BM20BTECH11001-Lab8

November 9, 2021

0.1 Generating pdf from data

[1]:

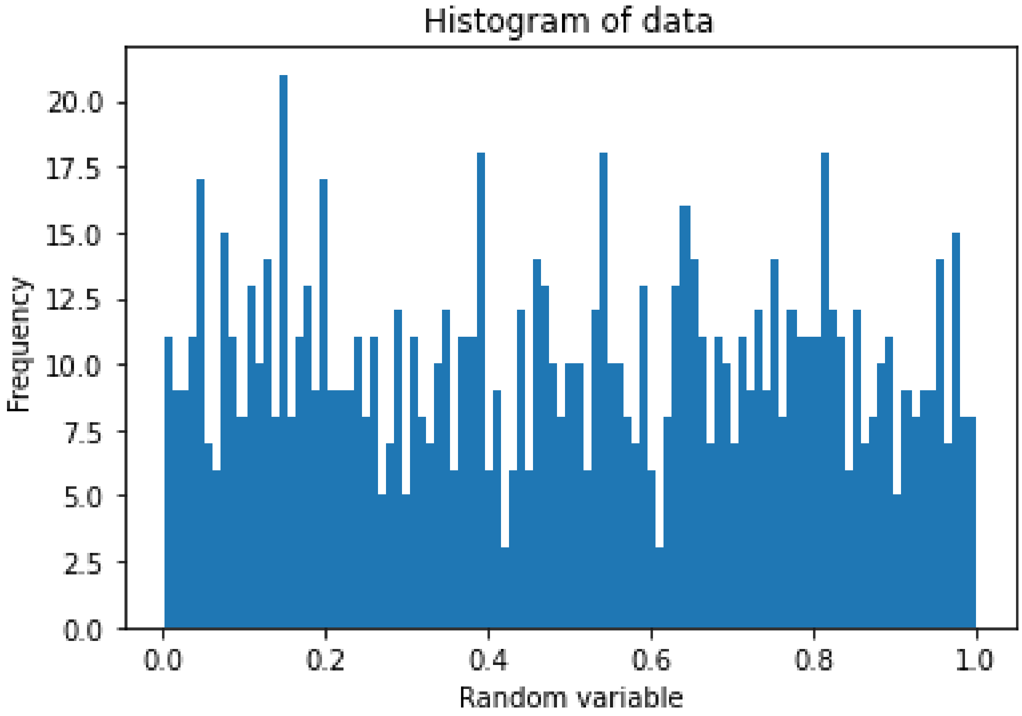
```
import numpy as np
import matplotlib.pyplot as plt
import random
```

[2]:

```
data = [np.random.uniform(0, 1) for t in range(0,1000)]
```

[3]:

```
fig, ax = plt.subplots()
ax.set_title('Histogram of data')
ax.set_xlabel('Random variable')
ax.set_ylabel('Frequency')
counts, bins, bars = ax.hist(data, bins=99);
```



[4]:

```
#Calculating actual probabilities of each random variable range
prob_array = counts/1000
```

[5]:

```
prob_array
```

[5]:

```
array([0.011, 0.009, 0.009, 0.011, 0.017, 0.007, 0.006, 0.015, 0.011,
       0.008, 0.013, 0.01 , 0.014, 0.008, 0.021, 0.008, 0.011, 0.013,
       0.009, 0.017, 0.009, 0.009, 0.009, 0.011, 0.008, 0.011, 0.005,
       0.007, 0.012, 0.005, 0.011, 0.008, 0.007, 0.01 , 0.012, 0.006,
       0.011, 0.011, 0.018, 0.006, 0.009, 0.003, 0.006, 0.012, 0.006,
       0.014, 0.013, 0.01 , 0.008, 0.01 , 0.01 , 0.006, 0.012, 0.018,
       0.01 , 0.01 , 0.008, 0.007, 0.013, 0.006, 0.003, 0.008, 0.013,
       0.016, 0.014, 0.011, 0.007, 0.011, 0.01 , 0.007, 0.011, 0.009,
       0.012, 0.009, 0.014, 0.008, 0.012, 0.011, 0.011, 0.011, 0.018,
       0.012, 0.011, 0.006, 0.012, 0.007, 0.008, 0.01 , 0.011, 0.005,
       0.009, 0.008, 0.009, 0.009, 0.014, 0.007, 0.015, 0.008, 0.008])
```

[6]:

```
#Calculating pdf from its definition: P.D.F = (Probability/(length of random␣
 ↪variable range)
pdf_array = prob_array/(bins[1]-bins[0])
```

[7]:

```
pdf_array
```

[7]:

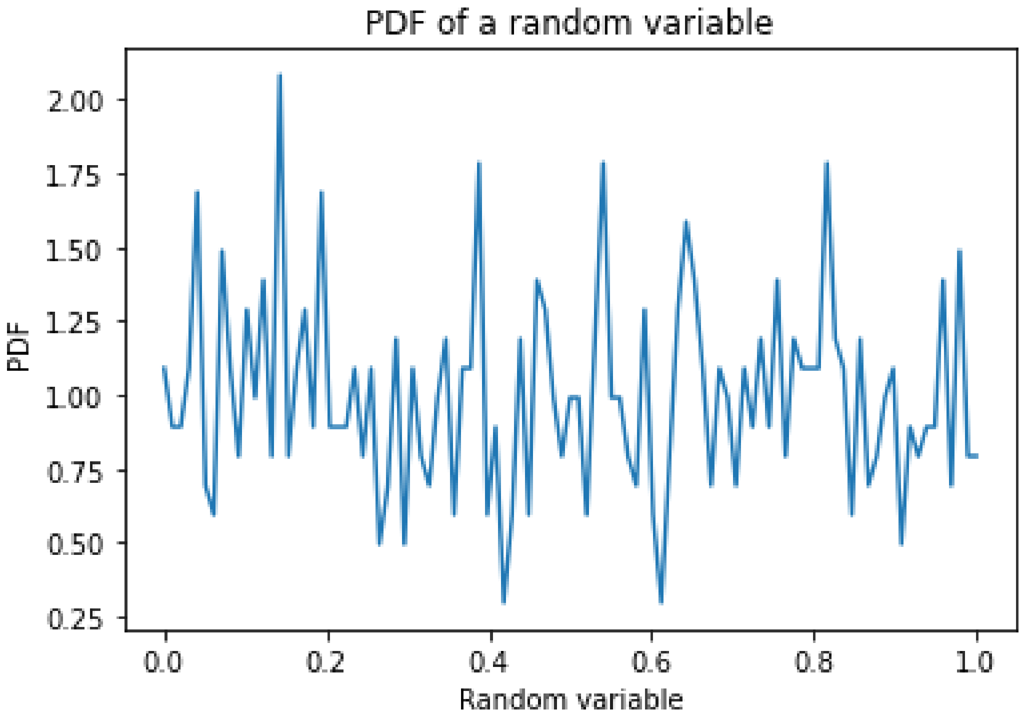
```
array([1.09069129, 0.89238378, 0.89238378, 1.09069129, 1.68561381,
       0.69407628, 0.59492252, 1.48730631, 1.09069129, 0.79323003,
       1.2889988 , 0.99153754, 1.38815255, 0.79323003, 2.08222883,
       0.79323003, 1.09069129, 1.2889988 , 0.89238378, 1.68561381,
       0.89238378, 0.89238378, 0.89238378, 1.09069129, 0.79323003,
       1.09069129, 0.49576877, 0.69407628, 1.18984504, 0.49576877,
       1.09069129, 0.79323003, 0.69407628, 0.99153754, 1.18984504,
       0.59492252, 1.09069129, 1.09069129, 1.78476757, 0.59492252,
       0.89238378, 0.29746126, 0.59492252, 1.18984504, 0.59492252,
       1.38815255, 1.2889988 , 0.99153754, 0.79323003, 0.99153754,
       0.99153754, 0.59492252, 1.18984504, 1.78476757, 0.99153754,
       0.99153754, 0.79323003, 0.69407628, 1.2889988 , 0.59492252,
       0.29746126, 0.79323003, 1.2889988 , 1.58646006, 1.38815255,
       1.09069129, 0.69407628, 1.09069129, 0.99153754, 0.69407628,
       1.09069129, 0.89238378, 1.18984504, 0.89238378, 1.38815255,
       0.79323003, 1.18984504, 1.09069129, 1.09069129, 1.09069129,
       1.78476757, 1.18984504, 1.09069129, 0.59492252, 1.18984504,
       0.69407628, 0.79323003, 0.99153754, 1.09069129, 0.49576877,
       0.89238378, 0.79323003, 0.89238378, 0.89238378, 1.38815255,
       0.69407628, 1.48730631, 0.79323003, 0.79323003])
```

[8]:

```
x_data = np.linspace(0,1,99)
```

[9]:

```
fig, ax = plt.subplots()
ax.set_title('PDF of a random variable')
ax.set_xlabel('Random variable')
ax.set_ylabel('PDF')
ax.plot(x_data, pdf_array);
```



0.2 Generating data from pdf

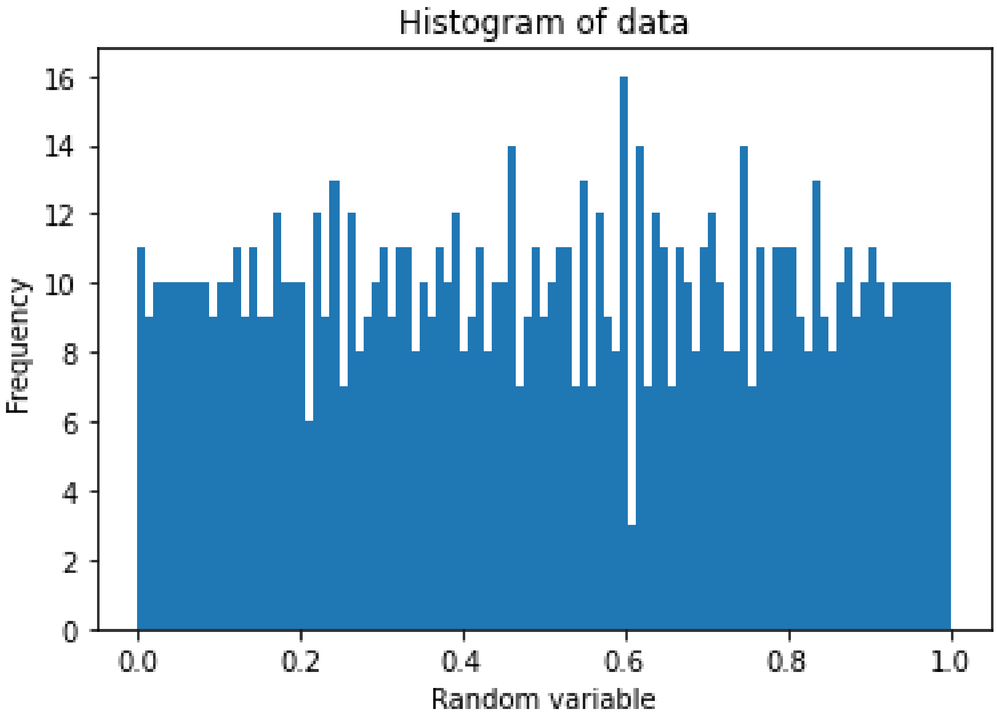
[10]:

```
#Function to generate data from PDF
def generate_data_from_pdf(pdf, intervals, no_of_observations):
    arr = []
    inter_length = intervals[1]-intervals[0]
    for x in range(0,len(intervals)-1):
        arr.append(int(pdf(intervals[x])*inter_length*no_of_observations))
    data_final = []
    i = 1
    for b in arr:
        if i<len(intervals):
            for a in range(0, b):
                data_final.append(np.random.uniform(intervals[i-1],␣
 ↪intervals[i]))
        i = i+1
    return data_final
```

0.2.1 Demonstrating the function with an example

```
[11]: data_1 = generate_data_from_pdf(lambda x: 1, np.linspace(0,1,101), 1000)
```

```
[12]: fig, ax = plt.subplots()
      ax.set_title('Histogram of data')
      ax.set_xlabel('Random variable')
      ax.set_ylabel('Frequency')
      ax.hist(data_1, bins=101);
```


Histogram of data
Frequency
Random variable