6 Hypothesis Testing on Non-Binary Categorical Variables

Overview

In previous modules have learned how to deal with a question like, "What is the proportion of people who are left-handed?," where relevant (sample of 20 subjects) data might look like this:

```
handDat <- tibble(row=c(1:20), dominantHand=sample(c("left","right"),size=20,prob=c(.1,.9),replace=TRUE)
kable(handDat[1:6,]) %>%
  kable_styling(position="center", full_width=FALSE)
```

row	dominantHand
1	right
2	right
3	left
4	right
5	right
6	right

A similar sort of question is "What proportion of people have blood type 'O'?" for which relevant data might look like this

```
bloodTypes <- read_csv("http://scofield.site/teaching/data/csv/blood.csv")[,1]
kable(sample(bloodTypes)[1:8,1]) %>%
  kable_styling(position="center", full_width=FALSE)
```

type
B
O
O
A
A
O
O
O

Both questions involve a proportion of interest, based on values from a single categorical variable, `dominantHand` in the first instance, and `type` (for blood type) in the second.

The blood type data points to a limitation in the 1-proportion approach. The `blood type` variable is *nonbinary*—it has more than 2 values, does more than simply answer `yes` or `no` to whether a subject has blood type 'O'. With data such as this—data for a categorical variable with three or more values—we could imagine asking subtler questions, like "Are the four blood types equally-likely?"

Or, consider this data from a package of M&M's:

A possible notion / hypothesis

H_0: $p_{Bl}=1/6$, $p_{Br}=1/6$, $p_G=1/6$, $p_O=1/6$, $p_R=1/6$, $p_Y=1/6$

frequency table

observed counts

color	Freq
blue	13
brown	7
green	12
orange	9
red	7
yellow	8

arises from raw data

case	color
1	Blue
2	Y
3	O
⋮	⋮
400	Brown

This data comes from Pack #1 at https://joshmadison.com/2007/12/02/mms-color-distribution-analysis-individual-pack-data/. While the Mars candy company makes no claim that these colors from a standard pack of M&M's are equally-likely, they might claim the colors are produced in various specified proportions. Our former methods do not offer us a straightforward of checking whether sample data (like from this pack of M&M's) offers evidence significant to doubt their claimed proportions. Our first task in this module is to remedy this with a new hypothesis test, called a **goodness-of-fit** test.

Whatever the proportions of M&M's, we might also wish to ask whether they are found in the same proportions in Canada as they are in the U.S. This is a question about association between two categorical variables, `M&M color` (response) and `country` (explanatory). We have addressed questions of that sort using 2-proportion inferential procedures, but they come with a built-in limitation that both explanatory and response variables must be binary. Later in the module we use an approach similar to the goodness-of-fit test to address whether associations exist between two nonbinary categorical variables (ones, for example, like `country` and `M&M color`).

Learning outcomes

The student completing this unit should be able to

- Identify 1- and 2-categorical-variable settings which are non-binary, and call for a different approach from previous units.
- State hypotheses as suggested in a research question surrounding the distribution of a single categorical variable.
- Compute expected counts for the various cells of the table.
- Compute the χ^2-statistic from sample data in both 1- and 2-categorical-variable instances.
- Desribe how, using tangible objects like cards, slips of paper, well-shaken bags, etc., one can produce a resampled χ^2-statistic.
- Use software to generate many resampled χ^2-statistics, view the resulting distribution, and use it to calculate a P-value.
- Recognize when a theoretical distribution can give an accurate P-value, and identify which theoretical distribution.
- Rightly interpret a statistically significant result.

Univariate categorical data

What do you expect from a fair coin? Say you a coin 50 times—i.e., take a sample of size $n = 50$ of its values? The raw data, arranged horizontally here for considerations of space, might appear as

$$\text{H},\ \text{H},\ \text{T},\ \text{H},\ \text{T},\ \text{H},\ \ldots,$$

providing 50 observations in all. We are dealing with a categorical variable, `coinFlip`, which has $k = 2$ possible values (making it *binary*). A frequency table would reveal how often each value occurred in our sample of 50. Would it surprise you if your sample had this frequency table?

c.rep..H...23...rep..T...27..	Freq
H	23
T	27

Between that frequency table and this next one, which one would surprise you more?

c.rep..H...34...rep..T...16..	Freq
H	34
T	16

Most likely you would find the second table more surprising. Perhaps the reason is is that we tend to suppose we have a fair coin, with the corresponding assertion, written here as a null hypothesis, that

$$\mathbf{H}_0 \colon p_H = p_T = 0.5,$$

where these two symbols represent population parameters, the proportion of `heads` this coin would produce, p_H, and the proportion p_T of `tails` the coin has in it. In some sense, in 50 flips we "expect"

$$(50)(0.5) = 25 \text{ heads, and } (50)(0.5) = 25 \text{ tails.}$$

Not that any of us really expects to get exactly 25 heads and 25 tails, but these numbers serve as a benchmark against which we compare actual results. And the results of the second table are, frankly, farther from these benchmarks than in the first table.

Expected counts for other scenarios.

Univariate Categorical Data: Goodness-of-Fit

The context for goodness-of-fit testing

We apply goodness-of-fit tests when

- considering univariate categorical data with k values. When $k = 2$ the variable under consideration is *binary*, and *can* be treated using 1-proportion procedures from earlier chapters.
- we want to test the strength of evidence against certain predisposed notions about the probability distribution of the variable. This predisposition, when written as a null hypothesis, take the form

  $$\mathbf{H}_0 \colon p_1 = p_{10}, \;\; p_2 = p_{20}, \;\; \ldots, \;\; p_k = p_{k0}.$$

  Here we have numbered the different values of the value 1, 2, ..., k, the p_i stand for the relative frequency (proportion) of value i in the population, and p_{i0} stands for the null value for p_i, the number we "believe" it to be. Note that, for these null values to make sense, we should have

  $$p_{10} + p_{20} + \cdots + p_{k0} = 1.$$

  For example, in rolls of a fair die, we have $k = 6$ possible values and would take as our null hypothesis

  $$\mathbf{H}_0 \colon p_1 = \frac{1}{6}, \;\; p_2 = \frac{1}{6}, \;\; p_3 = \frac{1}{6}, \;\; p_4 = \frac{1}{6}, \;\; p_5 = \frac{1}{6}, \;\; p_6 = \frac{1}{6}.$$

  So, the null values p_{10}, p_{20}, p_{30}, p_{40}, p_{50}, p_{60} are all 1/6. In a setting like simple genetics where there are 4 different expressions, D, C, B and A which occur in ratios of 1:3:3:9, we would have

  $$\mathbf{H}_0 \colon p_A = \frac{9}{16}, \;\; p_B = \frac{3}{16} = p_C, \;\; p_D = \frac{1}{16},$$

  so, the null values are 9/16, 3/16, 3/16 and 1/16 respectively (and we might refer to them as p_{A0}, p_{B0}, p_{C0}, p_{D0}).

In raw data form, sample data to test the hypothesis might look like it does in the Lock 5 data set **APMultipleChoice**:

```
head(APMultipleChoice)
```

```
##   Answer
## 1      B
## 2      B
## 3      D
## 4      A
## 5      E
## 6      D
```

We obtain a frequency table using the command

```
tally(~Answer, data=APMultipleChoice)
```

```
## Answer
##  A  B  C  D  E
## 85 90 79 78 68
```

There are five cells, corresponding to the $k = 5$ possible responses to questions on an AP exam. One would tend to believe that officials involved in authoring test questions would want each option A–E to be equally likely, occurring one-fifth of the time. That means for each test option, we propose a null value of 0.2:

$$\mathbf{H}_0 : p_A = \frac{1}{5},\ \ p_B = \frac{1}{5},\ \ p_C = \frac{1}{5},\ \ p_D = \frac{1}{5},\ \ p_E = \frac{1}{5}.$$

Quick question:

What, logically, should be the alternative hypothesis?

Now, for computations in RStudio, let's create some useful vectors:

```
sampleSize = 400
observed = tally(~Answer, data=APMultipleChoice)
nullVals = rep(0.2, 5); nullVals
```

```
## [1] 0.2 0.2 0.2 0.2 0.2
```

```
expected = sampleSize * nullVals; expected
```

```
## [1] 80 80 80 80 80
```

Our test statistic, called χ^2, is computed according to the formula

$$\chi^2 = \sum_i \frac{[(i\text{th observed value}) - (i\text{th expected value})]^2}{(i\text{th expected value})}.$$

Using the vectors above, this is

```
sum( (observed - expected)^2 / expected )
```

```
## [1] 3.425
```

Randomization to simulate the null distribution of χ^2

To physically produce one randomization sample, we might have a bag containing slips of paper, where each slip has a single value of the variable written on it, and the proportion of slips containing the same value matches that stated in the null hypothesis. In the case of our **APMultipleChoice** data, this would mean our bag had precisely 20% of its slips containing the letter "A", 20% containing "B", and so on. We would sample 400 times (equal to the original sample size), and count observed frequencies for the different values.

To simulate this in RStudio, note first that

```
names( tally(~Answer, data=APMultipleChoice) )
```

```
## [1] "A" "B" "C" "D" "E"
```

contains the 5 possible labels on slips of paper. We can sample 400 times from these slips, using the null values of the null hypothesis, with the command

```
resample(names(observed), p=nullVals, size=400)
```

```
##   [1] "D" "E" "C" "C" "D" "C" "D" "B" "A" "C" "C" "B" "E" "C" "E" "D" "D" "B"
##  [19] "B" "D" "C" "C" "B" "D" "D" "C" "C" "C" "A" "A" "A" "D" "C" "D" "C" "D"
##  [37] "D" "E" "B" "D" "C" "A" "D" "D" "D" "B" "C" "A" "D" "A" "E" "C" "B" "A"
##  [55] "E" "C" "C" "A" "C" "C" "C" "C" "C" "C" "A" "A" "B" "E" "B" "D" "D" "B"
##  [73] "B" "E" "E" "D" "C" "C" "B" "D" "B" "B" "D" "A" "B" "A" "B" "D" "B" "B"
##  [91] "B" "C" "A" "E" "B" "A" "E" "D" "B" "A" "A" "B" "C" "B" "C" "B" "D" "D"
## [109] "E" "B" "E" "E" "A" "B" "E" "A" "C" "E" "E" "D" "D" "C" "C" "A" "E" "D"
## [127] "A" "A" "D" "E" "E" "E" "C" "A" "B" "A" "C" "B" "E" "C" "B" "B" "D" "C"
## [145] "D" "B" "B" "B" "D" "D" "B" "A" "E" "D" "D" "D" "C" "D" "E" "C" "B" "A"
## [163] "A" "E" "A" "E" "D" "E" "D" "D" "A" "C" "E" "A" "A" "D" "E" "E" "C" "C"
## [181] "D" "E" "A" "B" "D" "D" "A" "D" "D" "B" "E" "A" "D" "B" "E" "E" "D" "D"
## [199] "E" "D" "C" "C" "E" "C" "B" "A" "C" "A" "B" "D" "A" "E" "C" "C" "D" "D"
## [217] "B" "C" "A" "B" "E" "A" "E" "D" "C" "C" "C" "A" "D" "A" "D" "B" "D" "A"
## [235] "C" "B" "E" "B" "B" "E" "A" "C" "E" "E" "D" "B" "D" "E" "A" "E" "A" "D"
## [253] "D" "B" "D" "E" "C" "D" "B" "C" "D" "E" "D" "D" "D" "D" "D" "E" "B" "E"
## [271] "D" "C" "A" "B" "A" "E" "C" "A" "B" "B" "A" "E" "D" "A" "E" "B" "A" "E"
## [289] "A" "D" "C" "C" "A" "D" "C" "E" "A" "D" "C" "A" "A" "C" "B" "B" "E" "B"
## [307] "D" "B" "A" "E" "A" "D" "B" "E" "A" "A" "E" "C" "B" "E" "B" "D" "C" "A"
## [325] "D" "C" "A" "B" "A" "B" "E" "C" "B" "D" "D" "A" "D" "B" "A" "B" "A" "D"
## [343] "B" "E" "A" "D" "B" "C" "A" "D" "E" "E" "E" "A" "C" "D" "D" "E" "A" "D"
## [361] "E" "D" "B" "E" "C" "D" "A" "E" "E" "D" "B" "E" "E" "B" "D" "A" "E" "E"
## [379] "B" "E" "D" "D" "D" "D" "A" "B" "E" "A" "D" "C" "B" "E" "B" "E" "B" "E"
## [397] "A" "E" "A" "E"
```

though an even more useful command is this one, that generates the observed counts of the randomized sample

```
tally( ~ resample(names(observed), p=nullVals, size=400) )
```

```
## resample(names(observed), p = nullVals, size = 400)
##  A  B  C  D  E
## 83 86 72 74 85
```

We need to use these observed counts to generate a randomization (χ^2) statistic:

```
sum( (tally( ~ resample(names(observed), p=nullVals, size=400) ) - expected)^2 / expected )
```

```
## [1] 2.25
```

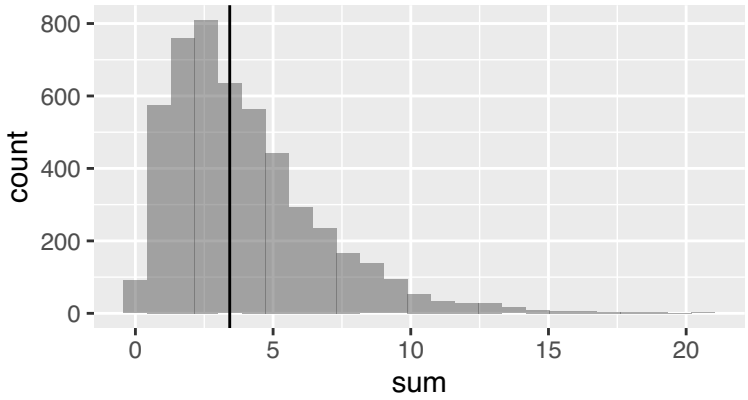
It is this command we would repeat often to generate a randomization distribution:

```
manyChiSqs <- do(5000) *
  sum( (tally( ~ resample(names(observed), p=nullVals, size=400) ) - expected)^2 / expected )
head(manyChiSqs)
```

```
##     sum
## 1 2.100
## 2 4.750
## 3 3.500
## 4 0.925
## 5 2.725
## 6 2.175
```

I plot this along with a vertical line at our test statistic, the χ^2-value from the original sample:

```
gf_histogram(~sum, data=manyChiSqs) %>% gf_vline(xintercept = ~3.425)
```



The approximate P-value is the proportion of times we see a test statistic at least as extreme as our test statistic which, means those instances that are *greater than or equal* to 3.425.

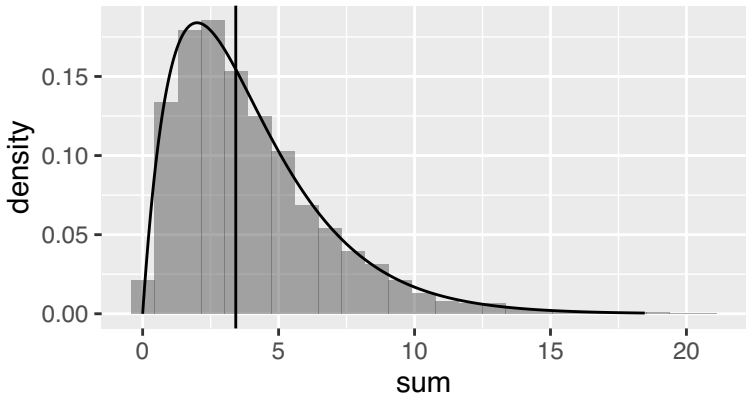
```
nrow( filter(manyChiSqs, sum >= 3.425) ) / 5000
```

```
## [1] 0.4912
```

Using a chi-square distribution

When all expected counts are at least 5, the null distribution is approximated fairly well by a chi-square distribution with **degrees of freedom** $k - 1$.

```
gf_dhistogram(~sum, data=manyChiSqs) %>% gf_vline(xintercept = ~3.425) %>%
  gf_dist("chisq", df=4)
```



That is, the area under the chi-distribution *curve* to the right of our test statistics $\chi^2 = 3.425$ is approximately the same as the proportion of times a χ^2 statistic at least that large occurred in the randomization distribution. Knowing the curve and the histogram have similar shape could save us a number of the steps performed above, steps that we meant to generate many randomization statistics. We could have found this area (our P-value) with the command

```
1 - pchisq(3.425, df=4)
```

```
## [1] 0.4893735
```

But there are settings in which this approach would give us a wildly different result than the actual P-value we seek. Consider this example. Some years ago, survivors of heart attacks in the Canada were polled a year later to ask them for self-assessments on how their quality of life was then in comparison with the quality of life before the heart attack. They were to choose among five options, displayed along with the proportion of respondents who selected the option.

- A: much better (24%)
- B: somewhat better (23%)
- C: about the same (31%)
- D: somewhat worse (16%)
- E: much worse (6%)

Admittedly these percentages all reflect *sample proportions*, not population parameters, but let us take them as null values in a null hypothesis:

$$\mathbf{H}_0 : p_A = 0.24, \;\; p_B = 0.23, \;\; p_C = 0.31, \;\; p_D = 0.13, \;\; p_E = 0.06.$$

Suppose the Canadian government implements changes to their health care system as a result of the survey, with an eye toward shifting results toward more people answering A, B or C. Some time later, another sample of $n = 50$ heart attack survivors is asked the same question, with resulting observed counts

survey	Freq
A	15
B	17
C	12
D	5
E	1

One might want to look at the strength of evidence in this data against the null hypothesis with null values posed above. The χ^2 test statistic is computed as follows

```
sampleSize <- 50
observed <- c(15, 17, 12, 5, 1)
```

```
nullVals <- c(.24, .23, .31, .16, .06)
expected <- sampleSize * nullVals; expected
```

```
## [1] 12.0 11.5 15.5  8.0  3.0
```

```
chisqStat <- sum( (observed - expected)^2 / expected ); chisqStat
```

```
## [1] 6.629091
```

Since there are $k = 5$ possible values, uninformed use of the `pchisq()` command would lead you to compute the P-value using the command

```
1 - pchisq(chisqStat, df = 4)
```

```
## [1] 0.1568362
```

But the rule of thumb, that all cells have expected counts at least 5, is not met here. Try out the app demonstrated in class at this link to see the two P-values, the one the from our command above, and the one from a randomization distribution. The discrepancy isn't horrible, but the conventional wisdom is that the randomization approach is more reliable, and therefore not to use `pchisq()`.

An all-in-one command

You guessed it–there is one. It's called `chisq.test()`. It needs you to send a vector of observed counts. For the raw data found in **APMultipleChoice**, that means you could either get the observed counts in advance:

```
observed <- tally(~Answer, data=APMultipleChoice)
chisq.test(observed, p = c(0.2, 0.2, 0.2, 0.2, 0.2))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  observed
## X-squared = 3.425, df = 4, p-value = 0.4894
```

Or, you could create the frequency table *as* you are calling the `chisq.test()` command:

```
chisq.test( tally(~Answer, data=APMultipleChoice), p = c(0.2, 0.2, 0.2, 0.2, 0.2) )
```

```
##
##  Chi-squared test for given probabilities
##
## data:  tally(~Answer, data = APMultipleChoice)
## X-squared = 3.425, df = 4, p-value = 0.4894
```

Compare the output of this command with things such as our sample χ^2-statistic, number of degrees of freedom, and P-value when we worked with this data above.

The default for this command is to compute its P-value using `pchisq()`. Nevertheless, if you use it on data where our rule of thumb is not met, it will alert you to the fact that its result may not accurately reflect the P-value. See the `Warning` in the output when using numbers from the example discussed involving a survey of heart attack survivors in Canada:

```
chisq.test( c(15, 17, 12, 5, 1), p = c(0.24, 0.23, 0.31, 0.16, 0.06))
```

```
## Warning in chisq.test(c(15, 17, 12, 5, 1), p = c(0.24, 0.23, 0.31, 0.16, : Chi-
## squared approximation may be incorrect
```

```
##
```

```
##  Chi-squared test for given probabilities
##
## data:  c(15, 17, 12, 5, 1)
## X-squared = 6.6291, df = 4, p-value = 0.1568
```

Question

Can you see why, in this instance, RStudio is offering this warning?