Lab 2

Due Thursday September 11 at 11:59 PM

Last week, you used the `sample` function to simulate random phenomena that had a finite sample space with equally-likely outcomes: coin flips, die rolls, playing cards. Using a loop, you repeated an experiment many many times, calculated the proportion of the time some event happened, and used that proportion as an approximation of the probability[1]. In class this week, we learned counting techniques for actually doing the math to compute these probabilities exactly.

In this lab, you will do both of these things, the math and the simulation, and verify that they agree. The examples today are small potatoes, but these generic skills are important. Being able to whip up computer simulations of random systems is one of the main skills that will empower you to continue self-studying probability and statistics after you leave the classroom. Hardly a day goes by when myself and my colleagues don't run a simulation of some kind.

Write-up instructions

The template for this assignment is in the Canvas files. You will submit solutions for Tasks 1 - 3 below (Task 0 is just an extra example for you to refer to). A complete solution has three pieces:

- the `R` code that runs the simulation and approximates the probability;
- the one line of code that uses `R` as a calculator to compute the exact decimal probability based on your mathematical solution;
- A little math formula (between double dollar signs in your `.qmd`) that summarizes the final answer.

Tasks 0 and 1 below provide a model for how this looks: simulation, calculation, formula for $P(A)$.

Task 0 (review)

Consider this question from Lab 1:

[1] The proportion will converge to the true probability by the *law of large numbers*, which we will study later in the course.

a five-card hand is randomly dealt to you from a well-shuffled deck. What is the probability that the hand contains *at least* one ace?

Last week, we didn't have the tools to do the math, but we could program a computer to approximate:

```
deck_of_cards <- c("AH", "2H", "3H", "4H", "5H", "6H", "7H", "8H", "9H", "10H", "JH", "QH",
                   "AD", "2D", "3D", "4D", "5D", "6D", "7D", "8D", "9D", "10D", "JD", "QD",
                   "AC", "2C", "3C", "4C", "5C", "6C", "7C", "8C", "9C", "10C", "JC", "QC",
                   "AS", "2S", "3S", "4S", "5S", "6S", "7S", "8S", "9S", "10S", "JS", "QS",

set.seed(123)

outcomes <- c()

for (i in 1:10000) {
  hand <- sample(deck_of_cards, size = 5, replace = FALSE)
  if (any(hand %in% c("AH","AD","AC","AS"))) {
    outcomes <- c(outcomes, 1)
  } else {
    outcomes <- c(outcomes, 0)
  }
}

mean(outcomes)
```

```
[1] 0.3473
```

Now let's see how the math goes. The complement of the target event "at least one ace" is "no aces at all." This is simpler and may be easier to count, so we will apply:

$$P(A) = 1 - P(A^c) = 1 - \frac{\#(A^c)}{\#(S)}.$$

The sample space S is the set of all five-card hands you could be dealt. We are selecting $k = 5$ from $n = 52$ without replacement and ignoring order, so $\#(S) = \binom{52}{5}$. The event A^c is the set of all five-card hands that do not include an ace. There are 48 non-ace cards, and so $\#(A^c) = \binom{48}{5}$. As such

$$P(A) = 1 - \frac{\binom{48}{5}}{\binom{52}{5}} \approx 0.34.$$

To get the actual number, we can use the `choose` function in `R`. Try typing this into the console:

```r
1 - choose(48, 5) / choose(52, 5)
```

```
[1] 0.341158
```

This is close to the number we approximated via simulation. Phew!

Task 1 (I do it)

- **Scenario**: roll three *fair* dice;
- **Event**: all three dice show different numbers.

This code simulates 7,500 trials and computes the proportion of the time the event occurs:

```r
set.seed(37367)

nreps <- 7500
counter <- 0

for(i in 1:nreps){

  rolls <- sample(1:6, 3, replace = TRUE)

  if( length(unique(rolls)) == 3 ){

    counter <- counter + 1

  }

}

counter / nreps # proportion
```

```
[1] 0.558
```

In order to do the math, we apply $P(A) = \#(A)/\#(S)$ directly. The sample space S is the set of all three-dice rolls. Thought of as three experiments in the sense of the counting principle, the number of ways we could mix and match the rolls is $\#(S) = 6 \times 6 \times 6 = 6^3 = 216$. The target event A is the set of all three-dice rolls where all numbers are different (ie sampling without replacement), and so $\#(A) = 6 \times 5 \times 4 = 120$. Putting it into `R` gives:

```r
120 / 216
```

```
[1] 0.5555556
```

We conclude then that

$$P(A) = \frac{6 \times 5 \times 4}{6^3} = \frac{120}{216} \approx 0.555,$$

which agrees with our simulation.

Task 2 (we do it)

- **Scenario**: 6 balls are randomly drawn from a jar containing 5 red, 4 blue, 3 green;
- **Event**: you draw exactly 3 red or exactly 2 green.

Here is a representation of the contents of the jar that you can sample from:

```
jar <- c(rep("R", 5), rep("B", 4), rep("G", 3))
jar
```

```
 [1] "R" "R" "R" "R" "R" "B" "B" "B" "B" "G" "G" "G"
```

Solution

```
set.seed(2025)
nsim <- 20000
count_event <- 0

jar <- c(rep("R", 5), rep("B", 4), rep("G", 3))

for (i in 1:nsim) {
  hand <- sample(jar, 6, replace = FALSE)
  n_red <- sum(hand == "R")
  n_green <- sum(hand == "G")

  if (n_red == 3 | n_green == 2) {
    count_event <- count_event + 1
  }
}

count_event / nsim
```

```
[1] 0.6606
```

- A = exactly 3 red, B = exactly 2 green
- Both could happen so $A \cap B$ is non-empty;
- $\#(S) = \binom{12}{6}$;
- $\#(A) = \binom{5}{3}\binom{7}{3}$;
- $\#(B) = \binom{3}{2}\binom{9}{4}$;
- $\#(A \cap B) = \binom{5}{3}\binom{3}{2}4$;
- So

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = \frac{\binom{5}{3}\binom{7}{3}}{\binom{12}{6}} + \frac{\binom{3}{2}\binom{9}{4}}{\binom{12}{6}} - \frac{\binom{5}{3}\binom{3}{2}4}{\binom{12}{6}}.$$

Calculate:

```
PA <- choose(5, 3) * choose(7, 3) / choose(12, 6)
PB <- choose(3, 2) * choose(9, 4) / choose(12, 6)
PAB <- choose(5, 3) * choose(3, 2) * 4 / choose(12, 6)

PA + PB - PAB
```

```
[1] 0.6580087
```

Task 3 (you do it)

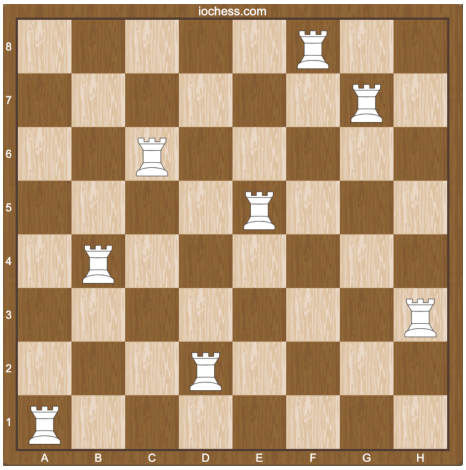

Figure 1: 8 safe rooks

- **Scenario**: you randomly place 8 rooks on *distinct* squares of a chess board;
- **Event**: all 8 rooks are safe from one another.

Figure 1 displays an example of a safe position. In chess, rooks are permitted to move up-and-down or side-to-side as far as they want, but not diagonally or anything else.

This code creates a *matrix* representing the chess board, from which you can sample the random positions:

```
board <- outer(letters[1:8], 1:8, paste0)
board
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
```

```
[1,] "a1" "a2" "a3" "a4" "a5" "a6" "a7" "a8"
[2,] "b1" "b2" "b3" "b4" "b5" "b6" "b7" "b8"
[3,] "c1" "c2" "c3" "c4" "c5" "c6" "c7" "c8"
[4,] "d1" "d2" "d3" "d4" "d5" "d6" "d7" "d8"
[5,] "e1" "e2" "e3" "e4" "e5" "e6" "e7" "e8"
[6,] "f1" "f2" "f3" "f4" "f5" "f6" "f7" "f8"
[7,] "g1" "g2" "g3" "g4" "g5" "g6" "g7" "g8"
[8,] "h1" "h2" "h3" "h4" "h5" "h6" "h7" "h8"
```

Here's an example of 8 random squares chosen for the rooks:

```
random_position
```

```
[1] "g6" "h4" "h6" "f4" "d1" "c6" "d7" "d6"
```

So that's where your 8 rooks live. In order to check whether or not they are safe from one another, you might find it helpful to separate out the row information and the column information of the placement using the `substring` command:

```
substring(random_position, 1, 1)
```

```
[1] "g" "h" "h" "f" "d" "c" "d" "d"
```

```
substring(random_position, 2, 2)
```

```
[1] "6" "4" "6" "4" "1" "6" "7" "6"
```

Solution

```r
board <- outer(letters[1:8], 1:8, paste0)

set.seed(2025)
nsim <- 100000
count_safe <- 0

for (i in 1:nsim) {
  rooks <- sample(board, 8)

  # split into rows (numbers) and cols (letters)
  r <- substring(rooks, 2, 2) # row = the number part
  c <- substring(rooks, 1, 1) # col = the letter part

  # safe if all 8 rows are different AND all 8 cols are different
  if (length(unique(r)) == 8 & length(unique(c)) == 8) {
    count_safe <- count_safe + 1
  }
}

count_safe / nsim
```

```
[1] 1e-05
```

- Recall that rooks can move up and down or side-to-side, but not diagonally. In order to be safe from one another, two rooks therefore need to occupy different rows *and* columns of the board. For the eight rooks to be mutually safe, they all must be in different columns and different rows from one another;
- There are $n = 8 \cdot 8 = 64$ squares on the board, and we are selecting $k = 8$ of them at which to place the rooks. The rooks are identical, so **order doesn't matter**, and it's **without replacement** since the squares need to be distinct, and so the total number of possible outcomes is $\#(S) = \binom{64}{8}$;
- The event A we care about is "the rooks are safe from one another," and this can happen $\#(A) = 8!$ ways.
- We know that a safe placement will have every row and column occupied by exactly one rook, so let us imagine building up a safe placement row-wise, starting from the bottom and working upward. There are 8 ways to place a single rook on the bottom row. No other rooks can go there, so we move to the next row. There are only 7 ways to place the next rook in the next row, because we must avoid the column occupied by the first rook. Similarly, there are only 6 ways to place the next rook in the next row, because we must avoid the two columns occupied by the first two rooks. And so on. As a

consequence of the counting principle, where we regard each row as an experiment, we see that $\#(A) = 8 \times 7 \times 6 \times \cdots \times 2 \times 1 = 8!$.

- So

$$P(A) = \frac{\#(A)}{\#(S)} = \frac{8!}{\binom{64}{8}} \approx 0.0000091095.$$

.

```
factorial(8) / choose(64, 8)
```

```
[1] 9.109465e-06
```