Lab 1: Randomness and Associated Algorithms

M. Muldoon <mark.muldoon@manchester.ac.uk>

Thursday 28 September 2023

The work for this week involves a set of three computer exercises described below. As many of you will just be starting to learn Python, I have provided a notebook, `Lab1.ipynb` to help guide you through the exercises. You needn't worry in detail about what the code does, but you should study it enough that you can modify it to perform variations on the simulations discussed below. You will also need a file of census data, `Data_AGE_UNIT.csv`. Both of these are available on BlackBoard alongside this sheet.

1 Generation of pseudo-random numbers

1.1 The logistic map

Consider a series of numbers $x_1, x_2, \ldots$generated using the rule

$$x_{n+1} = r x_n (1 - x_n) \tag{1}$$

If we start with $0 < x_0 < 1$, we are guaranteed that the subsequent x_i will remain in the range $0 < x_i < 1$ provided $0 < r < 4$. It turns out that if we choose r close to its upper limit, the behaviour of the sequence of x's becomes less and less "predictable". There are ways to quantify this unpredictability, but for now our aim is to develop intuition, simulate and visualise, so try different values of x_1 as well as increasing values of r such as $r = 3$, $r = 3.9$, $r = 3.99$, $r = 3.999$, etc. Plot the sequences of numbers generated, and consider how well you can predict $x_{i+1}, x_{i+2} \ldots$ given x_i for various values of r.

1.2 Use of "entropy"

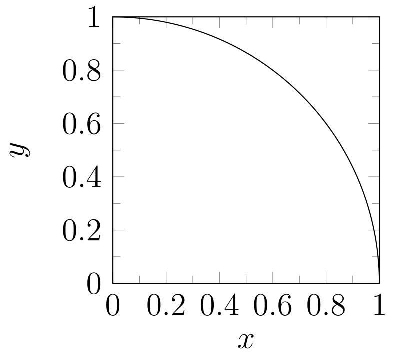
The logistic map, Eqn. (1), is an example of *deterministic chaos*, that is, it's a rule that the computer can follow, but whose results are apparently not easy to predict. An alternative approach is to generate randomness by drawing on factors *external* to the computer, for example the location and timing of mouse clicks. Here, we will consider using as an external factor the time at which the code is run. The sample code takes a particular approach to this, using the six decimal digits of the system clock's current time (which is available at microsecond resolution). Try to work out what the code is doing and decide whether you think it is sensible.

1.3 Random number generation in practice

A common approach to pseudo-random number generation in practice is for the computer to accumulate "entropy" from external events, then use these as a "seed" for a deterministic rule with good properties—for example the Mersenne Twister, which is the default random number generator in R and MATLAB. From now on, we will use the random number generation functions provided by software packages, and behave as though they are truly random; but the impossibility of generating truly random numbers remains a potential source of errors in data science.

2 Estimation of π using Monte Carlo methods

Suppose we *define* π to be the area of a circle whose radius is 1, and then want to estimate its numerical value. Here we will use a “Monte Carlo” method to estimate this number, given its definition. Consider the plot on the right—the quarter circle will have an area of $\pi/4$. Our intuition is that if we were to shoot an automatic paintball gun at this target (an experiment performed by Prof. D. Spiegelhalter and the BBC) the ratio of spots inside the quarter-circle to those inside the square would be $\pi/4$.



So our algorithm is as follows: we will pick random values of x and y independently from a uniform distribution between 0 and 1, then let the random variable Z equal 1 if the point (x, y) falls within the quarter-circle shown and 0 otherwise. This Z allows us to make an estimate of π in that its expected value, $\mathbb{E}[Z] = \pi/4$. We can then define a random variable A_n to be the average of n independent samples of Z. Formally:

$$A_n := \frac{1}{n}\sum_{i=1}^{n} Z_i = \frac{\pi}{4} + \varepsilon_n,$$

where the Z_i are copies of Z and ε_n is the *error*. But this is not much use without knowing how *confident* we can be in the estimate.

For a simple system such as this it is possible to calculate the error analytically, but in the spirit of Monte Carlo simulation, we will use a more general approach called *parametric bootstrapping*, where we make m simulations of A_n, and remove the largest and smallest $\alpha\%$ to give an estimate of the $(100 - 2\alpha)\%$ *confidence region*. Experiment with different values of m and n. In general m does not need to be large, but we want n to be as big as possible, or at least as big as is necessary to achieve a desired accuracy.

3 Working with Census Data

Finally, we will look at census data from 2011 on the distribution of ages in the Manchester Local Authority. These come in a format that requires some tidying up, which I have done for you. When these data are plotted they demonstrate a lot of “wiggles”’ (local minima and maxima) in the histogram. If developing policy for the city, one thing we might want to know is: How many of these local maxima are worth paying attention to—note, for example, the effect of students, which is certainly real and affects city policy—and how many are just noise? We will address this question using *bootstrapping* (which is related to, but different from, the parametric bootstrapping we’ve just met).

Suppose we have a sample of size n (in this case $n \approx 5 \times 10^5$, but you should work out the exact value from the data). In bootstrapping, we sample uniformly at random from the set of ages n times *with replacement* and treat this as a new dataset called the *bootstrap sample*. We take m such bootstrap samples and, for any quantity of interest that is derived from the original data, calculate it for each of the m bootstrap samples. We then remove the largest and smallest $\alpha\%$ of these to give an estimate of the $(100 - 2\alpha)\%$ confidence region for the quantity of interest.

Exercise:

- Bootstrap the data, generating m bootstrap samples.
- For each bootstrap sample, generate a histogram whose bins are one-year wide and count the number of people who are a given age (in years). Keep track of the number of people (in the bootstrap samples) who fall into each age-band. The result should be a big table whose rows are labelled by sample numbers and whose columns are labelled by ages in years: $0, 1, \ldots$.
- For each age, find a 95% confidence band for the number of people with that age. Plot these.
- Use the result to decide which local maxima appear “significant”.