Rodrigo Becerril Ferreyra
ID 017584071
E E 381 Section 12
Lab 1
Date: 18 September 2020

Introduction

The purpose of this lab is to get familiarized with Python, Anaconda, and modeling probabilities computationally.

1 Problem 1

1.1 Question

The purpose of this Problem is to create a function that takes in a 1-by-n probability vector p (whose elements $p_1, p_2, \dots, p_n$ are chosen such that $\sum_{i=1}^{n} p_i = 1$) and return a random integer in the interval $[1, n]$ according to the probability vector. This function is named `nSidedDie(p)`.

1.2 Results

The function was tested 10 000 times with the probability vector
$p = [0.10, 0.15, 0.20, 0.05, 0.30, 0.10, 0.10]$ in order to check that the function worked correctly. The expected result is a high number of 5s and 3s with a low number of 1s, 6s, and 7s. In short, the ratio of the number of is to the total number of values (10 000) should be approximate to the i-th index in the probability vector. Figure 1 is the image of the graph generated by the set of values. Note that this simulation took about 0.13 s.

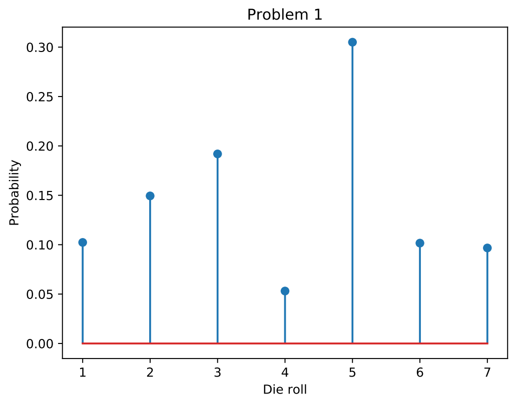


Figure 1: The Probability Mass Function graph.

2 Problem 2

2.1 Question

The purpose of this Problem is to create an experiment in which two fair, six-sided dice are rolled until the values on the top faces add up to 7 (which is defined to be a “success”). The number of rolls that were taken in order to reach a success is then recorded, and the experiment is repeated 100 000 times. The values of the successes are then plotted against the amount of times that they occurred out of 100 000 on a stem plot.

2.2 Results

Figure 2 is the image of the stem plot generated. It is easy to see that the majority of times, a success was reached on the first roll. The next points decline exponentially until hitting about one occurrence at the end. Note that this simulation took about 1.88 s.

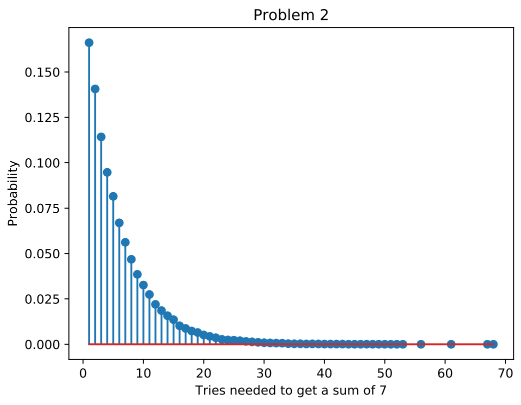


Figure 2: Stem plot of rolls.

3 Problem 3

3.1 Question

The purpose of this Problem is to devise and carry out an experiment that calculates the experimental probability of flipping 100 fair (two-sided) coins and having exactly 50 of them land on heads and 50 of them land on tails.

3.2 Results

To calculate this probability to a certain degree of precision, it was necessary to perform the experiment explained in Section 3.1 100 000 times. Table 1 describes the probability found in doing so. Note that this simulation took about 15 s.

Prob. of 50 heads in tossing 100 fair coins	
Ans.	$p \approx 8.000 \times 10^{-2} \pm 2 \times 10^{-3}$

Table 1: A table containing the probability.

4 Problem 4

4.1 Question

The purpose of this problem is to apply probability to real-life applications, such as security. The scenario is as follows: a hacker makes a list of m random lowercase four-letter words (repetitions allowed) to test on a password system. What is the probability that the hacker gets into the password system (in other words, the probability that the correct password matches one or more of the hacker's attempted passwords)? It is assumed that only four-letter words with lowercase letters are allowed, for simplicity.

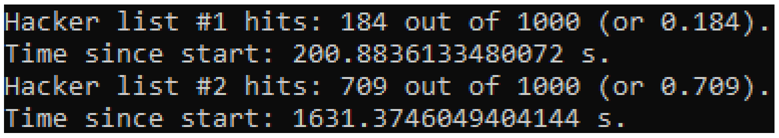
This Problem is split into three parts:

1. Let the hacker create a list of m words.
2. Let the hacker create a list of km words.
3. Find the length of the hacker's list in which the hacker has a 50% chance of hacking into the system.

For this problem, the values of $m = 80\,000$ and $k = 7$ were given.

4.2 Results

To achieve a certain degree of precision in the calculation of the probability, each test was performed 1000 times. Part 1's list was $m = 80\,000$ words long, and Part 2's list was $km = 560\,000$ words long. Figure 3 shows the output of these two experiments. Note that Test 1 took 200.88 s and Test 2 took 1430.49 s.



```
Hacker list #1 hits: 184 out of 1000 (or 0.184).
Time since start: 200.8836133480072 s.
Hacker list #2 hits: 709 out of 1000 (or 0.709).
Time since start: 1631.3746049404144 s.
```

Figure 3: Output for Problem 4 Tests 1 and 2.

The third part of this Problem is to find the value of m such that a list of m words gives the hacker a 50% chance of succeeding. I estimated the value of m in the following way.

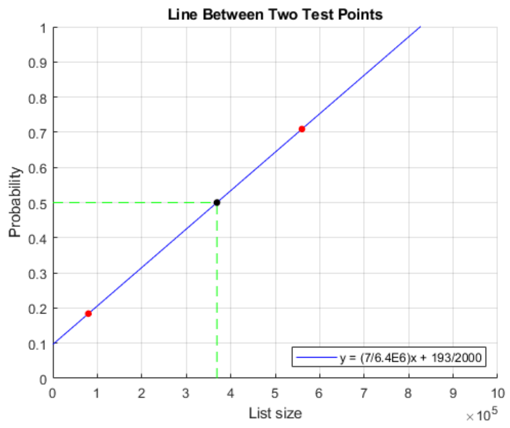


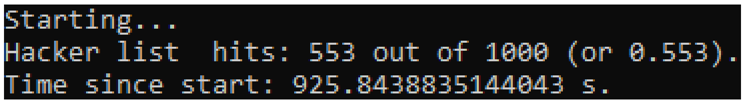
Figure 4: Estimation of m.

First, I plotted the two results that I achieved in Part 1 and Part 2 (the ordered pairs $(80\,000, 0.184)$ and $(560\,000, 0.709)$), and drew a line between them:

$$y = \frac{7}{6.4 \times 10^6}x + \frac{193}{2000}$$

Then, I solved the equation for x given that $y = 0.5$, to see how long the list of passwords needs to be to obtain the probability of 0.5; the answer comes out to be $x = 2\,582\,400/7 \approx 368\,914$. In doing this, I assumed that the probability can be described using a linear function.

Figure 5 is the output of the experiment using a hacker list of 368 914 words. The probability is 0.553, which is close to the goal of 0.500. Note that this simulation took about 925.84 s.



```
Starting...
Hacker list  hits: 553 out of 1000 (or 0.553).
Time since start: 925.8438835144043 s.
```

Figure 5: Output for Problem 4 Test 3

Table 2 is a compilation of all aforementioned results.

Prob. at least 1 word in m-sized list matches the password:	$p = 0.184$
Prob. at least 1 word inkm-sized list matches the password:	$p = 0.709$
Approx. num. of words in list needed to obtain $p = 0.500$	$m = 368\,914$

Table 2: Table of results for Problem 4.