# P_M1_1

September 1, 2022

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, that there are no sets of twins, and that there are 365 days in the year. Do not include leap years).

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.

$$P(\text{At least two have same birthday}) = 1 - \frac{\frac{365!}{(365-N)!}}{365^N} = 1 - \frac{365!}{365^N * (365-N)!} = 1 - \frac{\prod_{n=0}^{N-1} 365 - n}{365^N}$$

**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?

If we replace in the previous formula the value of N by N = 23 we can see that the probability is over 50%

[24]:
```
# Proof
N = 23
1 - (prod(seq(from = 365-N+1, to = 365, by = 1))/365^N)
```

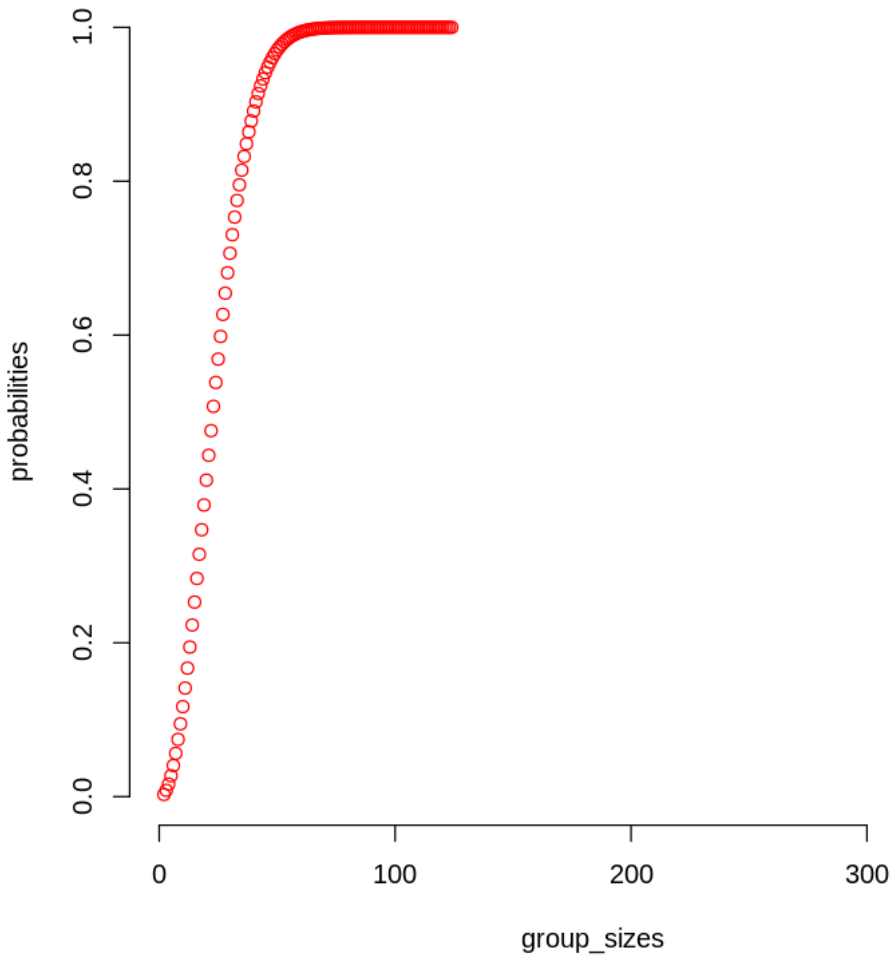
0.507297234323985

**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.

We will plot the probability for different size group from 2 to 365 (where obviously the porob is 100%)

[29]:
```
calculate_probs <- function(group_size) {
  return(1 - (prod(seq(from = 365-group_size+1, to = 365, by = 1))/
 ↪365^group_size))
}

group_size_limit = 365
group_sizes = seq(from = 2, to = group_size_limit, by = 1)
probabilities = lapply(group_sizes, calculate_probs)
plot(group_sizes, probabilities, type = "b", frame = FALSE,
     col = "red")
```



**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?

YOUR ANSWER HERE

# 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.

[1]:
```
# Your Code Here
sample_space = c(1,2,3,4,5,6)
# We tell to function that we want generate 1 sample from the sample space
sample(sample_space, 1)
```

2

**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?

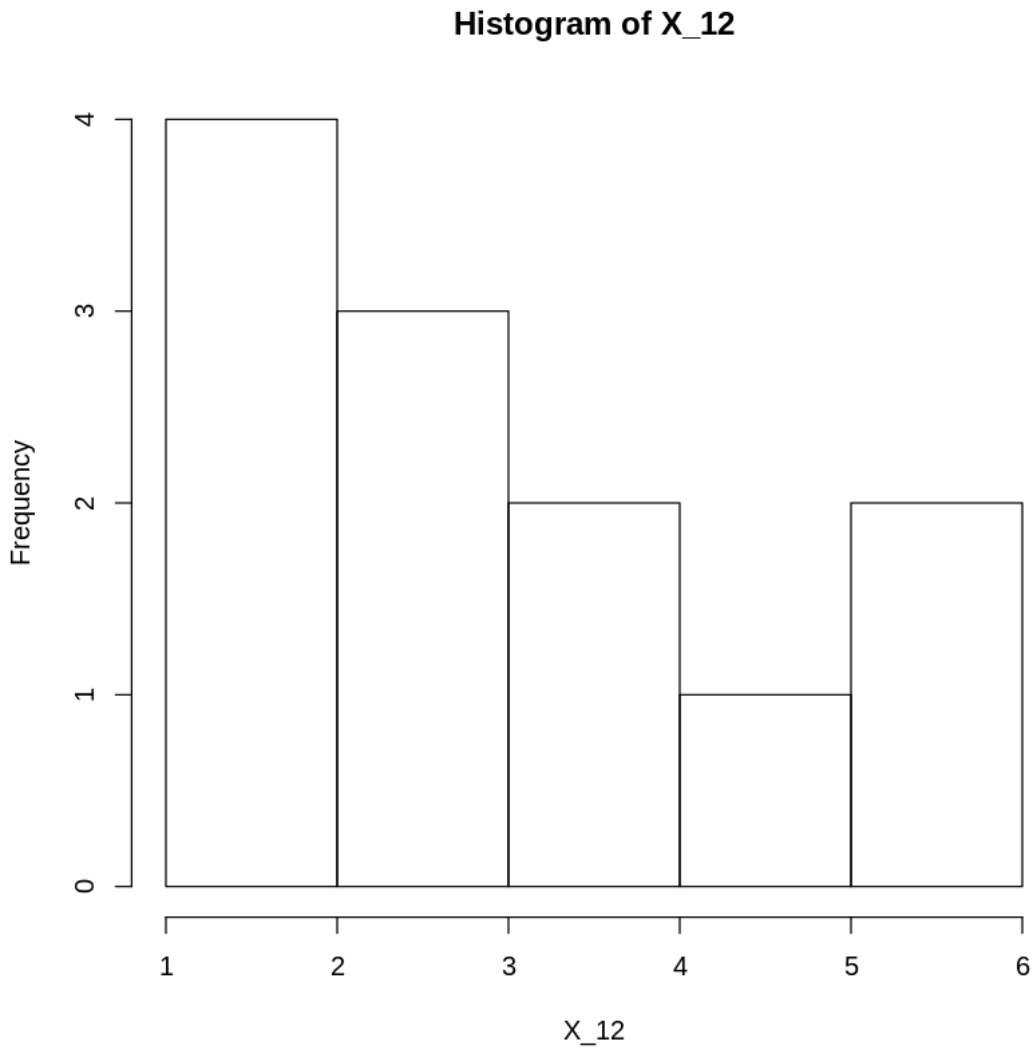
What if we generate multiple instances of $X$? That way, we could compare if the simulated probabilities match the theoretical probabilities (i.e. are all 1/6).

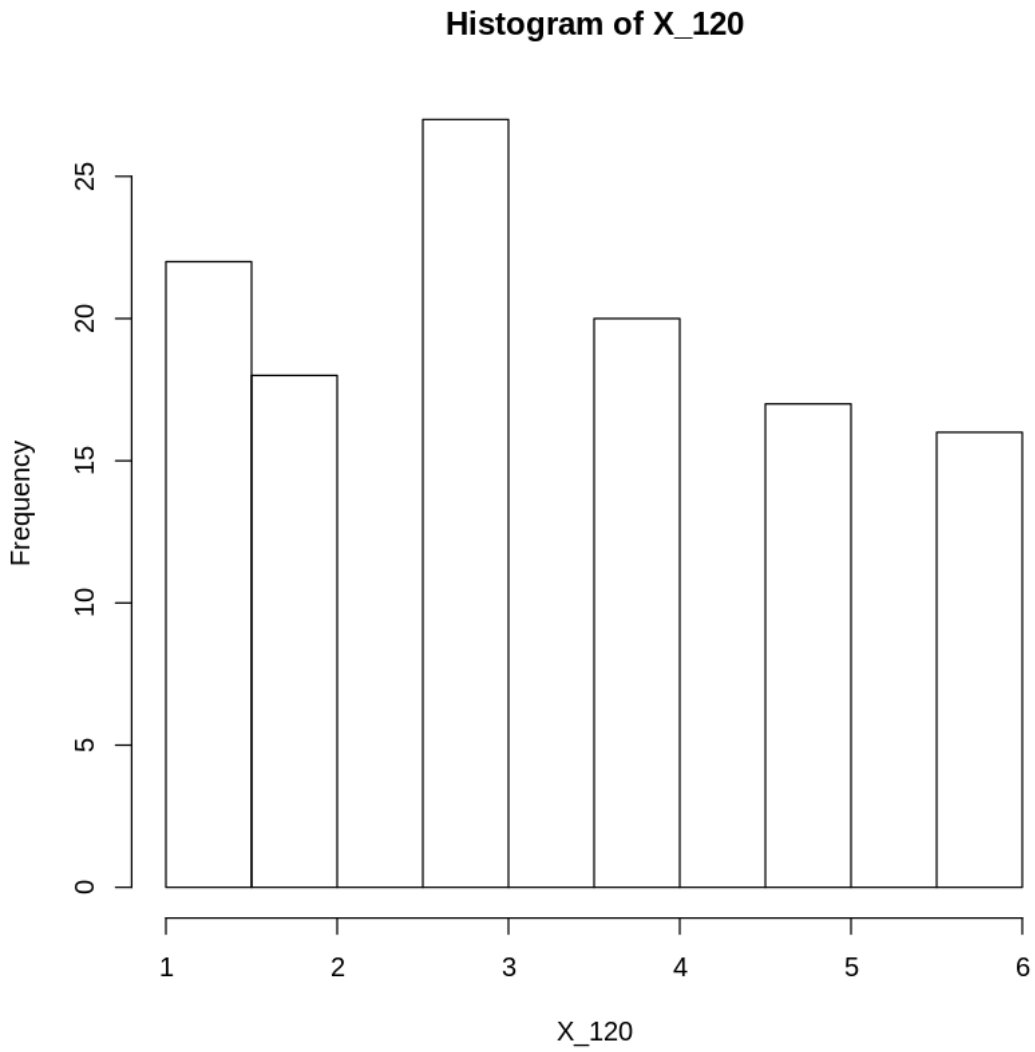
Generate 12 instances of $X$ and calculate the proportion of occurances for each face. Do your simulated results appear to come from a fair die? Now generate 120 instances of $X$ and look at the proportion of each face. What do you notice?

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.

[4]:
```
set.seed(112358)
# Your Code Here
X_12 = replicate(12,sample(sample_space, 1))
#
X_120 = replicate(120, sample(sample_space, 1))
```

```r
hist(X_12)
hist(X_120)
```






We can see that with bigger samples the distribution approach the uniform distribution and all the sides have the same frequencies and the same probability.

**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.

[9]:
```
# Your Code Here
prob_3 = 1/3
prob_rest = (1 - 1/3)/length(sample_space)
```

```
probs = replicate(6, prob_rest)
probs = replace(probs, 3, prob_3)
Y_1000 = replicate(1000, sample(sample_space, 1, prob=probs))
hist(Y_1000)
```