Week 6 R Lecture

Jeremy Foote and Aaron Shaw

October 20, 2020

Contents

Analyzing categorical data in R

The goal of this script is to help you think about analyzing categorical data, including proportions, tables, chi-squared tests, and simulation.

Estimating proportions

If a survey of 50 randomly sampled Chicagoans found that 45% of them thought that Giordano’s made the best deep dish pizza, what would be the 95% confidence interval for the true proportion of Chicagoans who prefer Giordano’s?

Can we reject the hypothesis that 50% of Chicagoans prefer Giordano’s?

```
est = .45
sample_size = 50
SE = sqrt(est*(1-est)/sample_size)

conf_int = c(est - 1.96 * SE, est + 1.96 * SE)
conf_int
```

```
## [1] 0.3121018 0.5878982
```

What if we had the same result but had sampled 500 people?

```
est = .45
sample_size = 500
SE = sqrt(est*(1-est)/sample_size)

conf_int = c(est - 1.96 * SE, est + 1.96 * SE)
conf_int
```

```
## [1] 0.4063928 0.4936072
```

Tabular Data

The Iris dataset is composed of measurements of flower dimensions. It comes packaged with R and is often used in examples. Here we make a table of how often each species in the dataset has a sepal width greater than 3.

```r
data(iris)
table(iris$Species, iris$Sepal.Width > 3)
```

```
##             
##              FALSE TRUE
##   setosa         8   42
##   versicolor    42    8
##   virginica     33   17
```

As described in the most recent *OpenIntro* reading, the chi-squared (χ^2) test is a test of whether the frequencies we see in a table differ from what we would expect if there was no difference between the groups. Base-R provides a `chisq.test()` function to conduct these tests. Here's an example of the test evaluating whether the frequency of Iris sepal widths greater than 3 differs by species (the null hypothesis is that there is no difference in the frequencies and the groups are drawn from the same population). It's always good to examine the contingency table before conducting the test!

```r
table(iris$Species, iris$Sepal.Width > 3)
```

```
##             
##              FALSE TRUE
##   setosa         8   42
##   versicolor    42    8
##   virginica     33   17
```

```r
chisq.test(table(iris$Species, iris$Sepal.Width > 3))
```

```
## 
##  Pearson's Chi-squared test
## 
## data:  table(iris$Species, iris$Sepal.Width > 3)
## X-squared = 50.225, df = 2, p-value = 1.241e-11
```

While we know little about the dataset, sample, or measurement, the incredibly low p-value means that the observed frequencies likely did not come from the same distribution and that the frequency of sepal widths more/less than 3 differs by species.

Note that you can create tables to conduct χ^2 tests by hand. The `chisq.test()` function will need the object to be of "class" `table` or `matrix` in order to run the test.

BONUS: Using simulation to test hypotheses and calculate "exact" p-values

Section 6.1.4 of *OpenIntro* mentions a simulation-based approach to inference when assumptions of χ^2 tests aren't met. You possess all the tools to implement something like this in R, so let's do it. Basically, the idea is that we use simulated data to construct a null distribution of outcomes and then compare our observed test statistic against the null to estimate an "exact" p-value corresponding to the proportion of samples drawn from the null would produce statistics at least as large as our observed data.

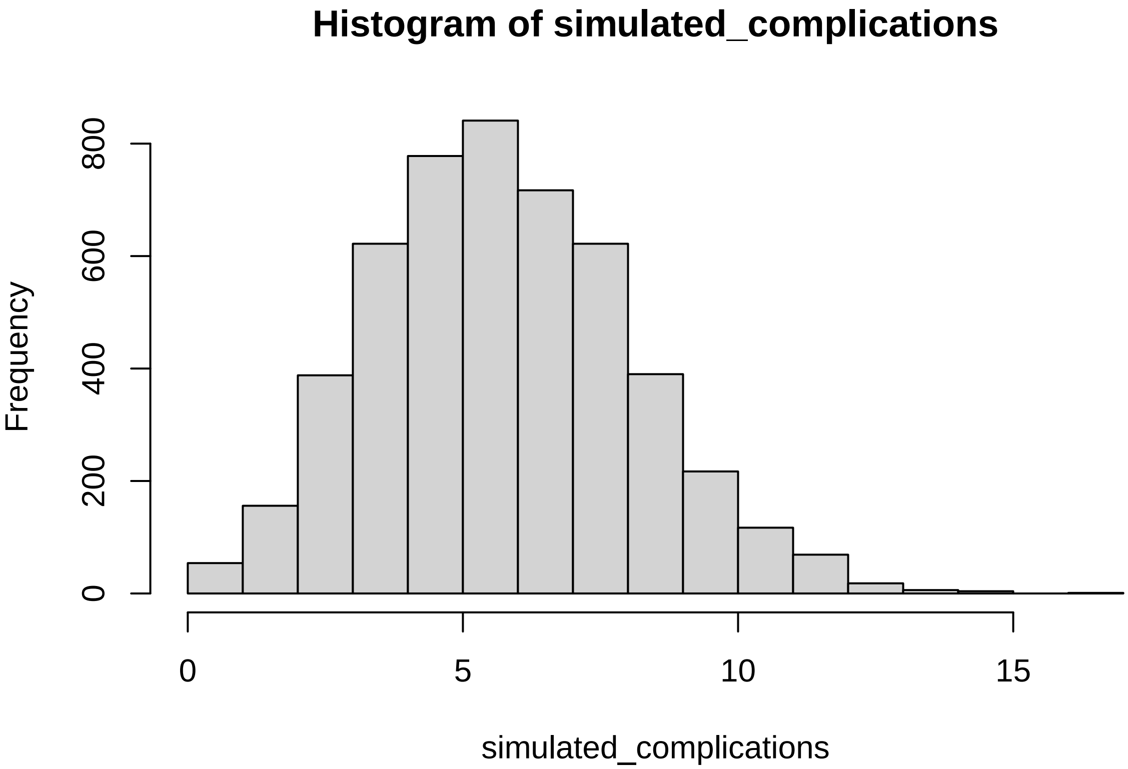
The earlier (3rd) edition of Chapter 6 of the *OpenIntro* textbook uses an example of a medical practitioner who has 3 complications out of 62 procedures, while the typical rate is 10%. We can use that information to create an example here based on the proportions. The null hypothesis is that this practitioner's true rate is also 10%, so we're trying to figure out how rare it would be to have 3 or fewer complications, if the true rate is 10%.

```r
# We write a function that we are going to replicate
simulation <- function(rate = .1, n = 62){
  # Draw n random numbers from a uniform distribution from 0 to 1
  draws = runif(n)
  # If rate = .4, on average, .4 of the draws will be less than .4
  # So, we consider those draws where the value is less than `rate` as complications
  complication_count = sum(draws < rate)
  # Then, we return the total count
  return(complication_count)
}

# The replicate function runs a function many times
simulated_complications <- replicate(5000, simulation())
```

We can look at our simulated complications

```
hist(simulated_complications)
```



And determine how many of them are as extreme or more extreme than the value we saw. This is the "exact" p-value.

```
sum(simulated_complications <= 3)/length(simulated_complications)
```

```
## [1] 0.1196
```