Physics 77 Capstone: Logistic Map Random Number Generation

Riley Wong

August 12, 2022

Abstract

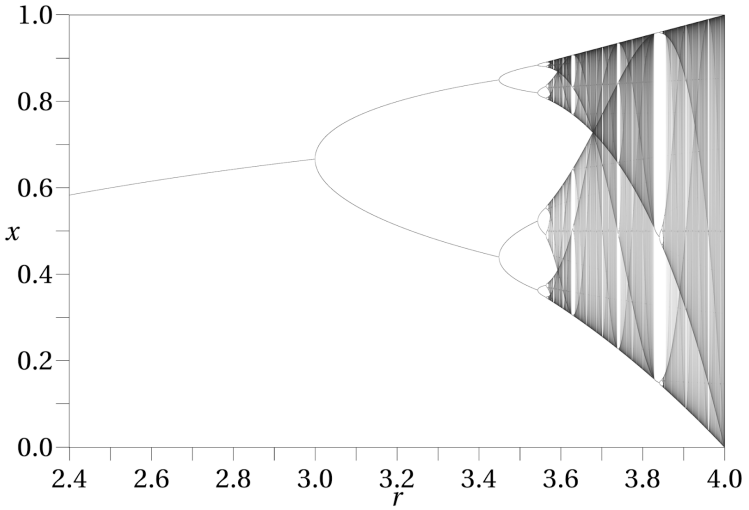
Functions with simple underlying rules can have very complex behaviors. For one, as the logistic map progresses and bifurcates at an increasing rate, the periods of oscillations become chaotic. This project will use such chaos to create a pseudo-random number generator, then test the function through a series of statistical tests to see if it is truly random.

Introduction

The logistic map is an extremely simple non-linear, first-order difference equation; it is used to model density dependent growth. The equation is as follows:

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

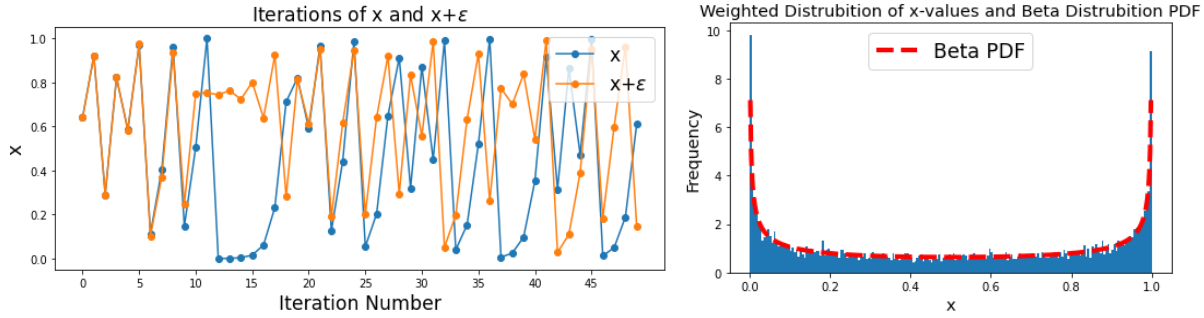
This is an iterative process. Here, x_{n+1} is the value of the next iteration yielded from the function call, x_n is the value of the current iteration, and r is the steepness parameter of this quadratic equation. Purposefully, when the current value (x_n) is low, the next iteration tends to increase, and when x_n is high, the next tends to decrease. Additionally, when r is greater than 3.57 (but less than 4), the bifurcated sequence converges and the proceeding oscillations have infinite fixed points—many describe this as the onset of chaos. However, there are exceptions: for example, when r = 3.8284, the sequence settles into a three cycle momentarily. Finally, when r = 4, the logistic map's range reaches (0, 1); because of this range, r = 4 seems to be the best parameter in generating random values.



The plot of fixed points vs. r for the logistic map (image credit: Pierce, Jordan)

Two properties of the logistic map sequence are notably relevant. First, sensitive dependence on initial conditions: chaotic systems amplify the discrepancies between initial

conditions through multiple stretches and folds. Second, while the logistic map is not uniformly distributed, research for this project has shown that when $r = 4$, the distribution follows a beta probability distribution function (PDF) with parameters a and b as 0.5 (the exact formula is unimportant as the scipy.stats.beta function is used).

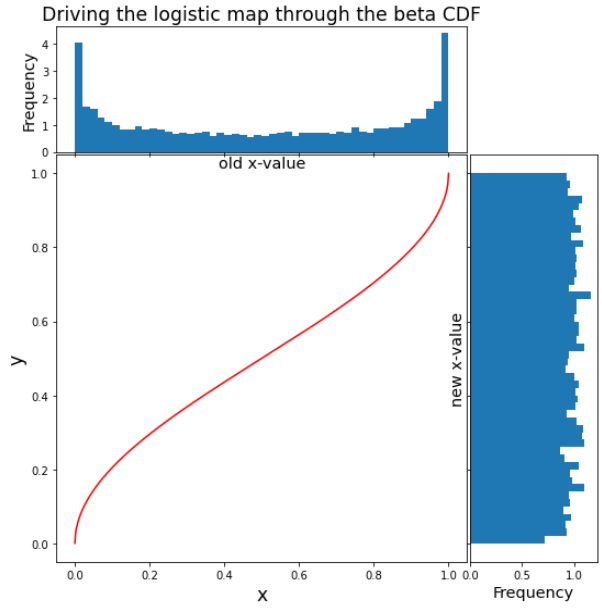


In the left plot, it is visualized that with originally close initial values of x and $x+\varepsilon$, where ε equals 0.001, the sequences deviate after about 9 iterations. On the right histogram, the distribution of a $r = 4$ logistic map sequence of 10,000 steps is graphed under the beta PDF with $a = 0.5$ and $b = 0.5$.

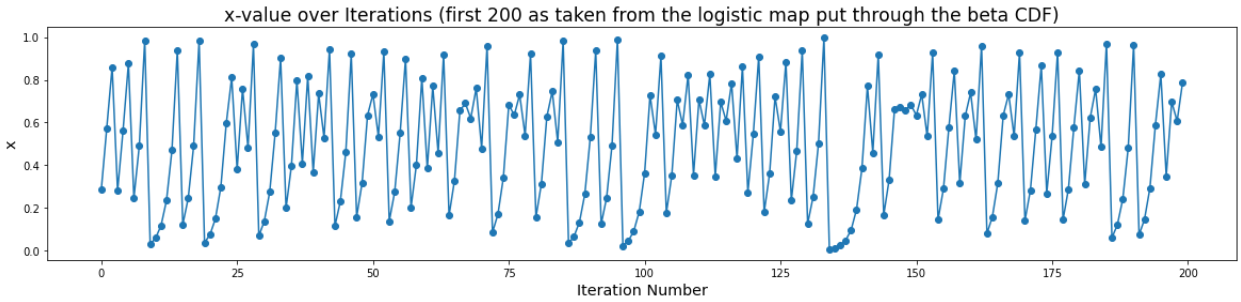
Creating the (Intrinsic) Generator

This generator will rely on the "built-in" (or intrinsic) randomness believed to exist in the chaotic region of the logistic map at $r = 4$.

As outlined, the first problem is retrieving a uniform distribution of numbers from the logistic map. Luckily, it is known that the beta distribution with $a = 0.5$ and $b = 0.5$ describes the distribution (when $r = 4$), so plugging the x-values into the beta cumulative distribution function (CDF) outputs a uniform distribution. This idea comes from reverse-engineering the methods of producing a normal distribution from uniform data.

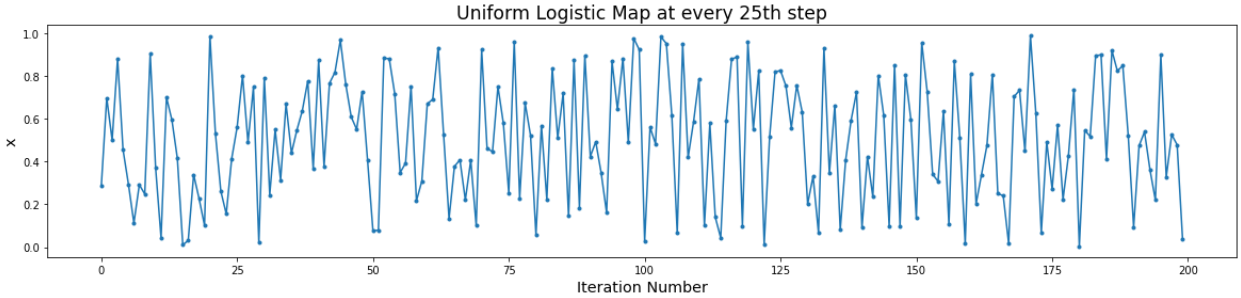


However, while the data appears uniform over many iterations (in this example 10,000 iterations), the values still seem to follow a predictable pattern locally:



Visualized from this sequence, the x-values under 0.66 always increase and x-values over 0.66 always decrease. Furthermore, the x-values seem to get fixated around the two points 0 and 0.66 (where originally before being put through the beta CDF, the "fixation" points were 0 and 0.75, the two equilibrium states of the model at $r = 4$).

A possible solution, and the one taken, is to yield an output only after multiple iterations. Plotting the same sequence for every 25th value yields this graph:



At this point, the sequence appears random. One way to support this temporarily is by acknowledging that it seems impossible to describe any relevant rules and features of this sequence more briefly than just repeating each data point; in information theory, this is defined as maximum entropy (of course, all this is without citing the simple underlying steps in this just-created function—in fact, there is no simple method that can reverse a complex sequence to find its underlying behavior if there exists one).

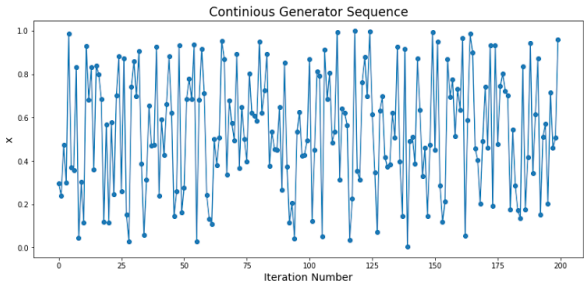
Seeding an Initial Value

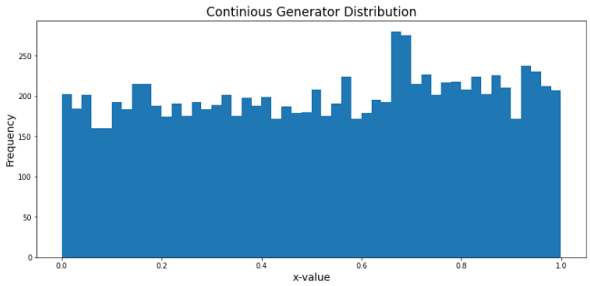
A seed initializer is needed in order to generate the initial values to start each sequence. In this project the sample environment is the computer's time: the seed is the millisecond since the last millisecond. This gives values from 0 to 1 with 13 leading digits measured by the program.

A Side Note: "Continuous" Generation

A continuous generator ("continuous" being a self-named description, the credit for the idea goes to Stephen Wolfram) samples an environment each time the generator function is called, rather than continuing from an ongoing sequence. One benefit to this may be the innate uncorrelation between consecutive iterations. Furthermore, it may seem at first more intuitive to create a sequence that is not entirely determined by one initial seed like in the former method.

With the above seed initializer used, an iteration amount of 10 before yielding an output, and put through the beta CDF, a continuous generator's sequence appears as so:





This seems to produce the same unpredictable behavior and overall uniformity as the former generator. Nevertheless, if it is believed that the logistic map at r =4 exhibits an inherently random behavior, then this middle factor of sampling an environment continuously is pointless. And even if statistical tests show that generation from the logistic map is not intrinsically random, then the produced sequences will only ever be as random as the environment it samples from anyway. Thus this method was ignored.

Statistical Test 1: Chi-square Goodness of Fit

To test that the distributions produced by the intrinsic generation method are in fact uniform, the chi-squared fitness test can be used. For this process, when binning a sequence of random values, it is expected that each bin has equal quantities.

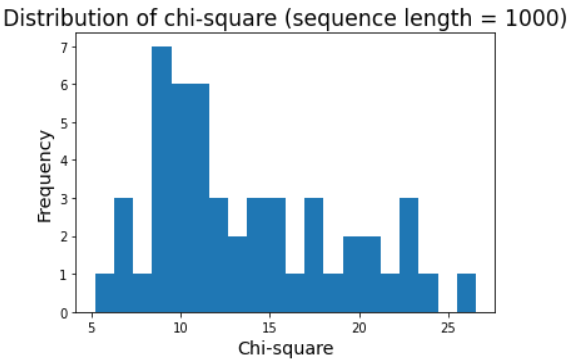
The chi-square formula used is:

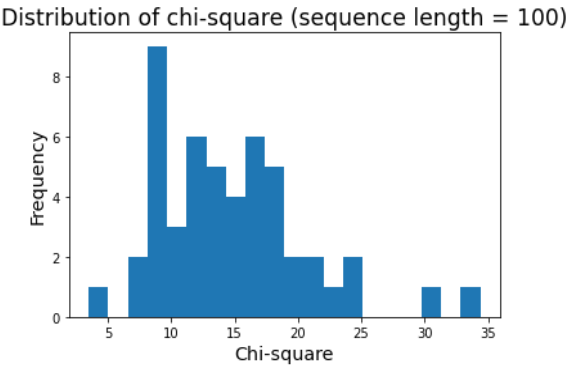
$$\chi^2 = \sum_i [(o_i - e)^2/e]$$

Where o_i is the observed quantity per bin and e is the expected quantity for a bin.

The average value of chi-square for 50 tests of 1,000 long randomly generated sequences, each using 15 bins, was found to be 13.672. Using a lookup table for this chi-square method (for 14 degrees of freedom), the test fails to reject the null hypothesis that the distribution is uniform with a significance of 0.05.

Running the same test for sequence lengths of 100 (this smaller sequence length tests uniformity more locally) yielded an average chi-square of 14.852. Similarly, the null hypothesis cannot be rejected with enough confidence.

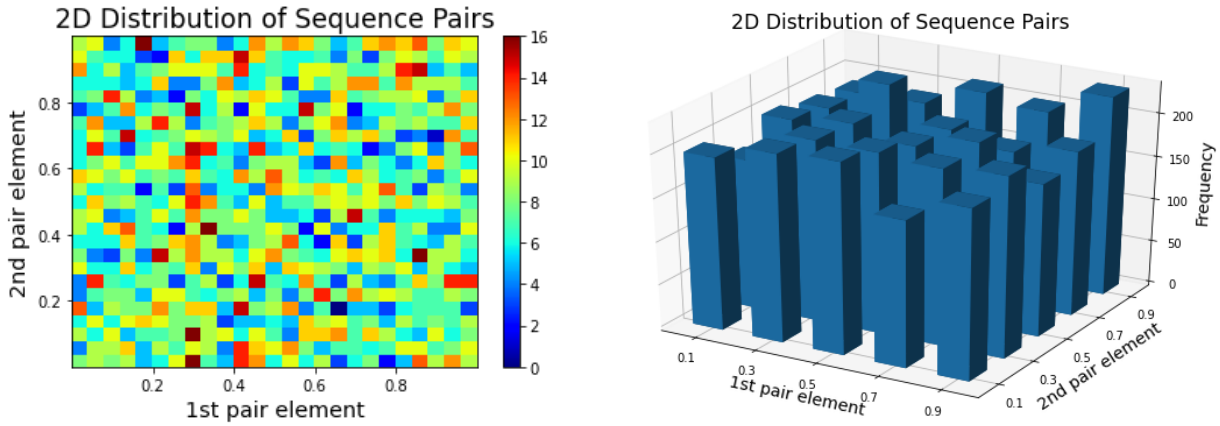




Although these tests fail to disprove uniformity in the sequence, they also fail to support uniformity in the sequence with high confidence.

Statistical Test 1 Continued: Chi-square in Higher Dimensions

It is also possible to test uniformity in higher dimensions. The Serial Test pairs consecutive elements in a sequence together and bins them as vectors. For example, with 2-dimensional pairs (x_1, x_2), (x_3, x_4) . . . (x_{n-1}, x_n), the plots are graphed so:



These are both plots of the same distribution of an originally 10,000 long sequence.

In a similar fashion, one should expect the distribution of values to be uniform. Although it is easy to visually conclude that they appear relatively uniformly distributed, again, chi-square tests do not seem to produce enough confidence for this conjecture.

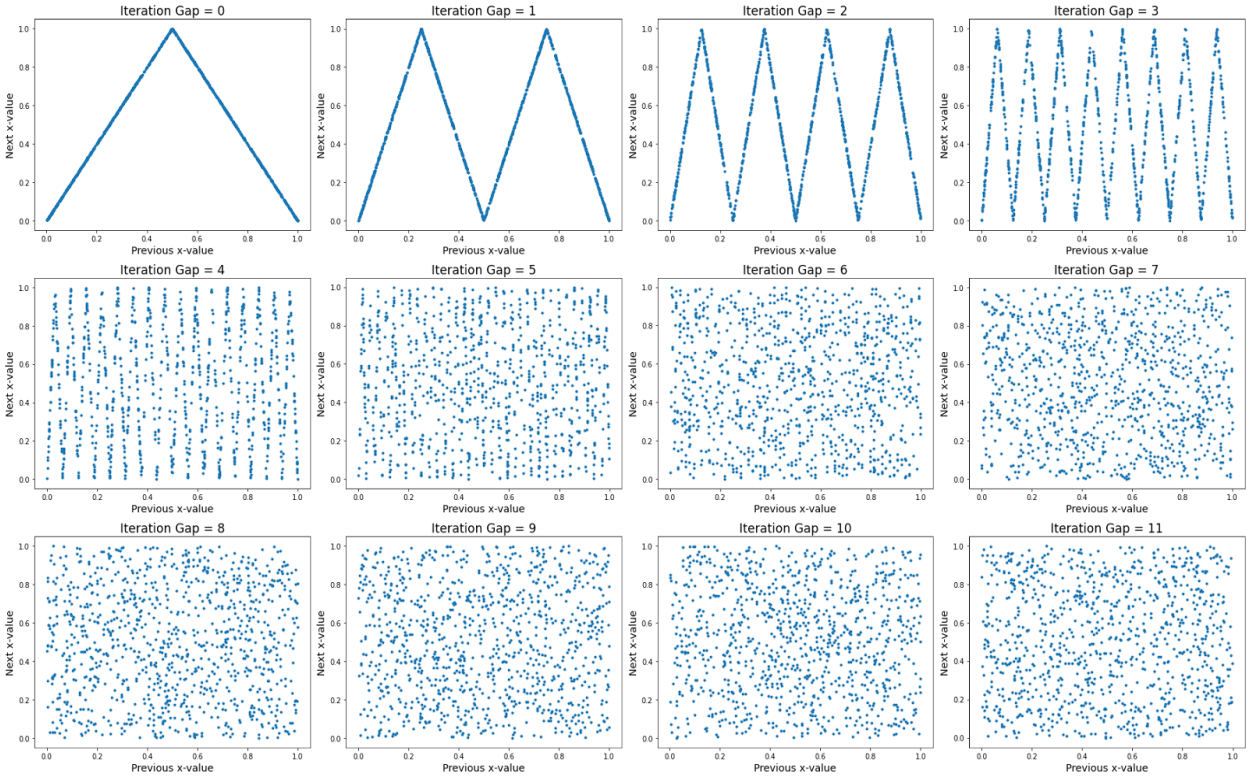
Statistical Tests 2: Correlation

By comparing two distinct random sequences produced by the same generator, Pearson's correlation coefficient can be used to make conclusions about a generator's randomness. In this project, numpy.corrcoef was used on two different 10,000 long lists and produced this correlation matrix:

1.00	1.13e-5
1.13e-5	1.00

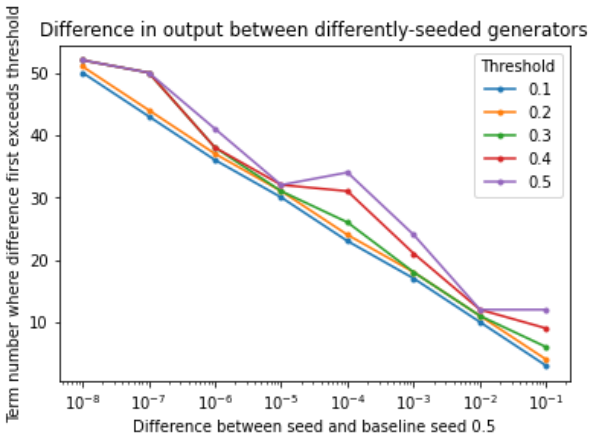
The off-diagonal boxes hold the correlation coefficient between the differing sequences. Since these relevant coefficients are essentially 0, this test supports the notion that the generator is random.

Another interesting visualization (that was come across independently in this project) is the correlation within one sequence—more specifically, when plotting the current values (x_n) against the next values (x_{n+1}). Fundamentally, this is plotting the original sequence shifted backward by 1 vs. the original sequence. This is done for a series of generators with different iteration amounts, or iteration gaps, before yielding an output:



For the first few scatter plots above where the iteration gap is low, there is visually a significant correlation between x_n and x_{n+1}. However, as the iteration gap increases to about 7 and higher, a correlation between x_n and x_{n+1} is hardly visible and can therefore be described as random. This supports the method of iterating over multiple values before outputting a result. In fact, this suggests that the original iteration gap of 25 could be lower for most cases.

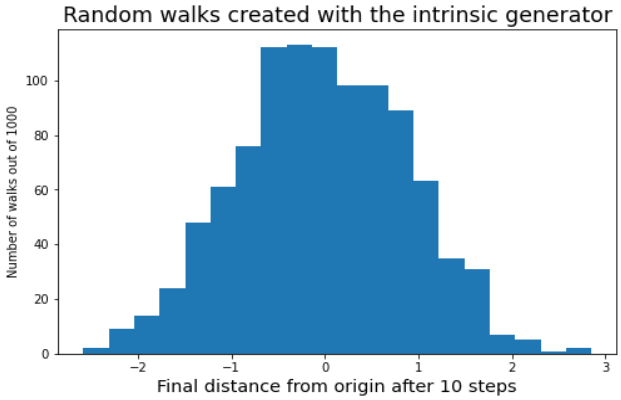
However, for small differences in initial seeds, it may still take a long iteration amount for sequences to first diverge from each other. In the graph below, it is seen that it takes up to 50 iterations for two sequences with a difference in seed of 0.00000001 to initially separate beyond even a threshold difference of 0.1:



While this latter point is not necessarily relevant to the statistical certainty of the random number generator, it does describe an edge case that would undermine the current method of generation—although a simple fix could possibly be to throw out the first 50 or so initial iterations before beginning to yield values.

Statistical Tests 3: A Random Walk

Lastly, a one dimensional random walk was used to test the randomness of the generator. In order to implement walking backward and forward, the generator's range of (0,1) is shifted to (-0.5, 0.5) by subtracting 0.5 from each yielded value. For 1,000 random walks of step length 10, the distribution of end points was as follows:



This produces a normal distribution, as expected from plotting multiple randomly generated walks. The symmetric centralization around the origin supports the conjecture that the generator is uniform, especially at an extremely low local level of just 10 yields. The fall-off as the values trend outward from the origin also supports the idea that the generator is random in nature rather than predictable, allowing for the possibility of producing rare peripheral cases, just as the normal distribution should describe.

Conclusion

The complex behavior produced from these simple rulesets can be difficult to understand intuitively, especially from a computer programming standpoint. How can a computer that follows rigid tasks create random numbers that are impossible to predict or even approximate? Such topical questions in the field of random number generation are still in contention to this day. Physicist and computer scientist Stephen Wolfram (mentioned briefly before) is a famous figure in this discussion who offers an over one-thousand page textbook "A New Kind of Science" dedicated to exploring such a contrasting way of thinking. One point Wolfram argues is that randomness is actually a common phenomena in the physical world: take the constant pi for example, it is an endless sequence of random digits recorded in nature.

Similarly, the tests undertaken in this project tend to support the notion that the logistic map at $r = 4$ is a naturally random model. And, with a few alterations, one can create a uniform and unpredictable generator from it. Accordingly, even if an original seed

is not relatively random, due to the suggested intrinsic randomness of the logistic map, a number (or more strictly, a *sequence*, since a single number alone is always ordinary) generated from the function will still be random through its inherent complexity.

Nevertheless, while the logistic map at $r = 4$ appears to function relatively well as an intrinsic-type generator under these statistical tests, there are countless other robust tests for randomness that may prove otherwise. In fact, such abundance and modernization of random sequence tests have led to the obsolescence of many other, once useful generators in the past—the logistic map might not be an exception.

Contributors

This capstone project was done in collaboration with Austin Carroll.
Course instructors: Professor Amin Jazaeri. and C.K. Wolfe.
Lectures taught by Professor Heather Gray.

References

[1] "Create 3D histogram of 2D data." The Matplotlib development team, matplotlib.org/stable/gallery/mplot3d/hist3d.html
[2] May, Robert M. "Simple Mathematical Models with very Complicated Dynamics." Nature Vol. 261, Springer Nature, 10 June 1976, physics111.lib.berkeley.edu/Physics111/Reprints/NLD/04-Simple_Mathematical_Models.Pdf
[3] Mellor-Crummey, John. "Testing Random Number Generators." Rice University Department of Computer Science, 12 April 2005, www.cs.rice.edu/~johnmc/comp528/lecture-notes/Lecture22.pdf
[4] Muller, Derek. "This equation will change how you see the world (the logistic map)." YouTube, uploaded by Veritasium, 29 Jan. 2020, www.youtube.com/watch?v=ovJcs7vyrk
[5] Muller, Derek. "What is NOT Random?." Youtube, uploaded by Veritasium, 16 Jul. 2014, www.youtube.com/watch?v=sMb00lz-IfE
[6] Pierce, Jordan. *Logistic Bifurcation map High Resolution*. Wikipedia, 12 September 2011, en.wikipedia.org/wiki/Logistic_map#/media/File:Logistic_Bifurcation_map_High_Resolution.png
[7] "Pseudorandom number generators | Computer Science | Khan Academy." YouTube, uploaded by Khan Academy, 28 Apr. 2014, www.youtube.com/watch?v=GtOt7EBNEwQ
[8] "Random Numbers - Numberphile." Youtube, uploaded by Numberphile, 13 Apr. 2013, www.youtube.com/watch?v=SxP30euw3-0
[9]"Scatter plot with histograms." The Matplotlib development team, matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_hist.html
[10] Stevens, Michael. "What is Random?." Youtube, uploaded by Vsauce, 16 Jul. 2014, www.youtube.com/watch?v=9rIy0xY99a0
[11] "The Feigenbaum Constant (4.669) - Numberphile." Youtube, uploaded by Numberphile, 16 Jan. 2017, www.youtube.com/watch?v=ETrYE4MdoLQ
[12] Turney, Shaun. "Chi-Square (X^2) Table | Examples & Downloadable Table." Scribbr, 31 May 2022, www.scribbr.com/statistics/chi-square-distribution-table/
[13] Wikipedia contributors. "Logistic map." Wikipedia, The Free Encyclopedia, 4 Jul. 2022, en.wikipedia.org/w/index.php?title=Logistic_map&oldid=1096404124
[14] Wolfram, Stephen. "A New Kind Of Science." Wolfram Media, 14 May 2002, www.wolframscience.com
[15] Wolfram, Stephen. "Mitchell Feigenbaum (1944-2019), 4.669201609102990671853203820…." Stephen Wolfram, 23 July 2019, writings.stephenwolfram.com/2019/07/mitchell-feigenbaum-1944-2019-4-669201609102990671853203820/