EE 381 Project 3

Binomial and Poisson Distributions

by Alexandra McBride

October 16, 2019

California State University, Long Beach

Dr. Anastasios Chassiakos

Problem 1

Introduction

This problem involves the rolling of three dice n times, where n is the number of times. In this scenario, the experiment will be considered successful if the first die lands on one, the second die lands on two, and the third die lands on three. We will be conducting Bernoulli trials for a given experiment, and each experiment will have 1000 trials. We use a random variable called X, which will be the number of successes for the experiment. We will then repeat the experiment 10,000 times, assigning their X values, and generate a PMF plot using these random variable results. This plot result will be for experimental Bernoulli trials.

Methodology

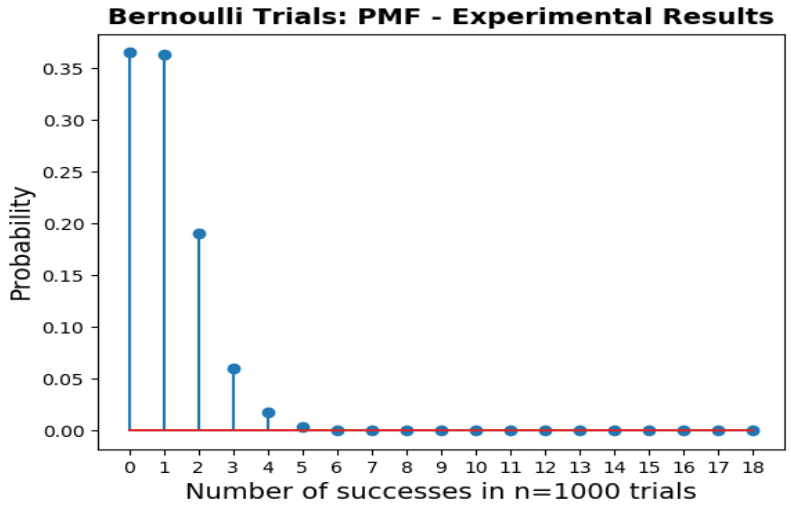
For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy` and `mathplotlib.pyplot` libraries and a method called `nSidedDie`.

In the program, I will save three die variables, which hold the values that the die landed on using the `nSidedDie` function. From there, I will check if the first die is one, the second die is two, and the third die is three. If this occurs, I will save it to a counter which keeps track of the random variable X for each experiment, which is the number of successes for the experiment. For one experiment, we will loop through this 1000 times and the number of successes will be saved in a results array.

We will execute the experiment 10,000 times and use the result for each experiment to create the PMF plot, which will generate the probability of a given experiment having a certain number of successes, or having a certain X value. The X values for the graph range from zero successes to eighteen successes.

Results and Conclusion

Here is the PMF plot, which show the probability of X values ranging from zero to eighteen:



In the results, we see that the number of successes X in a given experiment is usually zero or one, and that it is not very often that the number of successes X will go beyond those values. To conclude, the three dice landing in that pattern will not normally happen if done multiple times.

Appendix

File: problem1.py

```
import numpy as np
import nSidedDie as nsd
import plotting as plt

#PROBLEM 1: Bernoulli Trials - Experimental

def problem1():
    #The probability vector for the multi - sided dice is:
    p = [0.1, 0.1, 0.1, 0.3, 0.2, 0.2]
    N=10000
    n=1000
    results=np.zeros((N,1))

    for i in range (N): #peforms 10000 experiments
        numOfSucesses=0
        for j in range(n): #performs 1000 trials per experiment
            #roll three dice
            die1=nsd.rollDie(p)
            die2=nsd.rollDie(p)
            die3=nsd.rollDie(p)
```

```
        #if succesful, will add to a counter
         if die1==1 and die2==2 and die3==3:
            numOfSucesses+=1

      results[i]=numOfSucesses #storing number of successes
      print(results[i])

   #plotting results
   plt.plotting(18,results,N)

problem1()
```

File: plotting.py

```
def plotting(endpoint,outcomes,N,):
   # Setting the x values
   xRange = range(0, endpoint + 2)
   xSize = np.size(xRange)  # number of x values

   # Histogram to find final probabilities of each outcome
   hist, bin_edges = np.histogram(outcomes, bins=xRange)
   ticks = bin_edges[0:xSize - 1]
   plt.close('all')
   prob = hist / N

   # Plotting stem plot for PMF
   plt.stem(ticks, prob, use_line_collection=True)  # stem plot (x,y,...)

   # Labels for the plot
   plt.title('Bernoulli Trials: PMF - Experimental Results', fontsize=14
        fontweight='bold')
   plt.xlabel('Number of successes in n=1000 trials', fontsize=14)
   plt.ylabel('Probability', fontsize=14, )
   plt.xticks(ticks)
   filename=input("Enter a name and extension (.pdf) to save the file as :")
   plt.savefig(filename)
```

File: nSidedDie.py

```
import numpy as np
def rollDie(p):
```

```
#Getting n (number of sides)
n=len(p)

#Setting up the cumulative sum
cumulProbs = np.cumsum(p)  # array of sums
cumulProbs0 = np.append(0, cumulProbs)

#random number between 0 and 1
randNum = np.random.rand()
for j in range(0, len(p)):
   # checks for the range that the dieRoll is in and get the probability for that range
   if randNum > cumulProbs0[j] and randNum <= cumulProbs0[j + 1]:
      side = j+1
      return side
return 0
```

Problem 2

Introduction

This problem involves the same scenario as in the Problem 1 Introduction where we conducted Bernoulli trials. In the problem, we will now use the binomial formula instead of gathering our results experimentally as in Problem 1. We will use this formula to get the probability for the outcome of each random variable X, or the number of successes. We will use these results to create a PMF plot for this problem, which we will then compare with the results from Problem one where we used an experimental approach.

Methodology

For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy` and `mathplotlib.pyplot` libraries .

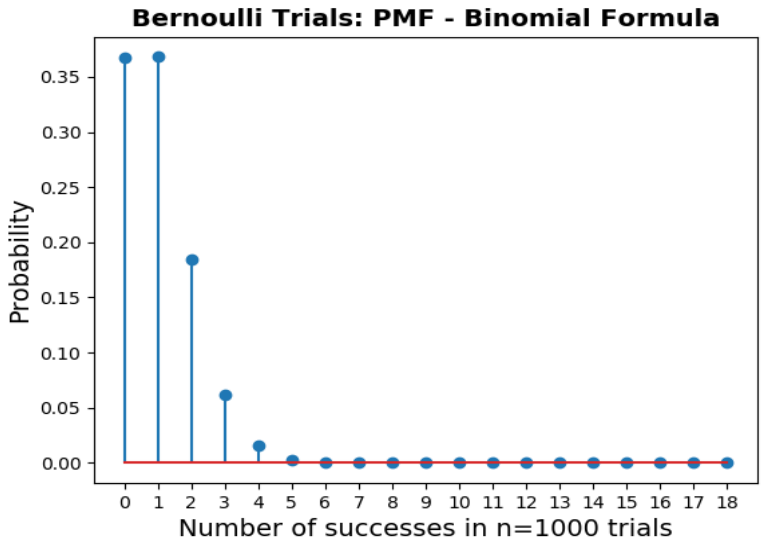
In this problem, we will a formula called the binomial formula. This formula is as follows:

$$p(X = x) = \binom{n}{x} p^x q^{n-x}$$

where n is the number of trials, x is the number of successes, p is the probability of success in the trial, and q is the probability of failure in the trial. The program will perform this calculation for numbers zero through eighteen, which will each be stored in a list to represent the x values. It will then perform the calculation for the given x, then store the result in another list to represent the corresponding $f(x)$ values. Lastly, the program will plot the values using the x list and the $f(x)$ list, which will give the PMF plot.

Results and Conclusion

Here is the PMF plot, which show the probability of X values ranging from zero to eighteen:



In the results, we see that the number of successes X in a given experiment is usually zero or one, and that it is not very often that the number of successes X will go beyond those values. Compared with the experimental results, the probabilities are about the same, thus verifying the binomial formula and allowing for a shorthand way to calculate the probability of a given X value.

Appendix

File: problem2.py

```
import numpy as np
import math as m
import matplotlib.pyplot as plt

#PROBLEM 2: Bernoulli Trials using Binomial Formula

def problem2():
    n = 1000
    p=0.001
    q=1-p
    x=np.zeros((19,1))
    y=np.zeros((19,1))

    #Performing the calculation using the binomial formula
    for k in range(19):
        f=m.factorial(n)/(m.factorial(k)*m.factorial(n-k)) * (p**k) * (q**(1000-k))
        x[k]=k
        y[k]=f
```

```
    #Plotting the PMF
    plotting(x,y)

def plotting(x,y):
    # Setting the x values
    xRange = range(0,19)

    # Plotting stem plot for PMF
    plt.stem(x,y, use_line_collection=True)  # stem plot (x,y,...)

    # Labels for the plot
    plt.title('Bernoulli Trials: PMF - Binomial Formula', fontsize=14,
        fontweight='bold')
    plt.xlabel('Number of successes in n=1000 trials', fontsize=14)
    plt.ylabel('Probability', fontsize=14, )
    plt.xticks(xRange)
    filename=input("Enter a name and extension (.pdf) to save the file as :")
    plt.savefig(filename)

problem2()
```

Problem 3

Introduction

This problem involves the same scenario as in the Problem 1 Introduction where we conducted Bernoulli trials. In the problem, we will now use the Poisson Distribution formula instead of gathering our results experimentally as in Problem 1. We will use this formula to get the probability for the outcome of each random variable X, or the number of successes. We will use these results to create a PMF plot for this problem, which we will then compare with the results from Problem1 where we used an experimental approach.

Methodology

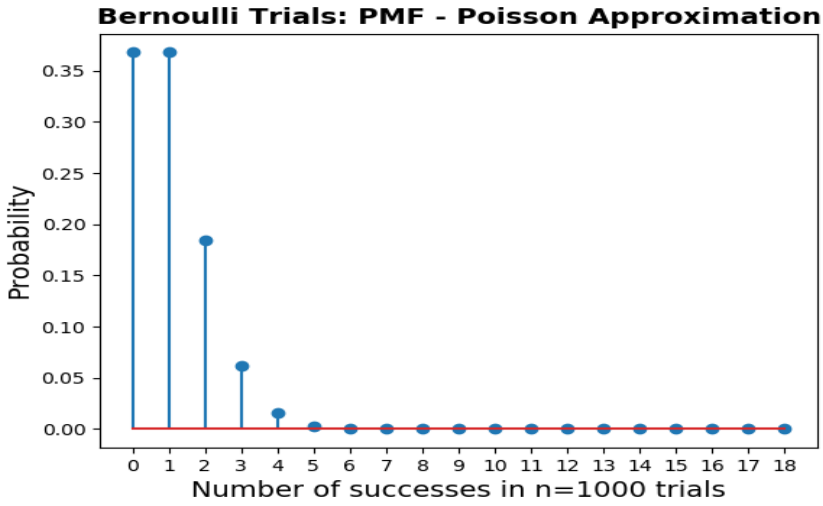
For this problem, I will be using Python in the PyCharm IDE. The tools used will include methods from the `numpy` and `mathplotlib.pyplot` libraries . In this problem, we will a formula called the Poisson distribution formula. This formula is as follows:

$$p(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

where x is the number of successes, λ is np, where n the number of trials multiplied and p is the probability of success in a given trial. The program will perform this calculation for numbers zero through eighteen, which will each be stored in a list to represent the x values. It will then perform the calculation for the given x, then store the result in another list to represent the corresponding $f(x)$ values. Lastly, the program will plot the values using the x list and the $f(x)$ list, which will give the PMF plot.

Results and Conclusion

Here is the PMF plot, which show the probability of X values ranging from zero to eighteen:



In the results, we see that the number of successes *X* in a given experiment is usually zero or one, and that it is not very often that the number of successes *X* will go beyond those values. Compared with the experimental results, the probabilities are about the same, thus verifying the Poisson distribution formula and allowing for a second shorthand way to calculate the probability of a given *X* value.

Appendix

File: problem3.py

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

#PROBLEM 3: Bernoulli Trials by Poisson distribution

def problem3():
    n = 1000
    p = 0.001
    x = np.zeros((19, 1))
    y = np.zeros((19, 1))

    #Performing calculations using Poisson distribution formula
    for k in range(19):
        lambdaVal=n*p
        f = (lambdaVal**k / math.factorial(k)) * math.exp(-lambdaVal)
        x[k] = k
        y[k] = f

    plotting(x, y)
```

```
def plotting(x,y):
    # Setting the x values
    xRange = range(0,19)
    xSize = np.size(xRange)  # number of x values

    # Plotting stem plot for PMF
    plt.stem(x,y, use_line_collection=True)  # stem plot (x,y,...)

    # Labels for the plot
    plt.title('Bernoulli Trials: PMF - Poisson Approximation', fontsize=14,
        fontweight='bold')
    plt.xlabel('Number of successes in n=1000 trials', fontsize=14)
    plt.ylabel('Probability', fontsize=14, )
    plt.xticks(xRange)
    filename=input("Enter a name and extension (.pdf) to save the file as :")
    plt.savefig(filename)

problem3()
```