EE 381 Project 3

William Thinh Luu
California State University of Long Beach
EE 381 Probability and Statistics, with Application on Computing
Behruz Revani, Ph.D.
11/3/2024

Part 1

Introduction

Given a probability vector for 3 identical 5-sided unfair dice to be p=[0.1, 0.2, 0.15, 0.25, 0.3], one experiment is when you roll the three dice n=1000 times. The experiment, also known as a Bernoulli Trial, is considered a success if and only if you roll “three ones” in a single roll. The goal is to find the number of successes in n rolls, which will be the random variable “X”. To generate a histogram for the probability distribution, repeat the experiment N = 10,000 times and record the number of successes in n rolls.

Methodology

First step was importing the random and matplotlib libraries to support randomness and plotting in python. I implemented the dice() function which rolls an unfair 5-sided die 3 times, and returns true if the three dice roll “three ones”. Then I implemented a function called bernoulliTrial() that counts the number of successes when dice() is called 1000 times and returns the number of successes. The experiment(N) function is created to run the bernoulliTrail() 10,000 times, keeping track of the results into an array. The function later determines the probability and creates a histogram to display the results using the matplotlib library.

Results

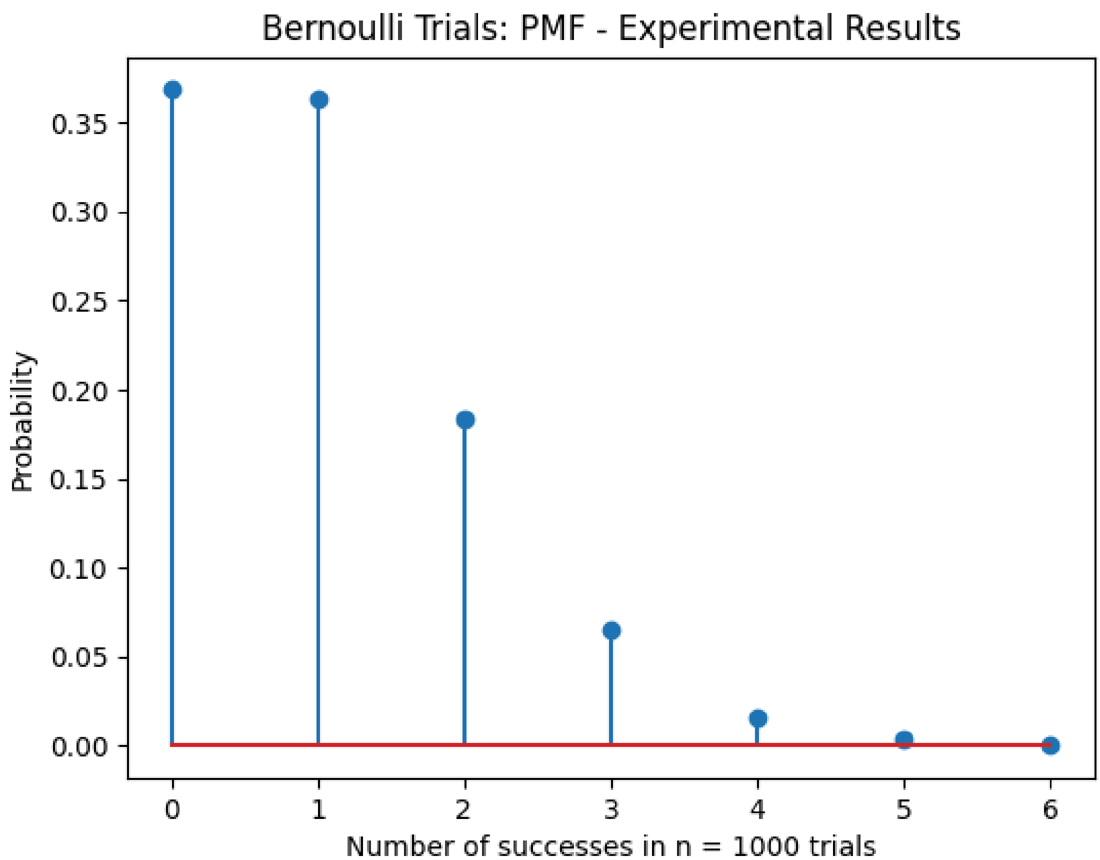

Figure 1: PMF of experimental results from Bernoulli Trials

Conclusion

The PMF results indicate that the number of successes in 1000 trials is low, aligning with the theoretical success rate of 0.001 (calculated as 0.1^3), which represents the probability of rolling three consecutive "1"s with the given biased 5-sided die. This low success rate suggests that obtaining even a single success in 1000 rolls is rare, as reflected by the high probability of zero or very few successes observed in the PMF. The skewed distribution toward fewer successes can be due to the unfair nature of the 5-sided die.

Appendix 1

```python
import random
import matplotlib.pyplot as plt #enables plotting

def dice():
    p = [0.1, 0.2, 0.15, 0.25, 0.3]
    dice =  list(range(1, 6)) #5-sided die
    roll3 = random.choices(dice, weights=p, k=3)

    return roll3[0] == 1 and roll3[1] == 1 and roll3[2] == 1

def bernoulliTrial():
    X = 0
    for i in range(1000):
        if dice() == True:
            X+=1

    return X #num of successes in 1000 rolls

def experiment(N):
    results = []
    for i in range(N):
        results.append(bernoulliTrial())

    # Generate histogram to approximate PMF
    max_X = max(results)
    frequency = [results.count(i) for i in range(max_X + 1)]
    probability = [f / N for f in frequency]  # Normalize to get probabilities

    # Create stem plot
    plt.stem(range(max_X + 1), probability)
    plt.xlabel("Number of successes in n = 1000 trials")
    plt.ylabel("Probability")
    plt.title("Bernoulli Trials: PMF - Experimental Results")
    plt.show()


if __name__ == "__main__":
    N = 10000
    experiment(N)
```

Part 2

Introduction

With the same experiment/Bernoulli trials in the previous part, we are to calculate the probability p of successes in a single roll of the three dice using the theoretical formula for the Binomial distribution. The Binomial formula is provided via Part 0 of the project instructions. Compare the difference between the PMF from part 1 with the PMF from part 2.

Methodology

By utilizing python's math, random, and matplotlib libraries, I rescued the same function for dice() and bernoulliTrial() from part 1 since part 2 is referring to the same experiment in part 1. The new function binomialDist(n, p, x) is created, which is the code for the Binomial formula using python's math library. It returns the probability of successes in a single roll of three dice. The experiment function is adjusted to call and plot the results of the binomial PMF using matplotlib library.

Results

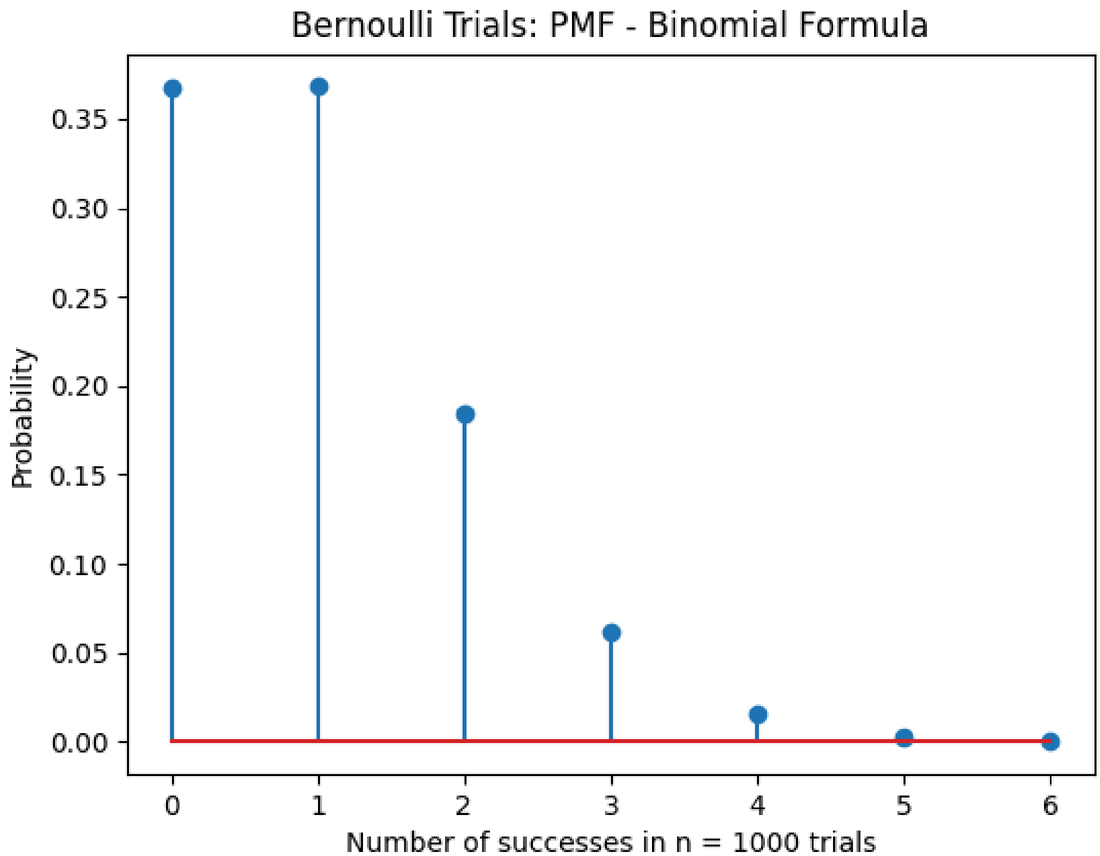

Figure 2: PMF of Binomial Distribution from Bernoulli Trials

Conclusion

Overall, the PMF of the Binomial formula and the Experimental results are about the same, with 0 and 1 being the highest. The leftward skewing of the graph is most likely due to the fact that both PMF are using the same Bernoulli trial with biased dice. This alignment between the theoretical and experimental PMFs confirms that the Binomial model accurately predicts the distribution of successes in this scenario.

Appendix 2

```python
import random
import math
import matplotlib.pyplot as plt  # Enables plotting

# Function to simulate rolling three dice and checking for three ones
def dice():
    p = [0.1, 0.2, 0.15, 0.25, 0.3]
    dice_faces = list(range(1, 6))  # 5-sided die
    roll3 = random.choices(dice_faces, weights=p, k=3)

    return roll3[0] == 1 and roll3[1] == 1 and roll3[2] == 1

# Function to perform a Bernoulli trial
def bernoulliTrial():
    X = 0
    for i in range(1000):
        if dice():
            X += 1
    return X  # Number of successes in 1000 rolls

# Function to calculate the binomial distribution PMF
def binomialDist(n, p, x):
    binom_coeff = math.comb(n, x)  # n choose x
    pdf = binom_coeff * (p ** x) * ((1 - p) ** (n - x))
    return pdf

# Function to perform the experiment and plot PMF
def experiment(N):
    results = []
    for i in range(N):
        results.append(bernoulliTrial())

    # Calculate theoretical PMF using the poisson distribution
    p = [0.1, 0.2, 0.15, 0.25, 0.3]

    # Probability of rolling three ones
    p_success = (p[0]) ** 3

    # Maximum number of successes
    max_X = max(results)

    binomial_pmf = [binomialDist(1000, p_success, x) for x in range(max_X + 1)]

    # Plot theoretical PMF
    plt.stem(range(max_X + 1), binomial_pmf)

    # Plot settings
    plt.xlabel("Number of successes in n = 1000 trials")
    plt.ylabel("Probability")
    plt.title("Bernoulli Trials: PMF - Binomial Formula")
    plt.show()

if __name__ == "__main__":
    N = 10000
    experiment(N)
```

Part 3

Introduction

Similarly to part 2, using the same experiment/Bernoulli trial from part 1, we are to calculate the probability p of successes in a single roll of the three dice using the theoretical formula of the Poisson distribution instead. The Poisson formula is provided via Part 0 of the project instructions. Finally, compare the difference between the PMF from part 1 and part 2 with the PMF from part 3.

Methodology

Since we are still using the experiment from part 1, functions dice() and bernoulliTrial() are to be reused. The function poissonDist(n, p, x) takes in the number of n trials, probability p of success, and x which is the number of occurrences during a unit time interval. Using the python library math, lambda is calculated and the poisson distribution function is implemented. Similarly to part 2, the experiment is run 10000 times and plotted onto a histogram using matplotlib library.

Results

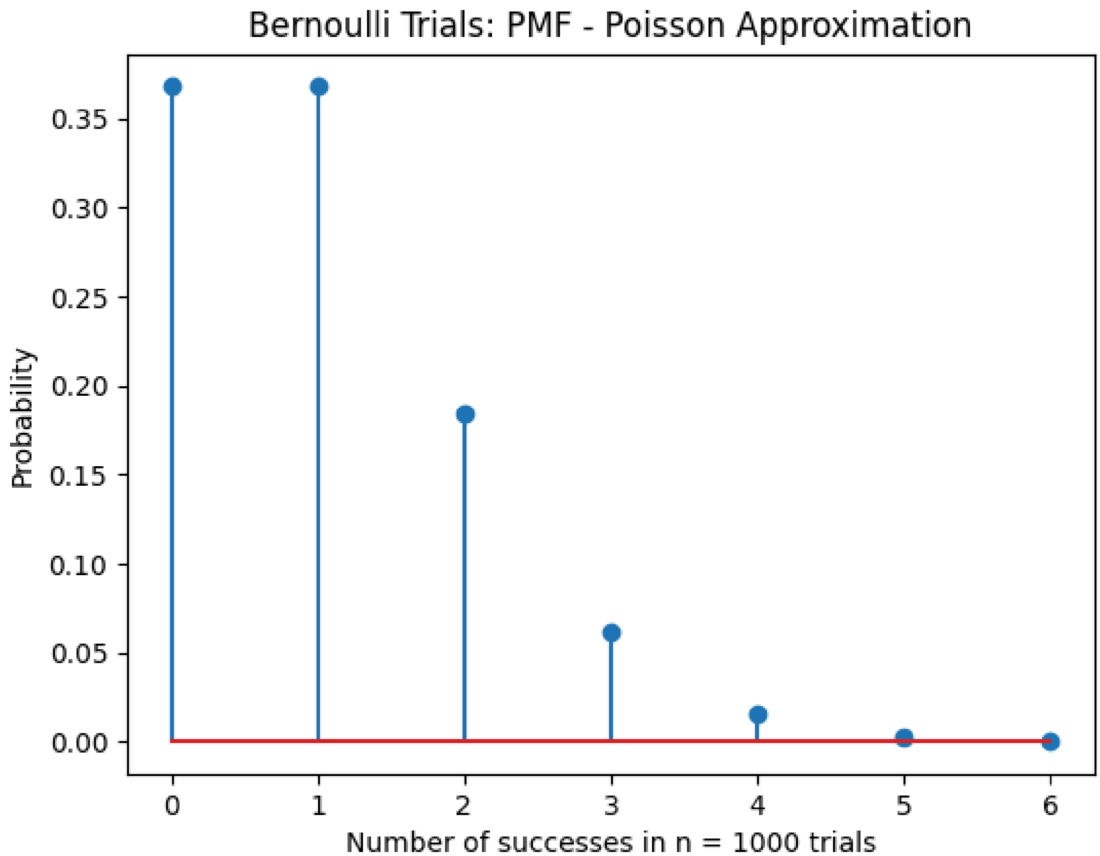

Figure 3: PMF of Poisson Distribution from Bernoulli Trials

Conclusion

The comparison between the PMFs from parts 1, 2, and 3 shows that the Poisson distribution provides a reasonable approximation to the experimental PMF, especially for events with low probability over a large number of trials. In this case, using the Poisson distribution aligns well with the nature of the experiment, as the event of rolling three ones is rare given the biased probabilities of each face on the 5-sided die. The results indicate that both the experimental and theoretical Poisson PMFs are skewed to the left, with the probability concentrated at low numbers of successes, which is consistent with the low likelihood of achieving three ones in any single roll.

Appendix 3

```python
import random
import math
import matplotlib.pyplot as plt  # Enables plotting


# Function to simulate rolling three dice and checking for three ones
def dice():
    p = [0.1, 0.2, 0.15, 0.25, 0.3]
    dice_faces = list(range(1, 6))  # 5-sided die
    roll3 = random.choices(dice_faces, weights=p, k=3)

    return roll3[0] == 1 and roll3[1] == 1 and roll3[2] == 1

# Function to perform a Bernoulli trial
def bernoulliTrial():
    X = 0
    for i in range(1000):
        if dice():
            X += 1
    return X

def poissonDist(n, p, x):
    λ = n*p
    pdf = ((math.e**(-λ)) * (λ**x)) / math.factorial(x)
    return pdf

def experiment(N):
    results = []
    for i in range(N):
        results.append(bernoulliTrial())

    # Calculate theoretical PMF using the poisson distribution
    p = [0.1, 0.2, 0.15, 0.25, 0.3]

    # Probability of rolling three ones
    p_success = (p[0]) ** 3

    # Maximum number of successes
    max_X = max(results)

    poisson_pdf = [poissonDist(1000, p_success, x) for x in range(max_X + 1)]

    # Plot theoretical PMF
    plt.stem(range(max_X + 1), poisson_pdf)

    # Plot settings
    plt.xlabel("Number of successes in n = 1000 trials")
    plt.ylabel("Probability")
    plt.title("Bernoulli Trials: PMF - Poisson Approximation")
    plt.show()

if __name__ == "__main__":
    N = 10000
    experiment(N)
```