Lesson 9 (week 12) - Tests and more tests!

YOUR NAME

2023-12-04

Outline

Today will be less about complicated coding and more about statistics. We will learn how to run:

- One categorical variable: Chi-square goodness of fit tests
- Two categorical variables: Risk ratio, Odds Ratio,Chi-square independence test; Fisher's test; G-test

---

1. One variable: Chi-square goodness of fit tests

Tools we will use

- `table()`
- `chisq.test()`

A χ^2 goodness-of-fit test compares the frequencies of values of a categorical variable to the frequencies predicted by a null hypothesis.

Example 1:

For example, the file `MandMlist.csv` contains a list of all the colors of M&M candies from one package, under the variable "color". Let's ask whether this fits the proportions of colors advertised by the candy company.

- read in the `MandMlist.csv` file and call it MMlist
- use `head()` and `str()` to have a look at the data

`table()`

```
#read in the file and call is MMlist
MMlist <- read_csv("MandMlist.csv")
```

```
## Rows: 55 Columns: 1
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): color
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#use head and str to check it
head(MMlist)
```

```
## # A tibble: 6 x 1
##   color
##   <chr>
## 1 brown
## 2 blue
## 3 blue
## 4 green
## 5 yellow
## 6 yellow
```

```
str(MMlist)
```

```
## spc_tbl_ [55 x 1] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ color: chr [1:55] "brown" "blue" "blue" "green" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   color = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Notice that `MMlist$color` contains the color of each of 55 M&Ms

Summarizing frequency data in a table is useful to see the data more concisely, and such a table is also necessary as input to the χ^2 test function. We can summarize the counts (frequencies) of a categorical variable with `table()`.

- Use the `table` function to summarize the colors of the M&Ms
- save the output to `MMtable`

```
#create MMtable
MMtable <- table(MMlist$color)
#look at it
MMtable
```

```
##
##   blue  brown  green orange    red yellow
##     10      5     12     11      6     11
```

This shows you how many M&Ms were blue, brown etc

```
chisq.test()
```

We can use a χ^2 goodness-of-fit test to compare the frequencies in a set of data to a null hypothesis that specifies the probabilities of each category. In R, this can be calculated using the function `chisq.test()`.

The company says that the percentages are 24% blue, 14% brown, 16% green, 20% orange, 13% red and 13% yellow. This is our null hypothesis. Let's test whether these proportions are consistent with the frequencies of the colors in this bag. R wants the proportions expected by the null hypothesis in a vector

- create a vector containing the expected proportions of each color of M&Ms according to the null hypothesis, in the following order: blue, brown, green orange, red, yello
- name this vector `expected_proportions`

```
#create expected proportions
expected_proportions <- c(0.24, 0.14, 0.16, 0.20, 0.13, 0.13)
#check it
expected_proportions
```

```
## [1] 0.24 0.14 0.16 0.20 0.13 0.13
```

Notice that R wants you to input the expected proportions, which it will use to calculate the expected frequencies.

The first thing we need to do is check whether the expected frequencies (expected by the null hypothesis) are large enough to justify using a χ^2 goodness of fit test.

Recall the χ^2 requirements:

- No more than 25% of the expected frequencies are less than 5
- None of the expected frequencies are less than 1

To obtain these expected counts, multiply the vector of the expected proportions by the total sample size. We can find the total sample size by taking the 'sum() of the frequency table.

- use the function `sum()` on your frequency table `MMtable`
- save the output as `n_total`, the total sample size
- check the value

```
#create n_total
n_total<-sum(MMtable)
n_total
```

```
## [1] 55
```

Now you can use the expected proportions and the total sample size to find the expected number (not proportion) of M&Ms of each color, given your sample size.

- multiply your smaple size by the expected proportions to find the expected absolute values for each color
- save the output as `expected_counts`
- look at the values
- decide whether it's ok to proceed with a χ^2 goodness-of-fit test or not, given the requirements above

```
#create expected counts
expected_counts<-n_total * expected_proportions
expected_counts
```

```
## [1] 13.20  7.70  8.80 11.00  7.15  7.15
```

If all looks good, go ahead and run a χ^2 goodness of fit test.

- The function `chisq.test()` requires two arguments in its input: a frequency table from the data (argument `x`) and a vector with the expected proportions from the null hypothesis (argument `p`). We have both.
- Input the parameter values using the objects you created above and run a χ^2 test. Save the output in `my_chisq_test` and print the results.

```
#your code here
my_chisq_test<-chisq.test(MMtable, p = expected_proportions)
my_chisq_test
```

```
##
##  Chi-squared test for given probabilities
##
## data:  MMtable
## X-squared = 5.1442, df = 5, p-value = 0.3985
```

Now let's run the χ^2 test manually for comparison! Remember the χ^2 formula:

$$\chi^2 = \sum_i = \frac{(O-E)^2}{E}$$

Follow the following steps. Always check each intermediate value to make sure you understand what's going on.

- First, calculate O-E and call it `step1`. You have both of these saved as `MMtable` and `expected_counts`, respectively
- Then, square `step1` and call it `step2`
- Then, divide `step2` by the expected counts and call it `step3`
- Finally, use `sum()` to add all values in `step3`. Call that `step4`. This is your chi-square value.

```
#step 1
step1<-MMtable-expected_counts
step1
```

```
##
##   blue  brown  green orange    red yellow
## -3.20  -2.70   3.20   0.00  -1.15   3.85
```

```
#step 2
step2<-step1^2
step2
```

4

```
##
##    blue   brown   green  orange     red  yellow
## 10.2400  7.2900 10.2400  0.0000  1.3225 14.8225
```

```
#step 3
step3<-step2/expected_counts
step3
```

```
##
##      blue     brown     green     orange        red     yellow
## 0.7757576 0.9467532 1.1636364 0.0000000 0.1849650 2.0730769
```

```
#step 4. This is your chi-square value
step4<-sum(step3)
step4
```

```
## [1] 5.144189
```

Lastly, we need to find the p-value corresponding to this value, so:

- now use the `pchisq` function to get the p-value. This function requires two arguments: the chi-square value (q) and the degrees of freedom (df). We have q but we need to calculate df.

- define the degrees of freedom. Remember that for goodness of fit tests this is nr of categories - 0 - 1. Use `length()` to find the number of categories in MMtable instead of manually typing that in. Save the degrees of freedom as `my_df`.

- run the `pchisq` function and save the output as `my_pchisq`. Make sure you change the argument `lower.tail` to FALSE. Then print `my_pchisq`.

```
#calculate df
my_df<-length(MMtable)-1

#run pchisq using my_df and step4 as arguments. Set lower.tail to FALSE
my_pchisq<-pchisq(q=step4, df=my_df, lower.tail=F)
my_pchisq
```

```
## [1] 0.398538
```

Now you can compare the results you saved as `my_chisq_test` to the ones you calculated manually:

- run `names(my_chisq_test)` to see all the values that get stored from running the chi-square test.
- use `$` to extract the `p.value` and `statistic` values and compare them to

```
#run names(my_chisq_test)
names(my_chisq_test)
```

```
## [1] "statistic" "parameter" "p.value"   "method"    "data.name" "observed"
## [7] "expected"  "residuals" "stdres"
```

```r
#check that the `statistic` value within my_chisq_test has the same value as step4
my_chisq_test$statistic==step4
```

```
## X-squared
##      TRUE
```

```r
#check that the `p.value` value within my_chisq_test has the same value as my_pchisq
my_chisq_test$p.value==my_pchisq
```

```
## [1] TRUE
```

Congrats! Now let's run a goodness of fit test where the null hypothesis is that data follow a Poisson distribution.

Poisson

A χ^2 goodness of fit test can be used to ask whether a frequency distribution of a variable is consistent with a Poisson distribution. This is a useful way to determine if events in space or time are not "random" but instead are clumped or dispersed.

It turns out that this is not a straightforward process in R. In this section, we'll highlight a couple of functions in R that will streamline doing the calculations by hand. We'll use the data on the numbers of extinctions per prehistoric time period described in Example 8.4 in the Whitlock and Schluter textbook (3rd edition).

Background (from the textbook): Do extinctions occur randomly through the long fossil record of Earth's history, or are there periods in which extinction rates are unusually high ("mass extinctions") compared with background rates? The best record of extinctions through Earth's history comes from fossil marine invertebrates, because they have hard shells and therefore lend to preserve well.

- read in the `MassExtinctions.csv` dataset and call it `extinctData`.
- use `head()` to have a look at the data.
- use `dim()` to get the dimensions. Each row corresponds to one block of time. The numbers record how many extinction events were recorded in that time block.
- save the column `numberOfExtinctions` as `number_of_extinctions`. Print it.
- run `table()` on this object to get the absolute frequency of 1,2, etc extinction events across all blocks of time. Call it `table_extinctions`. Print it.

```r
#read in the file as extinctData
extinctData <- read.csv("MassExtinctions.csv", stringsAsFactors = TRUE)

#use head understand the dataset

head(extinctData)
```

```
##   numberOfExtinctions
## 1                   1
## 2                   1
## 3                   1
## 4                   1
## 5                   1
## 6                   1
```

```
#use dim to get the dimensions
dim(extinctData)
```

```
## [1] 76  1
```

```
#create number_of_extinctions
number_of_extinctions <- extinctData$numberOfExtinctions
number_of_extinctions
```

```
##  [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  2
## [26]  2  2  2  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4
## [51]  4  5  5  5  5  5  5  5  5  5  5  6  6  6  6  7  7  8  9  9 10 11 14 16 16
## [76] 20
```

```
#run `table()` on this object to get the absolute frequency of 1,2, etc extinction events
↪  across all blocks of time.

table_extinctions<-table(number_of_extinctions)
table_extinctions
```

```
## number_of_extinctions
##  1  2  3  4  5  6  7  8  9 10 11 14 16 20
## 13 15 16  7 10  4  2  1  2  1  1  1  2  1
```

Interpreting `table_extinctions`:The first row lists each possible number of extinctions in a time period, ranging from 1 to 20. The second row shows the number of time periods that had each number of extinctions.

Questions:

- how many blocks of time are there?
- which frequency is most common across the time blocks?
- how many time periods had 2 or less extinctions? Hint: you can use the `which` function for this

```
# how many blocks of time are there?
nrow(extinctData)
```

```
## [1] 76
```

```
# which frequency is most common across the time blocks?
sort(table_extinctions)[length(table_extinctions)]
```

```
##  3
## 16
```

```
#how many time periods had 2 or less extinctions?
length(number_of_extinctions[which(number_of_extinctions<=2)])
```

```
## [1] 28
```

7

The mean number of extinctions per unit of time is not specified by the null hypothesis, so we need to estimate it from the data.

- get the mean number of extinctions per time period and save it as `mean_extinctions`. Print it

```
#get the mean
mean_extinctions<-mean(number_of_extinctions)
#print it
mean_extinctions
```

[1] 4.210526

We can now use this mean to calculate the probability according to a Poisson distribution of getting each specific number of extinctions.

Remember that under the Poisson, the probability of observing X successes occurring in any given block of time or space is:

$$P[X] = \frac{e^{-\lambda}\lambda^X}{X!}$$

Remember that lambda is simply the mean, which we've calculated.

Instead of using this equation to get the expected values for 0,1,... >20 extinction events, we will use the `dpois()` function.

dpois()

Fortunately, R has a function to calculate the probability of a given value under the Poisson distribution. This function is `dpois()`, which stands for "density of the Poisson". `dpois()` requires two values for input, the mean (which is called lambda in this function) and x, the value of the outcome we are interested in (here, the number of extinctions). For example, the probability of getting `x = 3` from a Poisson distribution with mean `4.21` can be found from the following:

```
dpois(x = 3, lambda = 4.21)
```

[1] 0.1846355

`dpois()` will accept a vector of x's as input, and return a vector of the probabilities of those x's. A convenient short hand to know here is that R will create a vector with a range of values with a colon. For example, 0:20 is a vector with the integers from 0 to 20.

```
0:20
```

[1] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

We'll need the probabilities for all possible values of x from 0 to 20 (because this was the largest value observed in the data). `dpois` rquires two arguments: a vector with the number of successes (0,1, and so on) and the lambda value, which we calculated above (it's the mean).

- Create a vector containing numbers 0:20 and save it in an object `vals`

- use the `dpois` function using as the `x` argument the vector you just created and the lambda value you calculated above to get the expected probabilities. Save this as `expected_extinction_probabilities`. Print it.

```
# create vector
vals<-0:20
# run dpois to create expected_extinction_probabilities
expected_extinction_probabilities<-dpois(x=vals, lambda=mean_extinctions)
expected_extinction_probabilities
```

```
##  [1] 1.483856e-02 6.247813e-02 1.315329e-01 1.846076e-01 1.943238e-01
##  [6] 1.636411e-01 1.148358e-01 6.907419e-02 3.635484e-02 1.700811e-02
## [11] 7.161310e-03 2.741171e-03 9.618145e-04 3.115189e-04 9.368989e-05
## [16] 2.629892e-05 6.920767e-06 1.714122e-06 4.009642e-07 8.885633e-08
## [21] 1.870660e-08
```

The odd notation here is a shorthand for scientific notation. 1.48e-02 means 1.48×10^2. The e stands for exponent.

Remember that we asked R to output the probability of values from 0 to 20. So the first value in this output corresponds to the probability of getting a zero, the second to the probability of getting a 1, etc.

In order to convert these probabilities into expected values for a χ^2 test, we need to multiply them by the total sample size (which you've figured out above). Call this `expected_extinction_counts`. Print it.

```
#calculate expected_extinction_counts
expected_extinction_counts<-length(number_of_extinctions)*
↪   expected_extinction_probabilities
#print expected_extinction_counts
expected_extinction_counts
```

```
##  [1] 1.127730e+00 4.748338e+00 9.996501e+00 1.403018e+01 1.476861e+01
##  [6] 1.243672e+01 8.727524e+00 5.249639e+00 2.762968e+00 1.292616e+00
## [11] 5.442596e-01 2.083290e-01 7.309790e-02 2.367543e-02 7.120431e-03
## [16] 1.998718e-03 5.259783e-04 1.302733e-04 3.047328e-05 6.753081e-06
## [21] 1.421701e-06
```

Most of these values would cause problems for the chi^2 test because they are too small (less than 1). We need to combine categories in some reasonable way so that the expected frequencies match the assumptions of the chi^2 test.

Let's combine the categories for 0 and 1 successes, and combine everything greater than or equal to 8 into one category. It is possible to do this all in R, but you can also do it by hand, using the expected frequencies we just calculated. Here is a table of the expected frequencies for these new combined categories, for both the observed and expected, summing over all the possibilities within each of these groupings. (E.g., the expected frequency for the category "0 or 1" is the sum of 1.128324 and 4.750244.)

```
knitr::include_graphics("fig1.png")
```

Number of extinctions	Expected	Observed
0 or 1	5.878568	13
2	9.999264	15
3	14.032300	16
4	14.768996	7
5	12.435494	10
6	8.725572	4
7	5.247808	2
8 or more	4.911998	9

Note that these expected values are large enough to match the conditions of the χ^2 test.

- Make vectors (using `c()`) for the observed and expected values for these new groups. You can do this one manually by typing in the values shown above.

```
#create expected_combined. I did this one for you

expected_combined <- c(5.878568, 9.999264, 14.032300, 14.768996, 12.435494,  8.725572,
→   5.247808, 4.911998)
#create observed_combined

observed_combined <- c(13, 15, 16, 7, 10, 4, 2, 9)
```

- From these we can calculate the χ^2 test statistic using `chisq.test()`.
- The data (x) are `observed_combined`. We will use `expected_combined` as the argument `p`. But note that we are giving the list of expected frequencies rather the expected probabilities. Therefore, we need to give it the option `rescale.p = TRUE`. (an obvious alternative is to simply obtain the proportions first, and then use that for p as we did in the previous example with no need to rescale).
- Finally, we want to run `chisq.test()$statistic`: by adding `$statistic` at the end, R will only give us the χ^2 value as output. (We don't yet want the full results of the test because R does not know to correct for the reduction in degrees of freedom caused by estimating the mean.)

```
#run chisq.test()$statistic using the appropriate parameters as listed above.

chisq.test(x=observed_combined, p=expected_combined, rescale.p=TRUE)$statistic
```

```
## Warning in chisq.test(x = observed_combined, p = expected_combined, rescale.p =
## TRUE): Chi-squared approximation may be incorrect
```

```
## X-squared
##  23.93919
```

This correctly calculates our χ^2 value as 23.939. The warning message in red is because one of the expected values is under 5, but because this occurred in less than 20% of the categories, it is fine to proceed.

The degrees of freedom for this test are the number of categories minus the number of parameters estimated from the data minus one. We ended up using 8 categories and estimated one parameter from the data, so we have $df = 8 - 1 - 1 = 6$.

pchisq()

To calculate the correct P-value, we need to know probability of getting a χ^2 value greater than 23.939 from a distribution that has 6 degrees of freedom. let's use it as we did before, with `lower.tail=F`:

```
#run pchisq
pchisq(q = 23.939, df = 6, lower.tail = FALSE)
```

```
## [1] 0.0005359236
```

Thus, our P-value for this test of the fit of these data to a Poisson distribution is approximately 0.0005. We reject the null hypothesis that these extinction data follow a Poisson distribution.

Try re-running the code in rex1.16 without $statistic

```
#code here
chisq.test(x=observed_combined, p=expected_combined, rescale.p=TRUE)
```

```
## Warning in chisq.test(x = observed_combined, p = expected_combined, rescale.p =
## TRUE): Chi-squared approximation may be incorrect
```

```
##
##  Chi-squared test for given probabilities
##
## data:  observed_combined
## X-squared = 23.939, df = 7, p-value = 0.001168
```

Notice that while the statistic value is correct, the p-value is different because it is using df=7, which is incorrect.

Two categorical variables

Here, we will learn how to look at, and test for, the association between two categorical variables. We'll use the Titanic data set as an example, looking at the association between the sex of passengers and whether they survived the accident. Let's read the data from the .csv file:

```
titanicData <- read.csv("titanic.csv", stringsAsFactors = TRUE)
```

The variables we'll use are "sex" and "survive". The variable "survival" contains a "yes" if the individual survived the sinking and a "no" for those that did not.

Categorical variables in R are often treated as "factors." You'lll notice that the read.csv() command above included an option stringsAsFactors=TRUE which tells R to treat all columns that include non-numeric characters to be treated as factors. There are several reasons why treating categorical variables in R as factors is beneficial, but here the most relevant is that factors allow us to set the order of different values or categories of the factors. This ordering will affect the order of appearance of these categories in tables and graphs. This order also matters for some calculations (such as odds ratios, below). The default order of categories is alphabetical, but you can change this.

To see the categories and their order for a factor variable, use the levels() command. For example, the default order of the categories "no" and then "yes" for the variable "survival" in the Titanic data set is alphabetical:

```
levels(titanicData$survival)
```

[1] "Died"   "Survived"

We want to change the order so that the "yes" category for survive comes first when we use R to calculate the odds ratio below. This is to ensure that the probability of success (here, "yes") is in the numerator. We change the order using the levels argument argument of the **factor()** command:

```
titanicData$survival <- factor(titanicData$survival, levels = c("Survived", "Died"))
```

The above command saves the new ordering of categories in the data frame titanicData, overwriting the original variable "titanicData$survive". To check the new order:

```
levels(titanicData$survival)
```

[1] "Survived" "Died"

Frequency table

A contingency table is an effective method to see the association between two categorical variables. Moreover, other R functions we will use in this exercise require a contingency table as input.

A frequency table can be created using `table()`. To create a contingency table that shows the relationship between two categorical variables:

- we simply give `table()` the vectors that contain those two variables. Put the explanatory variable first, and then the response variable after a comma.
- save this as `sex_survive_table`

```
sex_survive_table <- table(titanicData$sex, titanicData$survival)
sex_survive_table
```

```
##
##          Survived Died
##    Men        338 1329
##    Women      316  109
```

This shows us that in the Titanic data set, there are 109 female passengers who did not survive, 316 females who did survive, and analogous information for the males.

Sometimes for contingency analysis we have already summarized the counts for each case. In these cases it is useful to be able to create a data table directly from these counts. The following syntax will allow you to create a data table directly:

```
sex_survive_table_direct <- data.frame(yes = c(316,338), no = c(109,1329),
        row.names = c("female","male"))
sex_survive_table_direct
```

```
##        yes   no
## female 316   109
## male   338  1329
```

###Mosaic plots {-}

We learned in week 3 how to show associations between categorical variables like this using grouped bar graphs. The mosaic plot is another graphical technique for showing the association between two categorical variables. Here, each combination of the variables is represented by a rectangle, and the size of the rectangle is proportional to the number of individuals in that combination.

R has a function to calculate mosaic plots, with the sensible name `mosaicplot()`. In its most basic form, you just give it a frequency table as input.

```
mosaicplot(sex_survive_table)
```


sex_survive_table
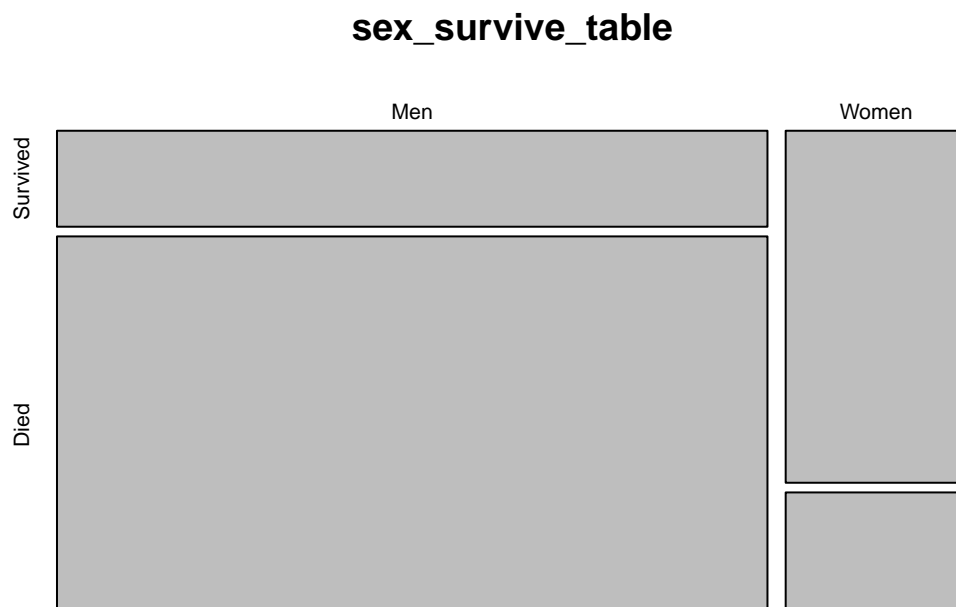
This shows the basic pattern. However, this plot can be greatly improved by adding a couple of extra options, to specify color and axes labels. We can add the option color = c("darkred", "gold") to tell R which colors to use for the different response variables. This is a vector of color names that R assigns in the same order as the order of the categories of the variable plotted on the vertical axis (the response variable) starting with the topmost. (R has many named colors, including all the basics like "red", "orange", "blue", etc.)

We would also like the axes to have labels. We can specify these with xlab and ylab as options. Let's simply call the x-axis "Sex" and the y-axis "Survival".

```
#make new mosaic plot with different colors and axis labels
mosaicplot(sex_survive_table, color = c("darkred", "gold"), xlab ="Sex", ylab =
↪    "Survival")
```

sex_survive_table

It is much easier now to see in the graph that the majority of females survived whereas the majority of males did not.

Odds ratios

One of the ways to measure the strength of the association between two categorical variables is an odds ratio.

In R, the simplest way to estimate an odds ratio is to use the command `fisher.test()`. This function will also perform a Fisher's exact test (more on that later). The input to this function is a contingency table like the one we calculated above. * We'll use the results in a couple of ways, so let's save the results in an object. Call it `sex_survive_fisher`. * then, call `names()` on this object to see what has been stored.

```
# run fisher's test and save results
sex_survive_fisher <- fisher.test(sex_survive_table)
#call names() on the object
names(sex_survive_fisher)
```

```
## [1] "p.value"     "conf.int"    "estimate"    "null.value"  "alternative"
## [6] "method"      "data.name"
```

The output of this function has several parts, two of which we'll want to look at now for estimating an odds ratio. We can see the specific elements of the output by using the $ notation.

Use $ to extract the `estimate` component from `sex_survive_fisher`. This will give youu the odds ratio.

```
#get odds ratio
sex_survive_fisher$estimate
```

```
## odds ratio
##  0.0878782
```

Now use $ to extract the `conf.int` component from `sex_survive_fisher`. This will give youu the 95% confidence interval for the odds ratio.

```
#get the 95% confidence interval for the odds ratio

sex_survive_fisher$conf.int
```

```
## [1] 0.06782129 0.11322761
## attr(,"conf.level")
## [1] 0.95
```

Risk Ratio

We'll use the epitools add on package to calculate relative risk and odds ratio. You might need to install it first if you have not already done so.

```
#install. Only need to run once
#install.packages("epitools")
#load
library("epitools")
```

Let's read in a different dataset for this. Let's estimate the relative risk to estimate association between aspirin treatment and cancer incidence in a 2x2 contingency table.

```
cancer <- read.csv("AspirinCancer.csv", stringsAsFactors = FALSE)
head(cancer)
```

```
##   aspirinTreatment cancer
## 1          Aspirin Cancer
## 2          Aspirin Cancer
## 3          Aspirin Cancer
## 4          Aspirin Cancer
## 5          Aspirin Cancer
## 6          Aspirin Cancer
```

Next, make a 2 x 2 contingency table showing association between aspirin treatment and cancer incidence (Table 9.2-1 from the textbook). R orders the categories of each variable alphabetically, which works here.

Make a contingency table with the disease status first and the treatment second in the table function.

```
#code here
cancerTable <- table(cancer$cancer, cancer$aspirinTreatment)
cancerTable
```

```
##
##              Aspirin Placebo
##   Cancer        1438    1427
##   No cancer    18496   18515
```

Make a mosaic plot:

```
#code here
mosaicplot( t(cancerTable), col = c("firebrick", "goldenrod1"), cex.axis = 1,
    sub = "Aspirin treatment", ylab = "Relative frequency", main = "")
```
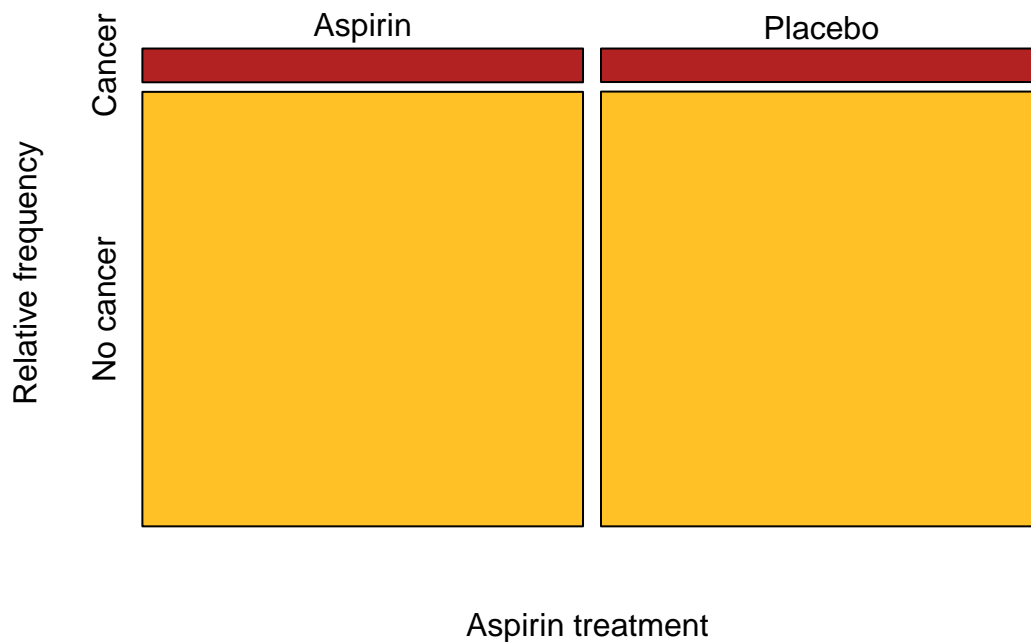

The relative risk (RR) is the probability of the undesired outcome (cancer) in the treatment group (aspirin) divided by the probability in the control (placebo) group.

Our contingency table `cancerTable`, has the undesired outcome in the treatment group in the top left position. The riskratio() function expects the opposite layout, so we need to flip the table and reverse the order of the rows to use it.. We can do this with the `t()` (as in, transpose) function. Do this and save your output as `cancerTable2`. Then, run `riskratio()` on `cancerTable2`. Set `rev=both` and `method="wald"`:

```
#make cancerTable2
cancerTable2<-t(cancerTable)
#run riskratio
RR <- riskratio(t(cancerTable2), rev = "both", method = "wald")
RR
```

```
## $data
##
##             Placebo Aspirin Total
##   No cancer   18515   18496 37011
##   Cancer       1427    1438  2865
##   Total       19942   19934 39876
##
## $measure
##           risk ratio with 95% C.I.
##             estimate     lower     upper
##   No cancer 1.000000        NA        NA
##   Cancer    1.004355 0.9670271 1.043124
##
## $p.value
##           two-sided
##             midp.exact fisher.exact chi.square
##   No cancer         NA           NA         NA
##   Cancer     0.8224348    0.8310911  0.8223986
##
## $correction
## [1] FALSE
##
## attr(,"method")
## [1] "Unconditional MLE & normal approximation (Wald) CI"
```

Chi-square contingency test

A χ^2 contingency (independence) test allows us to test the null hypothesis that two categorical variables are independent of each other.

Because this scenario appears frequently, it is one of the most common tests used in biology. However, remember that the χ^2 is an approximation, and requires that all of the expected values are greater than 1 and that at least 80% are greater than 5. When doing such a test of independence on a computer, it is probably better to use Fisher's exact test, which doesn't have this restriction.

The χ^2 contingency test can be done with the function `chisq.test()`, as we did before for one categorical variable. If we give a frequency table as input, this function will calculate the χ^2 for us.

Before we do the test, though, we need to make sure that the assumptions of the χ^2 test are met by our data. Fortunately, the chisq.test() function also provides a way for us to look at the expected values. If we give a frequency table as input, and then add $expected at the end of the function, it will show us the expected values for a test of independence, like this:

```
chisq.test(sex_survive_table)$expected
```

```
##
##            Survived      Died
```

```
##   Men    521.1367 1145.8633
##   Women  132.8633  292.1367
```

In this case all the expected values are greater than 5, so we have no problem meeting this assumption. Therefore, it is appropriate to do a χ^2 contingency test. Just give a frequency table as input to the chisq.test() function to do this test. We've added the option `correct = FALSE` to tell R to not do a Yate's correction, which can be overly conservative.

```
#run the chisq.test

chisq.test(sex_survive_table, correct=FALSE)
```

```
##
##   Pearson's Chi-squared test
##
## data:  sex_survive_table
## X-squared = 460.87, df = 1, p-value < 2.2e-16
```

Conclusion: what do you think?

Answer: We can reject the null hypothesis of no association between sex and survival on the Titanic.

Fisher's test

Another, more exact, option for testing for the independence of two categorical variables is Fisher's exact test. This is a test that is tedious to calculate by hand, but R can do it in a flash. This test makes no approximations and sets no minimum threshold for the sizes of the expected values.

To implement Fisher's exact test in R, use fisher.test(). This function is easy to use; just give it a frequency table as input.

```
#run fisher's test

fisher.test(sex_survive_table)
```

```
##
##   Fisher's Exact Test for Count Data
##
## data:  sex_survive_table
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   0.06782129 0.11322761
## sample estimates:
## odds ratio
##   0.0878782
```

Here there is little uncertainty about the presence of an association between sex and survival; the P-value is less that 2.2 x 10-16. (This strange value is the smallest number that can be recorded using the default numerical precision in R.)

Also given in the output here is information about the estimate of the odds ratio and its 95% confidence interval.

Unlike the version of Fisher's test described in the textbook, `fisher.test()` is able to calculate contingency tests even when there are more than two possible values for each variable. In such cases, though, it cannot calculate the odds ratios.

G-test   This one is another alternative to the two previous tests. It is not installed in base R, however, Let's load the `DescTools` package to run it.

```
#install. Only need to run once
#install.packages("DescTools")
#load
library("DescTools")
```

```
## Warning: package 'DescTools' was built under R version 4.3.1
```

As discussed in class, the G-test is very similar to the χ^2 test. It takes similar arguments, too. The required ones are the data (x) and the expected proportions (p). Since our data `sex_survive_table` is a table, we only need to give it x. Try it.

```
GTest(x=sex_survive_table)
```

```
##
##  Log likelihood ratio (G-test) test of independence without correction
##
## data:  sex_survive_table
## G = 434.07, X-squared df = 1, p-value < 2.2e-16
```

Voilà!

Case study: Investigating the association between taking Ribavirin and death due to SARS in Singapore"

Now let's apply what we've learned!

Motivation

Ribavirin is a drug that has been used to treat various respiratory disease caused by viruses. It has a very few adverse side effects and works on a large variety of viruses. During the SARS outbreak, Ribavirin was used in several countries, including Singapore, as a treatment. Researchers were interested in measuring the effectiveness of Ribavirin on treating SARS.

Questions of interest

Two questions of interest are:

1. Is there an association between the use of Ribavirin and survival?

2. What is the odds ratio of survival with Ribavirin use?

Data

CSV Data file: *sars.csv* (input files folder)

There were 229 individuals with SARS included in the retrospective study. Individuals either took Ribavirin or did not. The outcome of interest was whether an individual survived or died.

- Read in the dataset, which we'll call `sars`.
- Use `head()` and `str()` to learn a little about the dataset.

```
#read in the dataset
sars <- read_csv("sars.csv")
```

```
## Rows: 229 Columns: 3
## -- Column specification -----------------------------------------------------
## Delimiter: ","
## chr (2): Ribavirin, Survival
## dbl (1): id
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
#do the basic checks
head(sars) # Shows the first six rows of the data set
```

```
## # A tibble: 6 x 3
##       id Ribavirin      Survival
##    <dbl> <chr>          <chr>
## ## 1     1 No Ribavirin Died
## ## 2     2 No Ribavirin Died
## ## 3     3 No Ribavirin Died
## ## 4     4 No Ribavirin Died
## ## 5     5 No Ribavirin Died
## ## 6     6 No Ribavirin Died
```

```
str(sars)
```

```
## spc_tbl_ [229 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ id       : num [1:229] 1 2 3 4 5 6 7 8 9 10 ...
## $ Ribavirin: chr [1:229] "No Ribavirin" "No Ribavirin" "No Ribavirin" "No Ribavirin" ...
## $ Survival : chr [1:229] "Died" "Died" "Died" "Died" ...
## - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   Ribavirin = col_character(),
##   ..   Survival = col_character()
##   .. )
## - attr(*, "problems")=<externalptr>
```

The variable *id* is an ID variable for each individual in the study. The variable *Ribavirin* is equal to *No Ribavirin* when an individual did not receive Ribavirin and equal to *Ribavirin* when an individual did received Ribavirin. The variable *Survival* is equal to *Died* when an individual died and equal to *Survived* when an individual survived.

Exploring the Data

Let's begin by looking at the two variables of interest, Ribavirin use and survival.

Of the 229 individuals included in the study:

1. How many received ribovarin? How many did not?
2. How many died? How many survived?
3. Are there incorrect values or additional levels to either categorical variable? Is cleaning the data necessary?

- use `table()` to count the observed frequencies in each of the columns (Ribavirin and Survival) and answer these questions.
- check for NAs

```r
# Table of Ribavirin use
table(sars$Ribavirin)
```

```
##
## No Ribavirin    Ribavirin
##          132           97
```

```r
# Table of Survival
table(sars$Survival)
```

```
##
##     Died Survived
##       27      202
```

```r
#check for NAs
table(is.na(sars$Ribavirin))
```

```
##
## FALSE
##   229
```

```r
table(is.na(sars$Survival))
```

```
##
## FALSE
##   229
```

```r
#Answers: Received ribavirin: 87, did not receive ribavirin: 132. 27 died, 202 survived.
↪  No NAs
```

Investigating the association between Ribavirin use and survival

How many individuals died/survived within each treatment group?

- to answer this, look at a two-way table of Ribavirin use by survival. Call it `sars_table`

```
#make a 2x2 table. Ribavirin should come first in the function
sars_table<-table(sars$Ribavirin, sars$Survival)
sars_table
```

```
##
##                 Died Survived
##   No Ribavirin   17      115
##   Ribavirin      10       87
```

This table does not give a complete view of the relationship because it does not show proportions. Make a proportions table from the table you created above. Call it `sars_prop_table`.

```
sars_prop_table<-sars_table/sum(sars_table)
```

You could also make a plot. A mosaic plot shows probabilities associated with survival and Ribavirin use.

- Make a mosaicplot with different colors and axis labels ("Treatment Group" and "Survival"), as before

```
#make new mosaic plot with different colors and axis labels
mosaicplot(sars_table, color = c("darkred", "gold"), xlab ="Treatment Group", ylab =
↪  "Survival")
```

sars_table
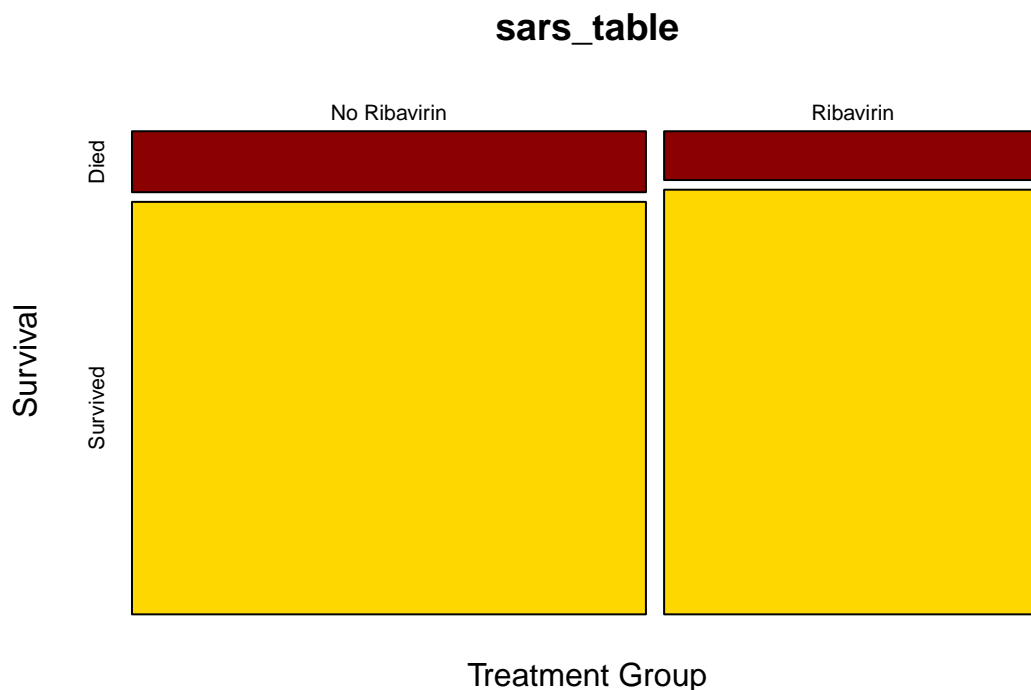
Describe what your mosaic plot is showing you: does there seem to be an asymmetric distribution of individuals who dies amongst the two treatments?

Let's test it!

First, do we meet the χ^2 requirements?

- no more than 20% of the cells can have an expected frequency less than five
- no cell can have an expected frequency less than one

If so, we can perform a χ^2 contingency test to see if there is a statistically significant association between Ribavirin and survival. Set `correct=FALSE`.

Describe the result of your hypothesis test: what is the p-value? Is it statistically significant? What do you conclude about the evidence of an association between Ribavirin use and death?

```
chisq.test(sars_table, p=sars_prop_table, correct=F)
```

```
## 
##  Pearson's Chi-squared test
## 
## data:  sars_table
## X-squared = 0.35495, df = 1, p-value = 0.5513
```

Since the *P*-value is larger than 0.05, there is no statistically significant evidence of an association between Ribavirin use and death.

Odds ratio of survival

To better quantify this lack of association, we'll next calculate the odds ratio of survival with Ribavirin.

What is the main difference between Fisher's exact test and the chi square contingency test we did above?

Answer:

Calculate the OR of survival using the `fisher.test` function and obtain the OR of survival with Ribavirin and 95% CI for the OR.

Describe your results: OR, CI of OR, what do they show and what do they mean.

```
#Obtain the OR

fisher.test(sars_table)$estimate
```

```
## odds ratio
##   1.284696
```

```
#obtain the 95% CI of the OR
fisher.test(sars_table)$conf.int
```

```
## [1] 0.5250188 3.3053466
## attr(,"conf.level")
## [1] 0.95
```

Conclusions

What can you conclude about the effect of Ribavirin use on survival in patients with sars? Discuss any limitations you detected in this study.

References

H-N. Leung, et al., "Investigational Use of Ribavirin in the Treatment of Severe Acute Respiratory Syndrome, Singapore, 2003", *Tropical Medicine and International Health*, 9 [2004]: 923-927 https://onlinelibrary.wiley.com/doi/epdf/10.1111/j.1365-3156.2004.01281.x