EE 381
Project 3
Alexander Fielding

Problem 1

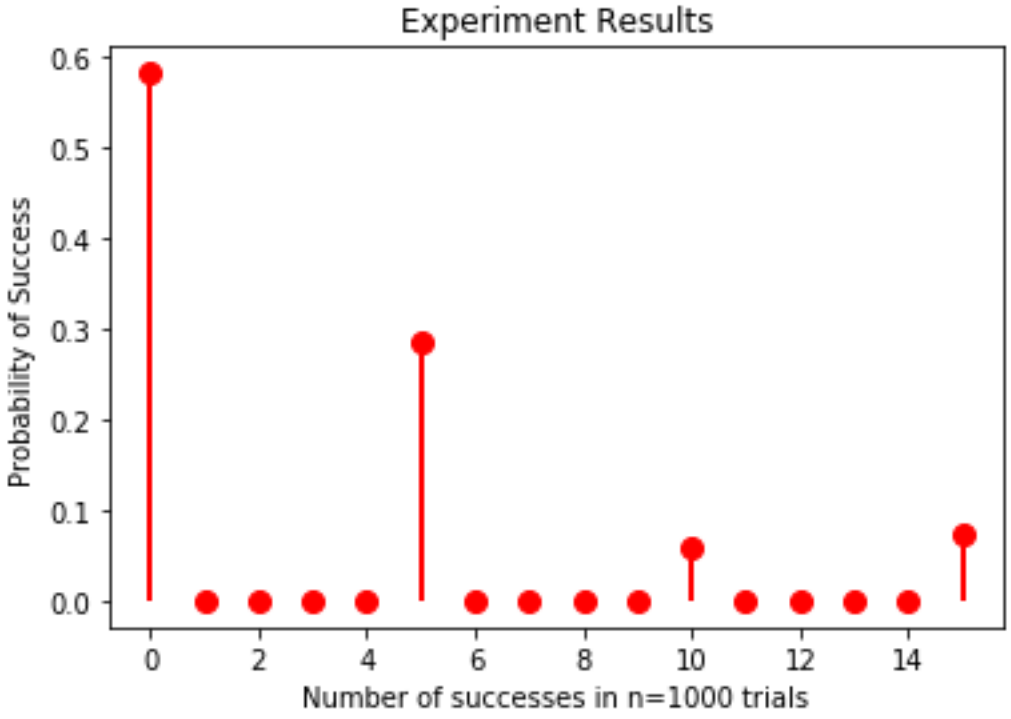
Introduction

This problem simulates the rolling of three 6-sided dice a thousand times to determine the probability of rolling a six on each die. The simulation is repeated 10,000 times and the resulting success out of a thousand are stored in a list. The data of the list is then plotted in a PMF graph.

Methodology

The simulation is built around a single function that handles the random dice roll and then a series of loops used to determine the number of successes out of a thousand and then repeating this process 10,000 times. At any point, if a roll results in anything other than 6 the simulation is determined as a failure and moves to the next attempt.

Results



Appendix

```
import random
import numpy as np
import pylab as plt
from scipy import stats

def diceroll():
    die = random.randint(1,6)
    return die

#Main
X = []
count = 0
countb = 0

while( countb < 10000):
    countb +=1
    success = 0
    while(count < 1001):
        count +=1
        alist = []
        i = 0
        while(i < 3):
            i+=1
            nxt = diceroll()
            if nxt != 6:
                i = 3
            else:
                alist.append(nxt)
                if len(alist) == 3:
                    success +=1
        X.append(success)

num_bins = 16
counts, bins = np.histogram(X, bins=num_bins)
bins = bins[:-1] + (bins[1] - bins[0])/2
probs = counts/float(counts.sum())

xk = np.arange(len(probs))
pk = probs
custm = stats.rv_discrete(name='custm', values=(xk, pk))

fig, ax = plt.subplots(1, 1)
ax.plot(xk, custm.pmf(xk), 'ro', ms=8, mec='r')
ax.vlines(xk, 0, custm.pmf(xk), colors='r', linestyles='-', lw=2)
plt.title('Experiment Results')
plt.ylabel('Probability of Success')
plt.xlabel('Number of successes in n=1000 trials')
plt.show()
```

Problem 2

Introduction

Here me attempt to avoid the simulation all together by utilizing the binomial theorem to determine the probability of rolling a specified amount of success (x) over the same sample size of a thousand rolls (n). We know that the probability of rolling a six on a die is 1/6, therefore the probability of this occurring three times in a row is 1/216 (p). The probability of failure used is 1-p (q).

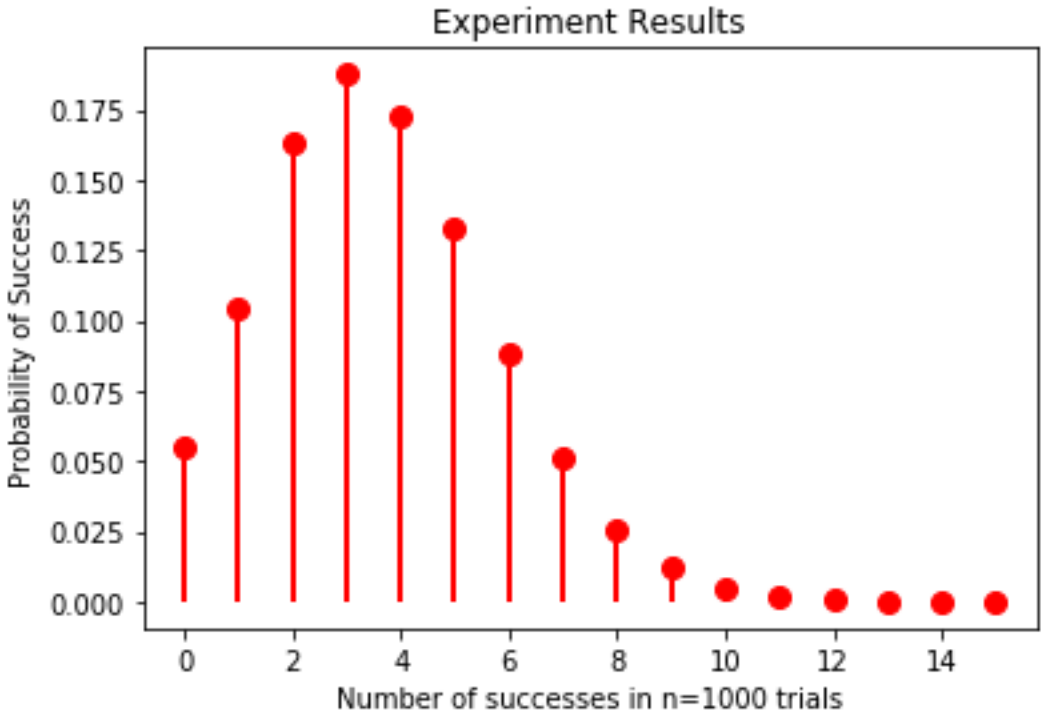
By using the binomial theorem, we can avoid having to run numerous simulations to determine the result which can be costly. The binominal theorem saves us both time and energy while theoretically giving us the same results of the many simulations.

Methodology

To accomplish this, I utilized the binomial function included in the numpy library and performed 1000 trials, 10,000 times as before with the same probability of success (1/216).

$$(x+a)^n = \sum_{k=0}^{n} \binom{n}{k} x^k a^{n-k}$$

Results



Appendix

```
import random
import numpy as np
import pylab as plt
from scipy import stats

#Main
p = (1/216)
x= 0
X = []

while(x < 20):
    result = np.random.binomial(1000, p, 10000)
    X.append(result)
    x = x + 1

#xk = ints with nonzero probabilities
#pk = the probabilities which sum to zero

num_bins = 16
counts, bins = np.histogram(X, bins=num_bins)
bins = bins[:-1] + (bins[1] - bins[0])/2
probs = counts/float(counts.sum())

xk = np.arange(len(probs))
pk = probs
custm = stats.rv_discrete(name='custm', values=(xk, pk))

fig, ax = plt.subplots(1, 1)
ax.plot(xk, custm.pmf(xk), 'ro', ms=8, mec='r')
ax.vlines(xk, 0, custm.pmf(xk), colors='r', linestyles='-', lw=2)
plt.title('Experiment Results')
plt.ylabel('Probability of Success')
plt.xlabel('Number of successes in n=1000 trials')
plt.show()
```

Problem 3

Introduction

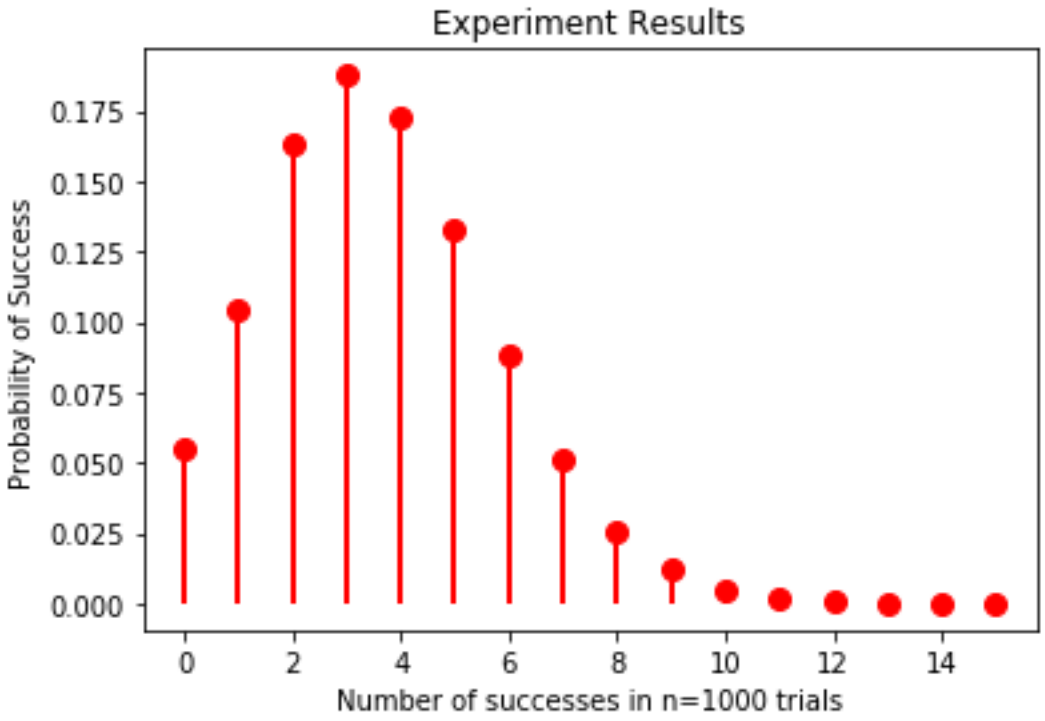
Problem 3 attempts to do the same as in problem 2 by using a formula to calculate the probabilities of success in the simulation while avoiding having to do the simulation itself. The difference here being that we utilize the Poisson distribution formula rather than the Binomial formula.

Methodology

Similar to the binomial formula implementation, I've implemented the Poisson distribution using the numpy library and plotting the results in the PMF. As expected it is a very similar distribution to the previous problem. One difference is that we utilize a new value, to represent the probability (1/216) and the number of trials (1000).

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

Results



Appendix

```
import numpy as np
import pylab as plt
from scipy import stats

#Main
p = (1/216)
numsuccesses = 0
X = []
lamda = p * 1000

while(numsuccesses < 20):
    result = np.random.poisson(lamda,1000)
    X.append(result)
    numsuccesses += 1

#xk = ints with nonzero probabilities
#pk = the probabilities which sum to zero

num_bins = 16
counts, bins = np.histogram(X, bins=num_bins)
bins = bins[:-1] + (bins[1] - bins[0])/2
probs = counts/float(counts.sum())

xk = np.arange(len(probs))
pk = probs
custm = stats.rv_discrete(name='custm', values=(xk, pk))

fig, ax = plt.subplots(1, 1)
ax.plot(xk, custm.pmf(xk), 'ro', ms=8, mec='r')
ax.vlines(xk, 0, custm.pmf(xk), colors='r', linestyles='-', lw=2)
plt.title('Experiment Results')
plt.ylabel('Probability of Success')
plt.xlabel('Number of successes in n=1000 trials')
plt.show()
```