Sampling
Mathematical Programming with Python

MATH 2604: Advanced Scientific Computing 4
Spring 2025
Monday/Wednesday/Friday, 1:00-1:50pm

https://people.sc.fsu.edu/~jburkardt/classes/python_2025/sampling/sampling.pdf

---

A fly lands at a random spot on a plate. What is the average distance to the center?

Probability

- *Probability suggests parameterized models of random processes;*
- *Statistics estimates the parameters from sample data.*
- *Uniform sampling means every outcome has an equal chance of being sampled.*
- *Errors are easy to see when sampling geometric regions.*
- *Rejection and other methods can improve sampling.*
- *Histograms can be constructed from data.*
- *A histogram estimates the probability density function.*

1 Probability Models Uncertainty

When a process includes some random component, then we may not be able to predict its future behavior completely. However, we may believe we understand the underlying system well enough that we can still consider a range of outcomes and assign a likelihood to each one. Whether our view of the system is accurate can be tested if we are able to observe many occurrences of the events.

Mathematical probability includes the creation of models of random processes, and methods for describing the range and likelihood of their behavior. We will be mainly interested in those aspects of probability that allow us to create models from a large set of observations, and then to use such a model to simulate the original system.

Our observations might come from a data file, or from an existing computer model. Our analysis of the data will seek to determine the average behavior and its variance, the range of outcomes, and the likelihood of a particular outcome. One of our vital tools in this process will be the use of random quantities for sampling, and for simulation.

2 A random fly

Consider a circular plate P that has a radius of $R = 1$ unit. Suppose we use a coordinate system whose origin is the center of the plate. Now let a fly land somewhere at a random spot (x, y) on the plate. We wish to study the typical distance of the fly from the center of the plate. In polar coordinates, this is simply the value of the radial coordinate r.

Since we are assuming that every point on the plate is equally likely as the fly's landing spot, we can compute the average distance to the center by the ratio of the integral of r against the elemental circle area, versus the area integral, that is:

$$\bar{r} = \frac{\int_P r \cdot r \, dr \, d\theta}{\int_P r \, dr \, d\theta} = \frac{2}{3} \approx 0.666$$

3 A simulated fly

Suppose we wish to simulate this process? We need to construct a procedure which can produce a random point in the unit circle, in such a way that every point is equally likely to be chosen. If we can do that, then we can consider generating n experimental results, computing the radial distance r_i for each of them, and estimating the true average distance by a sampled estimate:

$$\bar{r} \approx \frac{1}{n} \sum_{i=0}^{i<n} r_i$$

So how do we pick a point on the plate at random? We have already claimed that, if the plate were square, then we could correctly generate a point by picking two uniform random values:

$$-1 \leq x_i \leq +1$$
$$-1 \leq y_i \leq +1$$

Although this model does not describe the plate we are interested in, let's go ahead and run it just to get a feeling for how this would work.

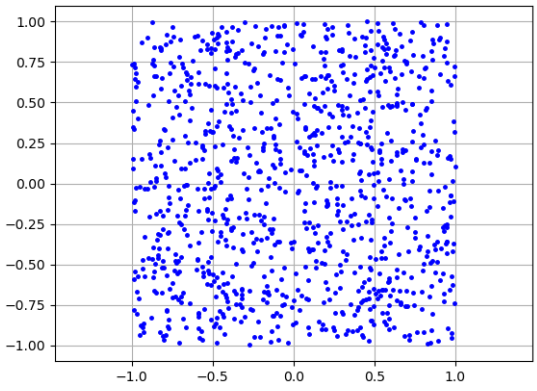
```
import numpy as np
n = 1000
dbar = 0.0
for i in range ( 0, n ):
  x = np.random.uniform ( low = -1.0, high = +1.0 )
  y = np.random.uniform ( low = -1.0, high = +1.0 )
  d = np.sqrt ( xy**2 + y**2 )
  dbar = dbar + d

dbar = dbar / n
```

Running this simulation produced the results:

```
dbar for square plate =  0.7609255248444867
Theoretical           =  0.7651957164642127
```

where the theoretical value was obtained by integration as $(\sqrt{2} + \log(1 + \sqrt{2}))/3$.

And here is our plot for $n = 1000$:



It is good to see that we got a rough approximation to the theoretical value. It is also reasonable that the value of `dbar` should be bigger for the square plate, since the added regions around the four corners offer points that are further from then center than the maximum unit distance on the circular plate.

However, now we need to determine how we can repeat this experiment for the circular plate itself.

4 Naive Sampling for the Circle

Now we have to consider the problem of generating random points inside a circle, in a uniform fashion. The circle has a natural polar coordinate system, and it would seem logical to use one random number $0 \leq r \leq 1$ as the radius, and another $0 \leq \theta \leq 2\pi$, compute (x, y) and then our distance. (Yes, we are actually recomputing r every time in this case, but let's not notice this minor inefficiency.)

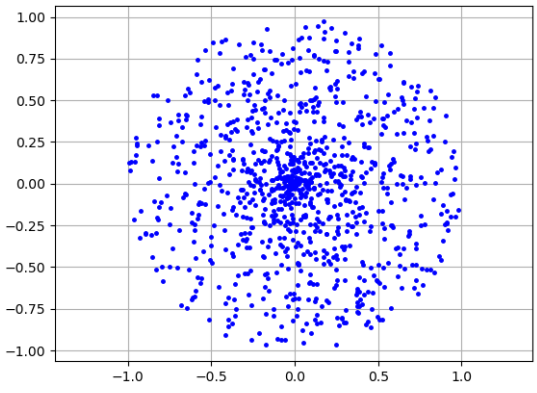
Here's a code that uses this sampling technique to estimate `dbar`:

```
  dbar = 0.0
  for i in range ( 0, n ):
    r = np.random.uniform ( low = 0.0, high = 1.0 )
    theta = np.random.uniform ( low = 0.0, high = 2.0 * np.pi )
    x = r * np.cos ( theta )
    y = r * np.sin ( theta )
    d = np.sqrt ( x**2 + y**2 )
    dbar = dbar + d

  dbar = dbar / n
```

Unfortunately, when we run this code with $n = 1000$, our estimate doesn't come close to the theoretical value we expect. What's worse is that, as we increase n, our estimate seems to converge, but to a different, wrong, result:

```
   n      dbar

  1000   0.5025577928132363
 10000   0.5017885255743197
100000   0.5018450485227093

Theory   0.6666666666666666
```

To get an idea of what's wrong, let's actually plot 1,000 random points as generated by our sampling method:



Now it should be clear that our naive sampling method produces many more points near the origin. It is not a uniform sampling method, and this has caused our estimates to be wrong.

5 Rejection Sampling for the Circle

Now our sampling method for the square plate worked OK. Perhaps we could somehow adapt it to the circular plate. Suppose we generate a sample point for the square. If it also fits in the circle, it's a valid sample point. If it's out near the corners of the square, we could reject it and try again. This would be accomplished by a Python `while (True):` loop that repeats until we get what we want.

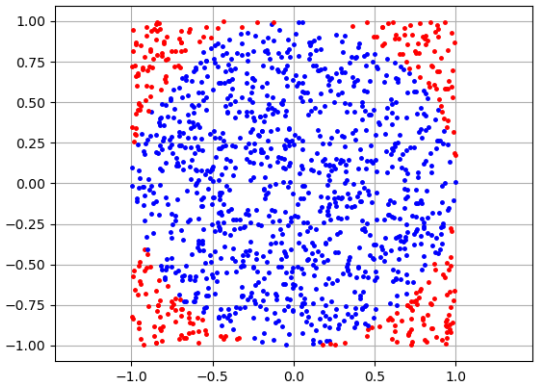
Here's how the code would look:

```
  dbar = 0.0
  for i in range ( 0, n ):

    while ( True ):
      x = np.random.uniform ( low = -1.0, high = +1.0 )
      y = np.random.uniform ( low = -1.0, high = +1.0 )
      d = np.sqrt ( x**2 + y**2 )
      if ( d <= 1.0 ):
        break
    dbar = dbar + d

  dbar = dbar / n
```

Now our results look much more convincing, although, in order to get n sample points, we also had to compute and reject many other points:

```
   n     rejected   dbar

  1000        254   0.6575594542913391
 10000       2709   0.6683102116066916
100000      27494   0.6662616703199041

Theoretical         0.6666666666666666
```

To get a feeling for how this works, and how many sampling points we have to reject, we can plot the case for when we get $n = 1000$:



If you are a picky programmer, it might bother you that we throw away a significant portion of our sample points. Also, you should notice that, to do rejection, we have to compute our points one at a time, rather than in a vectorized statement.

Let's try one more time to tackle the problem of uniform sampling in a circle, without throwing away any points.

6 An Efficient Sampling for the Circle

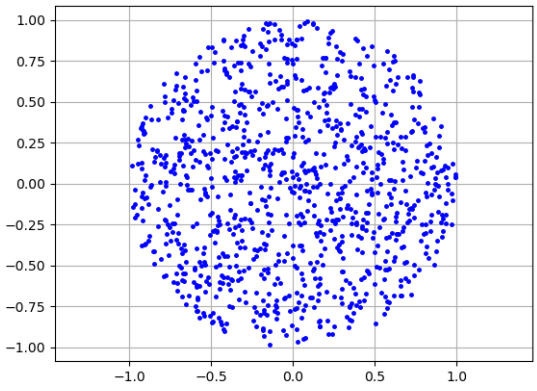
Let's think about what went wrong with our random sampling. Certainly, it's right to choose our angle uniformly at random. But what about the radius? We are just as likely to choose $r = 0$ as to choose $r = 1$, but there is only a single point at the origin, while $r = 1$ corresponds to the whole circumference. We are ignoring the fact that, as we move in the radial direction, the area is increasing like r^2.

The value of r at which exactly half the circle points are further from the center is not $r = 0.5$ but rather $r = \sqrt{0.5} \approx 0.7071$. So to pick circle points uniformly, we must pick a random number z and then set $r = \sqrt{z}$. Let's code this up:

```
r = np.sqrt ( np.random.uniform ( low = 0.0, high = 1.0, size = n ) )
theta = np.random.uniform ( low = 0.0, high = 2.0 * np.pi, size = n )
x = r * np.cos ( theta )
y = r * np.sin ( theta )
d = np.sqrt ( x**2 + y**2 )
dbar = np.mean ( d )
```

```
   n          dbar

  1000        0.6722899554731946
 10000        0.669202081480905
100000        0.6675643448803363

Theoretical   0.6666666666666666
```

And here is our plot for $n = 1000$:



We are able to approximate the theoretical results, we don't reject sample points, and we are able to compute much faster because we can use vectorized statements.

7 Histograms and Variance

We have been able to estimate the mean or average value for the distance of a fly to the center of a circular plate. Even though this is a random process, there is more we might want to know. In particular, knowing the average value, we would also like to know the average deviation. We know the fly will generally not land in the center, and so we'd like some measure for a typical distance. Similarly, if a darts player, on average, hits the center of the board, we would very much like to know how far away from the center a typical shot lands!

Along with the mean, the statistical quantities of maximum, minimum, and variance or standard deviation, give us some idea of the range and variability of the quantity we are studying. If we have saved our data as a vector, then these are easily computed:

```
np.min ( r )
np.mean ( r )
np.max ( r )
np.var ( r )
np.std ( r )
```

The variance and standard deviation are often symbolized by σ^2 and σ, since variance is the square of the standard deviation, and has the mathematical definition

$$\sigma^2 = \frac{1}{n}\sum_{i=0}^{i<n}(r_i - \mu)^2$$

where μ is the average or mean of the random variable r. (There is a minor religious dispute over whether to divide by n or $n-1$, which we will ignore.)

The statistics for our data make sense.

n	min	mean	max	var	std
1000	0.02737	0.66453	0.99945	0.05436	0.23315
10000	0.01313	0.66780	0.99985	0.05558	0.23575
100000	0.00245	0.66698	0.99999	0.05550	0.23558

In particular, the value of `std` suggests that for the bulk of our `d` values are within one standard deviation of the mean, that is

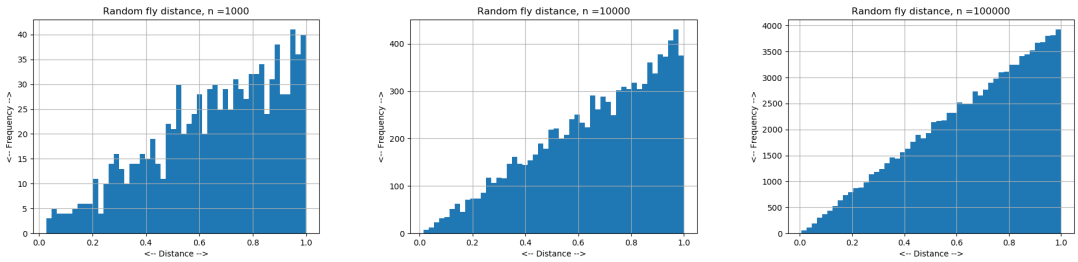
$$0.433 = 0.666 - 0.233 \leq d \leq 0.666 + 0.233 = 0.899$$

so that now we have not only an average value for our data, but an estimate of the typical variation from that average.

8 The Data Histogram

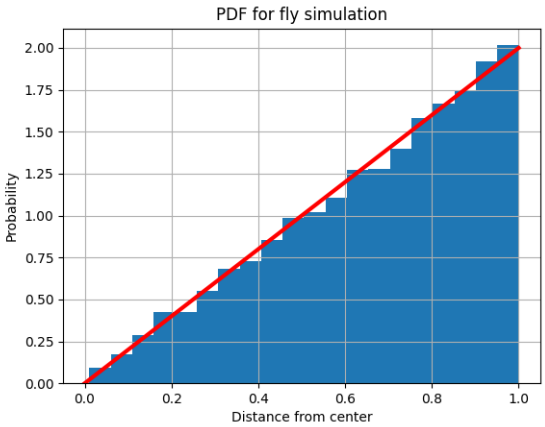
While these statistical quantities are important, often a histogram gives the most vivid picture of what is going on. This is a plot in which the range of the variable has been divided into equal intervals, and a count has been made of the number of times the variable took on each value. Usually, the result is presented as a kind of bar graph.

Here are three histograms for our `d` variable. As n, the number of samples, increases, the histograms reveal an increasingly regular pattern, suggesting that there is some sort of linear relationship betweeb the value of `d` and the frequency with which it is observed.



For instance, we can conclude from the histogram that, compared to a distance of $d = 0.2$, a distance of $d = 0.4$ is about twice as likely to be observed, and a distance of $d = 0.6$ three times as likely.

To make this relationship more clear, we can change the plots so that they are normalized to have area 1. In that case, we are exhibiting the probability density function, or PDF. We can see that the histogram is following the line $pdf(x) = 2 * x$, which indeed integrates to 1 over the interval.



It's best to think of a PDF as a function reporting the probability of a value x between x_1 and x_2 as the

integral of $pdf(x)$ over that interval:

$$\text{Prob}(x_1 <= x <= x_2) = \int_{x_1}^{x_2} \text{pdf}(x)\, dx$$

For the fly example, we started with a mathematical understanding of how the random process worked. In real life situations, we might only have the data, that is, perhaps 10,000 observations of the value of some quantity. By computing the statistical quantities, and creating a histogram, we could nonetheless try to understand some of the hidden patterns in the process that generated the data.