Birthday paradox - Coding

Ali Ghasemi - s289223

Introduction

In probability theory, the birthday problem asks for the probability that, in a set of n randomly chosen people, at least two will share a birthday. The birthday paradox is that, counterintuitively, the probability of a shared birthday exceeds 50% in a group of only 23 people.

In this lab, the whole problem has been implemented both for a Uniform distribution and a distribution which is based on real-life data.

Implementation and Data Structures

Different functions have been defined to achieve the distributions both for uniform distribution and also the real-life scenario.

In the function used for uniform distribution, first, we get a boolean list with a length of 366. This boolean list consists of zeros which will be turned into one in case of birth on a specific day of the year. The births are generated using functions provided by Numpy. The same goes for the real-life scenario but instead of random numbers, birthdays are achieved using real-life data.

The trials (experiments) are executed 1000 times and each time, functions that are mentioned before (to create the distributions) are called. Those functions return the number of people that have to enter the class before a conflict happens (by conflict, we mean two people having the same birthday).

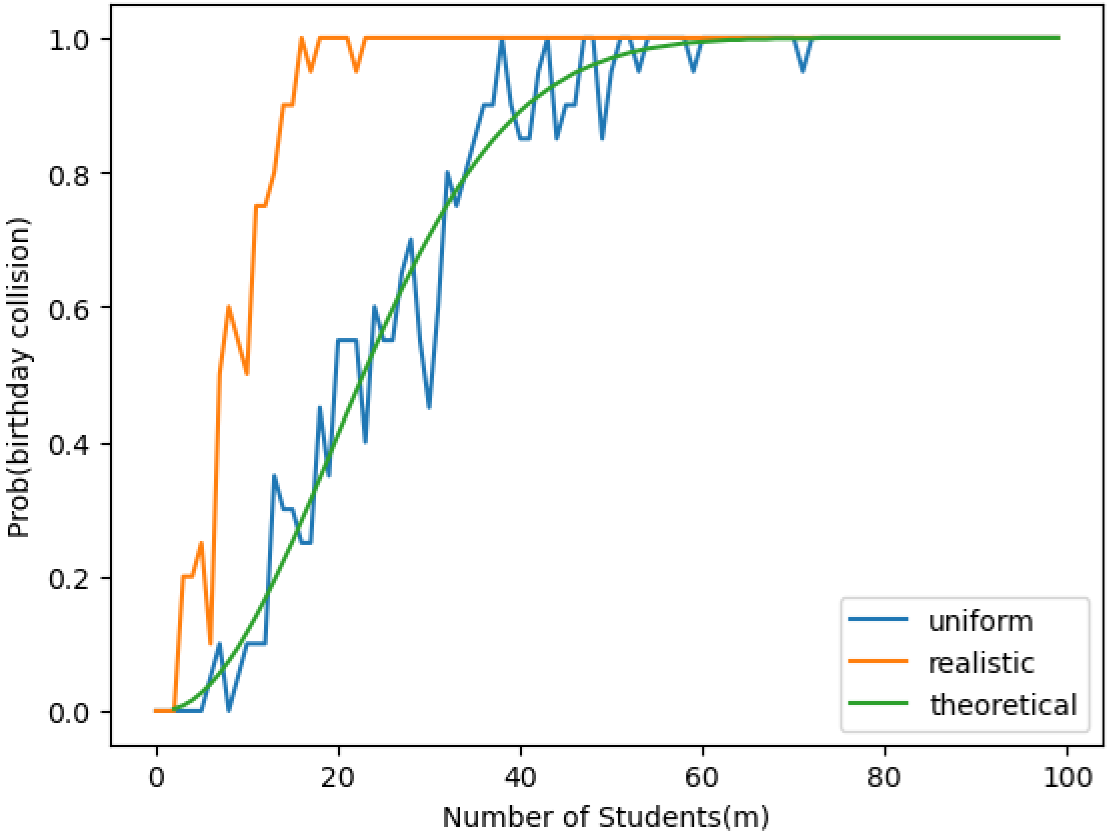
The outputs of these two function are stored in two different lists and after the end of the trials, the average of these outputs are calculated (answer to **task 1**)

The average number of people it takes until a conflict happens is roughly equal to 23.3 in theory and the empirical outputs that we got for the uniform distribution and real-life data distribution are respectively equal to 24.04 and 23.907.

For the **second task**, the probabilities of birthday conflicts are desired and for that matter, we have to create classes for students(by class, we don't mean the term "class" used in programming, we mean a normal hypothetical classroom). In order to perform the simulation, we consider 100 students (anything larger than this wouldn't be necessary). We also have to take into account that the number of students shouldn't be less than 2 (since there won't be any conflicts) and the number of trials should be higher than 32 (based on the central limit theorem).

We create the classes by making lists of zeros with lengths of 20. Again we create random birthdays (just like task 1) and check if there are any conflicts and we count the number of conflicts and compute the probability(the number of conflicts in each class divided by the size of each class)

You can see the comparison between the uniform distribution, real-life data distribution, and the theoretical scenario in the figure below.



Confidence Intervals

In order to find the confidence interval, first we need to find the mean of the number of people it takes until a conflict happens both for the uniform distribution and also the real-life data distribution.

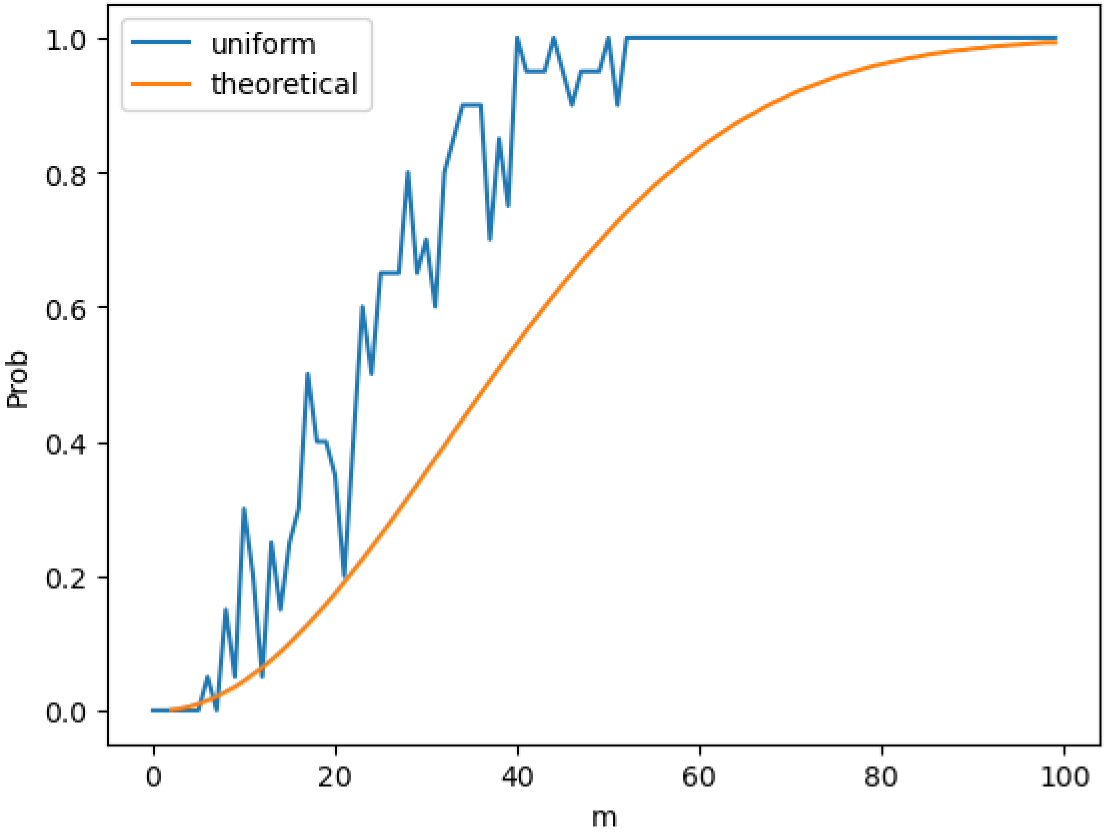
After that, the standard deviation has to be calculated. The degree of freedom is equal to 999 (n - 1) which is the number of trials minus one. We can calculate the confidence interval using different values for confidence. In this code, 0.9 has been chosen.

the obtained confidence interval for the real-life distro is equal to: (23.29160084090188, 24.52239915909812) and for the uniform distro, it is equal to: (23.37612603750964, 24.70387396249036).

Optional section

For this part of the lab, we can try to implement the simulation with values higher than 366 for “n” in order to generalize the problem and use it for other scenarios.

In this code, the experiment has been implemented with m = 1000 on the uniform distribution and comparing it to the theoretical output.


uniform
theoretical
Prob
m