Vinh Vu

Dr. Chassiakos

EE 381-04

March 13th, 2019

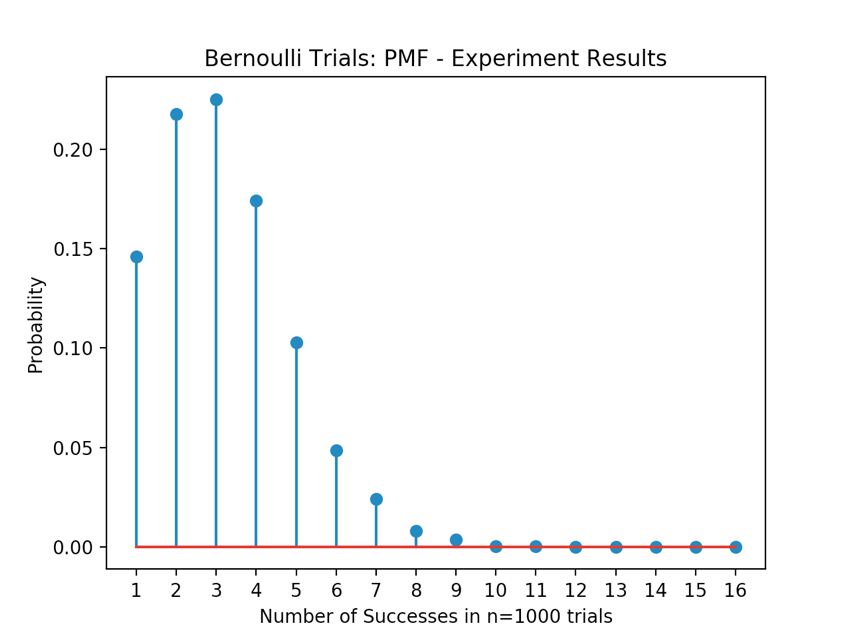
Project 3: Binomial and Poison Distributions

**Problem 1: Experimental Bernoulli Trials**

**Introduction:** This experimental simulates the rolling of three unfair dice n times. Random variable “X” is the number of success in n rolls. In order for one experiment to be considered success, first die lands on a “one,” second die lands on a “two,” and third die lands on a “three”. The three unfair die rolls 1000 times and the number of success in 1000 rolls is the random variable X. The experiment is then repeated 10,000 times and the record values for “X” is used to create a probability mass function plot.

**Methodology:** Since the three dice are unfair, the probability array is given as [0.2, 0.1, 0.15, 0.3, 0.2, 0.05] for the 6-sided die. For this experiment, we are only focusing on the probability for “one”, “two”, and “three.” To simulate the roll of the unfair die, function nSidedDie from project 1 is utilized with probability array p passed into. A nested for-loop is implemented to repeat the experiment 10,000 times with 1000 rolls each. Success time resets for each new experiment with variable assignment to 0. Three variables are used to call nSidedDie: Roll\_1, Roll\_2, and Roll\_3. Success only increments when Roll\_1 equal to 1, Roll\_2 equal to 2, and Roll\_3 equal to 3. Number of success is then saved into an array to be calculated into a probability array. It then being graphed using histogram.

**Results and Conclusion:**

****

**Appendix:**

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

p = [0.2, 0.1, 0.15, 0.3, 0.2, 0.05]

N = 10000

n = 1000

X = []

def nSidedDie(p):

if sum(p) != 1:

print('Probability values are incorrect!')

sides = len(p)

cs = np.cumsum(p)

cp = np.append(0,cs)

r = np.random.rand()

for k in range(0,sides):

if r > cp[k] and r<= cp[k+1]:

sides = k+1

return sides

for i in range(N):

success = 0

for j in range(n):

Roll\_1 = nSidedDie(p)

Roll\_2 = nSidedDie(p)

Roll\_3 = nSidedDie(p)

if (Roll\_1 == 1) and (Roll\_2 == 2) and (Roll\_3 == 3):

success += 1

X.append(success)

b=range(1,18)

sb=len(b)

h1, bin\_edges = np.histogram(X,bins=b)

b1=bin\_edges[0:sb-1]

fig2=plt.figure(2)

p1=h1/N

plt.stem(b1,p1)

plt.title('Bernoulli Trials: PMF - Experiment Results')

plt.xlabel('Number of Successes in n=1000 trials')

plt.ylabel('Probability')

plt.xticks(b1)

plt.show()

**Problem 2: Calculations using the Binomial Distribution**

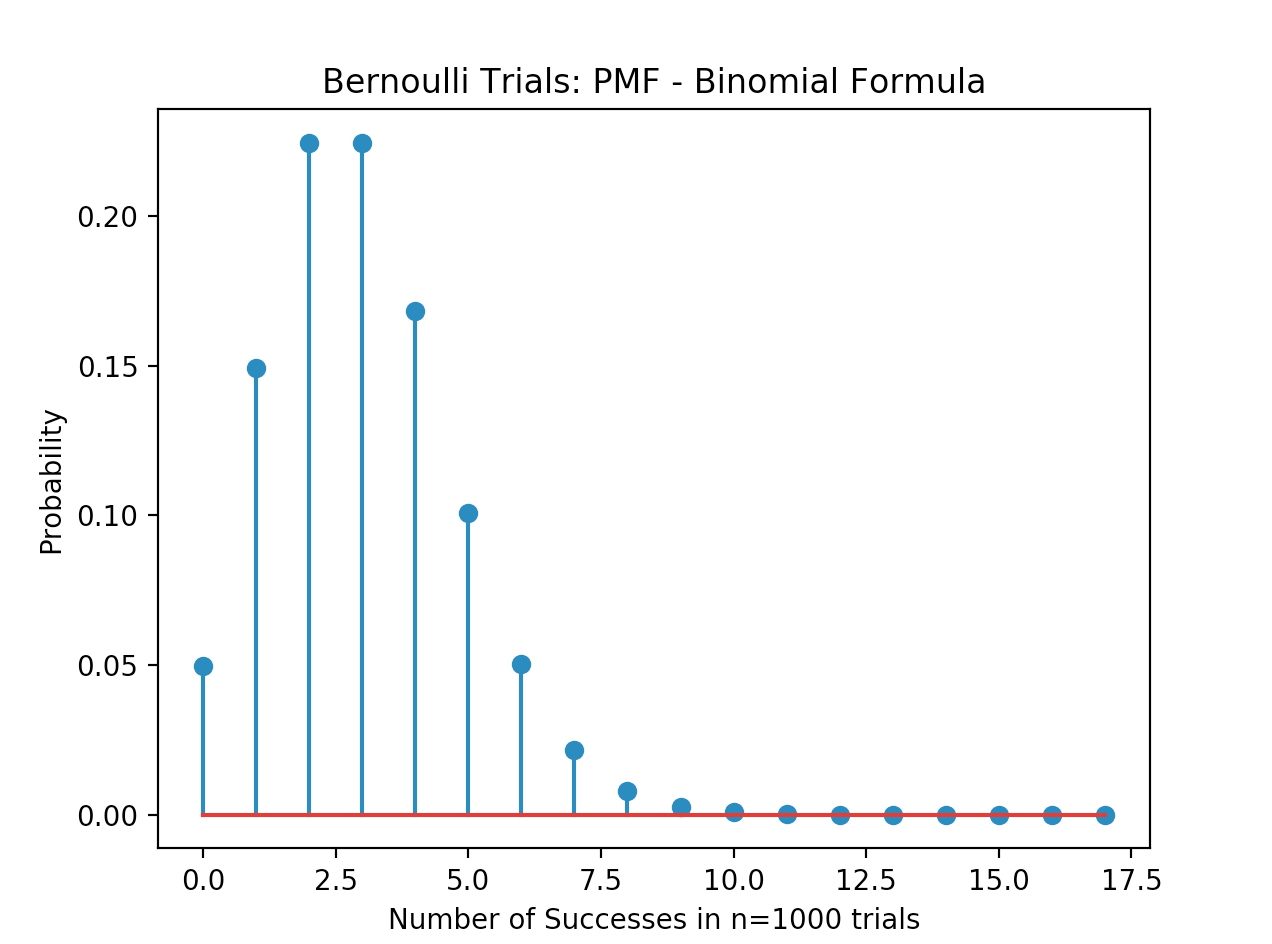
**Introduction:** In this problem, we are using the theoretical Binomial Distribution formula to calculate the probability p of success in a single roll of the three dice. We use the Binomial formula to generate the Probability Mass Function plot of the random variable X, where X is the number of success in n Bernoulli trials. PMF graph is generated from Binomial distribution formula is being compared to PMF generated from the actual experiment.

**Methodology:** Initially, to find the probability of success in a single trial, probabilities of rolling a 1,2 and 3 are multiplied. The result is 0.003, less than 1% for success in a single toss. With that probability, the expected number of success in n trials can be calculated using E(X) = np, where n=1000 and p=0.003. Based on theoretical formula, the number of success is 3 times out of 1000. To generate the PMF from binomial formula, p (X = x ) = pxqn-x, where . n is the total number of trials, in this case 1000, and x represents the number of successes, ranges from 0 to 18 to keep coherency with number 1. Probability is calculated and saved into a success probability array. The array is then graphed and compared to the PMF from number 1.

**Results and Conclusion:** PMF from Binomial distribution formula vs PMF from actual experiment show very similar results. The probability is the highest between 2 and 3 successes and the graph exponentially decreases until close to 0.

P=Prob(success in a single trial)= 0.0030000000000000005

The expected number of success in n trials is 3



**Appendix:**

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

import math

rolls = 1000

success = []

prob = [0.2, 0.1, 0.15, 0.3, 0.2, 0.05]

p = prob[0] \* prob[1] \* prob[2]

q = 1 - p

for m in range(0,18):

n = math.factorial(rolls)

n\_m = math.factorial(rolls-m)

m\_f = math.factorial(m)

X = float((n/(m\_f\*n\_m))\*(p\*\*m)\*(q\*\*(rolls-m)))

success.append(X)

print("P=Prob(success in a single trial)=",p)

print("The expected number of success in n trials is", int(1000\*p))

plt.stem(success)

plt.title('Bernoulli Trials: PMF - Binomial Formula')

plt.xlabel('Number of Successes in n=1000 trials')

plt.ylabel('Probability')

plt.xticks()

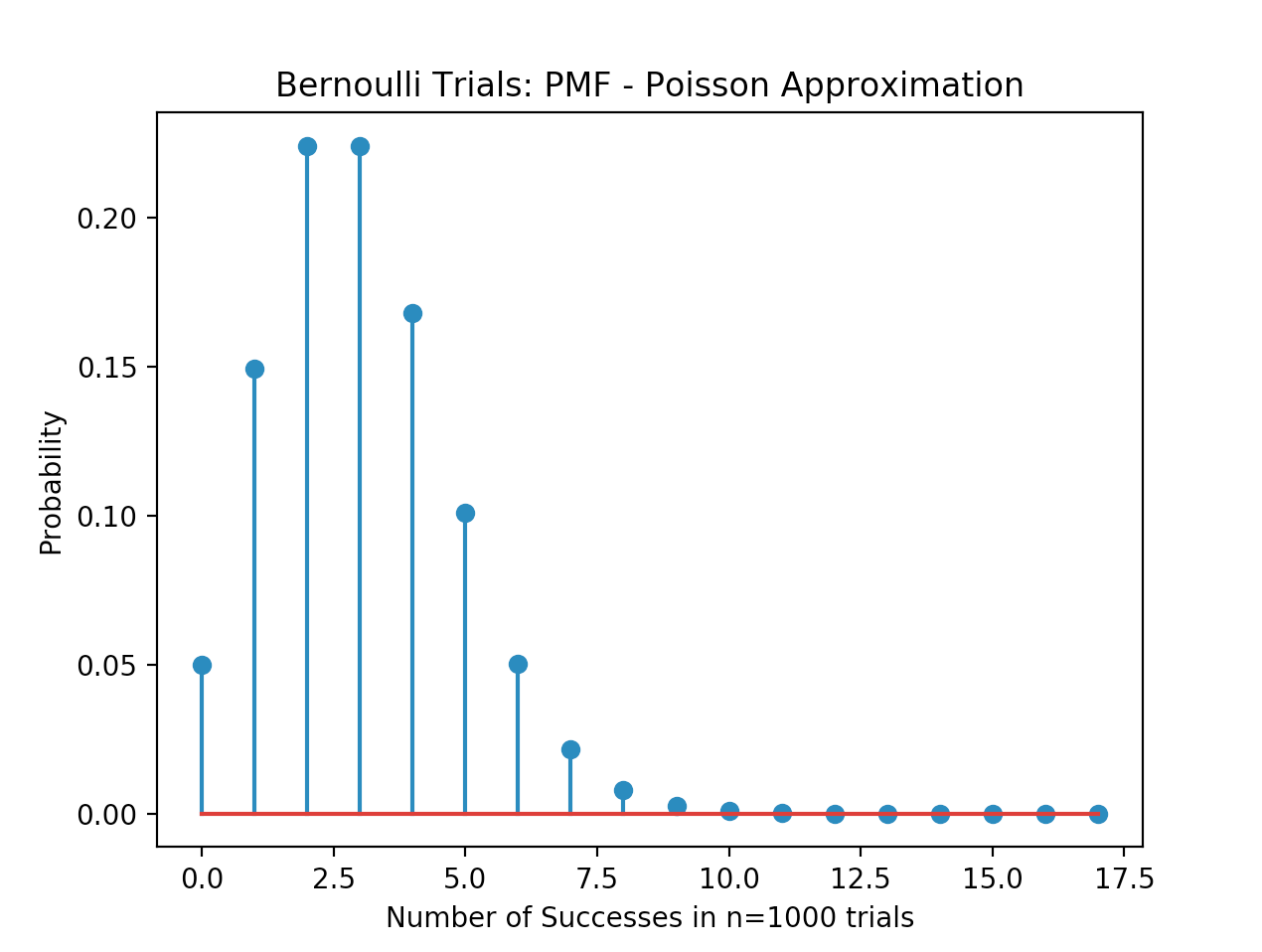
plt.show()

**Problem 3: Approximation of Binomial by Poisson Distribution**

**Introduction:** Poisson Distribution approximation is used when the probability of success is small and the number of trials is large, like our case of n = 1000 and p = 0.003. This method is used to approximate the probability of success in n trials, as an alternative to Binomial formula. Lamda, λ a parameter in the formula, is obtained from λ = np. The problem is asked to use the Poisson distribution formula to create the plot of a PMF plot of the random variable X, where X X is the number of success in n Bernoulli trials. PMF generated from Poisson distribution formula is being compared to PMF generated from the actual experiment.

**Methodology:** Poisson distribution approximation formula is given as p(X = x) = where λ = np. x value, representing the number of successes, ranges from 0 to 18 to keep coherency with number for easier comparison. Probability is calculated and saved into a success probability array. The array is then graphed and compared to the PMF from number 1.

**Results and Conclusion:** PMF from Poisson distribution approximation vs PMF from actual experiment show very similar results. The probability is the highest between 2 and 3 successes and the graph exponentially decreases until close to 0.

****

**Appendix:**

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

import math

rolls = 1000

success = []

prob = [0.2, 0.1, 0.15, 0.3, 0.2, 0.05]

p = prob[0] \* prob[1] \* prob[2]

lamda = rolls \* p

for m in range(0,18):

X = ((lamda\*\*m)\*(math.exp(-1\*lamda)))/(math.factorial(m))

success.append(X)

plt.stem(success)

plt.title('Bernoulli Trials: PMF - Poisson Approximation')

plt.xlabel('Number of Successes in n=1000 trials')

plt.ylabel('Probability')

plt.xticks()

plt.show()