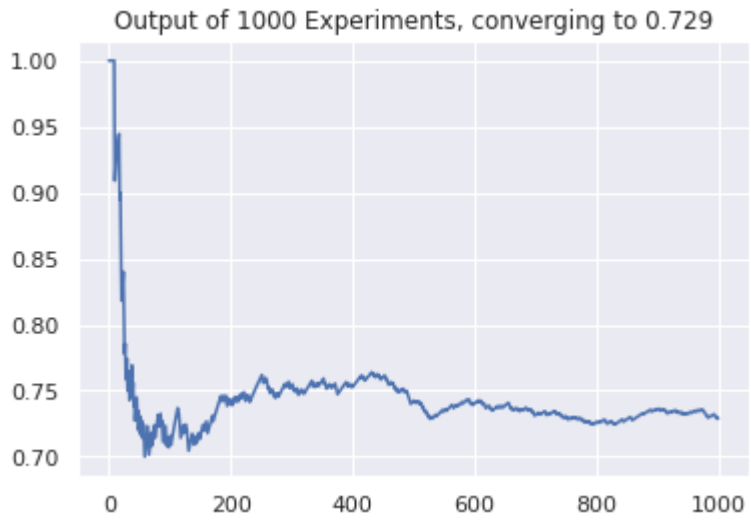
Muthukrishnan

AI, Computer Vision and Mathematics

May 15, 2021

Artificial Intelligence · Data science · Statistics

Birthday Problem and Monte Carlo Simulation



If you get on a plane that can carry 100 or more passengers at a time, there is a 99% chance that one of the passengers shares the same birthday as yours. If you get into a bus with a capacity of 50 passengers, you have a 97% chance of finding someone who shares your birth date! Surprising isn't it? This is exactly what the birthday paradox or the birthday problem is.

In this post, I will try to solve the birthday problem first the analytical way and then using the Monte Carlo simulation.

The Analytical Solution

The problem is to compute the probability of finding at least two people with the same birthday in a group of N people. Suppose we have a group of 10 people, the probability of two or more people sharing a birth date is equivalent to finding the probability of everyone having a different birthday and subtracting it from 1, using the idea, $P(A) = 1 - P(A')$, where A' represents "everyone having different birthdays"

We will ignore the leap year for simplicity's sake. If a person is born on a particular day, there are 365 ways of picking a date. Once the first date is picked, the 2nd person will be left with 364 days, the next one 363, and so on. So the number of ways everyone can have a different birthday is equivalent to 365x364x363x362.. 365-(N + 1), where N represents the number of people. In terms of probabilities, the person has 365 options to choose from so his/her probability will be 365/365, if the next person has a different birthday, his/her probability will be 364/365 and the list goes on. This leads us to the total probability as below.

$$P(A') = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \cdots \frac{365-(n+1))}{365}$$

so the probabability of two or more people sharing the same birthday is given by

$$P(A) = 1 - \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \cdots \frac{365-(n+1))}{365}$$

which is equivalent to,

$$P(A) = 1 - \frac{365 \times 364 \times 363 \times ..(365-(n+1))}{365^n}$$

Now that we have the formula, let's check the probability with different values of N using a simple python code:

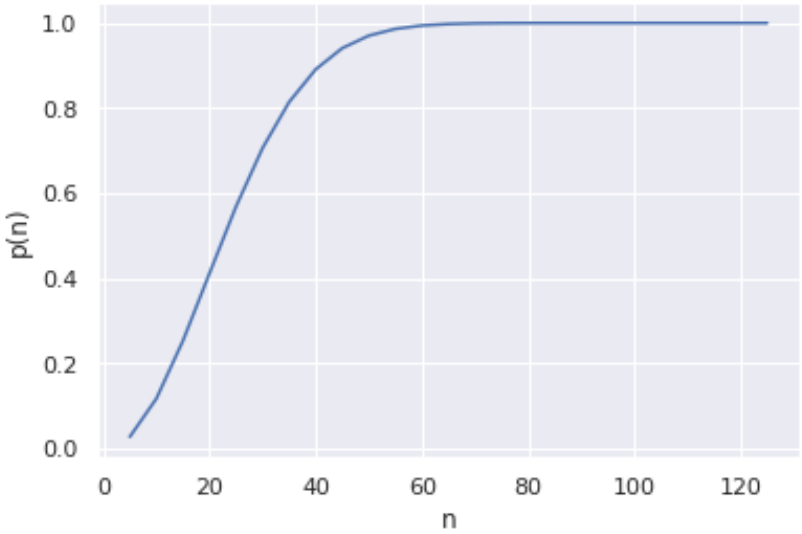
```
import numpy as np
import pandas as pd

results = []
for number_of_people in np.arange(5,130,5):
  probability = 1
  for count in range(1, number_of_people+1):
    probability *= (365-count+1)/365
  results.append({
      'n': number_of_people,
      'p(n)': (1-probability)
  })
print(pd.DataFrame(results))
```

The results are printed as below

N	P(N)
5	0.027136
10	0.116948
15	0.252901

As you can see in the above table, surprisingly, the probability of you sharing a birthday with someone else in a group of 100 and more people is nearly a 100%. The growth of this probability distribution can be seen in the below graph.

N	P(N)
20	0.411438
25	0.568700
30	0.706316
35	0.814383
40	0.891232
45	0.940976
50	0.970374
55	0.986262
60	0.994123
65	0.997683
70	0.999160
75	0.999720
80	0.999914
85	0.999976
90	0.999994
95	0.999999
100	1.000000
105	1.000000
110	1.000000
115	1.000000
120	1.000000
125	1.000000



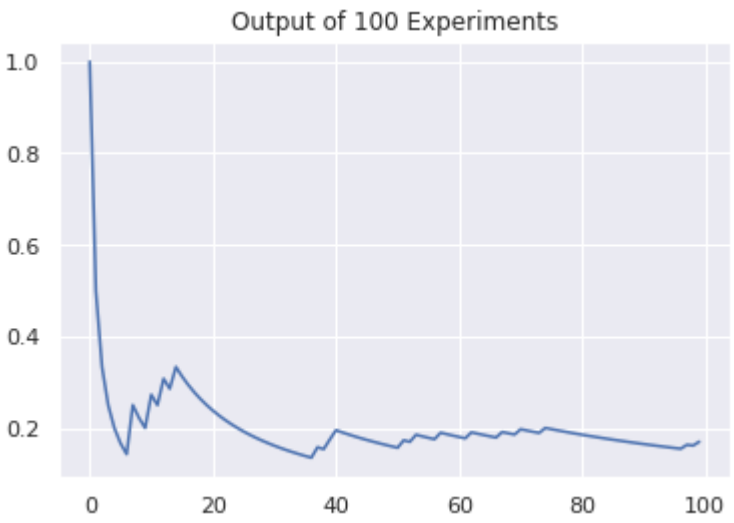
Monte Carlo Simulation

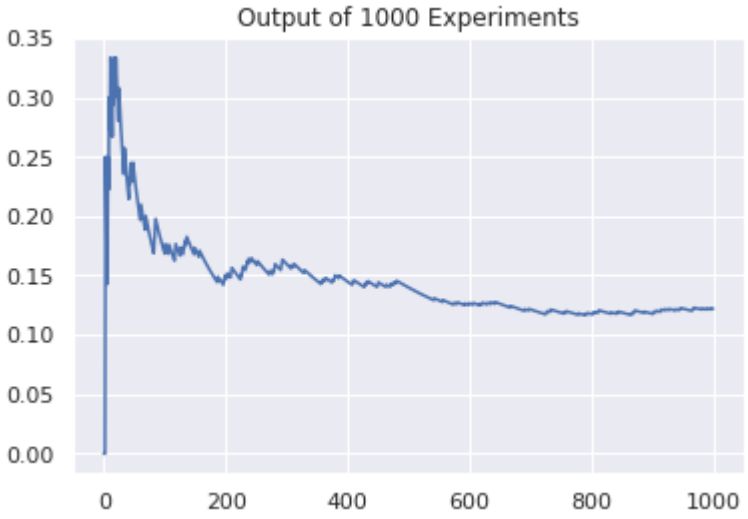
Monte Carlo simulation is a way to randomly simulate a large number of events and arrive at the probability of an outcome using the results of the experiments. The credit for the invention of Monte Carlo simulation is given to Stanislaw Ulam, a Polish-born mathematician who worked for John von Neumann. One day while trying to find the probability of winning a game of solitaire with pure combinatorial calculations, he wondered if he could just play the game of solitaire a large number of times and arrive at an opinion based on his observations. But it's nearly impossible to play the game a million times, that is when he called up John, his colleague and they decided to run this simulation on a computer. This way of using a simulation is called the Monte Carlo simulation named after a Casino in Monaco where Ulam's uncle would borrow money from relatives to gamble.

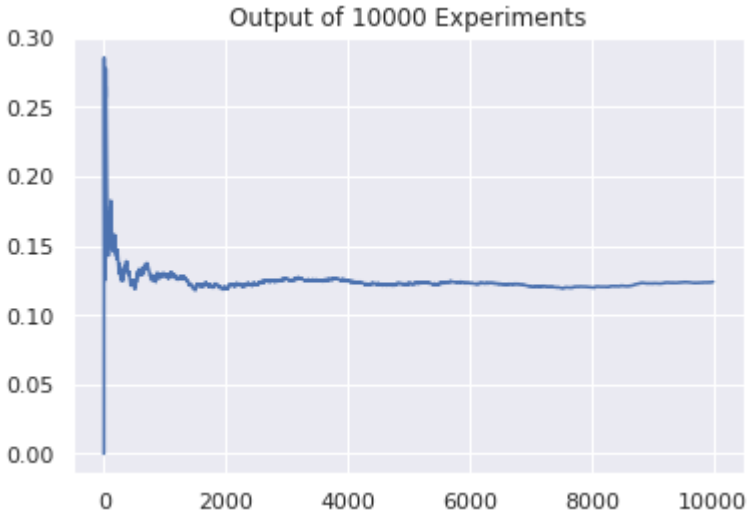
Let's look at a simple simulation before jumping into the birthday problem. Given 3 fair coins, the probability of getting 3 heads (HHH) in a row is 1/8, this is something we already know. Can we arrive at the same value using Monte Carlo Simulation? Let's give it a try. The code below simulates the tossing of three coins and stores the probabilities in an array. We plot the probabilities towards the end of our experiment.

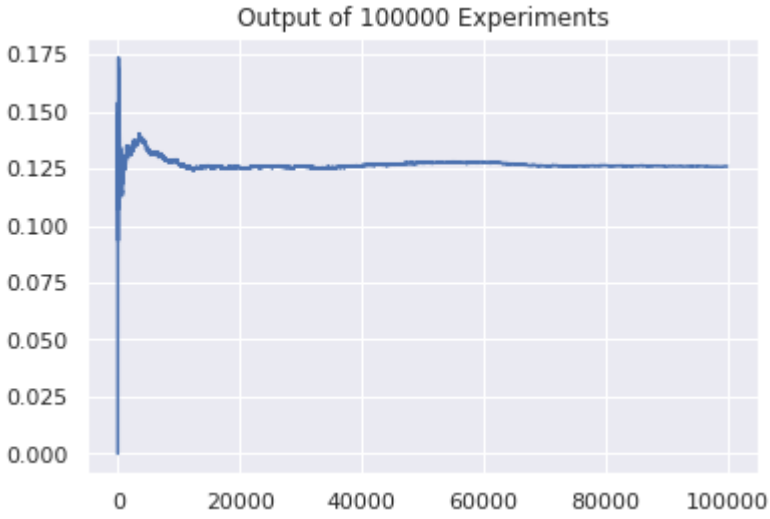
```
import matplotlib.pyplot as plt
import numpy as np
import random

# Assuming 0 to be Heads, 1 to be Tails
def toss_three_coins():
  coin1 = random.choice([0,1])
  coin2 = random.choice([0,1])
  coin3 = random.choice([0,1])

  if coin1 == coin2 and  coin2 == coin3 and coin1 == 0:
    # this is equivalent to HHH
    return 'HHH'
  else:
    return '' # empty as I am only interested in HHH

number_of_times_HHH = 0
total_number_of_experiments = 0
probabilities = []
for experiment_index in range(0,100): # increase this value to run more
experiments
  result = toss_three_coins()
  if result == 'HHH':
    number_of_times_HHH+=1 # increment HHH occurence.

  total_number_of_experiments += 1 # increment count

  probability = number_of_times_HHH/total_number_of_experiments
  probabilities.append(probability)

plt.plot(probabilities)
plt.show()
```

The outputs are as shown below:




Output of 1000 Experiments
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
200
400
600
800
1000


Output of 10000 Experiments
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2000
4000
6000
8000
10000


Output of 100000 Experiments
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
20000
40000
60000
80000
100000

As can be observed in the above charts, as we keep running the simulation, the probabilities are converging to the theoretical value of 0.125 (1/8). S, now that we know how Monte Carlo Simulation works, let's apply it to our birthday problem.

Birthday Problem Monte Carlo Simulation

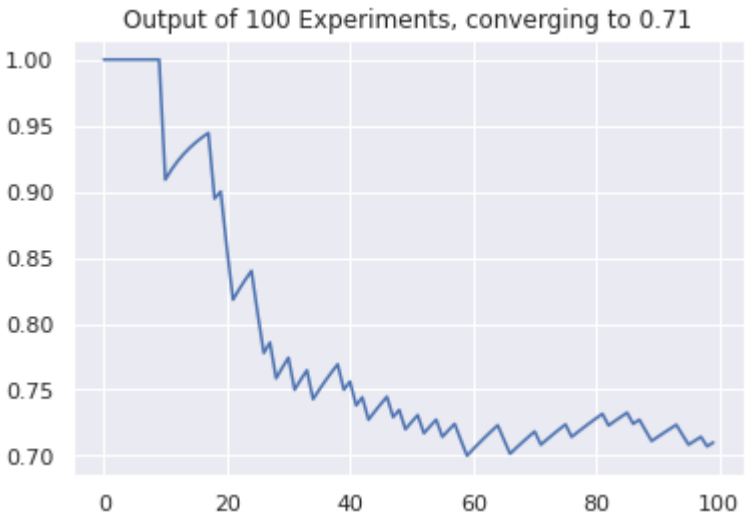
It's a lot simpler to simulate this problem over arriving at it analytically.

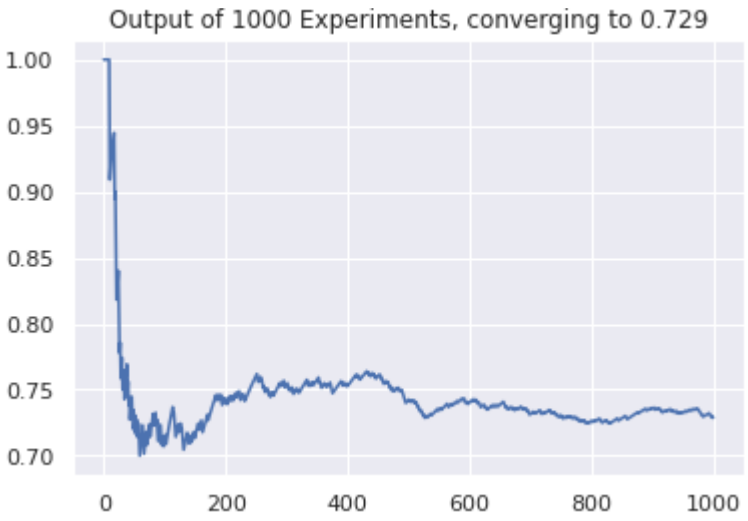
- First, choose the group size as N.
- Then randomly assign a date to each person in that group. Instead of assigning a date in terms of day and month, we can assign a number from 1 to 365.
- Check if there are any duplicate dates in the group.
- Divide the count of occurrences with the total number of experiments performed.

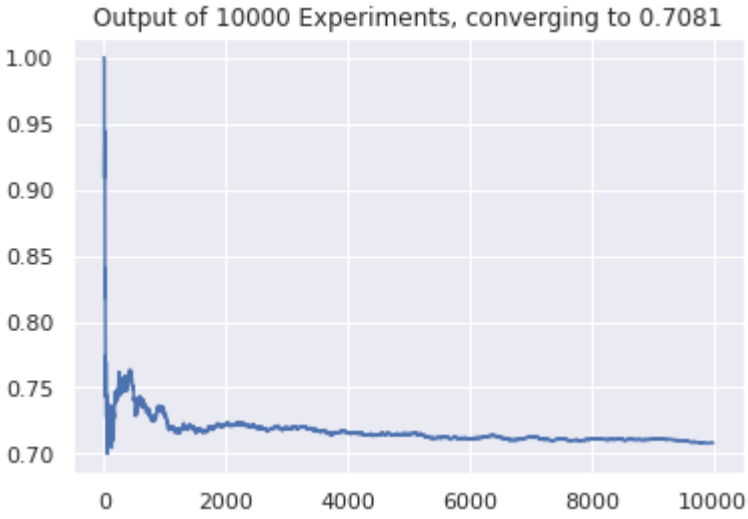
From the table above that we derived analytically, we can see that at 30 people, the probability of having a shared birthday is nearly 70%. So, for our experiment, let's run the simulation with N=30.

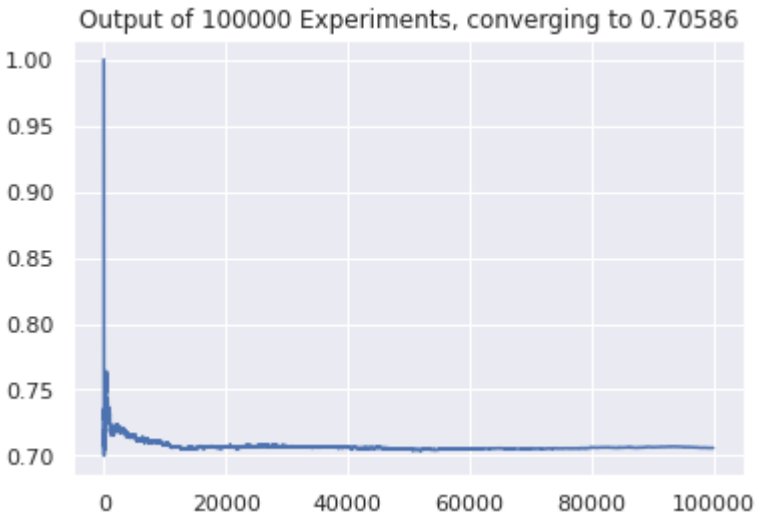
```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(10)

def contains_duplicates(X):
    return len(np.unique(X)) != len(X)


shared_birthday = 0
total_number_of_experiments = 0
probabilities = []
number_of_people = 30
for experiment_index in range(0,100):

  # assign dates to people
  dates = np.random.choice(range(1,366), size=(number_of_people))

  # check if there are duplicates
  if contains_duplicates(dates):
    shared_birthday+=1 # increment shared birthday

  total_number_of_experiments += 1 # increment count

  probability = shared_birthday/total_number_of_experiments
  probabilities.append(probability)

plt.plot(probabilities)
plt.title("Output of 100 Experiments, converging to " +
str(probabilities[-1]))
plt.show()
```

The outcomes are shown below




Output of 1000 Experiments, converging to 0.729
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
200
400
600
800
1000


Output of 10000 Experiments, converging to 0.7081
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
2000
4000
6000
8000
10000


Output of 100000 Experiments, converging to 0.70586
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
20000
40000
60000
80000
100000

As can be seen in the charts above, as more and more experiments are performed, the probabilities are slowly converging to our theoretical value of 0.70 for 30 people.

Key Ideas

- The Birthday problem or Birthday paradox states that, in a set of n randomly chosen people, some will have the same birthday. In a group of 23 people, the probability of a shared birthday exceeds 50%, while a group of 70 has a 99.9% chance of a shared birthday.
- We can use conditional probability to arrive at the above-mentioned probabilities.
- Monte Carlo simulation is a way to randomly simulate a large number of events and arrive at the probability of an outcome using the results of the simulated experiments.
- Monte Carlo simulation must perform a large number of experiments, over a million usually to find where the probability is converging. The value of convergence is usually very close to the real probability.

> Retrieved June 23, 2021 at 1:34 am (website time).
> Available at: 192.168.31.181/muthu/?p=1808