Project 1 - Stochastic Experiments

California State University Long Beach

EE 381

Probability and Statistics

Christopher Masferrer

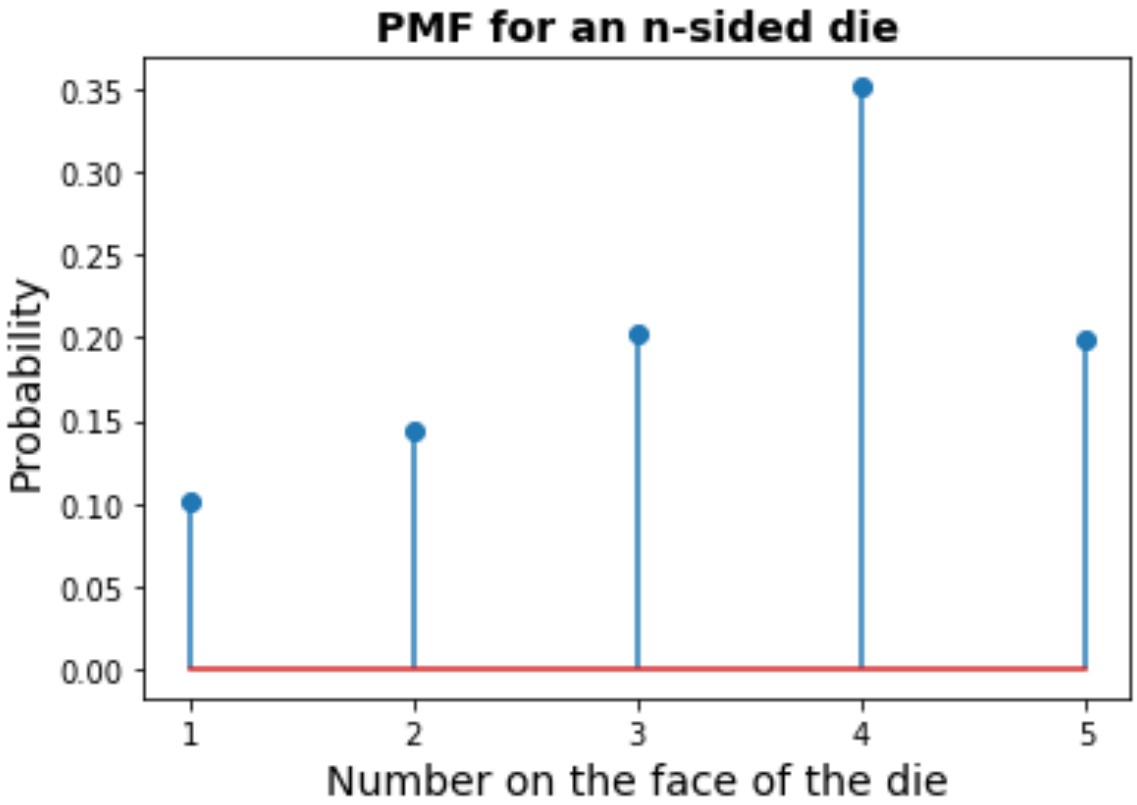
T/TH 5:30 - 6:20

Experiment 1: n-sided die

Intro: For this experiment I will be rolling a n sided die. This experiment will be repeated 10,000 times and the results on the chances of each values from the die will be displayed on a PMF chart.

Methodology: I created a function that rolls the dice N times, with N being 10,000. In addition, I created an additional function that allows the graph to store the rolls and display the information on the PMF chart.

Results:



Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  3 15:12:05 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 1
import numpy as np
import matplotlib.pyplot as plt

def nSidedDie(p):
    N = 10000
    n = np.size(p)
    cs = np.cumsum(p)
    cp = np.append(0,cs)

    for j in range(0, N):
        r = np.random.rand()
        for k in range (0, n):
            if r > cp[k] and r <= cp[k + 1]:
                d = k+1
        return d

def graph(p):
    N = 10000
    s = np.zeros((N,1))
    #
    for i in range(0,N):
        r=nSidedDie(p)
        s[i]=r

    #Plotting
    b= range(1,np.size(p)+2)
    sb = np.size(b)
    h1, bin_edges = np.histogram(s, bins = b)
    b1 = bin_edges[0 : sb - 1]
    plt.close('all')
    prob = h1/N
    #Plots and labels
    plt.stem(b1, prob, use_line_collection= True)
    plt.title('PMF for an n-sided die', fontsize = 14, fontweight = 'bold')
```

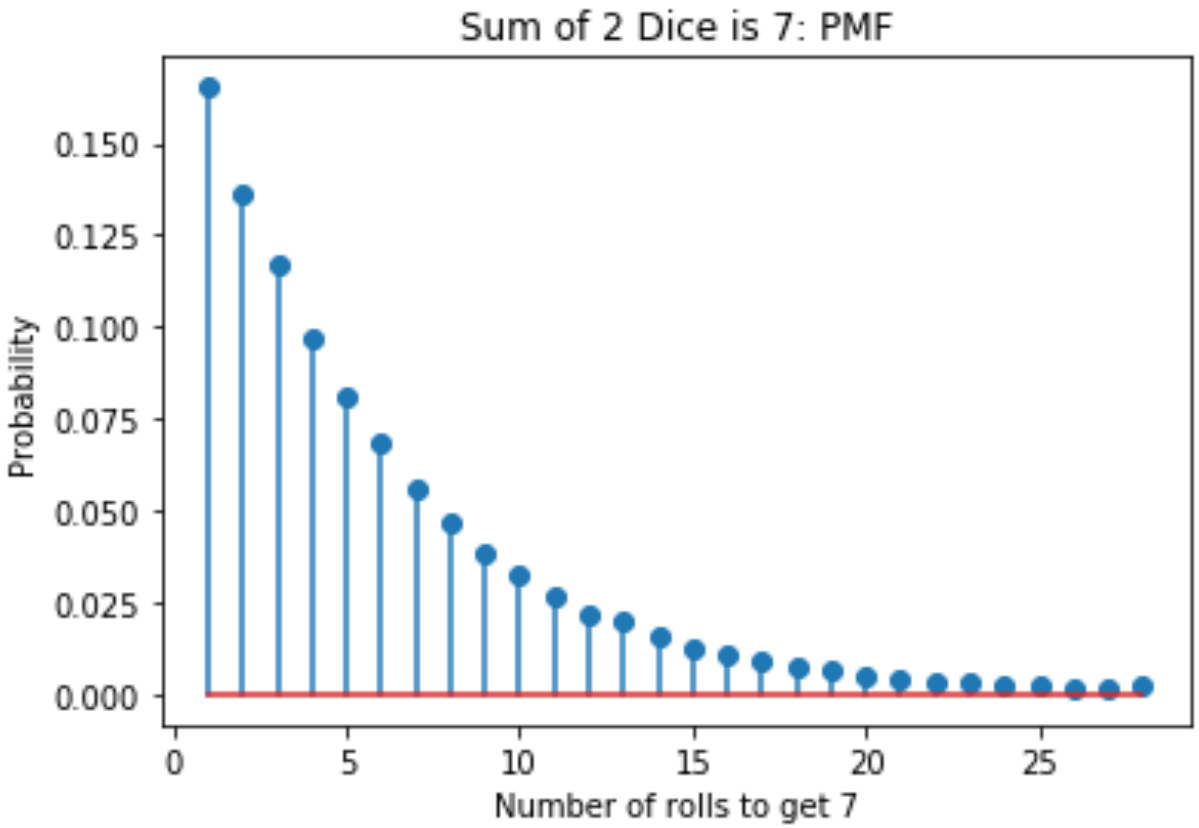
```
    plt.xlabel('Number on the face of the die', fontsize = 14)
    plt.ylabel('Probability', fontsize = 14)
    plt.xticks(b1)
    plt.show()

p = [0.10, 0.15, 0.20, 0.35, 0.20]
graph(p)
```

Experiment 2: Rolling for a 7 using 2 dice

Intro: For this experiment, I will be rolling two fair dice and see how many times it takes until I get a seven. I will be performing this 100,000 times and displaying the result on a stem plot.

Methodology: I created a method that performs all calculations and displays the chart. I used a counter to indicate how long it took before that run resulted in a seven and then recorded it on an array that was used for the stem plot.

Results:



Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  3 15:12:05 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 1
import numpy as np
import matplotlib.pyplot as plt


def sum2dice(N):
    count = [None] * N
    for i in range(0, N):
        complete = 1
        counter = 0
        while complete:
            dice1 = np.random.randint(1, 7)  # represents 6 sided dice
            dice2 = np.random.randint(1, 7)
            sum = dice1 + dice2  # summation of 2 dice
            counter += 1
            if sum == 7:  # if sum is 7 success
                complete = 0  # end while
                count[i] = counter

    b = range(1, 30)
    sb = np.size(b)
    h1, bin_edges = np.histogram(count, bins=b)
    b1 = bin_edges[0:sb - 1]
    plt.close('all')

    fig1 = plt.figure(1)
    p1 = h1 / N
    plt.stem(b1, p1)
    plt.title('Sum of 2 Dice is 7: PMF')
    plt.xlabel('Number of rolls to get 7')
    plt.ylabel('Probability')


print(sum2dice(100000))
```

Experiment 3: Coin Toss

Intro: For this experiment, I will be tossing 100 coins to see if I can get at least 50 heads. I will repeat the experiment 100,000 times to see how many times the experiment was performed successfully.

Methodology: I used a single function to perform the experiment with the number of times it performs being inputed as N. A for loop runs the experiment N times. In the for loop the coin toss is performed and then verified if the experiment was successful. Then the probability is calculated and returned

Results:

Probability of 50 heads in tossing 100 fair coins	
Ans.	**p =** 0.08029

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Sep  3 15:12:05 2019

@author: christophermasferrer
"""

#Christopher Masferrer
#EE 381
#Lab 1

import numpy as np

def MultiCoinToss(N):
    accum = 0
    for i in range(0, N):
        coin = np.random.randint(0, 2, 100)
        heads = sum(coin)
        if heads == 50:
            accum += 1
    prob = accum / N
    return prob


print(MultiCoinToss(100000))
```

Experiment 4: Password Hacking

Intro: For this experiment I will be generating a password that I will attempt to crack using a hacker list. This experiment will be performed 1000 times and the probability will be the result.

Methodology: I kept track on the number of successful attempts while a for loop ran the experiment 1000 times. Each run generated a new password and a new hacker list. The probability is then calculated and returned.

Results:

Hacker created m words Prob. that at least one of the words matches the password	p = 0.132
Hacker creates k*m words Prob. that at least one of the words matched the password	p = 0.653
p = 0.5 Approximate number of words in the list	m = 322000

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Sep  5 09:06:58 2019

@author: christophermasferrer
"""
import numpy as np
n = 456976

def password_cracker(N, k, m):
    success = 0
    for i in range(0,N):
        pw = np.random.randint(0,n)
        hacker = np.random.randint(0,n,m * k)
        if pw in hacker:
            success += 1
    prob = success / N
    print('The chance that a hacker has at getting the password is: ' + str(prob))

#1st experiment
password_cracker(1000, 1, 70000)

#2nd experiment
password_cracker(1000, 7, 70000)

#3rd experiment
password_cracker(1000, 1, 322000)
```