R Lab #7 REPLACEMENT - Contingency Analyses

Contents

Why "replace" R Lab #7?

The online R Lab #7 shows you how to create a frequency table, make a mosaic plot, run an odds ratio test, and run a chi-square contingency test. However, there are a couple of things in this lab that are inconsistent with the chapter text.

First, there is no tutorial on running a relative risk analysis.

Second, they use the fisher.test() function to run an odds ratio analysis, which uses a slightly different formula than presented in the chapter.

Using your own functions

To keep our work in R consistent with the textbook, I wrote functions to calculate the relative risk and odds ratio using the same formulas in the textbook. Writing a function in R is fairly easy. You essentially write the function as an R script. My function scripts are called relrisk.R and oddsratio.R, and these need to be downloaded and moved into your ABDLabs folder in order to go through this lab.

It's not enough, however, to have the functions in your working directory. In order for them to be available to you for running you need to import them into your environment. This is done with the source() function, which just requires the name of your function/script in quotes:

```
source("relrisk.R")
source("oddsratio.R")
```

Now our functions relrisk() and oddsratio() will be available to run.

Aspirin and cancer example

We will use the data from the study investigating a contingency between aspirin and cancer to demonstrate some analyses. Download the file "chap09e2AspirinCancer.csv", put it in your working directory, and read it into R:

```
aspirinData <- read.csv("chap09e2AspirinCancer.csv")
```

Notice that the file has two columns: "aspirinTreatment" gives info on whether participants received an aspirin or a placebo, and "cancer" gives info on whether or not the participants developed cancer.

Creating a properly formatted 2x2 contingency table

Next we need to create a 2x2 contingency table, but it needs to be in a *specific format* to work with our functions.

	Treatment	**Control**
Success (focal outcome)	a	b
Failure (alternative outcome)	c	d

Recall that in these biomedical examples that a "success" is the bad outcome (e.g., death or disease), and whether or not there is a "success" should be in the rows with success in top row and failure in the bottom row.

The treatments should be in the columns, with the treatment of interest in the left column and the placebo/control treatment in the left column.

R does not know this conventional format, so when you are making the table you need to make sure that it is *exactly* how you want it. If the rows and/or columns are mixed up then you will get the wrong answer.

To make the 2x2 contingency table we will use the table() function. The syntax for this is table(row_variable,column_variable).

Note that R orders levels in alphabetically order!

Here the variable that we want in rows is aspirin$cancer, which has the levels "cancer" and "no cancer". This turns out to be the same order that we want (i.e., cancer is the "success" and should be in the top row).

Here the variable that we want in columns is aspirin$aspirinTreatment, which has the levels "Aspirin" and "Placebo". This also turns out to be the same order that we want (i.e., Aspirin is the treatment and should be in the left column).

So since alphabetical order happens to be what we want in this case we do not change the order of the levels (see below for an example of that), and we can go straight to making the table.

```
aspirinTable <- table(aspirinData$cancer, aspirinData$aspirinTreatment)
aspirinTable
##
```

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

Notice that the table is in the correct 2x2 contingency format.

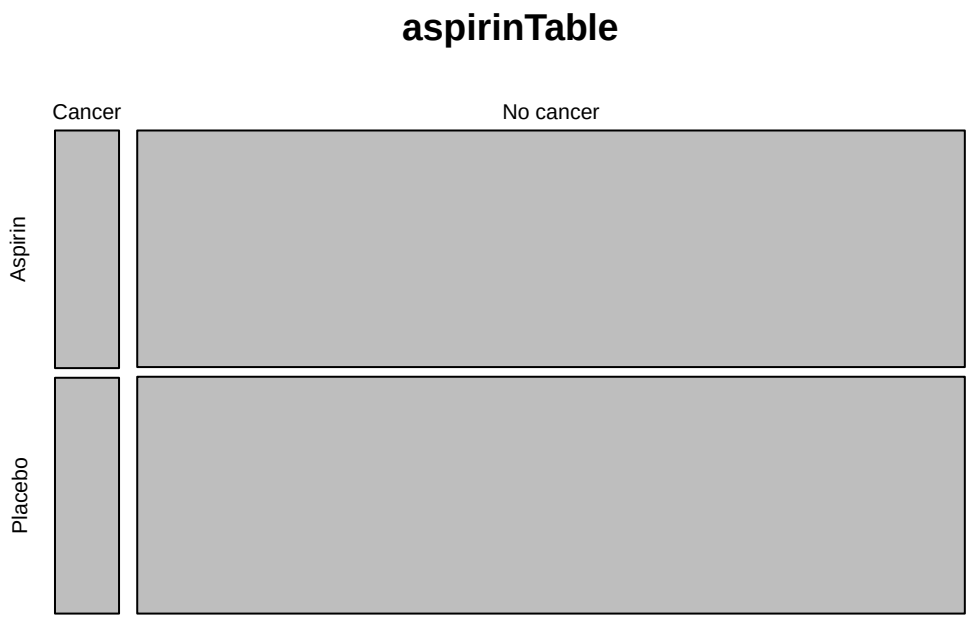
Mosaic plot

Before we run the analyses, however, let's view the observed data with a mosaic plot. The recommended arrangement is to have the "treatment" variable on the x-axis and the success/failure variable on the y-axis. So in the aspirin study we want to have aspirin/placebo on the x-axis and cancer/no cancer on the y-axis.

A mosaic plot is created with the mosaicplot() function, and the first argument is the frequency table that you want to graph. Within the function you can adjust other attributes such as the x-axis label (xlab), y-axis label (ylab), main label above plot (main), and colors (col).

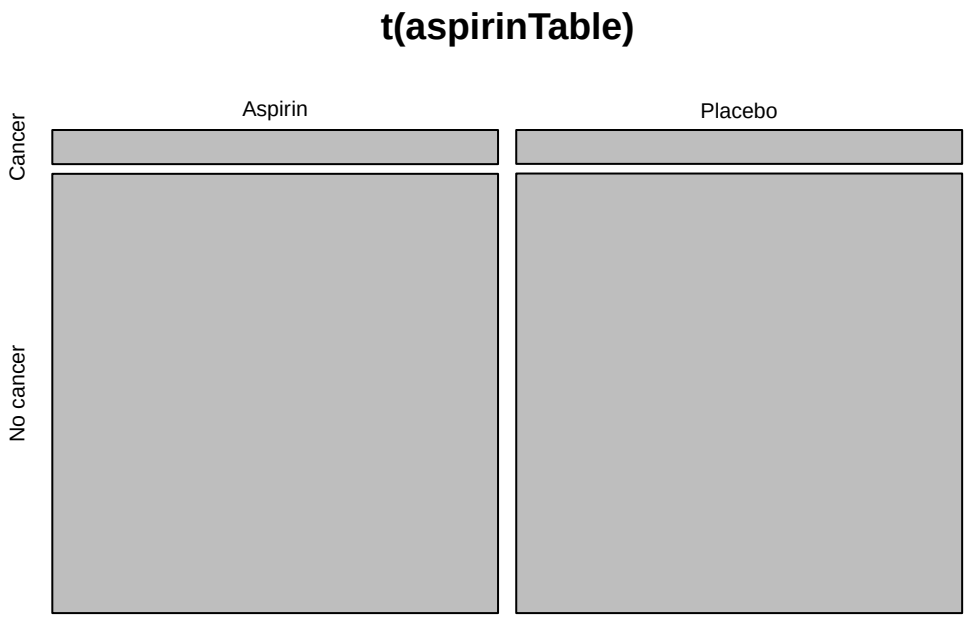
Let's see the default mosaic plot:

```
mosaicplot(aspirinTable)
```



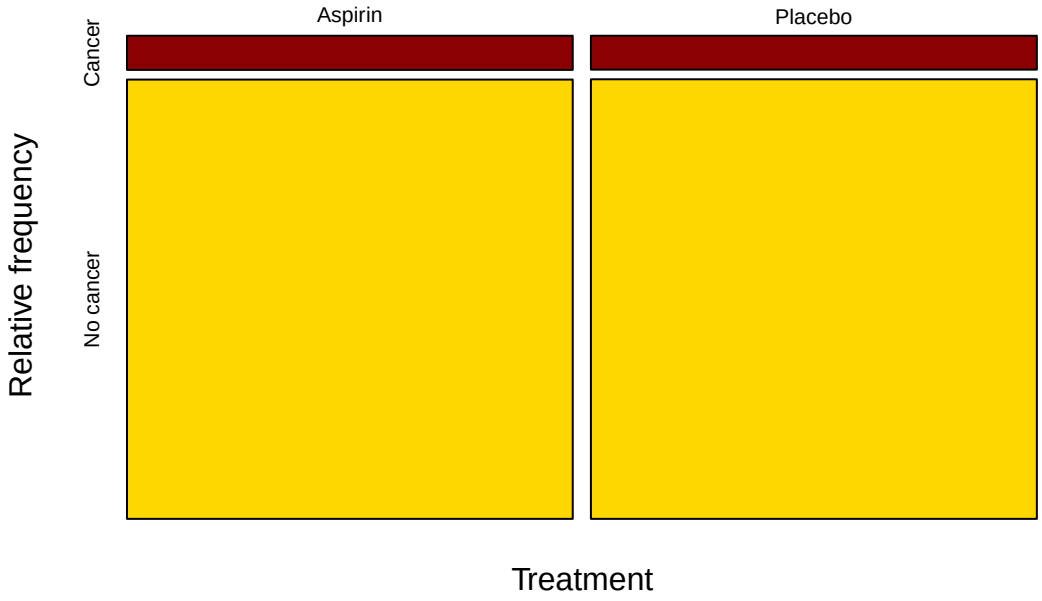
Notice that if we have a properly formatted 2x2 contingency table that the mosaic plot command has the variables on the wrong axes. To fix this we need to "transpose" the table with the t() function:

```
mosaicplot(t(aspirinTable))
```



Now the variables are on the right axes, and we can further improve the graph to make it look more like Figure 9.2-1. Here I add axis labels, remove the main label, and add color:

```
mosaicplot(t(aspirinTable), xlab = "Treatment", ylab = "Relative frequency",
    main = "", col = c("darkred", "gold"))
```



From the mosaic plot it looks like there is an equal probability of getting cancer in the aspirin and placebo treatments, but let's run the analyses...

Relative risk analysis

Since we have our relative risk function imported and a properly formatted 2x2 contingency table it is easy to run a relative risk analysis. The function is called relrisk() and it just takes one argument: a properly formatted 2x2 table. So to run the analysis:

```
relrisk(aspirinTable)
##
```

```
## Relative Risk
##
## Risk of Cancer in Aspirin group: 0.072138
## Risk of Cancer in Placebo group: 0.071558
##
## Relative risk: 1.008113
##
## 95% CI: 0.939435 - 1.081811
```

The output reports the risk in each group (columns in table; Aspirin and Placebo), the relative risk, and the 95% confidence interval of the relative risk. These are all calculated in the same way outlined in the chapter (without rounding). The relative risk is ~1, meaning that there is equal risk of the aspirin and placebo groups of getting cancer.

Odds ratio analysis

Since we have our relative risk function imported and a properly formatted 2x2 contingency table it is also easy to run an odds ratio analysis. The function is called oddsratio() and it just takes one argument: a properly formatted 2x2 table. So to run the analysis:

```
oddsratio(aspirinTable)
##
## Odds Ratio
##
## Odds of Cancer in Aspirin group: 0.077747
## Odds of Cancer in Placebo group: 0.077073
##
## Odds Ratio: 1.008744
##
## 95% CI: 0.934903 - 1.088416
```

The output reports the odds of "success" (disease) in each group (columns in table; Aspirin and Placebo), the odds ratio, and the 95% confidence interval of the odds ratio. These are all calculated in the same way outlined in the chapter (without rounding). The odds ratio is also ~1, meaning that there is equal odds of getting cancer among those in the aspirin and placebo groups.

Pigeon feather color and predation example

What if alphabetical order does not match the required format?

Let's look at practice problem #5 at the end of chapter 9. This question is about pigeons, and the idea that white rump feathers might serve to distract falcons and reduce predation.

First let's import the data:

```
pigeonData <- read.csv("chap09q05PigeonRumps.csv")
```

Re-ordering groups in a variable

In this example there are two outcomes in the data file under the variable survivalWithFalcon: "killed" and "survived." Here we treat killed as the success (disease) and want it in the top row, and thus alphabetical order is okay.

In this example there are two treatments in the data file under the variable rumpColor: "blue" and "white." In the logic of the experiment the researchers were wondering if white feathers provided an advantage, and

thus white is the treatment and should be in the first column. Blue is the control and should be in the second column. So alphabetical order is opposite of what we want.

To change the order of the groups in rumpColor we use the factor() function in the same way that we did in Chapter 8 for re-setting the order of the days of the week.

```
pigeonData$rumpColor <- factor(pigeonData$rumpColor, levels = c("white",
    "blue"))
```

Now we can make a frequency table that will be in the correct format:

```
pigeonTable <- table(pigeonData$survivalWithFalcon, pigeonData$rumpColor)
pigeonTable
##
##             white blue
##   killed        9   92
##   survived     92   10
```

This is how we want it! If you make a table and it's in the wrong format then just use factor() to customize the group order and re-make the frequency table.

Relative risk analysis

We can now run the relrisk() function:

```
relrisk(pigeonTable)
##
## Relative Risk
##
## Risk of killed in white group: 0.089109
## Risk of killed in blue group: 0.901961
##
## Relative risk: 0.098795
##
## 95% CI: 0.052785 - 0.184909
```

Here the relative risk is ~0.10. Since the treatment is in the numerator, this means that pigeons with white rump feathers have ~1/10th the risk of being predated by falcons compared to those with blue rump feathers. The 95% CI ranges from 0.05 to 0.18, so it is far lower than 1 (equal risk).

Odds ratio analysis

We can now run the oddsratio() function:

```
oddsratio(pigeonTable)
##
## Odds Ratio
##
## Odds of killed in white group: 0.097826
## Odds of killed in blue group: 9.2
##
## Odds Ratio: 0.010633
##
## 95% CI: 0.00413 - 0.027379
```

Here the odds ratio is ~0.01. Since the treatment is in the numerator, this means that pigeons with white rump feathers have ~1/100th the odds of being predated by falcons compared to those with blue rump feathers. The 95% CI ranges from 0.004 to 0.027, so it is far lower than 1 (equal odds).

Chi-square contingency test

If one of your variables has more than one group (or level) then you cannot run a relative risk or odds ratio. This is because both of these tests generates a statistic that is a fraction (group1/group2), which is not possible with more than two groups. In such cases the frequency table has more than two rows, columns, or both.

However, you have the option of running a chi-square contingency test, which tests the null hypothesis of independence between your two variables. That is, outcomes for one variable are independent of the other.

In this case you make a table of observed frequencies, and these are compared to expected frequencies that are derived from the data (see section 9.4).

Example 9.4: the gnarly worm gets the bird

Let's go through the example in the chapter about the study analyzing whether fish being eaten by birds is contingent on being uninfected, lightly infected, or heavily infected by trematode parasites.

Import the data:

```
fishData <- read.csv("chap09e4WormGetsBird.csv")
```

Table format doesn't matter for this test

Next we need to make the frequency table. Unlike a relative risk or odds ratio analysis, the format of the table does not matter and there is a good built-in function that we can run.

To demonstrate that the table format does not affect the results, let's make the table two different ways:

```
fishTable1 <- table(fishData$fate, fishData$infection)
fishTable1
##
##             highly lightly uninfected
##   eaten         37      10          1
##   not eaten      9      35         49
fishTable2 <- table(fishData$infection, fishData$fate)
fishTable2
##
##              eaten not eaten
##   highly        37         9
##   lightly       10        35
##   uninfected     1        49
```

Notice that the first variable provided goes in the rows, such that fishTable1 has fate in rows and fishTable2 has infection rate in rows.

Running a chi-square contingency test

Now let's run the chi-square contingency test. This is run with the same chisq.test() function used for the goodness-of-fit test from chapter 8. The difference here is that we don't provide expected null proportions with the p= argument. Instead, we just provide the frequency table and R will calculate the expected frequencies for us. However, we do need one additional argument of *correct=F* to tell R not to apply the continuity correction, which can be overly conservative.

First we'll run the test with fishTable1:

```
chisq.test(fishTable1, correct = F)
##
##  Pearson's Chi-squared test
```

```
##
## data:  fishTable1
## X-squared = 69.756, df = 2, p-value = 7.124e-16
```

And now with fishTable2:

```
chisq.test(fishTable2, correct = F)
##
##  Pearson's Chi-squared test
##
## data:  fishTable2
## X-squared = 69.756, df = 2, p-value = 7.124e-16
```

Note that the results are exactly the same for both tables. This is because the sum of differences in observed and expected is the same no matter the orientation.

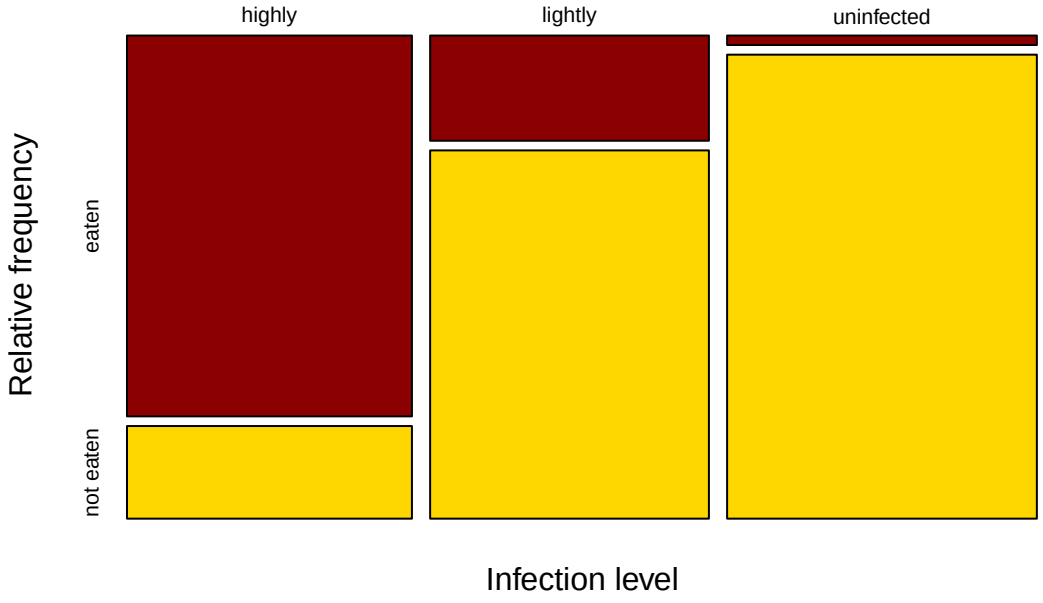
Also, the result is slightly different than what is presented in the textbook, but this is only because there is no rounding of values with R.

Interpreting the results

The P-value is reported as 7.124e-16, which is far below an alpha of 0.05. Therefore, we can reject the null hypothesis that being eaten is independent of infection status. But what is going on biologically?

One way to infer what is happening biologically is to look at the mosaic plot.

```
mosaicplot(t(fishTable1), xlab = "Infection level", ylab = "Relative frequency",
    main = "", col = c("darkred", "gold"))
```



Looking at the mosaic plot we can see the pattern that higher infection rate leads to a greater probability of being eaten.

It is also helpful to save the test as a variable because the observed and expected values will be stored and you can easily compare them.

```
fishTest <- chisq.test(fishTable1, correct = F)
fishTest$observed
##
```

```
##             highly lightly uninfected
##   eaten         37      10          1
##   not eaten      9      35         49
fishTest$expected
##
##               highly  lightly uninfected
##   eaten     15.65957 15.31915   17.02128
##   not eaten 30.34043 29.68085   32.97872
```

You can scan the differences in the output, or you can instead calculate the differences between observed and expected:

```
fishTest$observed - fishTest$expected
##
##                 highly    lightly uninfected
##   eaten      21.340426  -5.319149 -16.021277
##   not eaten -21.340426   5.319149  16.021277
```

This shows you that the significant result is driven by a surplus (+21.34) of highly infected fish that are eaten and a surplus (+16.02) of uninfected fish that survive.

Through both the plot and the differences between observed and expected frequencies you can conclude that parasite infection significantly increases the likelihood of being eaten.

R commands summary

- **Mosaic plot**
  - mosaicplot(table)
- **Relative risk (custom function)**
  - relrisk(table)
- **Odds ratio (custom function)**
  - oddsratio(table)
- **Chi-squared contingency test**
  - chisq.test(table,correct=F)