This week you should read Chapter 18 Random Numbers of the text book, watch the videos and do the video exercises for the following Learning Modules: 25 Numpy, 26 Scipy, and 27 Random Slicing.  Finally, you should complete Homework Assignment 10 which is due April 16th, and is found in the Assessments tab.  Assignment 10 will be posted later today.

Professor Ham

[**Assignment 10**](https://mymasonportal.gmu.edu/webapps/assignment/uploadAssignment?content_id=_7786194_1&course_id=_331470_1&group_id=&mode=view)

**Description:**

Python Data and Computations in the book, Computational Methods for Bioinformatics: Python 3.4, Third Edition by Jason Kinser.

Part 1

Problems 1 through 7 for Chapter 18

Part 2

Worksheet 2 and 3 exercises for learning module 25 Numpy

Part 3

Worksheet exercise for learning module 26 Scipy

Part 4

Worksheet exercise for learning module 27 Random Slicing

This is an individual assignment and should be your own work.

**Format:** Please submit the assignment in pdf or MSWord compatible format. Please show code used and output of the code in your submission.

**Point value:**

Part 1) 21 points (3 points per problem)

Part 2) 10 points (2 points per problem)

Part 3) 6 points total

Part 4) 4 points total

**Due Date:** April 16, 2018

Problems

1. Compute the average of sets of random numbers. The number of samples in the sets should be 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048 and 4096. Plot the average of the random values in each set versus the number of samples.

import numpy as np

samplenumbers=[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]

averages=[]

for n in samplenumbers:

averages.append(np.random.rand(n).mean())

print(samplenumbers)

print(averages)

*[2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]*

*[0.39603629631307374, 0.5264389438245813, 0.4213672402464441, 0.5663441021775738, 0.5177941810314792, 0.43905270708739663, 0.48414923994275866,*

*0.4951034247513098, 0.49717909029772556, 0.5039133940527643, 0.485678852371189, 0.48587576668544386]*

2. Compute the average of 10,000 samples of x^2 where x represents random numbers.

def prob2():

rands=np.random.rand(10000)

randsqr=[]

for i in rands:

randsqr.append(i\*i)

avg=np.array(randsqr).mean()

print(avg)

prob2()

*0.32587593533683673*

3. Compute the average of 10,000 samples of √ x where x represents random numbers. Is the result the same as √ 0.5?

def prob3():

rands=np.random.rand(10000)

randsqr=[]

for i in rands:

randsqr.append(math.sqrt(i))

avg=np.array(randsqr).mean()

print(avg)

print(math.sqrt(0.5))

prob3()

*0.6593395575100456*

*0.7071067811865476*

No, but they are very close!

4. Plot the histogram of 10,000 samples from a normal distribution with µ = 0.5 and σ = 0.3.

def prob4():

rands=np.random.normal(0.5, 0.3, 10000)

y, x = np.histogram(rands)

print(x)

print(y)

prob4()

*[-0.58649164 -0.3670978 -0.14770395 0.07168989 0.29108374 0.51047758*

*0.72987142 0.94926527 1.16865911 1.38805296 1.6074468 ]*

*[ 15 139 583 1626 2733 2679 1606 497 107 15]*

5. Plot the histograms of two normal distributions. The first has 10,000 samples with µ = 0.5 and σ = 0.4. The second has 9,000 samples with µ = 0.3 and σ = 0.2. What is the value of x where the two distributions cross over?

def prob5():

rands=np.random.normal(0.5, 0.4, 10000)

y, x = np.histogram(rands)

print(x)

print(y)

rands=np.random.normal(0.3, 0.2, 9000)

y, x = np.histogram(rands)

print(x)

print(y)

prob5()

*[-0.94865552 -0.65613039 -0.36360527 -0.07108014 0.22144498 0.51397011*

*0.80649523 1.09902036 1.39154548 1.68407061 1.97659573]*

*[ 15 139 583 1626 2733 2679 1606 497 107 15]*

*[-0.46774201 -0.32094188 -0.17414175 -0.02734162 0.11945851 0.26625865*

*0.41305878 0.55985891 0.70665904 0.85345917 1.00025931]*

*[ 13 77 404 1177 2216 2513 1730 687 158 25]*

6. Create a random DNA string with 1000 letters, but the probability of having an ’A’ is twice as much as the other three letters.

def prob6():

dna = list("AATCG")

dnastring = "".join(np.random.choice(dna, 10000))

print(dnastring)

prob6()

*AGAACGAAATAAATAGCAGCACTAAGACCAGAGAAGCTACGTAATAACACGAGTAAGGAATGTAATCAGGAAAGATACATAAAATCAGTACCATAATAAAAAATGTAAGAAAAGGATCCCTTCACACCACAAAACCTAAGCAAAAAGGCAACAGGGAGTATATACTCGGATTAATATTAGACTCCAGATAGGGTAGACGTGAAACTAAACTAGTACGATCGGAATGCACACACATCAATGACATAAGCGAACGAGAAATCTACTGCAACTATATTTGTAACAACCTAAAAAAACAATACGTAAATACAAAGCGGGCGGAATAATGGTTAAAATAAATTAAACACATAATTTAGAGCGCAGTCTTAGATAGTTCATATCTGGCTCCATATTGCACCATAAATCAGTATAAACAAATAGTACTAGCAAAACAATGTAATATAAGTAGAGTAAGCCAAATTCTAGCCTCGGTCAATAAGGGAACAAATTCCAATACAAAATGTAAATCGTGAAAGCAGGTATAAACCCTAAAAAATGAGGTAATCACTGATAAAGTAAACATTGAATTATACAGTGAAGGCCGGTCACTATAGTTTAATAGCGACGAAAATAATACACAAGATAAACCCACACCTTCGCAGTTTGTTCAAAAGAACGAACAAGTGGAAGTTACAGAATGAACGAAAAGGTGACAGAATAAGAACTAGTAATGTACAAGGCTCACTCAAGCAGTGAAGTGCACGGAGCAAACCTGACTGAGCAACCTGACAGTCCTCACACTGAAATAGGACTATTCCGCGATCCGGGTAAAGGAACCGTCTAGTTAAGTGCAAAAAGGAATTAGATAGATCTAATACTAGCGCGTTCAGGAAGGATTCACAATATGACGGAGGGGTATAGATAGAGCATAAAGGCAGACCATAGACACGCAGAAAAGTTTGGCATGACATGGTCGATAATAGATAATTTTCGTTTTATTATAGAAGAGGTAATAAGTCAGCAG*

7. Create a random amino acid string with 1000 letters.

def prob7():

aas="A R N D B C E Q Z G H I L K M F P S T W Y V".split()

proteinstring="".join(np.random.choice(aas, 1000))

print(proteinstring)

prob7()

