10)
    startvalue[2]=rnorm(1,0,5)
    startvalue[3]=runif(1,0,30)
    #determine the randomly selected start value for each loop, and save them into a vector

    chain=run_metropolis_MCMC(startvalue,iteration)
    #run the MCMC alagorithm defined previously, and save the result of each iteration in "chain"

    result_a[i, ] <- c(mean(chain[,1]), sd(chain[,1]))
    #save the mean value and sd of a in the result matrix
  }
  print(result_a)
}
```

Using the function to analyse accuracy

```
compare_outcomes(1000)
```

```
##            [,1]      [,2]
##  [1,] 4.391288 1.0277378
##  [2,] 5.036214 0.4074550
##  [3,] 5.029487 0.3761999
##  [4,] 5.140018 0.7366457
##  [5,] 4.989154 0.2451149
##  [6,] 4.872093 0.5726556
##  [7,] 4.870509 0.2745723
##  [8,] 4.665159 0.8079014
##  [9,] 4.989907 0.3392078
## [10,] 4.541498 1.0669030
```

```r
compare_outcomes(10000)
```

```
##            [,1]      [,2]
##  [1,] 4.935428 0.3007054
##  [2,] 4.949490 0.2409693
##  [3,] 4.986681 0.2329650
##  [4,] 4.960732 0.2883314
##  [5,] 4.987888 0.3865314
##  [6,] 4.993460 0.4652156
##  [7,] 4.921342 0.2905523
##  [8,] 4.943270 0.2835192
##  [9,] 4.932103 0.3081711
## [10,] 4.964830 0.2731777
```

```r
compare_outcomes(100000)
```

```
##            [,1]      [,2]
##  [1,] 4.953035 0.2329465
##  [2,] 4.951972 0.2302192
##  [3,] 4.953010 0.2408359
##  [4,] 4.952675 0.2450786
##  [5,] 4.952151 0.2332177
##  [6,] 4.954404 0.2403205
##  [7,] 4.954140 0.2380170
##  [8,] 4.946117 0.2350797
##  [9,] 4.951231 0.2458017
## [10,] 4.959137 0.2362687
```

The three above matrices shows the mean and standard deviation of a when conducting 1000,10000 and 100000 iterations. The first column refers to the mean of a, and the second column refers to the standard deviation. The ten rows in each matrix shows the result of each loop under different number of iterations.

The result matrix shows that as the times of iteration gets larger, the standard deviation of a gets lower significantly, showing that the value of a is becoming more stable. Additionally, the value of a tends to get closer to its true value ($a = 5$ in this case), when iterating for more times. In summary, the accuracy of the algorithm would increase significantly when the number of iteration gets larger.