Simulation exercise

Download this file's source (day-3-sim-ex.qmd) and open it in RStudio, Positron, or VS Code. Work through the code and associated questions. ***Render as you go to ward off problems as early as possible.***

For your own good, avoid AI tools for this task. Use only `?function_name` and R documentation or the useful links in the slides.

Reminder: the `mean()` of a `TRUE`/`FALSE` vector gives you the **proportion that are `TRUE`**.

Setup (given)

```
library(tidyverse)

set.seed(365)

counties <- left_join(
    read_csv("https://princeton-ddss.github.io/r-bootcamp/files/county_data.csv"),
    read_csv("https://princeton-ddss.github.io/r-bootcamp/files/county_population.csv"),
    join_by(GEOID == fips)
)
```

1 Part 1: the birthday problem

In this section, we are going to work with the "Birthday Problem." *Suppose k people gather in a room. Assuming birthdays are independent and equally likely on any day, what is the probability that at least two of them share a birthday?*

We won't derive a closed-form solution. Instead, we will simulate results.

1.1 One room

Draw $k = 23$ birthdays and check whether any two of them match. Represent a birthday as a number from 1 to 365.

Hint: `duplicated()` and `anyDuplicated()` are useful here, but you don't *need* them.

1.2 Wrap it in a function

Write a function `shared_birthday(k)` that runs your last answer once for an arbitrary k and returns a single `TRUE` or `FALSE`. Call it a few times to convince yourself it varies.

1.3 Many rooms

Use `replicate()` to run `shared_birthday(23)` 10,000 times and estimate the probability.

1.4 How many replications did you need?

Estimate the same probability using 10, 100, 1,000, 10,000, and 100,000 replications. How many replications do you need before the answer stops moving in the second decimal place?

1.5 Sweep across room sizes

Now vary the room. Estimate the probability for every k from 2 to 60 and store the results in a data frame with columns `k` and `prob`. (For this, you can drop down to 1,000 replications if your laptop takes a while.)

1.6 Plot it

Plot the number of people in the room against the probability of a shared birthday. Roughly how many people do you need for the probability to pass 50%?

1.7 Check against the exact answer

There is a closed form. The probability that *nobody* shares a birthday among k people is

$$\prod_{i=0}^{k-1} \frac{365 - i}{365}$$

so the probability at least two people do is one minus that. In R:

```
prod((365 - 0:(k - 1)) / 365)
```

Add the exact values to your plot as a second line to see how close 1,000 replications got you to the real answer.

2 Part 2: sampling and the bootstrap

For this part, **pretend the 3,135 counties are the population.** We'll draw samples from it to watch what sampling does.

2.1 The population

A handful of counties are missing `pop`. Build a vector called `pop_all` holding the county populations with those dropped, and use it for the rest of this part.

Then, plot its distribution and report its mean and standard deviation. Note that it is not symmetric/nothing like a "bell curve."

2.2 Sampling distributions

For each sample size $n \in \{2, 5, 30, 100, 1000\}$, draw 10,000 samples of that size **with replacement** and record each sample's mean. Plot the five distributions of sample means, preferably using `facet_*()`.

(Sampling with replacement keeps each draw independent of the others, which makes the arithmetic in §2.3 exact.)

Hint: to facet, all five sets of means need to live in *one* data frame, with a column recording which sample size produced each mean. `map_dfr()` over the five sizes will build that, and so will a `for` loop.

Watch the shape change as n grows. FYI – It's converging to a symmetric distribution centered around the "population" (county) mean. (Later, someone will explain the relationship between this and the "normal distribution.")

2.3 The width of those distributions

For each n, compute the standard deviation of your 10,000 sample means. Then print a data frame with one row per sample size and four columns: the sample size, that simulated standard deviation, $\sigma/\sqrt{n}$, and the ratio of the last two (as a comparison between the two quantities). (σ is the population standard deviation from §2.1.)

2.4 The middle 95%

Take your 10,000 sample means for $n = 100$ and find the values that cut off the bottom 2.5% and the top 2.5%. `quantile()` does this.

2.5 The bootstrap

Everything above relied on having the whole population, which let us draw as many fresh samples as we liked, at any size we liked. **You will almost never be in that position.** You usually only have one sample. Above, we knew the population and characterized repeated sample means. Now we have a sample and try to characterize the population.

Run this, and from here on pretend `my_counties` is all the data that exists:

```
my_counties <- counties |>
    filter(!is.na(income)) |>
    slice_sample(n = 50)
```

Report the mean `income` in `my_counties`. That is your estimate.

Then, to see how much it would have varied had you drawn a different 50 counties: treat `my_counties` as if it were the population and resample from *it*, with replacement, **at size 50**. Do that 10,000 times, plot the distribution of these estimates, and report its standard deviation and its middle 95%.