CP 3

Introduction to Monte Carlo Methods

Nathan Ngqwebo

NGQLIN011

Department of Science, University of Cape Town

PHY2004W | Intermediate Physics

Dr Spencer Wheaton

20 March 2023

CP 3

Introduction to Monte Carlo Methods

This computational activity is a starter on computational algorithms used for probability distribution sampling and numerical integration. We begin with generating normally distributed samples and plotting histograms with their associated gaussian curves. We end with the Monte Carlo method for approximating π. All code was produced using Python (Van Rossum & Drake, 2009) with the packages: NumPy(Harris et al., 2020), and Matplotlib(Hunter, 2007) used for data wrangling and plotting.

Monte Carlo Data Generation

Here, we create a histogram of the 'Activity1Data.txt' data and 'fit' a gaussian curve on the same plot. The histogram bins are made using the integer min, and max values of the dataset. The '*arange'* function then produces a list of numbers within that range each one-'binwidth' apart. See Figure 1. The resulting histogram is shown in Figure 2

```
bins = np.arange(np.floor(min(data)),
                 np.floor(max(data)) + 1, binwidth)
```

Figure 1

Python Code Sample of the 'arange' Function Call to Produce Histogram Bins

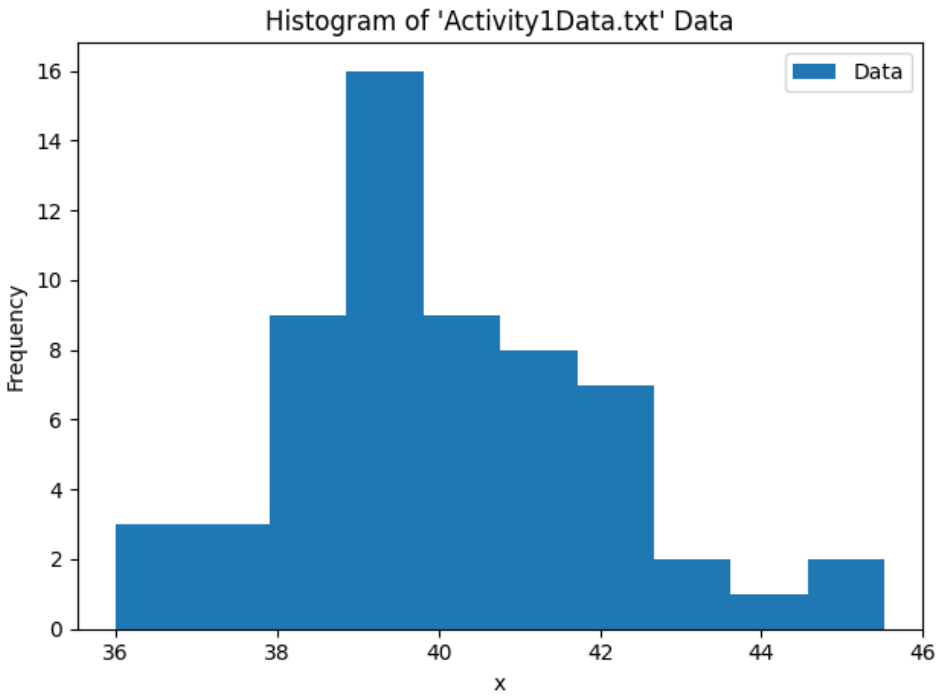

Figure 2

Histogram of 60 Samples with $\mu = 40.0$ *and* $\sigma = 2.0$. *Bin Width* $= \sigma$

To correctly display the gaussian curve over this data, the normal curve definition in (1) needs to be scaled such that the area under the curve is equal to area under the histogram, instead of 1 (in the case of a normal distribution). See (2).

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{1}$$

The area under histogram is the summed area of n rectangles of width b_i and height (frequency) f_i , where n is the number of bins.

$$A = \sum^{n} f_i \, b_i \tag{2}$$

The bins are the same size, $= b \sum^{n} f_i$

$= b \times$ Number of Data Points

We can then scale $f(x, \mu, \sigma)$ by A to produce the plot in Figure 3

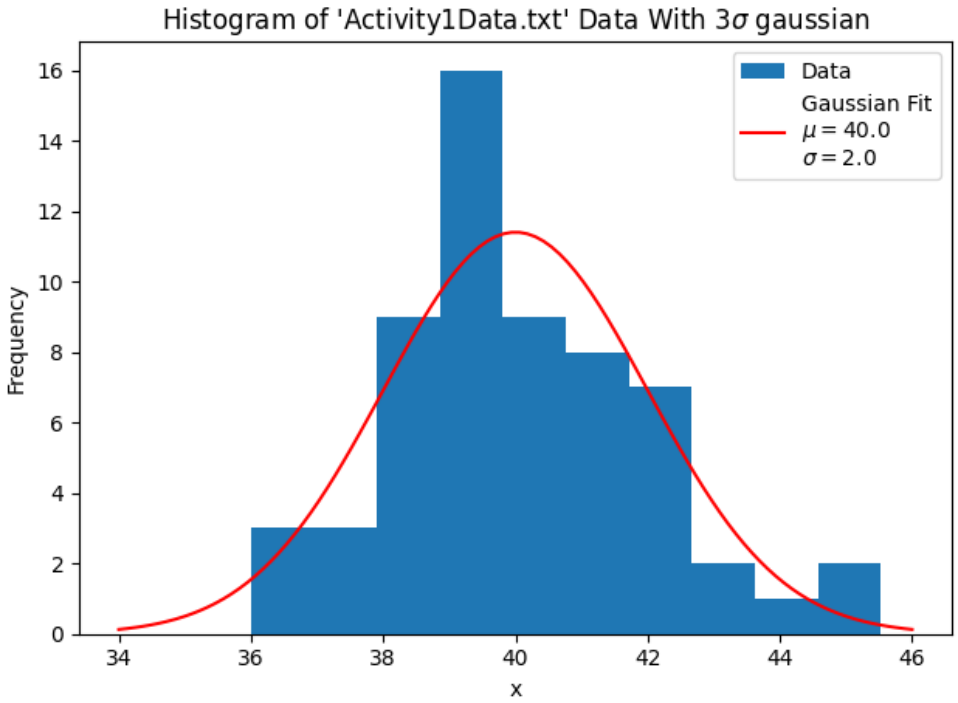

Figure 3

Previous Histogram with Plotted Gaussian 'Fit'

Using out own sample of 60 data points with $\mu = 40.0$ and $\sigma = 2.0$, we produced Figure 4.

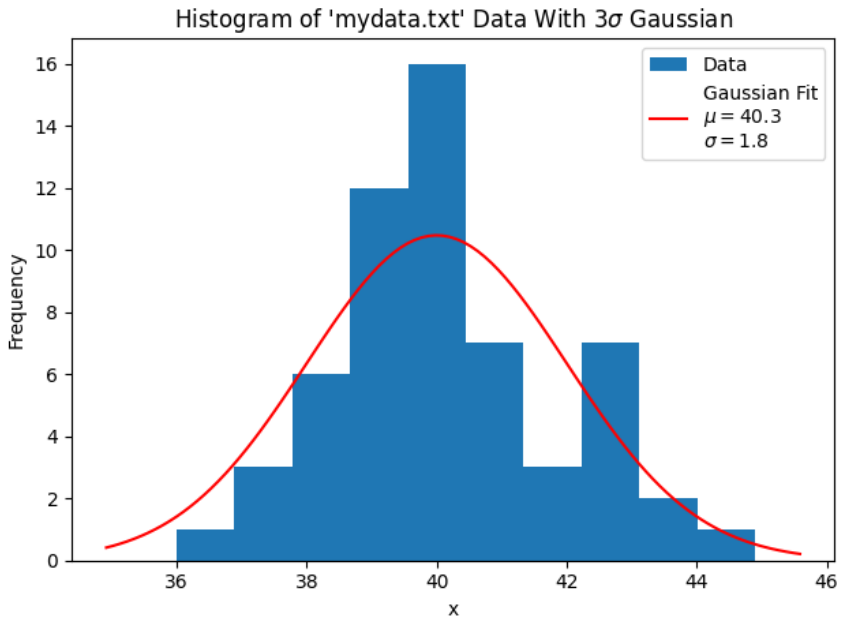

Figure 4

Histogram of 'mydata.txt' with Fitted Gaussian Curve.

Monte Carlo π Approximation

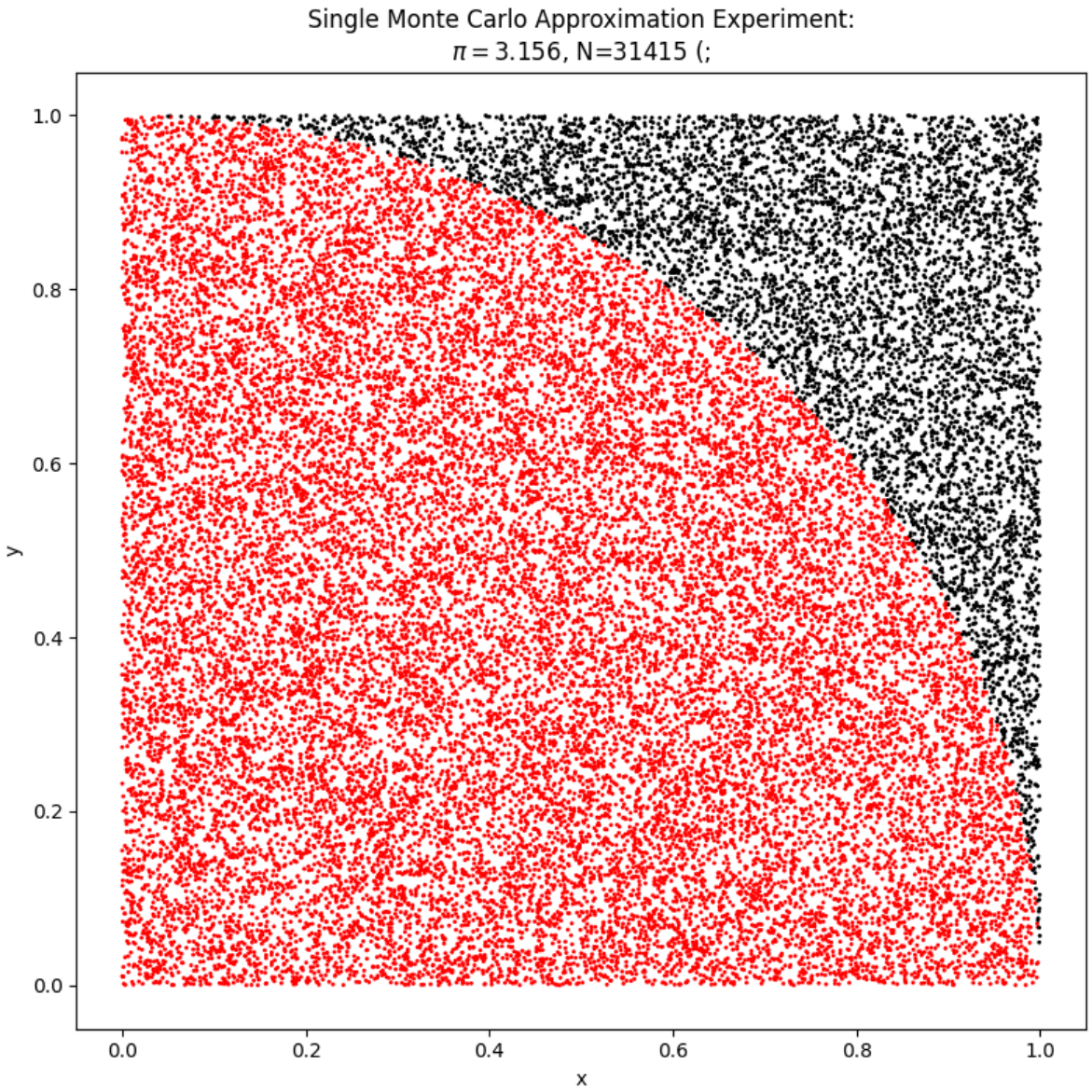

Figure 5

Visual Example of the Monte Carlo Method for Approximating π

Note: The above plot does not represent the actual data used for our measurement. It is only meant as a visual aid.

We present the result of the experimental method outlined in the practical handout where we approximate π as four times the ratio of the number of randomly generated points falling within the unit circle to those generated overall, according to (3).

$$\text{Area of a circle,} \quad C = \pi R^2 \tag{3}$$

$$\text{Area of a square of length } 2R, \quad S = 4R^2$$

$$\frac{C}{R} = \frac{\pi}{4}$$

$$\Rightarrow \pi = 4\frac{C}{R}$$

Using NumPy's vectorization, we were able to perform 50 of these Monte Carlo experiments, each with 100,000 uniformly sampled points, to calculate our estimate of π within a few seconds. This would take a while longer using loop methods.

Using this method, we found π to be 3.14146 ± 0.00076 with a 68% coverage probability, using a Type A uncertainty evaluation, that is, assuming our observations are sampled from an underlying gaussian distribution.

While our meqsuement appears accurate to within 3 digits of what we know is the true value, we express concern that the uncertainty of our measurement is significantly underestimated. We believe that the 50 experiments we performed were not entirely independent of each other. In total, the procedure we used makes 100 (2×50) total calls to the 'np.random.uniform' function producing two 100,000-element arrays of pseudorandom numbers each time and we are unsure of how NumPy handles multiple such calls using the same random seed.

References

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science \& Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Van Rossum, G., & Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace.