For Loops

Contents

Simulating the Law of Large Numbers

The Law of Large Numbers says that as sample sizes increase, the mean of the sample will approach the true mean of the distribution. We are going to simulate this phenomenon!

We'll start by making a vector of sample sizes from 1 to 50, to represent increasing sample sizes.

Create a vector called `sample_sizes` that is made up of the numbers 1 through 50. (**Hint**: You can use `seq()` or `:` notation).

We'll make an empty tibble to store the results of the for loop:

```
estimates <- tibble(n = integer(), sample_mean = double())
```

Write a loop over the `sample_sizes` you specified above. In the loop, for each sample size you will:

1. Calculate the mean of a sample from the random normal distribution with mean = 0 and sd = 5.
2. Make an intermediate tibble to store the results
3. Append the intermediate tibble to your tibble using bind_rows().

```
set.seed(60637)
for (___ in ___) {
  # Calculate the mean of a sample from the random normal distribution with mean = 0 and sd = 5.
  ___ <- ___
  # Make a tibble with your estimates
  this_estimate <- tibble(n = ___, sample_mean = ___)
  # Append the new rows to your tibble
  ___ <- bind_rows(estimates, ___)
}
```

We can use ggplot2 to view the results. Fill in the correct information for the data and x and y variables, so that the `n` column of the `estimates` tibble is plotted on the x-axis, while the `sample_mean` column of the `estimates` tibble is plotted on the y-axis.

```
# your data goes in the first position
___ %>%
  ggplot(aes(x = ___, y = ___)) +
  geom_line()
```

1. As the sample size (n) increases, does the sample mean becomes closer to 0, or farther away from 0?

Rewrite the loop code without looking at your previous code and use a wider range of sample sizes. Try several different sample size combinations. What happens when you increase the sample size to 100? 500? 1000? Use the `seq()` function to generate a sensibly spaced sequence.

```
set.seed(60637)
sample_sizes <- ___
estimates_larger_n <- ___

for (___ in ___) {
  ___ <- ___
  ___ <- ___
  ___ <- ___
}

___ %>%
  ggplot(___(___ = ___, ___ = ___)) +
  geom_line()
```

1. How does this compare to before?

Extending Our Simulation

Looking at your results, you might think a small sample size is sufficient for estimating a mean, but your data had a relatively small standard deviation compared to the mean. Let's run the same simulation as before with different standard deviations.

Do the following:

1. Create a vector called `population_sd` of length 4 with values 1, 5, 10, and 20 (you're welcome to add larger numbers if you wish).
2. Make an empty tibble to store the output. Compared to before, this has an extra column for the changing population standard deviations.
3. Write a loop inside a loop over `population_sd` and then `sample_sizes`.
4. Then, make a ggplot graph where the x and y axes are the same, but we facet (aka we create small multiples of individual graphs) on `population_sd`.

```
set.seed(60637)
population_sd <- ___
# use what every you came up with in the previous part
sample_sizes <- ___
estimates_adjust_sd <- ___

for (___ in ___){
  for (___ in ___) {
    ___ <- ___
    ___ <- ___
    ___ <- ___
  }
}
```

```
___ %>%
  ggplot(___) +
  geom_line() +
  facet_wrap(~population_sd) +
  theme_minimal()
```

How do these estimates differ as you increase the standard deviation?

Making the code more efficient.

At this point in your coding career, efficiency is minimally important. However, we discussed that loops in R are much slower if we build our output item by item (or row by row in this case.) We do better if we preallocate space.

1. Rewrite your loops where we start with `estimates <- vector("list", n)`. This creates a list of length `n` "filled" with NULL.
2. During each iteration of your loop fill one of the spaces in `estimates` with a named vector (which we think of as a row.) Use `[[` to index into your list.
3. Finally, when your loop terminates, you'll have `estimates` filled with `n` "rows". `bind_rows(estimates)` will convert the list to a tibble.

Want to improve this tutorial? Report any suggestions/bugs/improvements on here! We're interested in learning from you how we can make this tutorial better.