Stat 243: Lab, Monday Oct-17

OOP in R: Rolling a Die

The purpose of this lab is to practice Object-Oriented Programming (OOP) using either S3 or S4 classes in R.

The goal is to program two classes of objects: a regular `"die"` with six sides, and an object `"roll"` (i.e. the rolls of a `"die"` multiples times).

- Use an `.R` script file to write code for the class objects and methods
- If you are already familiar with S3 classes, try using S4 classes
- If you are new to both S3 and S4 classes, choose one of them

Object `"die"`

The object `"die"` should have two attributes:

- `sides`: vector with numbers 1, 2, 3, 4, 5, 6.
- `prob`: vector of probabilities for each side

For the `"die"` object write:

- a constructor function `die()` that creates a *fair* die by default
  - make sure that the argument `prob` has correct probability values
- a `"print"` method that displays the sides and the associated probabilities

You should be able to use `die()` like this:

```
# create a fair die by default
fair_die <- die()

# create a loaded die
loaded_die <- die(prob = c(0.075, 0.1, 0.125, 0.15, 0.20, 0.35))
```

Object `"roll"`

The object `"roll"` should have these attributes:

- `rolls`: vector with outputs of the rolls
- `sides`: vector with numbers 1, 2, 3, 4, 5, 6.
- `prob`: vector of probabilities for each side

For the "roll" object write:

- a constructor function `roll()` that *rolls* a "die" a given number of `times`
  - give `times` a default value of 1
- a `print()` method that displays just the rolls
- a `summary()` method that displays a table of frequencies: both the counts and proportions of each side of the rolled `die`
- a `plot()` method using a barchart of frequencies (relative frequencies of 1's, 2's, 3's, 4's, 5's, and 6's)
- an extraction method "[" to extract the value of a given roll
- a replacement method "[<-" to replace the value of a given roll
- an addition "+" method to add more rolls

You should be able to use `roll()` and its methods like this:

```
# roll fair die
fair500 <- roll(fair_die, times = 500)

# summary method
summary(fair500)
```

```
## summary "roll"
##
##   side count  prop
## 1    1    92 0.184
## 2    2   103 0.206
## 3    3    83 0.166
## 4    4    93 0.186
## 5    5    65 0.130
## 6    6    64 0.128
```

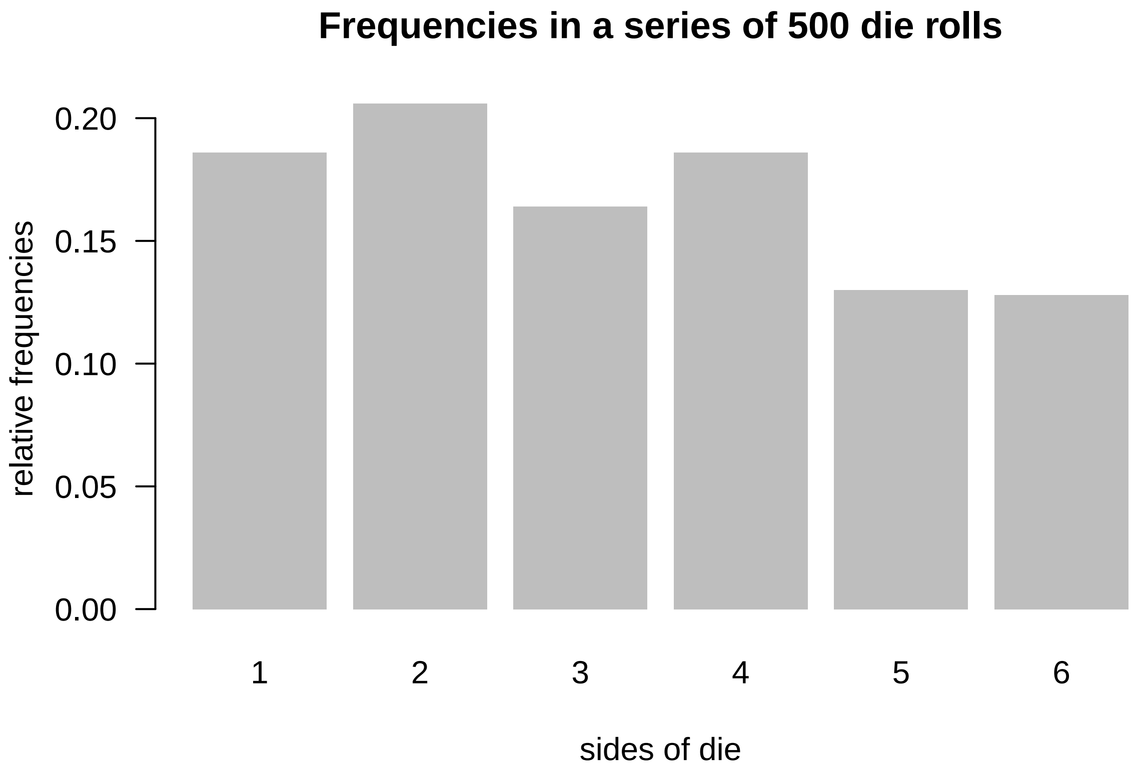
```
# retrieving roll in position 100
fair500[100]
```

```
## [1] 2
```

```
# replacing last roll
fair500[500] <- 1

# adding 100 rolls
fair600 <- fair500 + 100

# plot method
plot(fair500, 500)
```



De Mere's problem

Use your objects "`die`" and "`roll()`" to simulate a series of 1000 games for the famous *Chevalier De Mere's* dice problems:

- One gambling problem that launched modern probability theory (by Dan Ma).

The first problem involves computing the probability of getting at least one "6" in four rolls of a die. This probability can be computed analytically as:

$$1 - (5/6)^4$$

The other problem involves computing the probability of of getting at least two "6" in 24 rolls of a pair of dice. This probability can be computed analytically as:

$$1 - (35/36)^{24}$$

The goal is to use your `roll()` function to simulate a series of 1000 games for both of De Mere's problems:

- one series of 1000 games should involve rolling a die four times, and then count the number of games in which there is at least one 6.
- the other series of 1000 games should involve rolling a pair of dice 24 times, and then count the number of games in which there is at least one double 6.

Testing whether a die is fair

Here's one more challenge. The idea is to write R code, preferably a function, such that given a vector of outputs from rolling a die, it can be determined whether the die is fair.

One way to approach this problem is by using Pearson's Chi-square statistic:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} = N \sum_{i=1}^{n} \frac{(O_i/N - p_i)^2}{p_i}$$

where:

- n is the number of sides (i.e. 6)
- O_i is the observed frequency of side i, $i = 1, 2, 3, 4, 5, 6$
- $E_i = Np_i$ is the expected frequency of side i
- p_i is the theoretical relative frequency (1/6 assuming a fair roll)
- N is the total number of rolls

When testing for a fair six-sided die, the statistic χ^2 asymptotically approaches a chi-square distirbution with degrees of freedom $n - 1 = 5$.

You can generate a loaded die, and roll it 1000 times:

```r
set.seed(3625)
loaded_die <- die(prob = c(0.075, 0.1, 0.125, 0.15, 0.20, 0.35))
load_rolls <- roll(loaded_die, 1000)
```

Try implementing the chi-square statistic and compute a hypothesis test to see whether the rolls in `load_rolls` support the null hypothesis of a fair die.