P_M1_1

March 21, 2023

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.

0.0.2 Solution

1st person can have any birthday in the year the second person must have probability with 364/365

$$P(\text{At least two have same birthday}) = 1 - (364/365) = 0.003$$

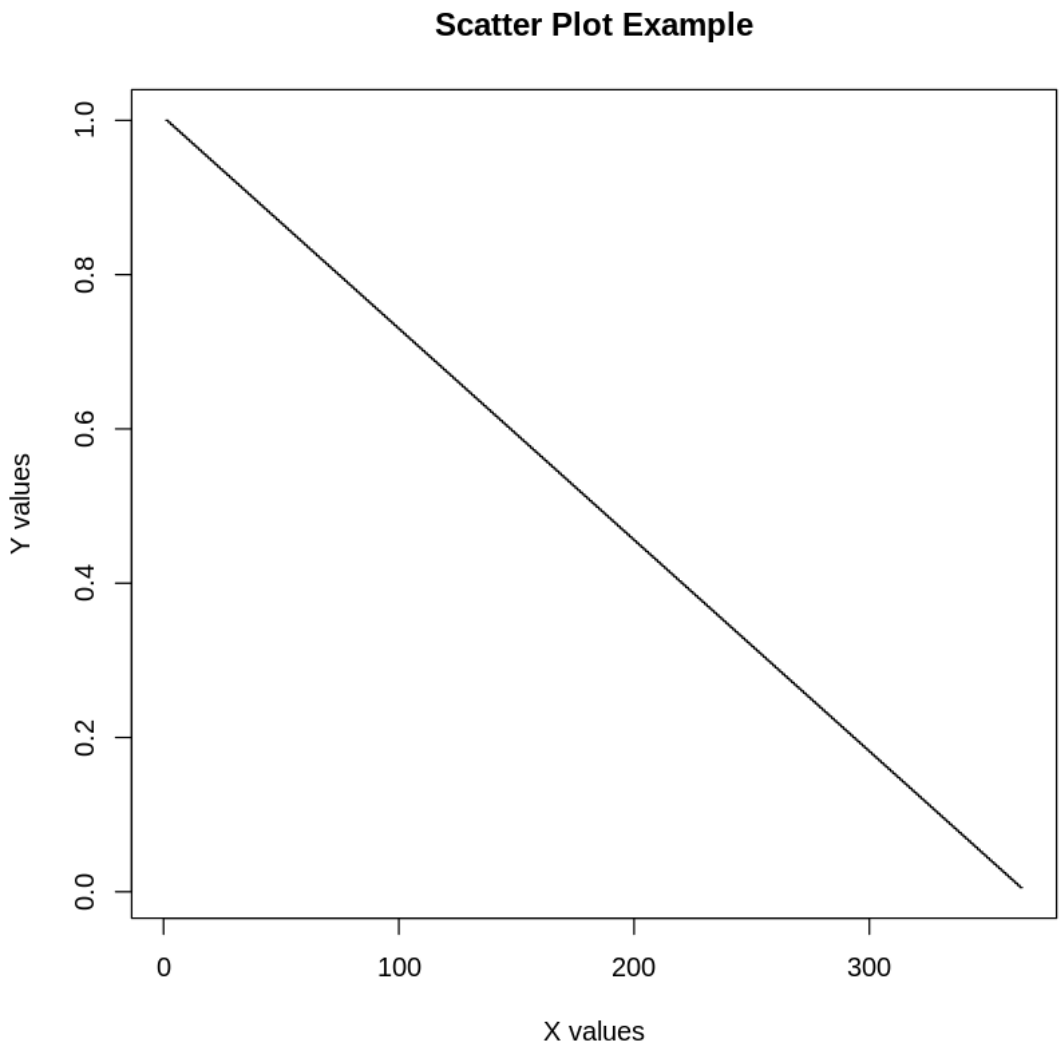
Part b) How large must N be so that the probability that at least two of them have the same birthday is at least 1/2?

0.0.3 Solution

We can calculate the probability for a given number of people using this equation: $1 - \left(\frac{365}{365}\right) \times \left(\frac{364}{365}\right) \times \left(\frac{363}{365}\right) \times \cdots \times \left(\frac{365 - n + 1}{365}\right)$ By placing N = 23 we can get probability that at least two of them have the same birthday is at least 1/2

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.

[93]:
```
N = 365
y = c()
x = c(1:365)
for (i in (1:N-1)){
    cal = ((N - i + 1)/N)
    y = append(probs,cal)
}
y = y[-1]
plot(x, y, type = "s", main = "Scatter Plot Example", xlab = "X values", ylab =␣
 ↪"Y values")
```



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.

[37]:
```
# Your Code Here
lst = c(1:6)
sample(lst)[1]
```

1

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?

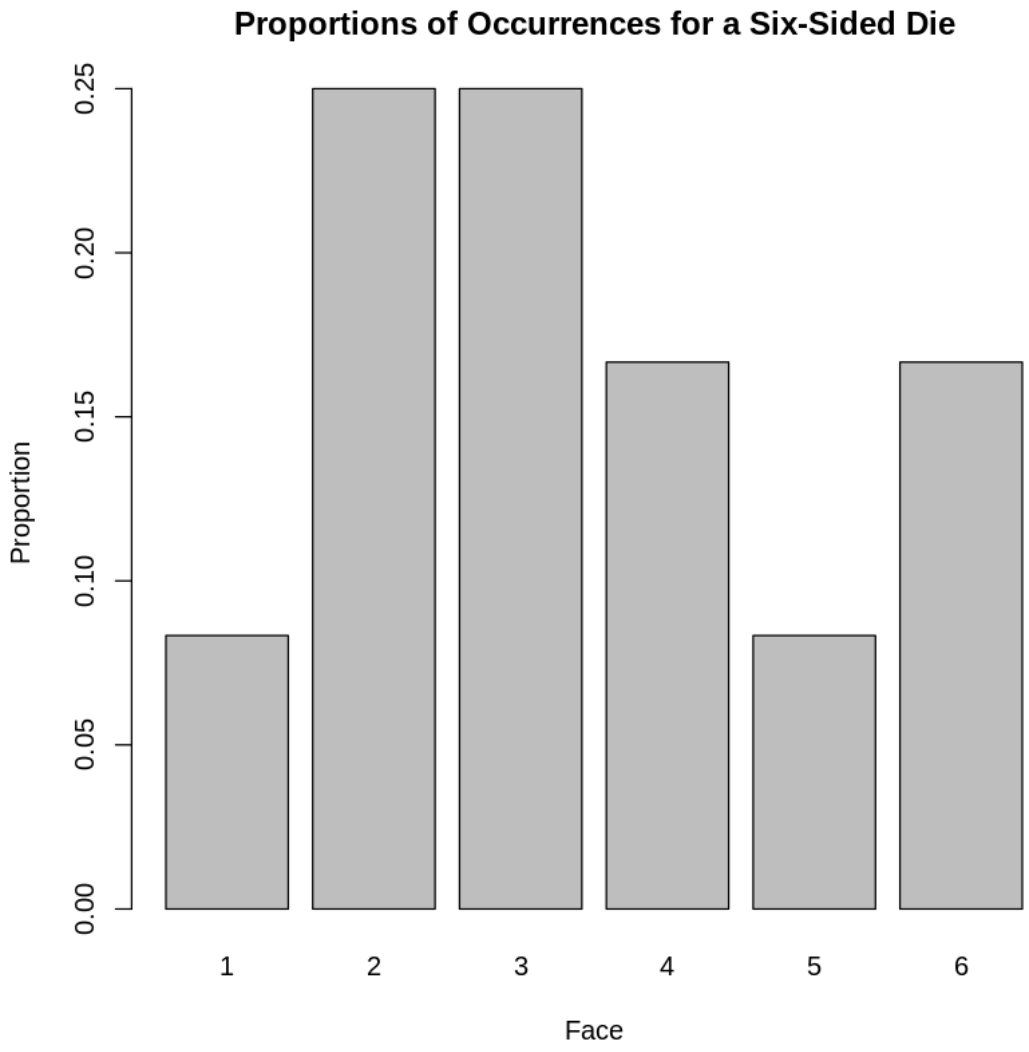
What if we generate multiple instances of X? That way, we could compare if the simulated probabilities match the theoretical probabilities (i.e. are all 1/6).

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.

[38]:
```
# Your Code Here
set.seed(112358)
X = c(1:6)
results = sample(X, 12, replace = TRUE)
proportions <- sapply(X, function(face) {
  sum(results == face) / length(results)
})
```

```
barplot(proportions, names.arg = outcomes, xlab = "Face", ylab = "Proportion",␣
↪main = "Proportions of Occurrences for a Six-Sided Die")
```

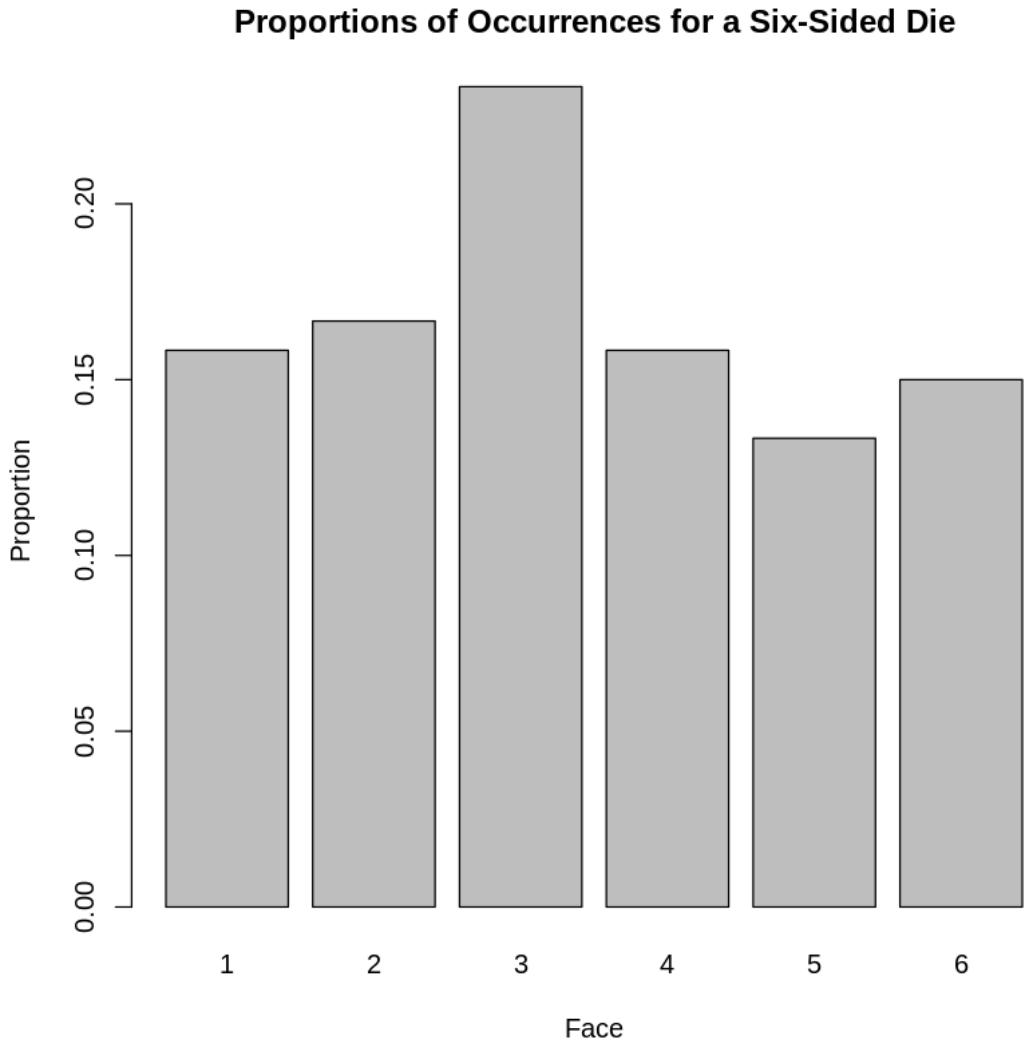


When we generate 12 instances of X, we may observe some variability in the proportions of occurrences for each face. However, since the sample size is relatively small, the proportions may not necessarily reflect the true probabilities of each face occurring. Therefore, we cannot definitively conclude whether the die is fair based on this small sample size

[39]:
```
# Your Code Here
set.seed(112358)
X = c(1:6)
results = sample(X, 120, replace = TRUE)
```

```r
proportions <- sapply(X, function(face) {
  sum(results == face) / length(results)
})

barplot(proportions, names.arg = outcomes, xlab = "Face", ylab = "Proportion",␣
↪main = "Proportions of Occurrences for a Six-Sided Die")
```



When we generate **120 instances of X**, we should observe the proportions of occurrences for each face converging towards the true probabilities of each face occurring (i.e., 1/6 for each face). This is because a larger sample size allows us to more accurately estimate the probabilities of each outcome occurring. Therefore, the results of simulating 120 instances of X should more closely resemble the true probabilities of a fair die.

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.

[43]:
```r
# Your Code Here
# Define the possible outcomes of rolling an unfair six-sided die
X = c(1, 2, 3, 4, 5, 6)

# Define the probabilities of each face value
probs = c(1/6, 1/6, 1/2, 1/6, 1/6, 1/6)

# Simulate 1000 instances of Y from the unfair six-sided die
results = sample(X, 1500, replace = TRUE, prob = probs)

# Display the proportion of each face value in the results
proportions <- sapply(outcomes, function(face) {
  sum(results == face) / length(results)
})

print(paste0("Proportions for 1500 instances of Y: ", proportions))
```

```
[1] "Proportions for 1500 instances of Y: 0.124"
[2] "Proportions for 1500 instances of Y: 0.13"
[3] "Proportions for 1500 instances of Y: 0.384"
[4] "Proportions for 1500 instances of Y: 0.124"
[5] "Proportions for 1500 instances of Y: 0.12"
[6] "Proportions for 1500 instances of Y: 0.118"
```