CSE 151: Programming Assignment #1

Due on Monday, April 11, 2016

Mangione-Tran,Carmine 9AM

Kyle Lee, Jaehee Park
A01614951, A11287366

Graphs and Diagrams

Figure 1: N-iterations vs. Normalized Mean & SD

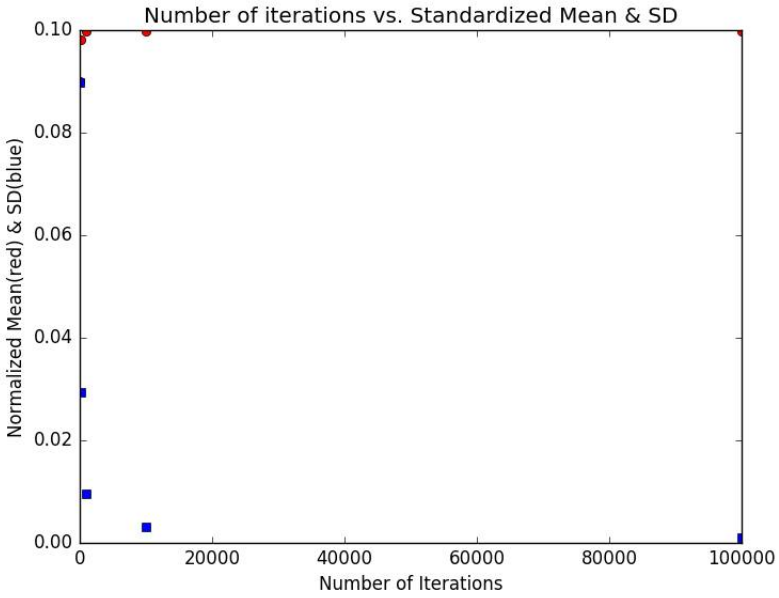


Figure 2: Log-Plot of N-iterations vs. Normalized Mean & SD

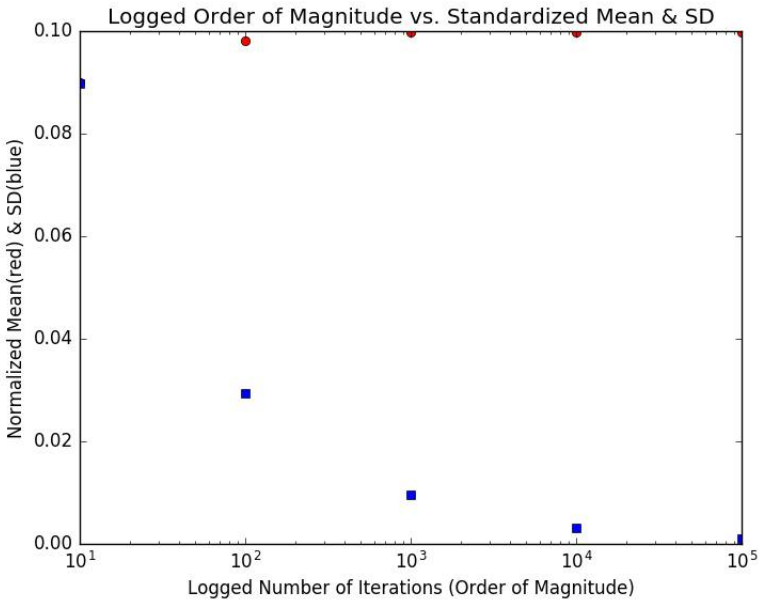


Figure 3: N-iterations vs. Normalized Mean

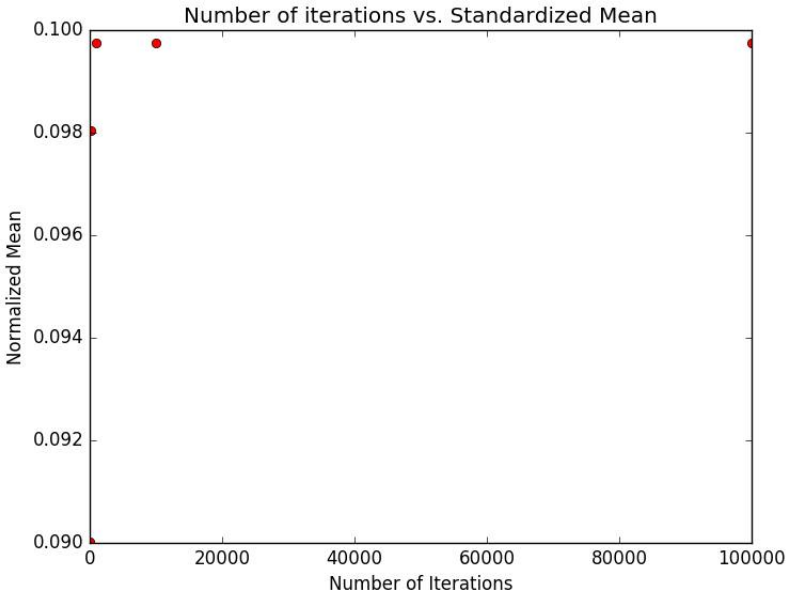
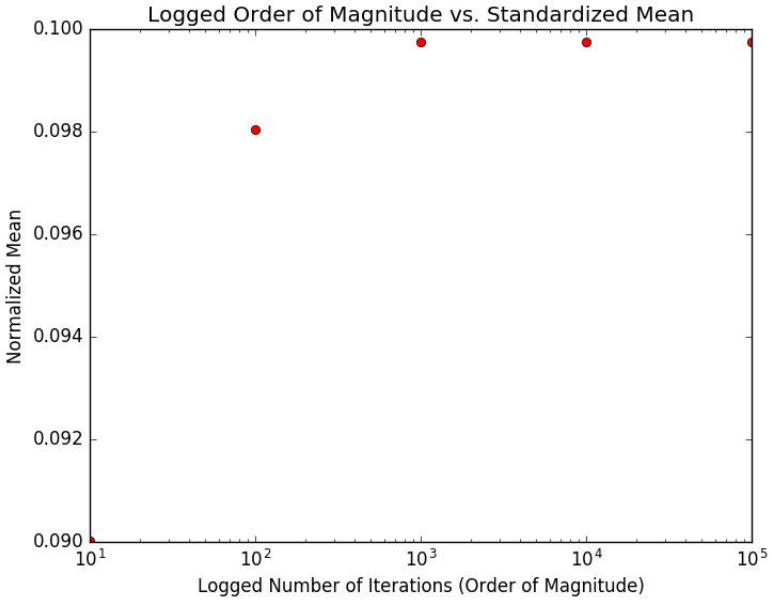


Figure 4: Log-Plot of N-iterations vs. Normalized Mean



By the **Law of Large Numbers**, we expect the mean to converge to the true mean as well as the standard deviation to approach 0.

Homework Code (Python)

```
import random
import csv
import math
from decimal import *
import numpy as np
import matplotlib.pyplot as plt

#read the csv file
inputFile = open('abalone.data')
inputReader = csv.reader(inputFile)
inputData = list(inputReader)    #inputData = list of our data (which is in lists)

#our set hit-rate
testSize = 0.1
#data size
size = len(inputData)
#set a random seed
random.seed(123451)

#initialize arrays
counter = [] #stores hit rates on index
meanA = [] #stores average means of trials
SDA = [] #stores standard deviation of trials

#method for generating hits
#@param:x => our data
#      :testSize => hit rate
#      :genList => list of # of hits per index
#@return:genList => updated list with hits per index
def count (x, testSize, genList):
    size = len(x)
    expectedDraws = int(round(size*testSize))
    x1 = Decimal(expectedDraws)/Decimal(size)

    j = 0

    for i in range(0, size):
        x2 = random.uniform(0,1)
        if x2 < x1:
            genList[i] = genList[i] + 1
            j = j+1
        x1 = Decimal(expectedDraws-j)/Decimal(size-i)
    return genList

#initialize counter array
for x in range(size):
    counter.append(0)

#Trial of 10 iterations
for i in range(1,10):
    counter = count(inputData, testSize, counter)
```

```
#store mean & standard deviation data 10
mean10 = np.average(counter)/10
sd10 = np.std(counter)/10
meanA.append(mean10)
SDA.append(sd10)

#Trial for 100 iterations
for i in range(11,100):
    counter = count(inputData,testSize, counter)
#store mean & standard deviation data 100
mean100 = np.average(counter)/100
sd100 = np.std(counter)/100
meanA.append(mean100)
SDA.append(sd100)

#Trial for 1,000 iterations
for i in range(101, 1000):
    counter = count(inputData, testSize, counter)
#store mean & standard deviation data for 1,000
mean1000 = np.average(counter)/1000
sd1000 = np.std(counter)/1000
meanA.append(mean1000)
SDA.append(sd1000)

#Trial for 10,000 iterations
for i in range(1001,10000):
    counter = count(inputData, testSize, counter)
#store mean & standard deviatoin data for 10,000
mean10000 = np.average(counter)/10000
sd10000 = np.std(counter)/10000
meanA.append(mean10000)
SDA.append(sd10000)

#Trial for 100,000 iterations
for i in range(10001, 100000):
    counter = count(inputData, testSize, counter)
#store mean & standard deviation data for 100,000
mean100000 = np.average(counter)/100000
sd100000 = np.std(counter)/100000
meanA.append(mean100000)
SDA.append(sd100000)

#plot mean & standard deviations
plt.figure(0)
plt.plot([10,100,1000,10000,100000], meanA,'ro')
plt.plot([10,100,1000,10000,100000], SDA,'bs')
plt.title('Number of iterations vs. Standardized Mean & SD')
plt.xlabel('Number of Iterations')
plt.ylabel('Normalized Mean(red) & SD(blue)')
plt.show()

#plot mean & standard deviations w/ logged iteration values
plt.figure(1)
```

```
plt.plot([10,100,1000,10000,100000], meanA,'ro')
plt.plot([10,100,1000,10000,100000], SDA,'bs')
plt.xscale('log')
plt.xlabel('Logged Number of Iterations (Order of Magnitude)')
plt.ylabel('Normalized Mean(red) & SD(blue)')
plt.title('Logged Order of Magnitude vs. Standardized Mean & SD')
plt.show()

#plot mean only
plt.figure(2)
plt.plot([10,100,1000,10000,100000], meanA,'ro')
plt.title('Number of iterations vs. Standardized Mean')
plt.xlabel('Number of Iterations')
plt.ylabel('Normalized Mean')
plt.show()

#plot mean only with logged iteration values
plt.figure(3)
plt.plot([10,100,1000,10000,100000], meanA,'ro')
plt.xscale('log')
plt.xlabel('Logged Number of Iterations (Order of Magnitude)')
plt.ylabel('Normalized Mean')
plt.title('Logged Order of Magnitude vs. Standardized Mean')
plt.show()
```