P_M1_1

October 19, 2021

This assignment will be reviewed by peers based upon a given rubric. Make sure to keep your answers clear and concise while demonstrating an understanding of the material. Be sure to give all requested information in markdown cells. It is recommended to utilize Latex.

0.0.1 Problem 1

The Birthday Problem: This is a classic problem that has a nonintuitive answer. Suppose there are N students in a room.

Part a) What is the probability that at least two of them have the same birthday (month and day)? (Assume that each day is equally likely to be a student's birthday and that there are no sets of twins.)

Note: Jupyter has two types of cells: Programming and Markdown. Programming is where you will create and run R code. The Markdown cells are where you will type out expalantions and mathematical expressions. Here is a document on Markdown some basic markdwon syntax. Also feel free to look at the underlying markdown of any of the provided cells to see how we use markdown.

$$P(\text{At least two have same birthday}) =$$
$$\text{1 - P(None have the same birthday)} =$$
$$1 - \frac{\frac{365!}{(365-N)!}}{365^N}$$

We can find the probability that at least two have the same birthday by subtracting the complement from 1. In other words, we want to find find 1 minus the probability that none of them have the same birthday.
To find the probability that None have the same birthday, we must divide 365! by (365 -N!) to represent the choices of days each member of the group has. For example, if there are two member of the group the first of the group would have 365 days to choose from and the second of the group would have 364 days. Simplifying $\frac{365!}{(365-2)!}$ would give use 365 * 364 on the numerator. Then, we must divide by the total number of possibilities (including if each person could have a birthday for any day of the year no matter what the person before them chose).

Part b) How large must N be so that the probability that at least two of them have the same birthday is at least 1/2?

[7]:
```
prob = function(n){1 - (choose(365,n) * factorial(n)) / 365^n}
#factorial multiplied back in so we don't divide by K! in our formula
prob(2)
```

0.00273972602739725

[8]:
```
for(n in 1:100){
    answer = prob(n)
    if(answer >= 0.50){
        print(n)
        print(answer)
        break
    }
}
```

```
[1] 23
[1] 0.5072972
```

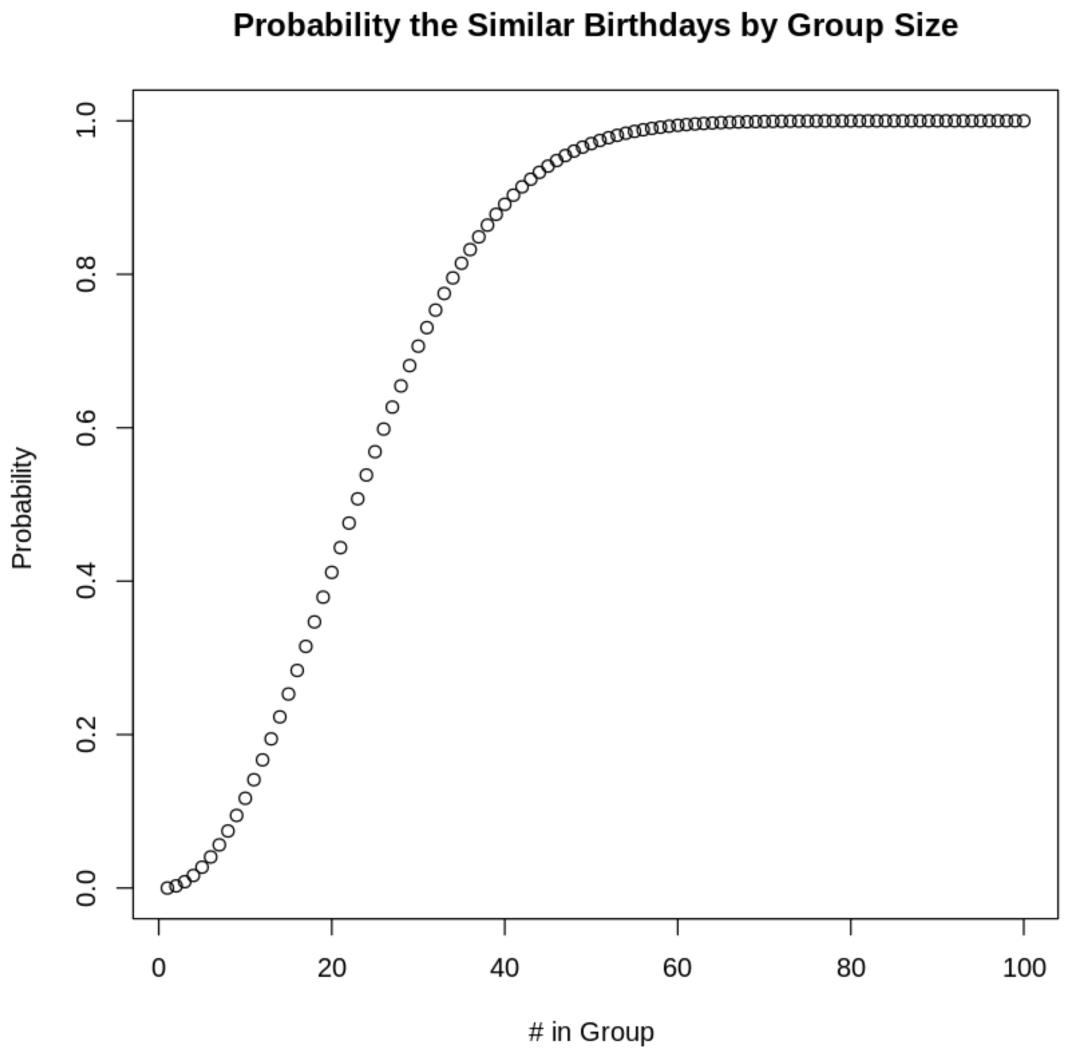
Due to the size of 365!, I used the 'choose function to find the combination and multiplied the factorial of N back into the equation. Therefore, we could get an efficient function for $\frac{365!}{(365-N)!}$ N must be at least 23 for the there to be a 0.50 probability or higher that two people share the same birthday.

Part c) Plot the number of students on the x-axis versus the probability that at least two of them have the same birthday on the y-axis.

[9]:
```
arrx = seq(1,100)
arrx = c(arrx)
arry = prob(arrx)
arry[23]
```

0.507297234323985

[10]:
```
plot(arrx, arry, main="Probability the Similar Birthdays by Group Size",
   xlab="# in Group ", ylab="Probability ")
```



Thought Question (Ungraded) Thought question (Ungraded): Would you be surprised if there were 100 students in the room and no two of them had the same birthday? What would that tell you about that set of students?

I would be very surprised. Maybe the students planned to deliberately not have two of the same people with the same birthday in the classroom?

1 Problem 2

One of the most beneficial aspects of R, when it comes to probability, is that it allows us to simulate data and random events. In the following problem, you are going to become familiar with these simulation functions and techniques.

Part a)

Let X be a random variable for the number rolled on a fair, six-sided die. How would we go about simulating X?

Start by creating a list of numbers [1, 6]. Then use the `sample()` function with our list of numbers to simulate **a single** roll of the die, as in simulate X. We would recommend looking at the documentation for `sample()`, found here, or by executing `?sample` in a Jupyter cell.

```
[11]: # Your Code Here
      die = 1:6
      sample(die, 1)
```

3

```
[12]: ?sample
```

Part b)

In our initial problem, we said that X comes from a fair die, meaning each value is equally likely to be rolled. Because our die has 6 sides, each side should appear about $1/6^{th}$ of the time. How would we confirm that our simulation is fair?

What if we generate multiple instances of X? That way, we could compare if the simulated probabilities match the theoretical probabilities (i.e. are all 1/6).

Generate 12 instances of X and calculate the proportion of occurances for each face. Do your simulated results appear to come from a fair die? Now generate 120 instances of X and look at the proportion of each face. What do you notice?

Note: Each time you run your simulations, you will get different values. If you want to guarantee that your simulation will result in the same values each time, use the `set.seed()` function. This function will allow your simulations to be reproducable.

```
[13]: set.seed(112358)
      # Your Code Here
      die_12_inst= sample(die, size=12, replace = TRUE)
      table(die_12_inst) / 12
```

```
die_12_inst
         1          2          3          4          5          6 
0.08333333 0.25000000 0.25000000 0.16666667 0.08333333 0.16666667 
```

To confirm if our simulation is fair, we would need to compare the distribution of our results of the experiment. Since we believe each side of the die should have an equal distribution, then each number should appear about $\frac{1}{6}$, or ≈ 0.16667, of the times we roll the die. In the experiment in which we ran 12 trials, some numbers show the exact percentage we were expecting. However, others showed way more or way less. It would help to run more trials. The more we trails we run, more of the true distribution of the experiment will be revealed.

[14]:
```
set.seed(112359)
die_120_inst= sample(die, size=120, replace = TRUE)
table(die_120_inst) / 120
```

```
die_120_inst
        1         2         3         4         5         6 
0.1666667 0.1916667 0.1750000 0.1583333 0.1250000 0.1833333 
```

as we can see in our second experiment with 120 trials, each of the outcomes have an outcome rate closer to what we expected (.16667)

Part c)

What if our die is not fair? How would we simulate that?

Let's assume that Y comes from an unfair six-sided die, where $P(Y = 3) = 1/2$ and all other face values have an equal probability of occuring. Use the `sample()` function to simulate this situation. Then display the proportion of each face value, to confirm that the faces occur with the desired probabilities. Make sure that n is large enough to be confident in your answer.

[15]:
```
# Your Code Here
die_unfair = sample(die, size=500, replace=TRUE, prob=c(.1, .1, .5, .1,.1,.1))
table(die_unfair) / 500
```

```
die_unfair
    1     2     3     4     5     6 
0.112 0.098 0.488 0.120 0.096 0.086 
```

To simulate an unfair die I initiated the 'prob' argument with each event's probability. The index of the vector represents the event on the die. For example, the .1 in the first index states that rolling a 1 has a .1 probability. By the results of the experiment, we can see the outcomes are close to matching the probabilities.