

BIO 480/680 Lab Exercise 9

Computers and random numbers

Due date: 18 April 2025

This exercise is intended to get you started working with the computer random number generator that we will need for an upcoming model of biological processes involving random events. There are two parts to this exercise that involve using a random number generator in different ways and testing if the results seem to fit our concept of randomness. You should choose one of these two parts to carry out for your lab project.

If you want to generate a random number between 0 and 1 and optionally scale it to some other range (see lecture slides):

```
x = random.random()      # Produce a random float between 0-1
```

You might want to generate a random integer within some range:

```
x = random.randint(0, 9) # Produce a random integer 0-9
```

1) One of our expectations for the random number generator is that it produces numbers which we called “uniform deviates”. In other words, every possible number is produced with equal likelihood. So if we generate a large set of random integers, for example, every number should tend to occur with equal frequency. The first part of this exercise is to create a simple program that generates a large number of random integers between 0 and 9. Keep track of how many times each one of those ten possible integers appears. There are many ways to carry out this exercise, but one possibility would use a series of “if” statements to count how often each integer appears. This example counts the number of times “2” appears:

```
number = 100
x = np.zeros(number, dtype=int)
count2 = 0
for i in range(number):
    x = random.randint(0,9)
    if x == 2 :
        count2 += 1
```

Then draw a graph showing the frequency of appearance on the y-axis versus the generated value on the x-axis. Repeat this process for sets of 100, 1000, and 10,000 random integers and see if your random number generator really seems to be “uniform”. One good way to plot all the data on one graph would be to express the frequency of appearance on the y-axis as a fraction of the total numbers generated: if you generate 1000 random integers between 0 and 9 and the value 2 appears 120 times, you would plot this value as 120/1000 and so forth. If you draw three separate graphs, make sure

that you draw all three graphs with the same y-axis range (possibly, 0 to 0.16) so as to make the pattern in your results more apparent. Do your results suggest uniformity?

2) The second part of the exercise is a test for sequential correlation. Every time the random number generator produces a number, the next number it produces should not be related (correlated) to the previous number. The easiest way to do this might be to create a loop that runs, say, 100 times. Each time around the loop, generate two random numbers between 0 and 1 – these should be floating point numbers, not integers. Store the first number in an array called `x1[]` and the second number in an array called `x2[]`. Then save these two arrays to a disk file. Import the file into your graphing software and plot `x2` versus `x1` as an xy-scatter plot. Now repeat this process for 100, 1000, and 10,000 pairs of random numbers. If the random number generator is uniform and free of sequential correlations, the points should be scattered about the graph with no apparent pattern. Does your random number generator seem to pass this test?

```
number = 100
x1 = np.zeros(number)
x2 = np.zeros(number)
for i in range(number):
    x1[i] = random.random()
    x2[i] = random.random()
```