EM384: Analytical Methods for Engineering Management

Lesson 33: Simulating Random Variables in Python

24 April 2023

Table of contents

Lesson Objectives

Lesson Objectives

- Simulate discrete and continuous random variables in Python
- Visualize the results of a simulation using a histogram in Python.
- Find the expected value of a simulation outcome in Python.
- Visualize the results of a simulation using an empirical cumulative distribution function (ECDF) in Python.
- Find the probability of a result using the empirical cumulative distribution function (ECDF) in Python.

Plotting in Python

User-Defined Plotting Functions in EM384

- In EM384 we generally use the **matplotlib** library to create plots.
- Since we create a lot of plots of the same kind (histograms, line plots, it makes sense to create some *user-defined* plotting functions that we can reuse.
- The examples in this lesson use the following functions which must always be stored in memory before you call them (a good rule of thumb is to put all your user defined functions after the import commands but before you start coding your script).
- Remember to execute your entire script at least once to store the plotting functions in memory!

User-Defined Python Plotting Function used in EM384

```
# A simple function to display a histogram
def simple_hist(values,num_bins,title):
    fig, axis = plt.subplots(figsize =(10, 5))
    axis.hist(values, bins = num_bins)
    plt.title(title)
    # Displaying the histogram
    plt.show()

#A simple function to display a line plot of x and y data
def simple_plot(x,y,x_label,title):
    fig, axis = plt.subplots(figsize=(10, 5))
    axis.plot(x,y)
    plt.title(title)
    plt.xlabel(x_label)
    plt.show()
```

Using the ECDF Function in Python

What is an ECDF?

The abbreviation ECDF means Empirical Cumulative Distribution Function. It is a cumulative distribution function for a sample of random data (such as the output of a Monte Carlo simulation). In fact, it tells you the same information about your sample output, as a CDF would tell you about a random variable.

If F is the ECDF function of some sample data, then:

$$F(x) = P(X \leq x)$$

Creating and using an ECDF function in Python

Make sure you import Numpy and the ECDF function first:

```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
```

Given some sample data, for example:

```
sample = np.random.uniform(-1,1,size=1000) +\
np.random.normal(2,0.5,size=1000)
```

We can create a **function** in Python that will serve as our ECDF function (we can pick any variable name):

```
F = ECDF(sample)
```

(Notice that *F* is not saved in the variable explorer since it is a function)

Creating and using an ECDF function in Python

Then we need can use the ECDF function to find any value $F(x) = P(X \leq x)$ from our sample.

```
F(1)
>>> 0.114

F(3.4)
>>> 0.963
```

The interpretation is:

- The probability that a value in our sample is less than or equal to 1 is 0.114.
- The probability that a value in our sample is less than or equal to 3.4 is 0.963.
- The probability that a value in our sample is greater than 1 is $1 - 0.114 = 0.886$.
- The probability that a value in our sample is between 1 (excluded) and 3.4 (included) is $F(3.4) - F(1) = 0.849$
- etc.

Creating and using an ECDF function in Python

To get a sense of the result of our simple MOnte-Carlo simulation, and the range of our sample data, we can calculate a few statistics, or plot a histogram.

We can find the mean:

```
np.mean(sample)
>>> 1.9971
```

We can find the min value:

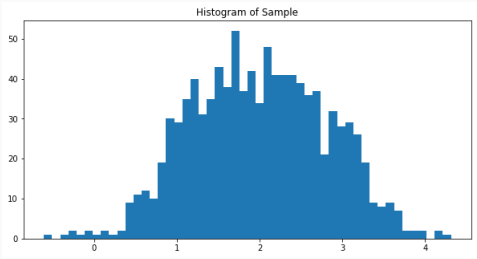
```
np.min(sample)
>>> -0.6064
```

We can find the max value:

```
np.max(sample)
>>> 4.3124
```

To make a histogram, I can use our user-defined function **simple_hist** using the arguments: data,bins,title.

```
simple_hist(sample,50,'Histogram of Sample')
```



Creating and using an ECDF function in Python

Now that we have a range, we create a list (or array) of *x* values for a plot using the np.linspace function:

```
x = np.linspace(-0.0606,4.312,1000) #Note: x is a list (or array)
```

The we use the ECDF function we created earlier to find the corresponding y values:

```
y = F(x)
```

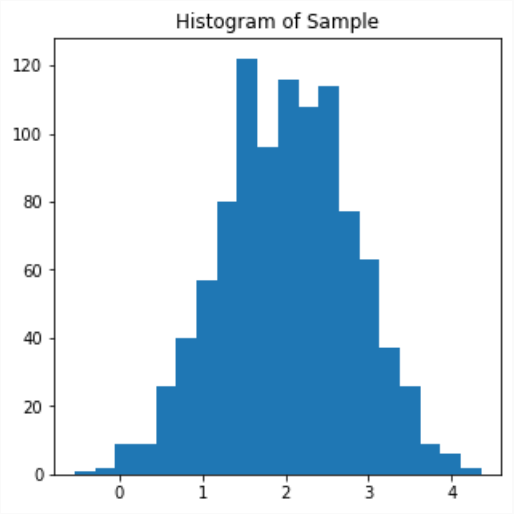
Finally, we plot the ECDF function with our user-defined simple_plot function:

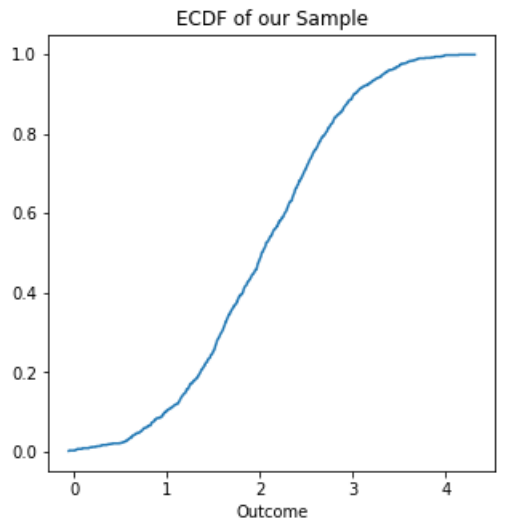
```
simple_plot(x,y,'Outcome','ECDF of our Sample')
```

Putting it all Together

```
#import necessary libraries
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF

#simulate 1000 random variable outcomes from the same distribution
sample = np.random.uniform(-1,1,size=1000) +\
np.random.normal(2,0.5,size=1000)

#Plot histogram, get the sample space
simple_hist(sample,20,'Histogram of Sample')
F = ECDF(sample) #Create the ECDF function
#Create a list of evenly spaced x values
x = np.linspace(-0.0606,4.312,1000)
y = F(x) #y is a list of ECDF values corresponding to x
simple_plot(x,y,'Outcome','ECDF of our Sample') #plot ECDF
```

Putting it all Together





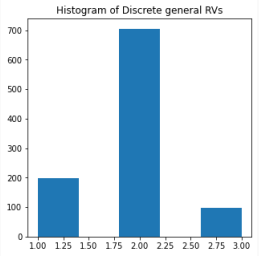
Simulating Discrete Random Variable Outcomes in Python

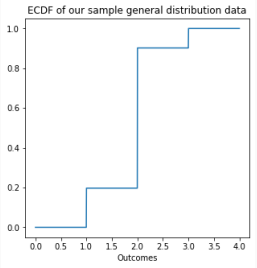
A Discrete General Distribution

A discrete general distribution is a distribution that is not described by any parameters, but by the full sample space and corresponding probabilities. In Python, I can simulate values from these types of distributions using loops and/or if statements, or I can use the np.random.choice() function.

```
#First define the sample space and probabilities
S = [1,2,3]
P = [0.2, 0.7, 0.1]

#Simulating a single outcome
np.random.choice(a = S, p = P)

#OR

#Simulating m outcomes
np.random.choice(a = S, p = P, size = m)
```

Simulating a Discrete General Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Discrete General Distribution
S = [1,2,3]
P = [0.2, 0.7, 0.1]
sample = np.random.choice(a = S, p = P, size = 1000)
simple_hist(sample, 5, 'Histogram of Discrete general RVs') #plot a histogram of the RVs
x = np.linspace(0,4,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Discrete General distribution data') #Plot the ECDF
```

The Discrete Uniform Distribution

The Discrete Uniform distribution is a *discrete* distribution defined on an integer sample space $S = \{a, a+1, .., b\}$ between a and b. It has two parameters: the lower limit of the domain a and upper limit of the domain b.

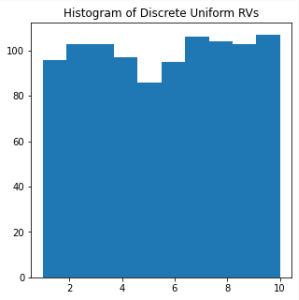
If X is a discrete Uniform RV with parameters a and b, we say that:

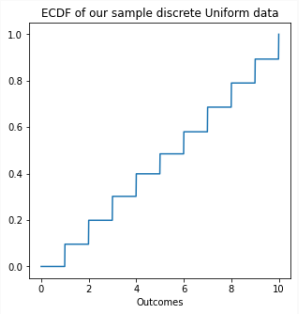
$$X \sim \mathcal{U}\{a, b\}$$

In Python, we can use the **np.random.randint()** function to simulate values from a discrete uniform distribution with parameters a and b:

```
#Simulating a single outcome
np.random.randint(a,b+1) #We need +1 to include b in the outcomes

#OR

#Simulating m outcomes
np.random.randint(a,b+1, size = m) #Also need to add +1
```

Simulating a Discrete Uniform Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Discrete Uniform distribution with parameters a=1, b=10
sample = np.random.randint(1,11,size=1000)
simple_hist(sample, 10, 'Histogram of Discrete Uniform RVs') #plot a histogram of the RVs
x = np.linspace(0,10,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Discrete Uniform data') #Plot the ECDF
```

The Bernoulli Distribution

The Bernoulli distribution is a *discrete* distribution defined on sample space $S = \{0, 1\}$. It has one parameter: the probability of success p. The mean of this distribution is also p.

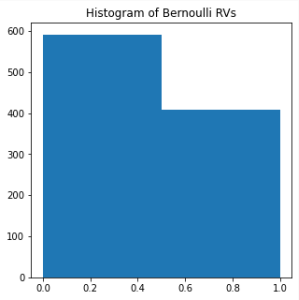
If X is a Bernoulli RV with parameter p, we say that:

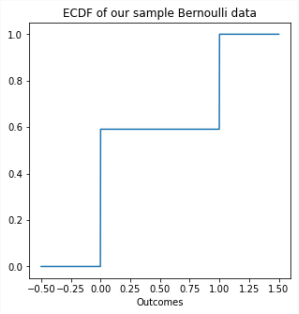
$$X \sim Bernoulli(p) \quad (\text{or} \quad X \sim \mathcal{B}(1, p), \text{see Binomial})$$

In Python, we can use the function np.random.binomial with parameter $n = 1$ (since a Binomial RV with $n = 1$ is Bernoulli) to simulate values from a Bernoulli distribution with parameter *prob*:

```
#Simulating a single outcome
np.random.binomial(n = 1, p = prob)

#OR

#Simulating m outcomes
np.random.binomial(n = 1, p = prob, size = m)
```

Simulating a Bernoulli Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Bernoulli distribution with parameter p = 0.4
sample = np.random.binomial(n=1,p=0.4,size=1000)
simple_hist(sample, 2, 'Histogram of Bernoulli RVs') #plot a histogram of the RVs
x = np.linspace(-0.5,1.5,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Bernoulli data') #Plot the ECDF
```

The Binomial Distribution

The Binomial distribution is a *discrete* distribution defined on sample space $S = \{0, 1, 2, ..., n\}$ where n is the number of Bernoulli trials. It has two parameters: the probability of success p in a single trial, and the number of trials n. The mean of this distribution is np.

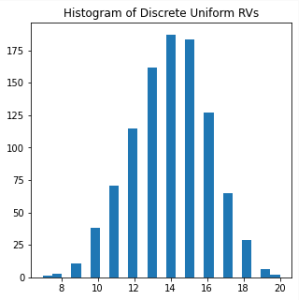
If X is a Binomial RV with parameters n and p, we say that:

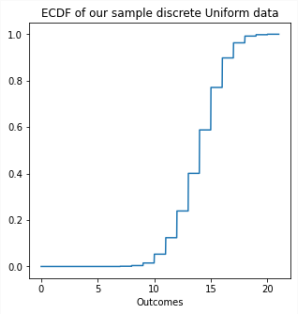
$$X \sim \mathcal{B}(n, p)$$

In Python, we can use the function **np.random.binomial** to simulate values from a Bernoulli distribution with parameters $n, prob$:

```
#Simulating a single outcome
np.random.binomial(n = n, p = prob)

#OR

#Simulating m outcomes
np.random.binomial(n = n, p = prob, size = m)
```

Simulating a Binomial Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Binomial distribution with parameters n=20, p=0.7
sample = np.random.binomial(n=20,p=0.7,size=1000)
simple_hist(sample, 25, 'Histogram of Binomial RVs') #plot a histogram of the RVs
x = np.linspace(0,21,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Binomial data') #Plot the ECDF
```

The Poisson Distribution

The Poisson distribution is a *discrete* distribution defined on sample space $S = \mathbb{Z} = \{0, 1, 2, ...\}$. It has one parameter: the mean arrival rate λ, and the mean of this distribution is λ.

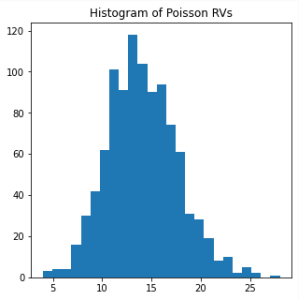
If X is a Poisson RV with parameter λ, we say that:

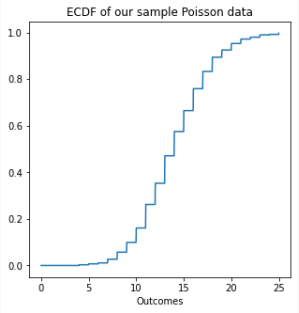
$$X \sim Pois(\lambda)$$

In Python, we can use the function **np.random.poisson** to simulate values from a Poisson distribution with parameter *Lambda*:

```
#Simulating a single outcome
np.random.poisson(lam = Lambda)

#OR

#Simulating m outcomes
np.random.poisson(lam = Lambda, size = m)
```

Simulating a Poisson Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Poisson distribution with mean arrival rate 14
sample = np.random.poisson(lam=14,size=1000)
simple_hist(sample, 25, 'Histogram of Poisson RVs') #plot a histogram of the RVs
x = np.linspace(0,25,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Poisson data') #Plot the ECDF
```

Simulating Continuous Random Variable Outcomes in Python

The Uniform Distribution

The Uniform distribution is a *continuous* distribution defined on sample space $S = [a, b]$. It has two parameters: the lower limit of the domain a and upper limit of the domain b. The mean of this distribution is $(b - a)/2$.

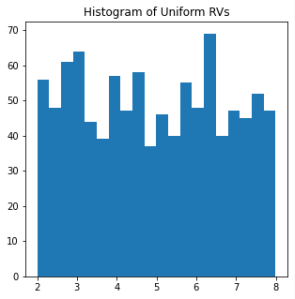
If X is a Uniform RV with parameters a and b, we say that:

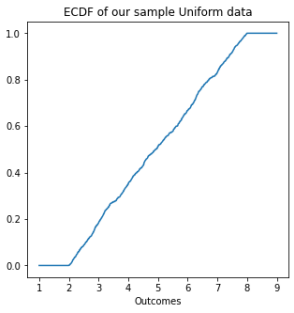
$$X \sim \mathcal{U}(a, b)$$

In Python, we can use the function **np.random.uniform** to generate an array of n values from a uniform distribution with parameters a and b:

```
#Simulating a single outcome
np.random.uniform(a,b)

#OR

#Simulating m outcomes
np.random.uniform(a,b,size = m)
```

Simulating a Uniform Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a Uniform distribution with parameters a=2,b=8
sample = np.random.uniform(2,8,size=1000)
simple_hist(sample, 20, 'Histogram of Uniform RVs') #plot a histogram of the RVs
x = np.linspace(1,9,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Uniform data') #Plot the ECDF
```

The Normal Distribution

The Normal distribution is a *continuous* distribution defined on sample space $S = \mathbb{R} = (-\infty, \infty)$. It has two parameters: the mean μ and standard deviation σ (When $\mu = 0$ and $\sigma = 1$, we refer to this as a standard normal distribution).

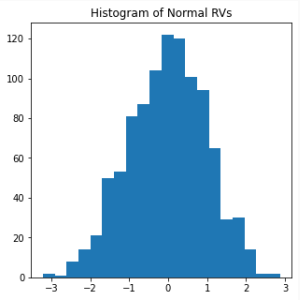
If X is a Normal RV with parameters μ and σ, we say that:

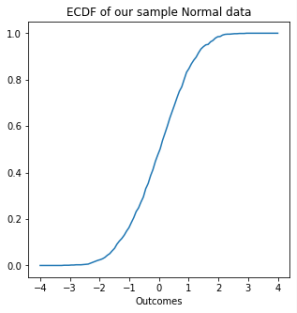
$$X \sim \mathcal{N}(\mu, \sigma)$$

In Python, we can use the function **np.random.normal** to simulate values from a normal distribution with mean *mu* and standard deviation *sigma* (the size argument can be omitted if generating a single value).

```
#Simulating a single outcome
np.random.normal(loc = mu, scale = sigma)

#OR

#Simulating m outcomes
np.random.normal(loc = mu, scale = sigma, size = m)
```

Simulating a Normal Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from a normal distribution with parameters mu = 0 and sigma = 1
sample = np.random.normal(loc=0, scale=1, size=1000)
simple_hist(sample, 30, 'Histogram of Normal RVs') #plot a histogram of the RVs
x = np.linspace(-4,4,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'ECDF of our sample Normal data') #Plot the ECDF
```

The Exponential Distribution

The Exponential distribution is a *continuous* distribution defined on sample space $S = \mathbb{Z}^* = [0, \infty)$. It has one parameter: the rate λ. The mean of this distribution is $1/\lambda$.

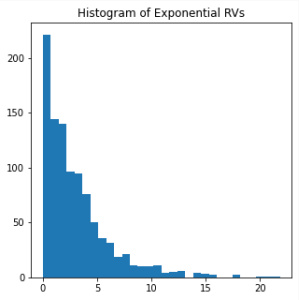
If X is an Exponential RV with parameter λ, we say that:

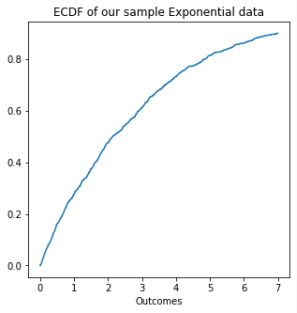
$$X \sim Exp(\lambda)$$

In Python, we can use the function ***np.random.exponential*** to simulate values from an exponential distribution with rate parameter *Lambda*, and mean *1/Lambda*:

```
#Simulating a single outcome
np.random.exponential(scale = 1/Lambda)

#OR

#Simulating m outcomes
np.random.exponential(scale = 1/Lambda, size = m)
```

Simulating an Exponential Distribution






```
import numpy as np
from statsmodels.distributions.empirical_distribution import ECDF
#create an array of 1000 RVs drawn from an exponential distribution with parameters lambda = 1/3
sample = np.random.exponential(scale=3, size=1000)
simple_hist(sample, 30, 'Histogram of Exponential RVs') #plot a histogram of the RVs
x = np.linspace(0,7,1000) #create the x values for our ECDF plot
F = ECDF(sample) #Create the ECDF function from our sample
y = F(x) #create y values for our ECDF plot
simple_plot(x,y,'Outcomes','ECDF of our sample Exponential data') #Plot the ECDF
```

Conclusion

Next Class

Homework:

- Review Block 4 readings and previous Python tutorial videos.

Next Lesson:

- Design and implement simple Monte Carlo simulations in python.
- Visualize the results of monte carlo simulation in python.
- Create and interpret an empirical cumulative distribution function (ECDF) from the results of the model.