Test for Association Between Categorical Variables

For Lock 5: 7.2

Thomas Scofield

April 28, 2020

Testing for an Association Between Two Categorical Variables

Two-way tables

Recall that a way to summarize bivariate categorical data is to construct a **two-way table**, also known as a **contingency table**. The `Lock5withR` data set called **WaterTaste** is the subject of Example 7.10 on p. 481.

```
head(WaterTaste)
```

```
##   Gender Age Class UsuallyDrink FavBotWatBrand Preference First     Second
## 1      F  18     F     Filtered      DEER PARK       CABD  Fiji SamsChoice
## 2      F  18     F          Tap           NONE       CABD  Fiji SamsChoice
## 3      F  18     F          Tap      DEER PARK       CADB  Fiji SamsChoice
## 4      F  18     F          Tap       AQUAFINA       CBDA  Fiji   Aquafina
## 5      M  19     J     Filtered           NONE       CBAD  Fiji   Aquafina
## 6      M  19     J     Filtered          SAM'S       CABD  Fiji SamsChoice
##        Third     Fourth    Sex
## 1   Aquafina        Tap Female
## 2   Aquafina        Tap Female
## 3        Tap   Aquafina Female
## 4        Tap SamsChoice Female
## 5 SamsChoice        Tap   Male
## 6   Aquafina        Tap   Male
```

It has a number of categorical variables, two being `UsuallyDrink` and `First`. We view the contingency table for these two variables using any one of the commands

```
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
```

We can have row and sum totals if we pipe this two-way table to the `addmargins()` command:

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

Pipe

```
##               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
```

12 cells

A two-way table is a likely first step toward seeing if two variables have an association. But what, exactly, should we be looking for?

Independent categorical variables

opposites

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

Ha

H₀

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.

Want this, so

$$\boxed{\frac{E_{1,1}}{2863} = \frac{3465}{11195}} = \frac{E_{1,2}}{8332}$$

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

	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

Can get $E_{2,1}$ either by same method

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

or by subtraction

$$E_{2,1} = 2863 - E_{1,1}$$

Answer:

In the above exercise, you were asked to come up with numbers for the empty cells so that Jeter's success in plate appearances is independent of the *handedness* of the pitcher he faced. The numbers you come up with are generally not the numbers observed in the sample (Here, I am viewing the record of at-bats in Jeter's career as a sample of all *possible* outcomes of his facing pitchers in Major League Baseball); they usually aren't even integers. But they serve as benchmarks for comparison, **expected counts** arising from a null hypothesis that the variables are independent. When we use these and the observed counts to calculate a χ^2-statistic, it will once again be the case that larger values of χ^2 correspond to stronger evidence against the null hypothesis. If our χ^2 (test) statistic is statistically significant, we will reject the null hypothesis ("the variables are independent") in favor of the alternative hypothesis: "the two variables have an association."

The expected counts, those that reflect perfect independence of the variables, are calculated via the formula:

$$(\text{expected count})_{\text{row } i, \text{ column } j} = \frac{(\text{total of row } i) \times (\text{total of column } j)}{(\text{grand total})}.$$

Exercise:

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.

Randomization distributions

As usual, we need a P-value, which is computed based on where the test stastistic is in the null distribution. The most natural thing is to approximate this null distribution via 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. Read up on it. Then use StatKey and observe that its randomization samples have different observed counts than the original sample, but have the same marginal totals.

Exercise:

Write a description of how you would generate *one* randomization statistic, in the case of variables `UsuallyDrink` and `First` in the **WaterTaste** data, using only bags and slips of paper.

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. As before, the **rule validating the use of a theoretical χ^2-distribution** is

Every cell has an expected count of at least 5.

Exercise:

In StatKey, go to the "χ^2 Test for Association". From the drop-down menu, select data for "Homes for Sale (Size by State)". Do not generate any randomization samples. Instead, look at "Show Details" to convince yourself the rule of thumb is met. Then

a. From the screen determine the test statistic.
b. Determine the number of degrees of freedom using the formula

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

c. Use a theoretical χ^2-distribution calculator or `pchisq()` to obtain the approximate P-value.

Computations in RStudio

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

For `chisq.test()` to give us meaningful calculations, it must be provided with the relevant contingency table. The **WaterTaste** data comes to you as raw data, not as a table. In such an instance, we build the table and send it to `chisq.test()`.

```
tally(UsuallyDrink ~ First, data=WaterTaste)     # builds the two-way table
```

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

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

```
##
##  Pearson's Chi-squared test
##
## data:  tally(UsuallyDrink ~ First, data = WaterTaste)
## X-squared = 4.9725, df = 6, p-value = 0.5473
```

These two commands both calculate χ^2 for the input two-way table, and both give warnings on this data.

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

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

Warning is repeated on each instance of chisq.test( ) using this data

```
## X-squared
##  4.972506
```

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

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

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

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

##             First
## UsuallyDrink Aquafina  Fiji SamsChoice Tap
##     Bottled     10.25 16.81       9.84 4.1
##     Filtered     6.50 10.66       6.24 2.6
##     Tap          8.25 13.53       7.92 3.3
```

The warning message we are getting is explained by the expected counts. There is a cell (more than one, in fact) whose expected count is less than 5. So, while the command gets its P-value from a theoretical χ^2-distribution, we should not trust it. We can put more faith in a P-value obtained via randomization. (For how to use RStudio to do that, see below).

The **WaterTaste** data frame supplied raw data. But sometimes we encounter bivariate categorical data already summarized in a two-way table. An example might be the "one-true-love" data in Table 2.3, p. 50. You can enter this two-way table into R using the commands

```
table2.3 <- rbind( c(372, 363), c(807, 1005), c(34, 44) )
table2.3
```

— This builds a contingency table directly from observed counts, but gives no row or column headings.

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

Exercise: Try building the same table using cbind( ).

You see the observed counts appear in cells in the correct positions, so the `chisq.test()` command has what it needs.

```
chisq.test(table2.3)
```

Doesn't produce any warning using this table. Why not?

```
##
##  Pearson's Chi-squared test
```

```
##
## data:  table2.3
## X-squared = 7.9878, df = 2, p-value = 0.01843
```

There are no warnings, because the expected counts are all larger than 5.

```
chisq.test(table2.3)$expected
```

```
##             [,1]      [,2]
## [1,] 339.64000 395.36000
## [2,] 837.31657 974.68343
## [3,]  36.04343  41.95657
```

If you want you can add headers which help keep the meaning of cells straight, doing this using `colnames()` and `rownames()` commands:

```
colnames(table2.3) <- c("Male", "Female")
rownames(table2.3) <- c("Agree", "Disagree", "Don't know")
table2.3
```

} gives row/column headers to the table built above

```
##            Male Female
## Agree       372    363
## Disagree    807   1005
## Don't know   34     44
```

Finally, if you know the test statistic, the number of degrees of freedom, and that it is safe to use a theoretical chi-square distribution, you can use the `pchisq()` command. For instance, for the Table 2.3 data, having χ^2 ~~chi^2~~ $= 7.9878$ and `df = 2`, the P-value is approximately

```
1 - pchisq(7.9878, df=2)
```

```
## [1] 0.01842771
```

Using RStudio 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
##  4.401549
```

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
```

— test stat: χ^2 from original 2-way table

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] 7.45602
```

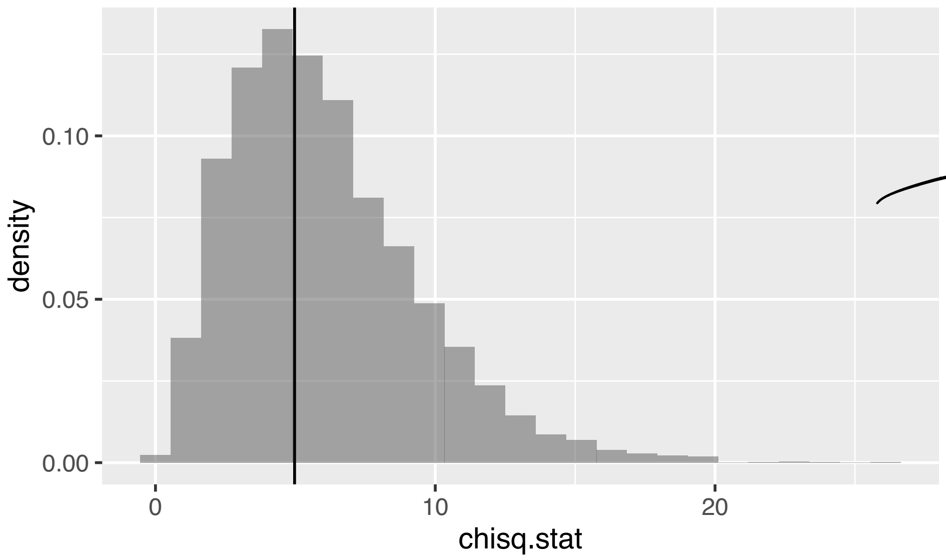
It's this we will repeat many times to simulate a randomization distribution, simulating the null distribution (the distribution of χ^2-statistics) when the null hypothesis of "independence of variables" holds:

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

```
##   chisq.stat
## 1   7.760171
## 2   3.941085
## 3   7.809024
## 4   2.205436
## 5   2.963984
## 6   4.961785
```

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

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



Randomization distribution for variables: "Usually Drink" and "First" from WaterTaste

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:

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

```
## [1] 0.5668
```

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()`.

```r
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 `table2.3`:

```r
table2.3
```

```
##             Male Female
## Agree        372    363
## Disagree     807   1005
## Don't know    34     44
```

```r
rawTable2.3dat <- dataFromTwoWayTable(table2.3, "opinion", "sex")
head(rawTable2.3dat)
```

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

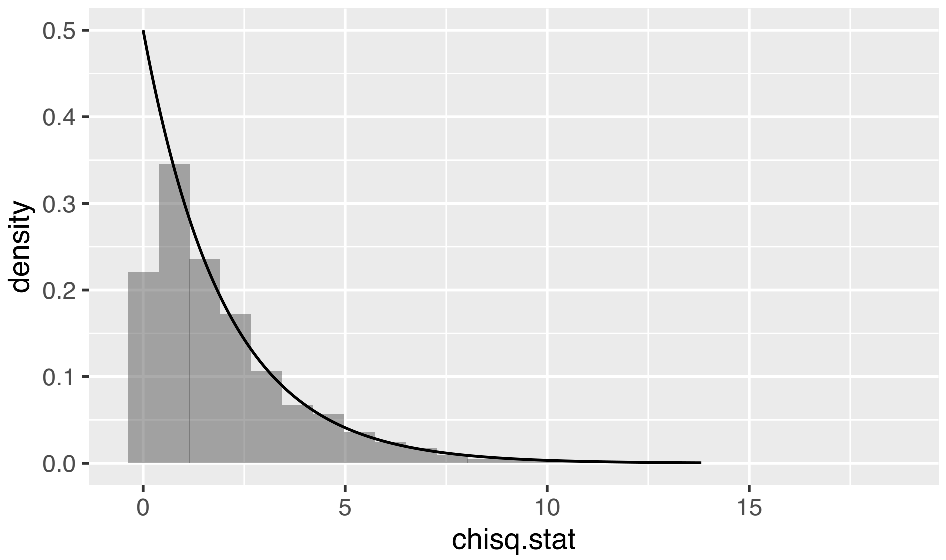
With `rowTable2.3dat` we can do steps like before to produce a randomization distribution:

```r
manyRandomizedChisqStats <- do(5000) *
  chisq.stat( tally(opinion ~ shuffle(sex), data=rawTable2.3dat) )
head(manyRandomizedChisqStats)
```

```
##   chisq.stat
## 1 0.09049019
## 2 4.63367916
## 3 0.30222057
## 4 1.46943684
## 5 0.74305656
## 6 0.64944222
```

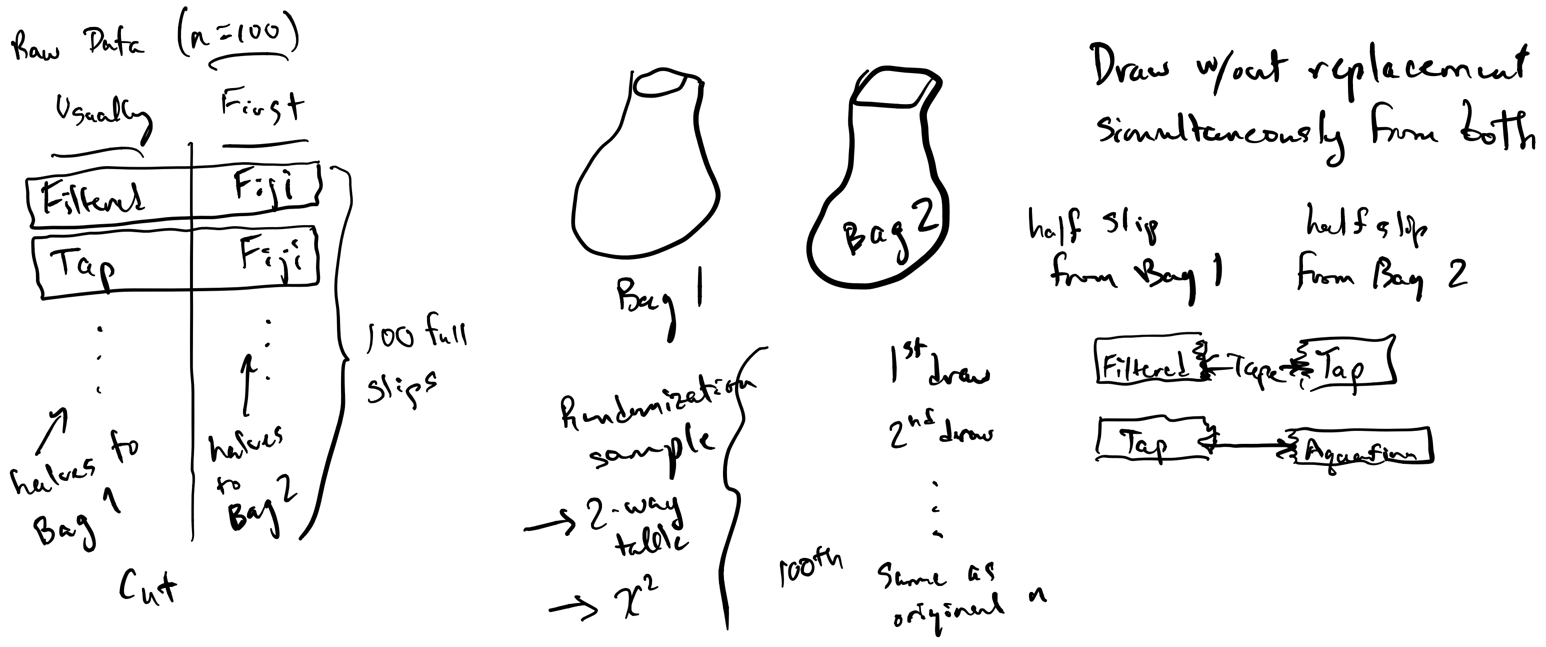
Then, perhaps a comparison between a histogram of the randomization distribution and the theoretical chi-square distribution with `df=2`. Since all cells have expected counts at least 5, we expect they should be similar.

```
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.

Physically selecting randomization sample for Water Taste


Raw Data (n=100)
Usually
First
Filtered
Fiji
Tap
Fiji
halves to Bag 1
halves to Bag 2
Cut
100 full slips
Bag 1
Bag 2
Randomization sample
→ 2-way table
→ χ^2
1st draw
2nd draw
100th
Same as original n
Draw w/out replacement simultaneously from both
half slip from Bag 1
half a slip from Bag 2
Filtered
Tap
Tap
Tap
Aquafina