Millie N White

4/26/2021

ISYE 6644 Project 2

Library of random variate generation routines.

I used python to create the libraries.

I started with the good random variable generator to produce 1 random variable.

The function desert_island_generator has 3 inputs: distribution_size and seed.

Distribution size is the number of random variables to be generated and I made sure the seed was large enough in order for the variables to be random.

The returns an array of pseudo random variables.

The function uniform produces an array of uniforms and I used that as my basis for obtaining the pseudo random variables of the rest of the distributions

A. Discrete Distributions

1. Geometric

Complete Source Code

```
def uniform(a = 0, b = 1, distribution_size = 1,seed = 123456789):
   return a + (b - a) * desert_island_generator(distribution_size,seed)
```

User Guide

To generate one uniform between 0 and 1

a and b are 0 and 1 respectively since the uniforms are from zero to 1.

The seed is 123456789 but could be any number that is larger.

Add the difference of b and a to a and multiply that value by the array generated by the desert_island_generator function.

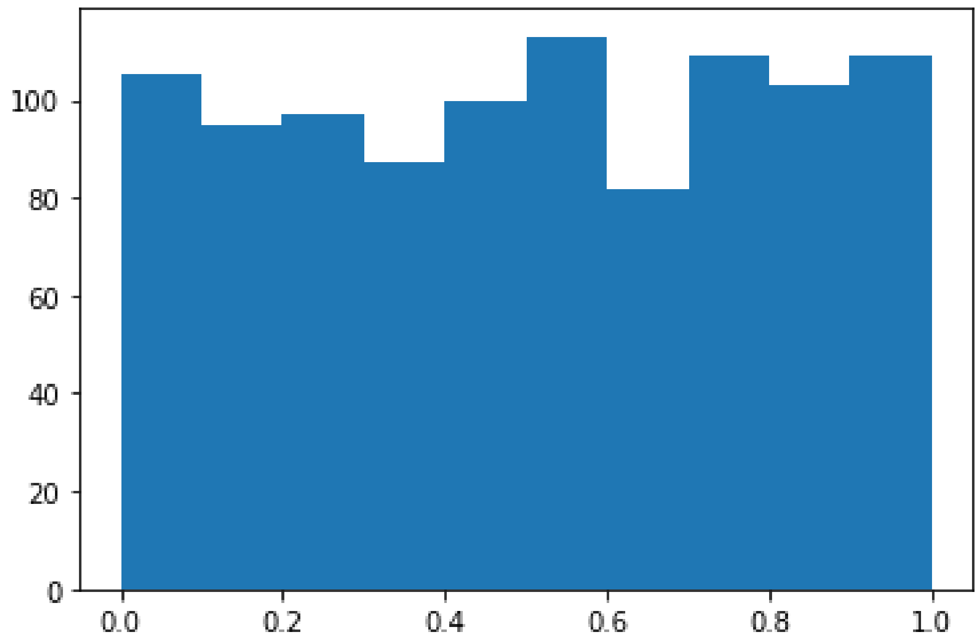
The uniform function returns an array

Example

unif = uniform(a = 0, b = 1, distribution_size = 1000,seed = 123456789)

The above code generates 1000 uniforms returned as an array.

The uniforms are plotted in the histogram below that shows that they are all between 0 and 1



2. **Bernoulli Distribution Bern(p)**

Complete Source Code

```
def bernoulli(num_coins = 1,distribution_size = 1,p = 0.5):
    seed = 123456789
    bern = uniform(0,1,distribution_size,seed)
   #check if random variable is less than or greater than probability of 0.5
    #then convert it to integer 0 if false or 1 if true
    bern = (bern < p).astype(int)
    return bern
```

User Guide

The bernoulli function calls the uniform distribution function with parameters a=0, b=1,the distribution_size and the p value

The array returned has either 0 or 1 values.0 if the values is less then p and 1 if the value is greater than p.

Example

bernoulli()

The above code calls the Bernoulli function and uses it's default values to generate 1 random variable of 0 or 1.

3. Binomial Distribution

Complete Source Code

```
def binomial(num_coins = 1,distribution_size = 1,p = 0.5 ):
    binomials = []  #list of binomial random variables
    for _ in range(distribution_size): #run this 1 time
        seed = random.randint(123456789,500000000000000)
        U = uniform(distribution_size = num_coins,seed = seed)

        #check if each value in the array is less than/equal to p and convert to 1 or 0
        ans = (U <= p).astype(int)
        #Add up the values of each trial
        binomials.append(np.sum(ans))
    return np.array(binomials)
```

User Guide

The binomial function calls the uniform distribution function with parameters a=0, b=1,the distribution_size and the p value

The array returned has the sum of how many of the pseudorandom values are greater or less then p, added up.

Example

num_coins = 50

distribution_size = 100

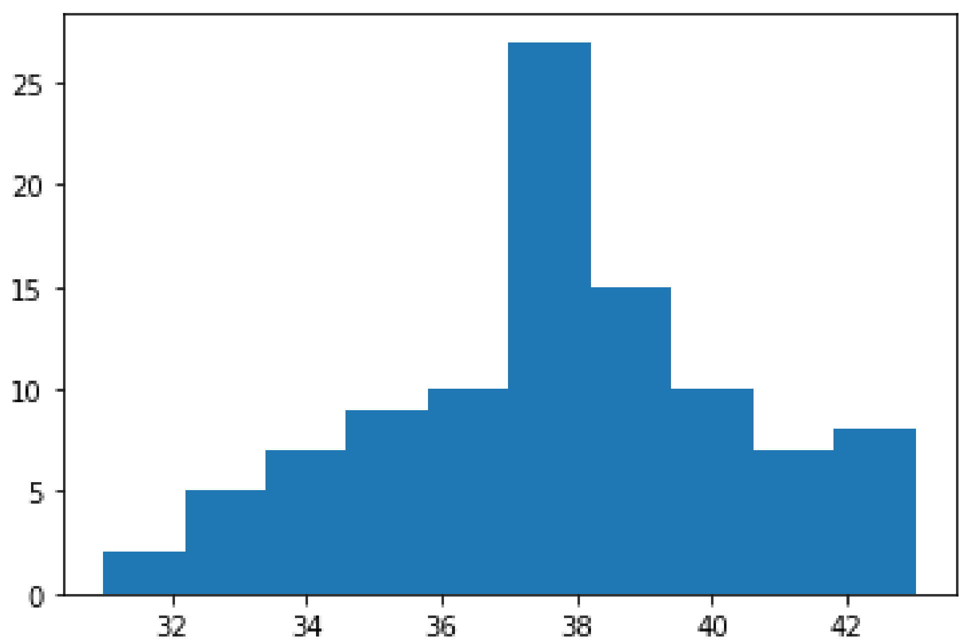
p = 0.75

binom = binomial(3,20,0.55)

binom

The example code shows 50 weighted coins flipped 100 times with probability of 0.75 of getting heads

Following is the resulting histogram



4. Geometric Distribution

Complete Source Code

def geometric(distribution_size = 1,p = 0.3):

seed = random.randint(123456789,500000000000000)

U = uniform(0,1,distribution_size = distribution_size,seed = seed)

geometric = np.log(1 - U) / np.log(1 - p)

return geometric

User Guide

The geometric function calls the uniform distribution function with parameters a=0, b=1,the distribution_size and the seed.

The seed is a large number generated using random.randint and is between the 2 large numbers shown.

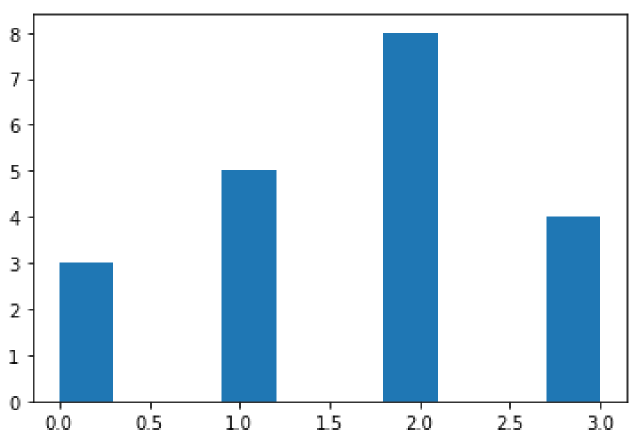
Each pseudo random number in the array returned by the uniform function is run through the formula above to generate a geometric distribution

Example

#Histogram of geometric distribution of size 10 with p = 0.3

geometric(10,0.3)

Following is the histogram for the geometric distribution



B. Continuous Distributions

1. Exponential Distribution

Complete Source Code

```
def exponential(lambd = 5,distribution_size = 1):
    seed = random.randint(123456789,500000000000000)
    U = uniform(0,1,distribution_size = distribution_size,seed = seed)
    U = U.tolist()
    exponential = []
    for u in U: #loop through list
        exp = -(1/lambd) * np.log(1 - u)
        exponential.append(exp) #append values in list
    return exponential
```

User Guide

The parameters of the exponential function are lambda and distribution_size.

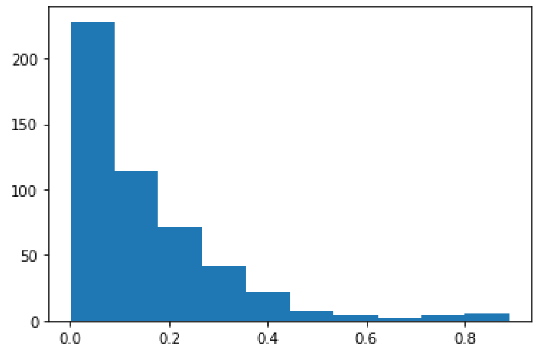
The exponential distribution calls the uniform distribution .

When the array of uniforms is returned by the uniform function they are run through the formula above to generate an exponential distribution

Example

```
exp = exponential(7,500)
exp
plt.hist(exp)
plt.show()
```

Following is the histogram for the exponential distribution



2. Normal Distribution Normal(μ,σ2)

Complete Source Code

```
def normal(mu = 0.0,stdev = 1.0,distribution_size = 1):
    seed1 = random.randint(123456789,500000000000000)
    seed2 = random.randint(123456789,500000000000000)
    U1 = uniform(0,1,distribution_size = distribution_size,seed = seed1)
    U2 = uniform(0,1,distribution_size = distribution_size,seed = seed2)

    #print(U1)
    z0 = (np.sqrt(-2 * np.log(U1)) )* (np.cos(2 * np.pi * U2))
    z1 = np.sqrt(-2 * np.log(U1)) * np.sin(2 * np.pi * U2)
     #z0 with a mean of 0 and standard deviation of 1
    z0 = z0 * stdev + mu
    z1 = z1 * stdev + mu

    return z0
```

User Guide

The parameters of the normal function are mean, standard deviation and distribution_size.

The normal distribution function calls the uniform distribution

The array of uniforms is ran through the Box-Muller transform and values with a normal distribution are returned

Example

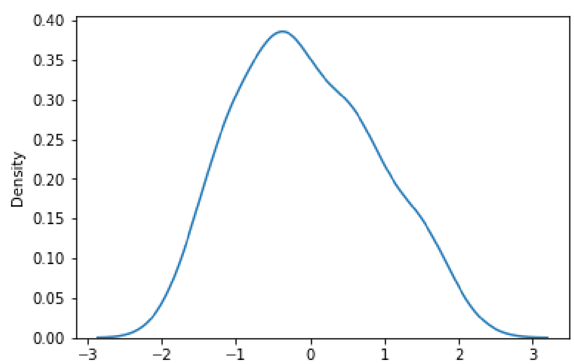
z0 = normal(0,1,100)

100 random variables with mean zero and standard deviation of 1

#Display distribution

sns.kdeplot(z0)

plt.show()



3. **Uniform Distribution**

Complete Source Code

```
def uniform(a = 0, b = 1, distribution_size = 1,seed = 123456789):
   return a + (b - a) * desert_island_generator(distribution_size,seed)
```

User Guide

The uniform function calls the desert_island_generator function which produced pseudo random numbers. The returned array is then manipulated using the formula *a + (b - a)*

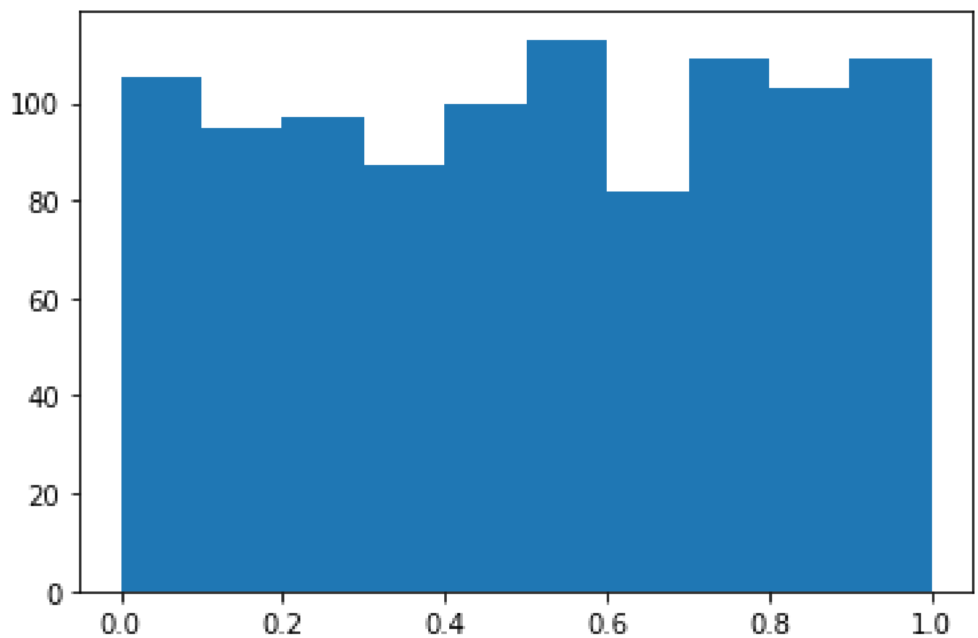
To ensure all the numbers in the array are between 0 and 1

Example

Generate 1000 uniform random variables between 0 and 1

unif = uniform(a = 0, b = 1, distribution_size = 1000,seed = 123456789)

The plot for this uniform distribution is below.



4. **Weibull**

Complete Source Code

```
def weibull(lambd = 5,beta = 2,distribution_size = 1):
    seed = random.randint(123456789,500000000000000)
    U = uniform(0,1,distribution_size = distribution_size,seed = seed)
    U = U.tolist()
    weibull = []
    for u in U: #loop through list
        weib = 1/lambd * (-(np.log(1 - u)))** 1/beta
        weibull.append(weib) #append values in list
  return weibull
```

User Guide

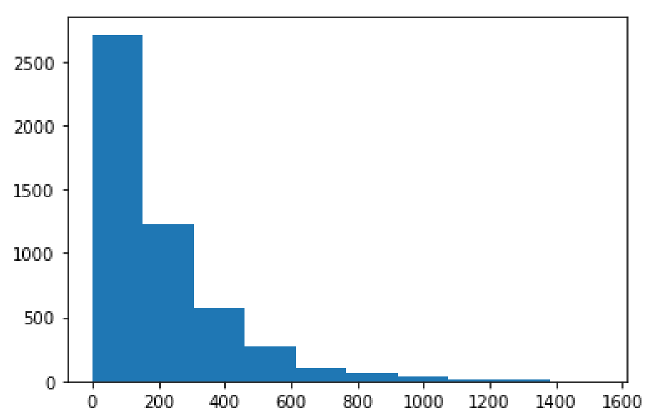
The parameters of the weibull function are lambda,beta and distribution_size.

The Weibull distribution function calls the uniform distribution

The returned array of uniforms is converted to Weibull random numbers by the formula above

Example

wei = weibull(lambd = 0.001,beta = 5,distribution_size = 5000)



5. **Triangular Distribution**

Complete Source Code

triangular(0,1,2) distribution

```
def triangular(distribution_size = 1):
   seed = random.randint(123456789,500000000000000)
   U = uniform(0,1,distribution_size = distribution_size,seed = seed)
   triangular = []
   for num in U:
     if num < 0.5:
         tr = np.sqrt(2 * num)
      elif num > 0.5:
         tr = 2 - np.sqrt(2-(1 - num))
      triangular.append(tr)
   triangular = np.asarray(triangular)
   return triangular
```

User Guide

The parameter of the triangular function is distribution_size.

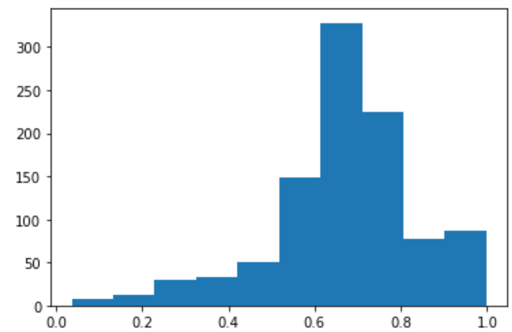
The triangular function calls the uniform function

The returned array of uniforms is converted to triangular by the if-elif statements which are obtained from calculating the cdf of a triangular(0,1,2) distribution.

Example

1000 values of a triangular(0,1,2) distribution

trian = triangular(1000)



References:

How to generate random variables from scratch (no library used)

https://towardsdatascience.com/how-to-generate-random-variables-from-scratch-no-library-used-4b71eb3c8dc7

Probability Distributions in Python

https://www.datacamp.com/community/tutorials/probability-distributions-python