hw1_S2023_pt2

January 26, 2023

1 Homework 1 Part 2

This is an individual assignment.

1.1 Description

Create or edit this Jupyter Notebook to answer the questions below. Use simulations to answer these questions. An analytical solution can be useful to check if your simulation is correct but analytical solutions alone will not be accepted as a solution to a problem.

Problem 9 Consider repeatedly rolling a fair 4-sided die.

1. Create a simulation to compute the probability that the top face will be 4 at least once on four rolls of the die?
2. Create a simulation to compute the probability that the top face will be 4 at least once on 20 rolls of the die?
3. Create a simulation to compute how many rolls of the die would you have to do to be 90% confident that you would see at least one 4?
4. Using the formula you have computed in problem 2 part 4, make a Python function that takes in the target value p and outputs the required number of rolls of an integer.
   1. Find the values for $p = 0.95$ and $p = 0.99$.
   2. Use your simulation to verify that the number of rolls you specified is sufficient to achieve $p \geq 0.95$.

1.The probability that the top face will be 4 at least once on four rolls of the die is 0.683

[1]:
```
import random
import numpy as np
import numpy.random as npr

#Problem 9 #1
def one_four(num_sims, num_rolls):
    #npr is numpy random function 1,5 gives a die with 4 faces
    #size(num_rolls, num_sims) num_rolls indicates how many dice rolls.␣
 ↪num_sims represents the number of simulatons
    dice = npr.randint(1,5, size=(num_rolls, num_sims))

    #axis 0 means for each column we sum all of the rows
    #axis 1 would mean for each row sum all of the columns
    np.sum(dice==4, axis=0)

    #check how many times a 4 was rolled
    event = np.sum(np.sum(dice==4, axis=0) >=1)

    #compute the probability
    event/num_sims
    return event/num_sims

#num_rolls = 4 to sim rolling the dice four times
#num_sims = 100_000 for the number of times to run
num_sims = 100_000
num_rolls = 4
print("Probability of getting at one roll as 4 when rolling a fair 4-sided die␣
 ↪four times = ", one_four(num_sims,num_rolls))
```

```
Probability of getting at one roll as 4 when rolling a fair 4-sided die four
times =  0.68306
```

2.The probability that the top face will be 4 at least once on 20 rolls of the die is 0.997

[2]:
```
#Problem 9 #2
#change the number of sims to twenty and input to the function
num_rolls = 20
print("Probability of getting at one roll as 4 when rolling a fair 4-sided die␣
 ↪20 times = ", one_four(num_sims, num_rolls))
```

```
Probability of getting at one roll as 4 when rolling a fair 4-sided die 20 times
=  0.99665
```

3. The number of rolls needed to guarantee a probability of at least 90% is 9 rolls.

[3]:
```
#Problem 9 #3
num_rolls = 0
Probability = 0.0

while Probability < 0.9:
    #compute the probability for the current number of rolls
    Probability = one_four(num_sims,num_rolls)

    #check if the probability is at least 90% if it is break out of the loop
    if Probability >= 0.9:
        break
    # if the probability is less than 90% continue to increase the rolls
    num_rolls +=1

print(num_rolls)
```

```
9
```

4A. The number of rolls needed to guarantee a probability of at least 95% is 11 rolls. The number of rolls needed to guarantee a probability of at least 99% is 17 rolls.

```
[4]: #4A
     def rolls_req(p):
         #formula found from problem 2
         num_rolls = np.ceil((np.log(1-p))/(np.log(0.75)))
         return num_rolls

     p = 0.95
     print(rolls_req(p))

     p = 0.99
     print(rolls_req(p))
```

```
11.0
17.0
```

Using the rolls_req() function 11 rolls is returned. This number of rolls yields greater than 95% probability.

```
[5]: #4B
     p = 0.95
     print(one_four(num_sims,int(rolls_req(p))))
```

```
0.95763
```

1.2 Problem 10

Create a simulation function where you will roll a fair 6-sided die twice. Use simulation to find out the probability of getting a 4,5, or 6 on the first toss and a 1,2,3 on the second toss.

The probability of getting a 4,5, or 6 on the first toss and a 1,2,3 on the second toss is $1/4 = 0.25$

```
[6]: def six_side_twice(num_sims):
         #npr is numpy random function 1,5 gives a die with 4 faces
         #size(2, num_sims) 2 rows means 2 dice rolls. columns represents the number␣
      ↪of simulatons
         return npr.randint(1,7, size=(2, num_sims))

     #create an list to contain the results of two dice rolls
     dice = six_side_twice(num_sims)

     #find the probabiltiy of getting a 4,5,6 on first toss and a 1,2,3 on the␣
      ↪second toss
     i = 0
     event_counter = 0
     for i in range(num_sims):
         if (dice[0,i] >=4 and dice[1,i] <=3):
             event_counter += 1
```

```
print("The probability of getting a 4,5, or 6 on the first toss and a 1,2,3 on␣
↪the second toss = ",event_counter/num_sims)
```

```
The probability of getting a 4,5, or 6 on the first toss and a 1,2,3 on the
second toss =  0.24872
```

1.3 Problem 11

Suppose that you have a bag with 3 coins. One of them is a fair coin, but the others are biased trick coins. When flipped, the three coins come up heads with probability $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{4}$, respectively.

Consider the experiment where you pick one coin at random and flip it three times. Let H_i be the event that the coin comes up heads on flip i. What is the probability of the outcome $H_1 \cap H_2 \cap \overline{H_3}$?

With small modification in your code, find out the probability of the outcome $H_1 \cap \overline{H_2} \cap \overline{H_3}$.

Use simulation to find out the probability.

The probability of the outcome $H_1 \cap H_2 \cap \overline{H_3}$ is 0.082

```
[7]: def three_flip(num_sims):
         coins = ['fair','2-tails','3-tails']
         head_count = 0
         event_count = 0.0
         for sim in range(num_sims):

             #Choose one of the 3 coins at random
             coin = random.choice(coins)
             if coin == 'fair':
                 # the probability of heads is 1/2
                 S = ['H','T']
             elif coin == '2-tails':
                 #The probability of heads is 1/3
                     S = ['H','T','T']
             else:
                 #The probability of heads is 1/4
                 S = ['H','T','T','T']

             #Flip the chosen coin three times and assign the result to values
             values = random.choices(S, k=3)

             #Check if the desired condition is met
             if (values[0] == 'H' and values[1] == 'H' and values[2] == 'T'):
                     event_count +=1
         return event_count

     print("The probability of the outcome Heads, Heads, Tails =␣
     ↪",three_flip(num_sims)/num_sims)
```

```
The probability of the outcome Heads, Heads, Tails =  0.08187
```

The probability of the outcome $H_1 \cap \overline{H_2} \cap \overline{H_3}$ is 0.138

```
[8]: def three_flip_two(num_sims):
         coins = ['fair','2-tails','3-tails']
         head_count = 0
         event_count = 0.0
         for sim in range(num_sims):

             #Choose one of the 3 coins at random
             coin = random.choice(coins)
             if coin == 'fair':
                 #The probability of heads is 1/2
                 S = ['H','T']
             elif coin == '2-tails':
                 #The probability of heads is 1/3
                     S = ['H','T','T']
             else:
                 #The probability of heads is 1/4
                 S = ['H','T','T','T']

             #Flip the chosen coin three times and assign the result to values
             values = random.choices(S, k=3)

             #check if the desired condition is met
             if (values[0] == 'H' and values[1] == 'T' and values[2] == 'T'):
                     event_count +=1
         return event_count

     print("The probability of the outcome Heads, Tails, Tails = ",␣
      ↪three_flip_two(num_sims)/num_sims)
```

```
The probability of the outcome Heads, Tails, Tails =  0.13733
```

2 Submit Your Solutions

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your PDF file to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.