Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

Module: STU22004
Date: 03/12/2021
Group:

Applied Probability Report
Group Project

Individuals participated in project

Abdelaziz Abushark : 20332134
Ali Al Ani :
Noel Peter :
Charlie Christiansson :

Work distribution

Abdelaziz Abushark : (Q2)
Ali Al Ani : Q1) part b
Noel Peter : Q1) part c
Charlie Christiansson : Q1) parts a&d

Q1) a&d = charlie , b = ali c = noel

Q2)

The process of doing the code for the 2nd Question

I started doing the second question using **Python (Jupyter notes)** as my software as I felt it is the easiest software to use and get the right answer as it has special libraries and functions that will save me some time doing my mathematical equations and double checking the answers.

The Functions and Libraries used to write the code

I used functions and libraries to make my work easier and simpler :

- Function append which was used to add the card to the list instantly
- Library numpy which I used to do the complex mathematical equations and functions as later I used it to verify my answer for example variance = math.var(shuffle) in which does the math equation for variance for me which also done manually but later verified by numpy as the manual equation I used for variance was var =sum(x-avg)**2 for x in shuffleCardList)/len(shuffleCardList) which can be done in a more simple way using numpy which is variance = math.var(shuffleCardList) . In addition I used numpy to verify my expectation equation and answer for me which I wrote in code which was avg = sum(shuffleCardList)/len(shuffleCardList) which was later verified by numpy using that code expectation=math.mean(shuffleCardList) .
- Library tensfor flow which I later used to double check numpy library and my manual code that I wrote in which my code for numpy and manual code were variance = math.var(shuffleCardList) , var =sum(x-avg)**2 for x in shuffleCardList)/len(shuffleCardList) which I later verified using tensor flow by variance=tf(shuffle) which gave me the same answer as well as I used it for double checking my expectation in which my numpy and manual code were avg = sum(shuffleCardList)/len(shuffleCardList) and expectation=math.mean(shuffleCardList) which was later verified by tensor flow using this code tf.mean(shuffleCardList)
- Function random which was used to generate random floating numbers and to randomize the cards process as we have 100 cards as it was used to randomly shuffle the cards
- Function shuffle which I was pretty surprised to find it on Python which helped me a lot to shuffle the cards as I linked function random with it to make the process random as function shuffle takes a sequence , eg list of items and change the original list then after than I linked the numpy library to it to do the mathematical calculations for me which resulted in math.random.shuffle(cards)
- Library matplotlib.pyplot which I later used to draw me the graphs of the expectation and variance of 10000 repetitions which I will be attaching below

The code and the methods used

- I used one method which is called gameOfCard and main method which is called questionTwo as I Interlinked the two methods to generate the final answer for me

- I started my main method def questionTwo() by writing an array list for the cards which is shuffle and
- I added a for loop for the 10000 repetitions that was asked for the question to do which was for x in range (0,1000)
- I used the function append to add the card list instantly and linked it to the 2nd method which is def gameOfCard() .
- I then wrote the math equation for the variance and the expectation which were avg = sum(shuffleCardList)/len(shuffleCardList) in which avg represents the expectation and then the variance which was var = sum(x-avg)**2 for x in shuffleCradList) / len(shuffleCardList).
- I finally ended my code by printing out the result which is print(“expectation) and print(“var)
- The 2nd method was was gameOfCards() in which I coded the first part of the question in which it asked that the deck has 100 cards so I created a card list that has range from (0,100)
- I then interlinked numpy library and the 2 functions random and shuffle to make the procedure random and simple as I explained what numpy , shuffle and random do in the previous point which is **The Functions and Libraries used to write the code**
- I then proceeded to calculate the number of hits as the question asked us to count the numbers of hits when the cards turned over so it would have the same range as the number of cards.
- I created a for loop to generate the number of hits as we will have a hit whenever a card is turned over and I printed out the different combinations of hits
- I followed it by an If statement to test as if the hit was not in the same range as cards which was (0,100) ro continue and not return anything but if it was in the same range it would return the number of hits which was referred as noHits
- I added the library matplotlib.pyplot to draw me the graphs by using plt.plot so I can visualize the final results and added a title to it by using plt.title
- I then added a random seed which will generate me specific random numbers to check both results from my manual calculations and numpy library
- Finally the code is now done and ready to be used to generate the 10000 simulations.

Pictures of the code used and the output to generate the final answer

Code I wrote that used my manual calculations :

- You can see the calculations clearly and the 2 methods used
- You can see the output of the function which is below and I will post it again below it to confirm it.

In [23]:
```
import numpy as np
import random
random.seed(12345678)

def gameOfCard():
    cards = list(range(1, 101))
    np.random.shuffle(cards)  # will create deck of 100 cards randomly and then shuffle them
    noHits: int = 0
    for x in range(0, 100):
        if cards[x] != x+1:
            continue
        noHits += 1
    return noHits


def questionTwo():
    shuffle = np.array([cardGame()for x in range( 10000)])
    avg = sum(shuffle) / len(shuffle)
    var = sum((x - avg) ** 2 for x in shuffle) / len(shuffle)
    expectation = avg



    print(" | expectation of 10000 repititions| " + str(expectation))
    print(" | variance of  10000 repititions  |  " + str(var))


questionTwo()
```

```
 | expectation of 10000 repititions| 1.0028
 | variance of  10000 repititions  |  1.0103921600000338
```

Sample output :

```
 | expectation of 10000 repititions| 1.0028
 | variance of  10000 repititions  |  1.0103921600000338
```

Code I wrote that using library numpy to confirm my answer :

- You can clearly see the numpy library
- You can clearly notice that the answers are the same
- Compare this output to me previous output to see they are the same

In [22]:
```
import random
random.seed(12345678)
import numpy as np
import matplotlib.pyplot as plt


def cardGame():
    cards = list(range(1, 101))
    random.shuffle(cards)
    noHits = 0
    for x in range(0,100):
        if cards[x] == x+1:
            noHits+=1
    return noHits


def questionTwo():
    cardSet = np.array([cardGame()for x in range( 10000)])
    variance = np.var(cardSet)
    expectation = np.mean(cardSet)
    print(" | expectation of 10000 repititions| " + str(expectation))
    print(" | variance of  10000 repititions  |  " + str( variance))
    plt.plot([cardSet[:i+1].var() for i in range(10000)])
    plt.plot([cardSet[:i+1].mean() for i in range(10000)])

    plt.title("The variance and expectation of 10000 repetitions ")
questionTwo()
```

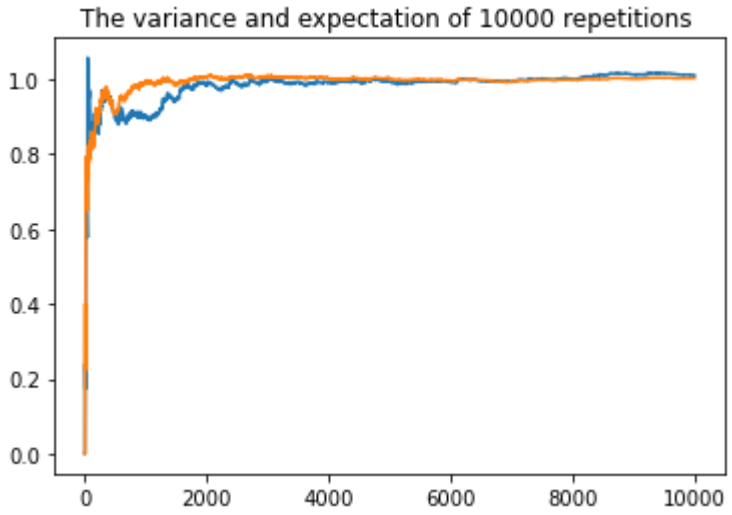
```
 | expectation of 10000 repititions| 1.0028
 | variance of  10000 repititions  |  1.01039216
```

Sample output :

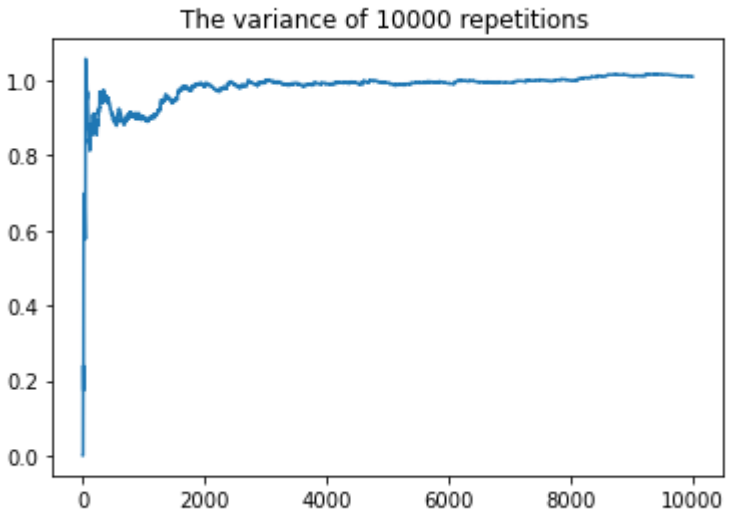
```
| expectation of 10000 repititions| 1.0028
| variance of  10000 repititions  |  1.01039216
```

Graphs that I drew to make me visualize the final results :

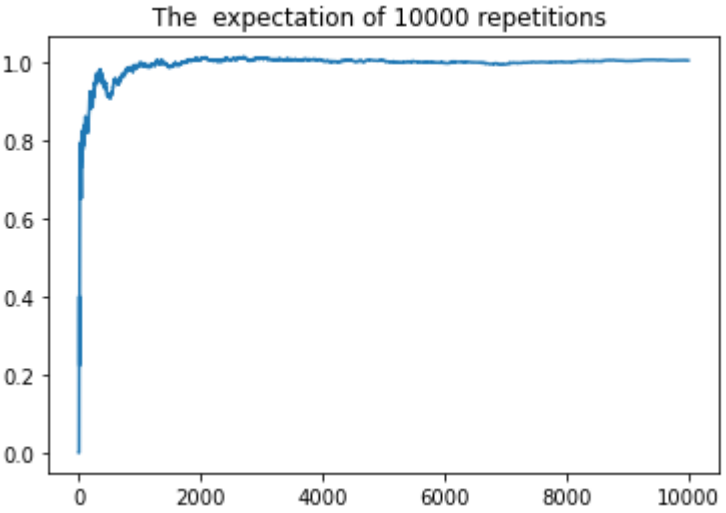
- The variance and expectation of 10000 repetitions



- The variance of 10000 repetitions



- The Expectation of 10000 repetitions



By adding a random seed function I can say that it limits the random numbers to a limitation to 10000 simulations and the final answer estimate is :

- Notice that if I did not add the random.seed it will keep generating different simulations of numbers and will give me different results every time I run my program so random.seed() function is very important that help you generate a limitation of 10000 simulations and get the estimation for the final answer.

- **The expectation of the total number of hits : 1.0028**
- **The variance of the total number of hits : 1.0103921600000338**

For the full code please visit : https://github.com/azizosharke/AppliedPorbability