Generating N Random Variables Using Poisson Distribution

Shamiul Hasan
1505038

September 21, 2020

1 Problem Description

In probability theory and statistics, the Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occurring in a fixed interval of time or space if these events occur with a known constant mean rate and independently of the time since the last event. The Poisson distribution can also be used for the number of events in other specified intervals such as distance, area or volume.
In this assignment, we have to generate N number of random variables using the Poisson Distribution. We also have to show the curve plotting the probability distribution and observed frequencies in fraction (frequency/N) i.e. observed probability.
For this problem, the parameter $\lambda = 1$ and $N = 1000$.

2 Definitions

A discrete random variable X is said to have a Poisson distribution with parameter $\lambda > 0$, if, for $k = 0, 1, 2, \cdots$ the probability mass function of X is given by,

$$Mass, p(x) = \begin{cases} \frac{e^{-\lambda}\lambda^x}{x!}, & \text{if } x \epsilon \{0, 1, 2, \cdots\} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$Distribution, F(x) = \begin{cases} \frac{e^{-\lambda}\lambda^x}{x!}, & \text{if } x < 0 \\ e^{-\lambda} \sum_{i=0}^{\lfloor x \rfloor} \frac{\lambda^i}{i!}, & \text{if } x \geq 0 \end{cases} \tag{2}$$

3 Code

Below is the Python code to simulate this problem.

```
#!/home/heisenberg/anaconda3/bin/python

import matplotlib.pyplot as plt
import math
from random import seed
from random import random
from random import uniform

# This function is to plot a single curve


def plotSingleGraph(x, y):
    plt.plot(x, y)
    plt.xlabel('X')
    plt.ylabel('CDF')
    plt.title('Poisson Distribution')
    plt.show()

# This function calculates the CDF or F(x) of the Poisson Distribution


def getCDF(x, Lambda):
    if x < 0:
        return 0
    else:
        return math.exp(-1.0 * Lambda) * sum([(pow(Lambda, i) / math.
   factorial(i)) for i in range(math.floor(x+1))])

# This function calculates the PDF or p(x) of the Poisson Distribution


def getPx(x, Lambda):
    if x < 0:
        return 0
    else:
        return math.exp(-1.0 * Lambda) * math.pow(Lambda, x) / math.
   factorial(x)


# This function calculates the Frequencies and Cummulative Frequencies of
   random numbers between 0 and 1.
def getExperimentalValues(cdfs):
    discreteValues = []

    seed(1)

    for _ in range(1000):
        # generate random numbers between 0-1
        value = random()

```

```
48          discreteVal = 0
49
50          # Finding the smallest number which is larger than 'value'. It is
        actually upper bound.
51          # It can be solved using both Bruteforce and Binary Search. I used
         the latter.
52
53          lo = 0
54          hi = len(cdfs)-1
55          while lo <= hi:
56              mid = int(lo + (hi-lo)/2)
57
58              if value > cdfs[mid]:
59                  lo = mid+1
60              else:  # value <= cdfs[mid]
61                  hi = mid-1
62                  discreteVal = mid
63
64          discreteValues.append(discreteVal)
65
66      freqencies = [0 for i in range(21)]
67
68      for val in discreteValues:
69          freqencies[val] = freqencies[val] + 1
70
71      for i in range(len(freqencies)):
72          freqencies[i] = freqencies[i] / 1000
73
74      noncumulative = [i for i in freqencies]
75
76      for i in range(1, len(freqencies)):
77          freqencies[i] = freqencies[i] + freqencies[i-1]
78
79      return discreteValues, freqencies, noncumulative
80
81
82  # In this function I calculated p(x), F(x), frequencies/N, cumulative
        frequencies/N and plot them.
83
84  def main():
85      ####### Theoratical values #######
86      x = [i for i in range(0, 21)]
87      Lambda = 1
88
89      ######### p(x) ############
90
91      Px = [getPx(i, Lambda) for i in x]
92      plt.plot(x, Px, label="x vs P(x)")
93
94      ########## F(x) ############
95      cdfs = [getCDF(i, Lambda) for i in x]
96      plt.plot(x, cdfs, label="x vs F(x)")
97
98      ######## Experimental values #########
```

```
    # print(cdfs)

    discreteValues, cummulative_freq, noncumulative_freq =
    getExperimentalValues(
        cdfs)
    plt.plot(x, cummulative_freq, label="x vs cummulative frequencies")
    plt.plot(x, noncumulative_freq, label="x vs non-cummulative
    frequencies")
    ############## Plot them ###############
    plt.xlabel('x - axis')
    plt.ylabel('y - axis')
    plt.title('Poisson')
    plt.legend()
    plt.show()


if __name__ == '__main__':
    main()
```

Listing 1: Python Code

4 Results

4.1 Table

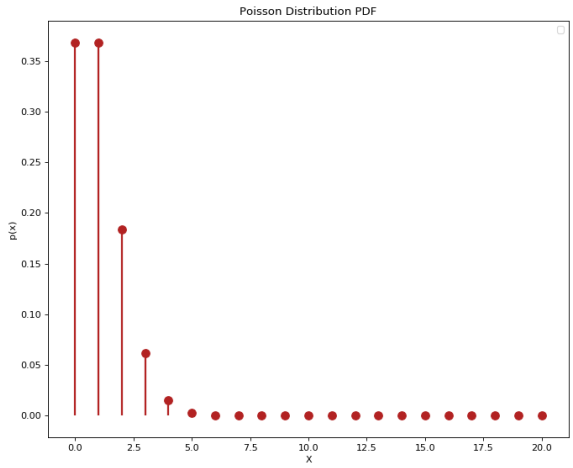
Here are the results I got from theoretical and experimental calculations.

x	p(x)	Frequency/N	F(x)	Cumulative Frequency/N
0	0.36787944117144233	0.344	0.36787944117144233	0.344
1	0.36787944117144233	0.37	0.7357588823428847	0.714
2	0.18393972058572117	0.207	0.9196986029286058	0.9209999999999999
3	0.06131324019524039	0.057	0.9810118431238462	0.978
4	0.015328310048810098	0.02	0.9963401531726562	0.998
5	0.003065662009762019 6	0.002	0.9994058151824182	1.0
6	0.0005109436682936699	0.0	0.999916758850712	1.0
7	7.299195261338141e-05	0.0	0.9999897508033254	1.0
8	9.123994076672677e-06	0.0	0.9999988747974021	1.0
9	1.0137771196302974e-06	0.0	0.9999998885745217	1.0
10	1.0137771196302975e-07	0.0	0.9999999899522337	1.0
11	9.216155633002704e-09	0.0	0.9999999991683893	1.0
12	7.68012969416892e-10	0.0	0.9999999999364023	1.0
13	5.907792072437631e-11	0.0	0.9999999999954803	1.0
14	4.2198514803125934e-12	0.0	0.9999999999997002	1.0
15	2.8132343202083955e-13	0.0	0.9999999999999816	1.0
16	1.7582714501302472e-14	0.0	0.9999999999999991	1.0
17	1.0342773236060278e-15	0.0	1.0000000000000002	1.0
18	5.745985131144599e-17	0.0	1.0000000000000002	1.0
19	3.0242027006024205e-18	0.0	1.0000000000000002	1.0
20	1.5121013503012103e-19	0.0	1.0000000000000002	1.0

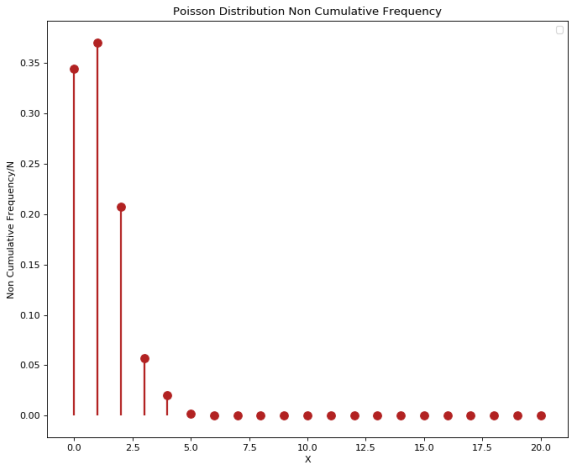
Table 1: Theoretical and Experimental Values Generated by Python Code

Table 1 shows us that $p(x)$ and $Frequency/N$ values are apporximately similar. Same goes for $F(x)$ and $CumulativeFrequency/N$. This is our expected result.

4.2 Graphs

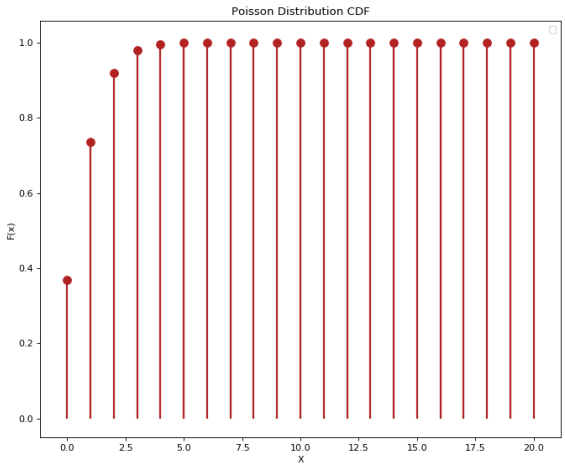


(a) p(x) vs x

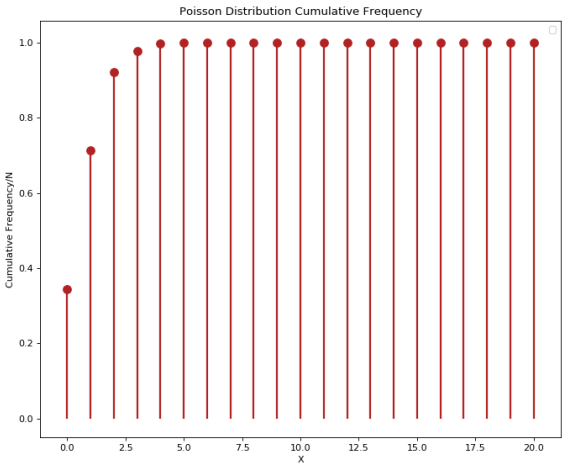


(b) Frequency/N vs x

Figure 1: PDF and Frequency Side by Side

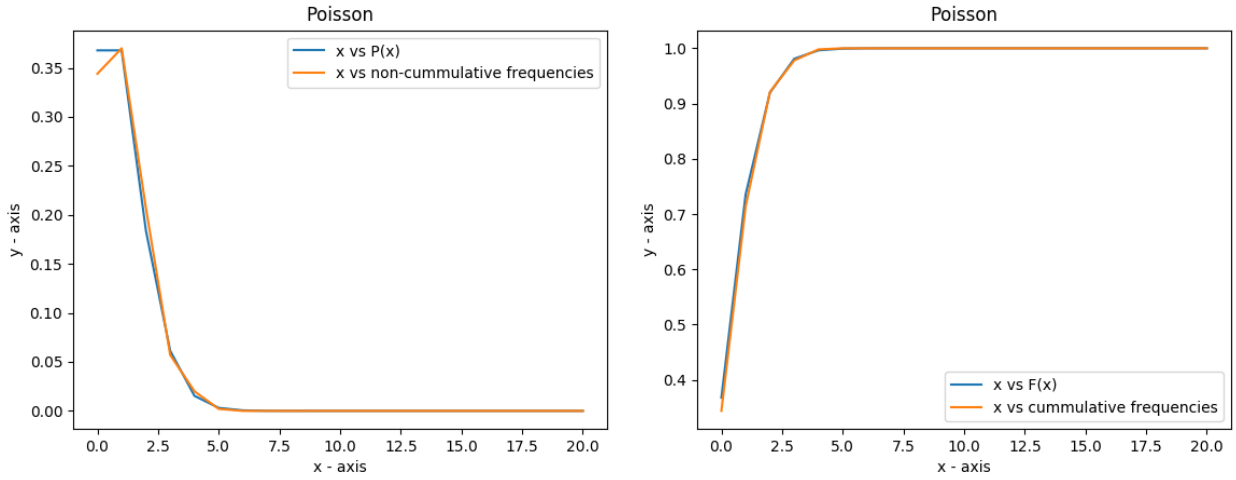


(a) F(x) vs x



(b) Cumulative Frequency/N vs x

Figure 2: CDF and Cumulative Frequency Side by Side



(a) Overlap of PDF and Frequency/N (b) Overlap of CDF and Cumulative Frequency

Figure 3: Similarities between graphs

4.2.1 Observation

From Graph 3 we can see that PDF almost overlaps with Frequency/N and CDF overlaps with Cumulative Frequency. That means, our experimental values are relevant with our theoretical values.

5 Conclusions

Graph 3 actually shows us that our experimental and theoretical results are compatible to each other. So, we can generate N random variables using Poisson distribution.