Stanley Ung

011743626

EE 381

Project 1 - Stochastic Experiments

**Problem 1**

**Intro**

This experiment involves rolling to fair die and observing how many rolls it takes to get a 7.

**Methodology**

In order to find the number of rolls it took to get a 7. I would roll the dice and record increment a count by 1 to keep track of the number of rolls until we rolled a 7. When I rolled a 7, we take the count and add it to an empty list that keeps a record of the number of rolls it takes for a successful experiment. These numbers would then be graphed by the number of rolls it takes by number of times the experiment took that many times. This experiment was ran 100000 times.

**Code**

import numpy as np

import matplotlib.pyplot as plt

# Number of experiments

N = 100000

def roll(N):

num = []

for i in range(0, N + 1):

sum = 0

count = 0

while(sum != 7):

sum = np.random.randint(1, 7) + np.random.randint(1, 7)

count = count + 1

num.append(count)

b = range(1, 30) # range of numbers

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

b1 = bin\_edges[0 : 28]

# close('all')

fig1 = plt.figure(1) # declare figure

plt.stem(b1,h1) # type of plot, stem plot

plt.title('Stem plot - Sum of two dice') # title of graph

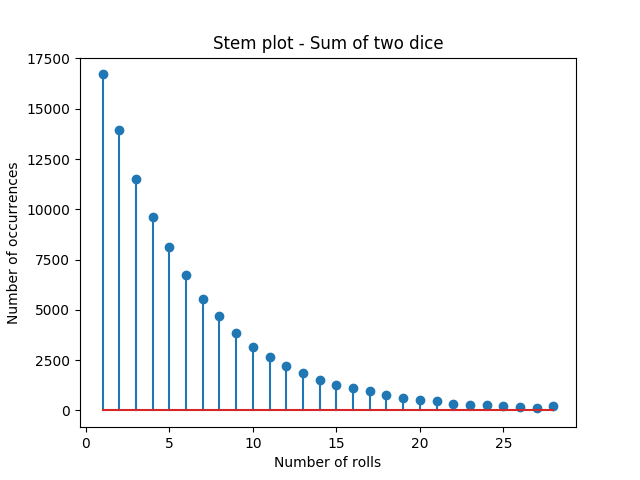
plt.xlabel('Sum of two dice') # x axis label

plt.ylabel('Number of occurrences') # y axis label

fig1.savefig('2 EE381 Proj Stoch Exper-1.png') # name of file saved

**Results**

Since 7 was the highest probable number to get out of a roll of two fair dice, it made sense that most of the time it only took one roll to get a 7 but taking 25 rolls did not occur very much.

****

**Problem 2**

**Intro**

In this problem, we are tossing a fair coin 100 times and observing if we can get exactly 50 heads. The single experiment is a success when we get exactly 50 heads.

**Methodology**

To determine if I had tossed and landed exactly 50 heads, I created a randomly generated list of length 100 and it would generate a list of 1’s and 0’s represent heads and tails respectively. I would sum the list after it is generated and if the sum was 50, then that means the experiment was success. This experiment was ran 100000 times.

**Conclusion**

**Code**

import numpy as np

# A single experiment: 100 tosses a coin

def toss():

# returns 0 or 1

# 0 for tails, 1 for heads

return np.random.randint(0, 2, 100)

# # of times we want to repeat the experiment

N = 100000

# # of successful experiments

success = 0.0

# Perform the experiment N times

for i in range(0, N + 1):

if (sum(toss()) == 50):

success = success + 1

# The probability is # of successful experiments / Total # of experiments

p = success / N

print('The probability of getting exactly 50 heads in 100 tosses of a coin is ' + str(p))

**Results**

After running 100000 trials, the probability of getting EXACTLY 50 heads in 100 tosses of a coin is roughly 0.08 or 8%.

|  |  |
| --- | --- |
| Probability of 50 heads in tossing 100 fair coins |  |
| **Ans.** | ***p =*** 0.07982 |

**Problem 3**

**Intro**

In this problem we want to find the probability of drawing a 4 of a kind from a standard 52 card deck.

**Methodology**

To create a deck, I creates a random permutation of numbers 1 through 52 and then shuffled them. This deck would be a list in python and I would take the first 5 elements as a hand to draw. I would then sort the hand and compare the first and fourth card and then compare the second and last card. Since the hands were sorted, doing these comparisons would tell me if I had drawn a 4 of a kind. This experiment was ran 100000 times.

**Code**

import numpy as np

N = 100000

success = 0

# deal a sorted hand of 5 cards

def deal():

# Create a deck from 1 - 52

deck = np.random.permutation(52) % 13

# deal a sorted hand

hand = np.sort(deck[:5])

# if the 1st and 4th card match OR if the 2nd and 5th card match, it is a 4 of a kind

# since the hand is in sorted order

if((hand[0] == hand[3]) or (hand[1] == hand[4])):

print(hand)

return ((hand[0] == hand[3]) or (hand[1] == hand[4]))

for i in range(0, N + 1):

if (deal()):

success = success + 1

p = success / N

print('The probability of drawing a 4 of a kind is ' + str(p))

**Results**

After running the experiment 100000 times, the probability of drawing a 4 of a kind is roughly 0.0002.

|  |  |
| --- | --- |
| Probability of 4-of-a-kind |  |
| **Ans.** | ***p =*** 0.00024 |

**Problem 4**

**Intro**

In this problem, we want to find the probability that a hacker can find our password through a randomly generated list of 4 letter passwords, assuming we also had a 4 letter password. The hacker would generate a list of 105 and 106 number of passwords that may or may not include duplicates.

**Methodology**

I first started off by creating a list of random numbers from 0 to 105. This random list contains 264 numbers. This list may contain duplicate numbers. I then generate a single number in the same interval for the user password. I compare everything in the list to the user password to see if the experiment was a success. I repeat these steps for 106 hacker list size as well. In order to achieve a 50% probability that he hacker catches your password, it was simply trial and error.

**Code**

import numpy as np

import string

possiblePasswords = 26\*\*4

# The number way

def generatePassword():

return np.random.randint(0, possiblePasswords)

def generateList(x):

return np.random.randint(0, possiblePasswords, x)

# # of trials

N = 1000

yourPassword = generatePassword()

success = 0

hackerListSize = 10\*\*6

for i in range(0, N):

hackerlist = generateList(hackerListSize)

if (yourPassword in hackerlist):

success = success + 1

p = success / N

print('The probability of a hacker having your password is ' + str(p))

**Results**

With 105 list size, the hacker had a roughly 0.2 or 20% probability of guessing your password correctly with the list. Increasing the list tenfold to 106 the hacker increased up to a roughly 0.9 or 90% chance of getting the user password.

|  |  |
| --- | --- |
| *m* = 105  Prob. that at least one of the words matches the password | *p =* 0.198 |
| *m* = 106  Prob. that at least one of the words matches the password | *p =* 0.883 |
| *p =* 0.5  Approximate number of words in the list | *m =* 317000 |