Present: Sophia, Caleb, Noah, Garrett, Anna J, Natalie F., Natalie V., Josiah, Elizabeth, Valeria

Test for Association Between Categorical Variables

Thomas Scofield

November 12, 2021

Testing for an Association Between Two Categorical Variables

The bare necessities:

- We are testing the hypothesis that two categorical variables are **independent** ($\mathbf{H}_0$) vs. the hypothesis that they are associated ($\mathbf{H}_a$)
- For two binary categorical variables, this is the same thing (though different techniques are used) as testing $p_1 - p_2 = 0$ vs $p_1 - p_2 \neq 0$.
- The test statistic is the χ^2-statistic, still found using formula

$$\chi^2 \;=\; \sum_{\text{cell}(i,j)} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}.$$

  Here, $O_{i,j}$ is the observed count in cell (i,j) (row i, column j) of the two-way table, and $E_{i,j}$ refers to the expected count for that same cell.
- Each expected count is computed as

$$E_{i,j} = \frac{(\text{total of row } i)(\text{total of column } j)}{\text{grand total}}.$$

Prominent R commands used in these notes

`rbind()`, for entering a two-way table directly
`rownames()`, to specify labels for the rows of a two-way table
`colnames()`, to specify labels for the columns of a two-way table
`tally()`, to create two-way tables from raw data involving 2 categorical variables
`addmargins()`, calculates row/column totals of table and adds them to the display
`pchisq()`, for a given cutoff and dfs, calculates area to the left of that cutoff
`qchisq()`, for a given number of dfs, finds a specified quantile
`chisq.test()`, to conduct, from start-to-finish, all calculations of a χ^2-test
`chisq.test()$statistic`, like above, but outputs just the χ^2-statistic
`chisq.test()$expected`, like above, but outputs just a table of expected counts

Creating two-way tables in R

Sometimes you have **raw data**, like that found in the `Lock5withR` dataset called **WaterTaste** (the subject of Example 7.10 on p. 481). So long as you have the `mosaic` package loaded, you can get a two-way (or **contingency**) table for categorical variables `UsuallyDrink` and `First` using ‘tally():

requires raw data

```
tally(UsuallyDrink ~ First, data=WaterTaste)
```

```
##              First
## UsuallyDrink Aquafina Fiji SamsChoice Tap
##     Bottled        14   15          8   4
##     Filtered        4   10          9   3
##     Tap             7   16          7   3
```

Piping the result to `addmargins()` results in the same table complete with row and column totals.

```
tally(UsuallyDrink ~ First, data=WaterTaste) %>% addmargins()
```

```
##              First
## UsuallyDrink Aquafina Fiji SamsChoice Tap Sum
##     Bottled        14   15          8   4  41
##     Filtered        4   10          9   3  26
##     Tap             7   16          7   3  33
##     Sum            25   41         24  10 100
```

Still the contingency table having 12 cells — the marginal sums are not part of the table.

At other times, data may be presented already **summarized as a two-way table**. A case in point is Example 7.9 on the first page of Section 7.2. (Open your book to look that data over.) You can create a table using `rbind()`:

```
table7.19 <- rbind(c(372,363), c(807,1005), c(34,44))
table7.19
```

```
##      [,1] [,2]
## [1,]  372  363
## [2,]  807 1005
## [3,]   34   44
```

The table isn't as informative as it would be if it also included labels. To add these, making it appear more like it does in the text, we use these subsequent commands:

```
rownames(table7.19) <- c("Agree","Disagree","Undecided")
colnames(table7.19) <- c("Male","Female")
table7.19 %>% addmargins()
```

```
##           Male Female  Sum
## Agree      372    363  735
## Disagree   807   1005 1812
## Undecided   34     44   78
## Sum       1213   1412 2625
```

When variables are independent: context for computing expected counts

If two variables are *not* associated, then they are *independent*. How does a two-way table for independent variables look?

Example. Consider the plate appearances (batting opportunities) for the baseball player Derek Jeter. You see, below, an incomplete table, one where you only know the marginal distributions (i.e., the sums of each row and column). In his career in Major League Baseball, Jeter faced a left-handed pitcher 2,863 times, a right-handed pitcher 8,332 times. In 11,195 plate appearances, he got a hit 3,465 times, and failed to get a hit the other 7,730 times.

Exercise:

Supply values made up by you in the empty cells of the table, cells that ought to include the number of hits Jeter had off of a right-hand pitcher, the number off a left-hand pitcher, etc. Do this in a manner that

- is consistent with the marginal distributions, and
- makes it so Jeter did precisely as well against right-hand pitchers as he did against left-hand pitchers.

	vs. LHP	vs. RHP	Total
Hits	$E_{1,1}$	$E_{1,2}$	3465
Misses	$E_{2,1}$	$E_{2,2}$	7730
Total	2863	8332	11195

$$\frac{E_{1,1}}{2863} = \frac{3465}{11195} \implies E_{1,1} = \frac{(2863)(3465)}{11195}$$

$$E_{2,1} = \frac{(2863)(7730)}{11195}$$

Any (i,j)-cell $E_{i,j} = \frac{(\text{col } j \text{ total})(\text{row } i \text{ total})}{\text{grand total}}$

Exercise:

For the contingency table we called `table7.19` (see above)

```
##           Male Female  Sum
## Agree      372    363  735
## Disagree   807   1005 1812
## Undecided   34     44   78
## Sum       1213   1412 2625
```

Male	Female
$\boxed{}$	$E_{1,2}$
$E_{2,1}$	$E_{2,2}$
$E_{3,1}$	$E_{3,2}$

$$E_{1,1} = \frac{(1213)(735)}{2625}$$

calculate a table of expected values, and the corresponding χ^2-statistic.

Write out the two-way table involving the `UsuallyDrink` and `First` variables from the Lock 5 data set **WaterTaste**. This table contains the observed counts in eight cells. Then, next to that table, draw another table with the same number of cells, but for these cells calculate the expected counts. Use your numbers from the two tables to calculate a χ^2-statistic.

Do on your own.

Randomization distributions

As usual, we need a P-value, which is computed based on where the test stastistic in the null distribution. That is the culmination of this hypothesis test, called the **chi-square test for association**. A natural way to approximate this null distribution is to use randomization.

Constructing a randomization distribution physically. We have already discussed, back in Chapter 4, how to do this when both categorical variables are *binary*. The process followed is very similar here.

Exercise:

Write a description of how you would generate *one* randomization statistic, in the case of the "One true love" data above, using only bags and slips of paper.

	Male	Female	Sum
Agree	372	363	735
Disagree	807	1005	1812
Undecided	34	44	78
Sum	1213	1412	2625

How one carries out a randomization sample for bivariate binary categorical data is described in Example 4.28. This description is almost the same for nonbinary categorical data, though calculating the randomization statistic is quite different here, as we need to compute χ^2, not $\hat{p}_1 - \hat{p}_2$, from our randomization sample. Here we would place 735, 1812, and 78 slips in one bag labeled "Agree", "Disagree", "Undecided". In another bag, 1213 and 1412 slips labeled "Male", "Female". For one randomization sample begin drawing (w/out replacement) simultaneous slips from both bags and identifying the two together as if for one case. Draw until both bags are empty. Produce the corresponding two-way table and χ^2-statistic.

For those interested in generating a randomization distribution using R, see details below in the section called "Using R to generate a randomization distribution". For now, let's use StatKey, noting that its randomization samples have different observed counts than the original sample, but have the same marginal totals.

Getting a P-value using a theoretical distribution.

As with the goodness-of-fit testing of the previous section, there is a rule of thumb which says we can turn to a theoretical χ^2 distribution instead of a randomization distribution to calculate the approximate P-value. Once again, the **rule validating the use of a theoretical χ^2-distribution** is

Every cell has an expected count of at least 5.

When it is justified, we choose the degrees of freedom to be

$$df = [(\# \text{ of rows}) - 1][(\# \text{ of columns}) - 1]$$

When the table is 3 rows, 2 columns and you know the row/column sums, you could be told just two cells' observed counts and be able to fill in the rest, like in the picture where 372, 44 are provided. This is a manifestation of $df = (3-1)(2-1) = 2$.

Exercise:

For the "One true love" data,

a. verify that it is appropriate to consult a theoretical χ^2-distribution.
b. use the `pchisq()` command to find the approximate P-value.

372		735
		1812
	44	78
1213	1412	

The all-in-one command: `chisq.test()`

For `chisq.test()` to give us meaningful calculations, it must be provided with the relevant contingency table.

```
chisq.test( table7.19 )
```

```
##
##  Pearson's Chi-squared test
##
## data:  table7.19
## X-squared = 7.9878, df = 2, p-value = 0.01843
```

What did `chisq.test()` do here? Pretty much everything, including

- computed the expected counts
- computed the χ^2-statistic
- used the theoretical χ^2 distribution to compute a P-value

Some of this was reported, other things we must ask for. For instance, to get just the χ^2-statistic, we can add "`$statistic`" to the end of the command:

```
chisq.test( table7.19 )$statistic
```

```
## X-squared
##  7.987828
```

```
#  chisq( tally(UsuallyDrink ~ First, data=WaterTaste) )   # this also works
```

To view the expected counts, we replace "`$statistic`" with "`$expected`"

```
chisq.test( table7.19 )$expected
```

```
##                Male    Female
## Agree     339.64000 395.36000
## Disagree  837.31657 974.68343
## Undecided  36.04343  41.95657
```

Note: If we use this command in cases where the expected counts do not satisfy the rule-of-thumb, we receive a warning. Try out

```
chisq.test( tally(UsuallyDrink ~ First, data=WaterTaste) )
```

Using R to generate a randomization distribution

A randomization distribution for the χ^2-statistic of a contingency table is one of the more challenging randomization distributions we might undertake in RStudio.

Raw data: Let's first suppose you have it, like the data provided in the **WaterTaste** data frame. Generating a randomization sample in such a setting isn't so bad; we simply select the two columns of interest, and shuffle one of them. What is more, you *can* use the `chisq.test()` command on that randomization sample to produce a randomization statistic. It would go like this:

```
chisq.test( tally( UsuallyDrink ~ shuffle(First), data=WaterTaste ) )$statistic
```

```
## Warning in chisq.test(tally(UsuallyDrink ~ shuffle(First), data = WaterTaste)):
## Chi-squared approximation may be incorrect
```

```
## X-squared
## 0.6954914
```

But the output produces a warning that is not so easily suppressed. What would help tremendously is to have a command that behaves like `chisq.test()$statistic` does, without producing the warning. I don't know of one existing already, but R is a programming language, and it isn't that hard to produce a rudimentary version that suits our needs. The sole purpose of the code in the next cell is to add a new command, called `chisq.stat()`, to our arsenal, tailored to our needs.

```
chisq.stat <- function(observedTable){
  expectedTable <- (apply(observedTable,1,sum) %o% apply(observedTable, 2, sum)) /
                     sum(observedTable)
  sum((observedTable - expectedTable)^2 / expectedTable)
  }
```

The sole purpose of `chisq.stat()` is to compute from a two-way table of observed counts the corresponding χ^2-statistic. Acting on the two-way table of original data in **WaterTaste**, it gives us our test statistic free of warning:

```
chisq.stat( tally(UsuallyDrink ~ First, data=WaterTaste) )
```

```
## [1] 4.972506
```

Shuffle the `First` column so as to produce a randomization sample, and `chisq.stat()` will produce one randomization statistic:

```
chisq.stat( tally(UsuallyDrink ~ shuffle(First), data=WaterTaste) )
```

```
## [1] 3.231574
```

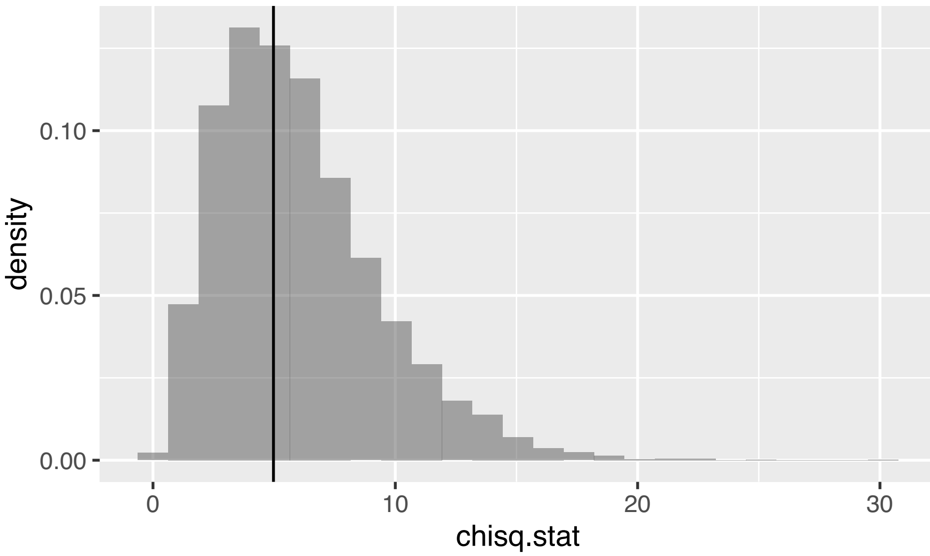
It's this we will repeat many times to simulate a randomization distribution, simulating the null distribution (the distribution of χ^2-statistics) when the null hypothesis of "independence of variables" holds:

```
manyRandomizedChisqStats <- do(5000) *
  chisq.stat( tally(UsuallyDrink ~ shuffle(First), data=WaterTaste) )
head(manyRandomizedChisqStats)
```

```
##   chisq.stat
## 1   3.736576
## 2  10.983149
## 3   6.445042
## 4   5.200464
## 5   5.360056
## 6   2.556962
```

The resulting histogram should look much like the randomization distribution obtained in Statkey.

```
gf_dhistogram( ~chisq.stat, data=manyRandomizedChisqStats ) %>%
  gf_vline(xintercept = ~4.9725)
```



Our approximate P-value corresponds the the relative frequency of randomization statistics at least as high or higher than our test statistic of 4.9725. We can determine this by counting occurrences:

```
nrow( filter(manyRandomizedChisqStats, chisq.stat >= 4.9725) ) / 5000
```

```
## [1] 0.5646
```

Summarized data: Now suppose you have *only* the contingency table, not the raw data that generated it, as with Table 2.3 on p. 50. The simplest approach—the one we took with two binary categorical variables back in Chapter 4—is to generate a raw data set from the information in the table. Again, it would be nice if there were an R command that did the tedious generation of rows and used `rbind()` to put those rows together. I do not know of such a command native to R, but the code below produces one called `dataFromTwoWayTable()`.

```
dataFromTwoWayTable <- function(conTable, rowVarName, colVarName){
  myDat <- data.frame()
  rNames = rownames(conTable)
  cNames = colnames(conTable)
  for (ii in 1:nrow(conTable)) {
   for (jj in 1:ncol(conTable)) {
     myDat <- rbind(myDat, do(conTable[ii,jj]) * data.frame(x=rNames[ii], y=cNames[jj]))
   }
  }
  myDat <- subset(myDat, select=c(x,y))
  colnames(myDat) <- c(rowVarName, colVarName)
  return(myDat)
 }
```

It is unpredictable what this command will do if you send it the wrong kind of information (that makes it *brittle*, as can happen when emphasis has been placed on programming quickly over conscientiously). What it expects is a contingency table with column and row headers that correspond to two variables, followed by the *name* of the row variable, followed by the *name* of the column variable. It should work just fine on `table7.19`:

```r
table7.19
```

```
##           Male Female
## Agree      372    363
## Disagree   807   1005
## Undecided   34     44
```

```r
rawTable7.19dat <- dataFromTwoWayTable(table7.19, "opinion", "sex")
head(rawTable7.19dat)
```

```
##   opinion  sex
## 1   Agree Male
## 2   Agree Male
## 3   Agree Male
## 4   Agree Male
## 5   Agree Male
## 6   Agree Male
```

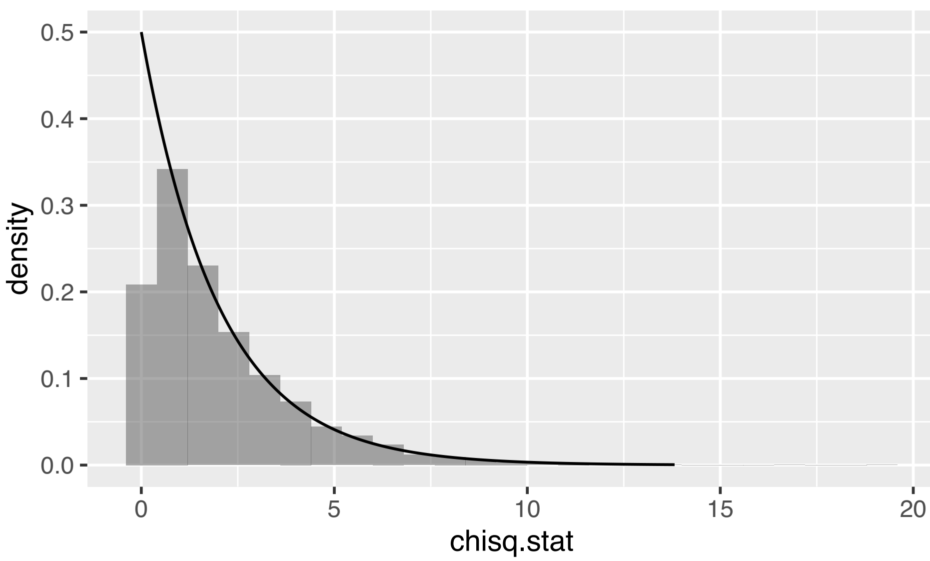
With `rowTable7.19dat` we can do steps like before to produce a randomization distribution:

```r
manyRandomizedChisqStats <- do(5000) *
  chisq.stat( tally(opinion ~ shuffle(sex), data=rawTable7.19dat) )
head(manyRandomizedChisqStats)
```

```
##   chisq.stat
## 1 3.63547704
## 2 0.18462945
## 3 0.05790213
## 4 1.85636116
## 5 4.56103779
## 6 0.13570693
```

Then, perhaps a comparison between a histogram of the randomization distribution and the theoretical chi-square distribution with `df=2`. Since all cells have expected counts at least 5, we expect they should be similar.

```r
gf_dhistogram( ~chisq.stat, data=manyRandomizedChisqStats ) %>%
  gf_dist("chisq", df=2)
```



The lesson here is that with a little programming ability, or with access to commands like `chisq.stat()` and `dataFromTwoWayTable()` whose programs were given above, the randomization distribution for χ^2 for a contingency table is made simpler than it would otherwise be in RStudio. It is seemingly simpler still in StatKey, where the sticking point is to make sure it has access to the data you want in a usable form.