Tables

Put your name here

Put the date here

Introduction

In this module, we'll learn how to summarize categorical data using tables.

Instructions

Presumably, you have already created a new project and downloaded this file into it. From the `Run` menu above, select `Run All` to run all existing code chunks.

When prompted to complete an exercise or demonstrate skills, you will see the following lines in the document:

ANSWER

These lines demarcate the region of the R Markdown document in which you are to show your work.

Sometimes you will be asked to add your own R code. That will appear in this document as a code chunk with a request for you to add your own code, like so:

```
# Add code here
```

Be sure to remove the line `# Add code here` when you have added your own code. You should run each new code chunk you create by clicking on the dark green arrow in the upper-right corner of the code chunk.

Sometimes you will be asked to type up your thoughts. That will appear in the document with the words, "Please write up your answer here." Be sure to remove the line "Please write up your answer here" when you have written up your answer. In these areas of the assignment, please use contextually meaningful full sentences/paragraphs (unless otherwise indicated) and proper spelling, grammar, punctuation, etc. This is not R code, but rather a free response section where you talk about your analysis and conclusions. You may need to use inline R code in these sections.

When you are finished with the assignment, knit to PDF and proofread the PDF file **carefully**. Do not download the PDF file from the PDF viewer; rather, you should export the PDF file to your computer by selecting the check box next to the PDF file in the Files plane, clicking the `More` menu, and then clicking `Export`. Submit your assignment according to your professor's instructions.

Load Packages

We load the `MASS` package to work with data on risk factors associated with low birth weight, and the `gmodels` package to make nice contingency tables.

```
library(MASS)
library(gmodels)
```

Working with factor variables

R uses the term "factor variable" to refer to a categorical variable. Your data set may already come with its variables coded correctly as factor variables, but often they are not. For example, our birth weight data `birthwt` has several categorical variables, but they are all coded numerically.

The code below is somewhat involved and technical. After the code chunk, I'll explain what each piece does.

```
race <- factor(birthwt$race,
               levels = c(1, 2, 3),
               labels = c("White", "Black", "Other"))
race_df <- data.frame(race)
```

First of all, because `birthwt` is a dataset defined in the `MASS` package, we don't want to modify it. Therefore, if we want to change something, we have to assign a new name to the resulting operation. That is why we have `race <-` at the beginning of the code line. The symbol `<-` is taking the result of the command on the right (in this case, the `factor` command) and giving it a new name.

The `factor` command converts `birthwt$race` into a factor variable. The `levels` of the variable are the pre-existing numerical values. The `labels` are the names we actually want to appear in our output.

The letter `c` in `c(1, 2, 3)` and `c("White", "Black", "Other")` is necessary whenever we want to combine more than one thing into a single expression. (In technical terms, the "c" stands for "combine" or "concatenate" and creates a "vector".)

The last line takes the single vector `race` and turns it into a data frame that we call `race_df`. Many of the commands we will use require that we analyze variables that are sitting inside of data frames. Let's see how this worked.

```
str(race_df)
```

```
## 'data.frame':    189 obs. of  1 variable:
##  $ race: Factor w/ 3 levels "White","Black",..: 2 3 1 1 1 3 1 3 1 1 ...
```

```
head(race_df)
```

```
##    race
## 1 Black
## 2 Other
## 3 White
## 4 White
## 5 White
## 6 Other
```

You can see from the output that this created a data frame called `race_df` containing a single factor variable called `race` sitting inside it. The values of race are no longer numbers 1, 2, or 3. Instead, those numbers have been replaced by human-readable words describing the three racial categories.

If a variable is already coded as a factor variable in its data frame, it is important **not** to use the `factor` command on it. This will mess up the data and make all future attempts at analysis break.

Summarizing one categorical variable

If you need to summarize a single categorical variable, a frequency table usually suffices.

```
table(race_df$race)
```

```
##
## White Black Other
```

```
##    96    26    67
```

If you want percentages, you have to create a table and then apply the `prop.table` command to it:

```r
prop.table(table(race_df$race))
```

```
##
##     White     Black     Other
## 0.5079365 0.1375661 0.3544974
```

The graphical analogues of these tables are the bar chart and the relative frequency bar chart. (See the module `Graphing_categorical_data.Rmd`.)

Exercise

Generate a frequency table like above, but this time use the original variable `birthwt$race`.

ANSWER

```r
# Add code here to create a frequency table with birthwt$race.
```

Explain the advantage of creating a factor variable with meaningful labels over using the original numerical variable.

ANSWER

Please write up your answer here.

Summarizing two categorical variables

A table summarizing two categorical variables is called a contingency table (or pivot chart, or cross-tabulation, or probably several other terms as well). There are multiple ways of getting contingency tables out of R, but the most flexible is the `CrossTable` command from the `gmodels` package.

First things first, though. We need to create one more factor variable. We'll use the `smoke` variable about whether the mothers smoked during pregnancy.

```r
smoke <- factor(birthwt$smoke,
                levels = c(1, 0),
                labels = c("Yes", "No"))
smoke_race <- data.frame(smoke, race)
```

The `smoke` variable is created in exactly the same way as the `race` variable was earlier. Now, though, because we want to analyze both `smoke` and `race` together, we create a data frame called `smoke_race` with both variables.

(When we work with two variables, typically we think of one variable as explanatory, and the other as response. For reasons that will become more clear later on in the course, we will adopt the convention of always listing the response variable first, followed by the explanatory variable. So we use `smoke_race` instead of `race_smoke`.)

Let's make sure the above commands did what we intended.

```
str(smoke_race)
```

```
## 'data.frame':    189 obs. of  2 variables:
##  $ smoke: Factor w/ 2 levels "Yes","No": 2 2 1 1 1 2 2 2 1 1 ...
##  $ race : Factor w/ 3 levels "White","Black",..: 2 3 1 1 1 3 1 3 1 1 ...
```

```
head(smoke_race)
```

```
##   smoke  race
## 1    No Black
## 2    No Other
## 3   Yes White
## 4   Yes White
## 5   Yes White
## 6    No Other
```

Examine the output from the above code to make sure we have a data frame with the two factor variables we want.

You may be wondering why the command looked like this:

`smoke <- factor(birthwt$smoke, levels = c(1, 0), labels = c("Yes", "No"))`

instead of like this:

`smoke <- factor(birthwt$smoke, levels = c(0, 1), labels = c("No", "Yes")).`

For most response variables, we often designate one category as a category of interest in our study. This category is often called the "Success" condition. For example, the question we're asking about this data is, "Within each racial category, what percentage of mothers smoked?" Therefore, the "Yes" condition in the `smoke` variable is considered the "Success" category (even though there's obviously nothing "successful" about smoking while pregnant!). When we list that success category first in the factor command, R will treat it a little differently than other categories. It's not so important to us here, but it will be very important in the future. This typically only matters for the response variable. (The explanatory variable is `race` and our research question does not single out one racial category as being of more interest to us than any other.)

And now for the contingency table. The first variable listed in the `CrossTable` command will be your row variable and the second, your column variable. There's no right or wrong way to do this, but I prefer to use the response variable for the rows and the explanatory variable for the columns. (This is consistent with our convention described above to list the response variable first, followed by the explanatory variable.) For example, I might be interested in knowing if a woman's race is associated with how likely she might have been to smoke. Therefore, `smoke` will be my row variable and `race` will be my column variable. For now, ignore all the extra options on the second and third lines of the code chunk below.

```
CrossTable(smoke_race$smoke, smoke_race$race,
           prop.r = FALSE, prop.c = FALSE,
           prop.t = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |-------------------------|
##
##
## Total Observations in Table:  189
##
##
```

```
##                  | smoke_race$race
## smoke_race$smoke |     White |     Black |     Other | Row Total |
## -----------------|-----------|-----------|-----------|-----------|
##              Yes |        52 |        10 |        12 |        74 |
## -----------------|-----------|-----------|-----------|-----------|
##               No |        44 |        16 |        55 |       115 |
## -----------------|-----------|-----------|-----------|-----------|
##     Column Total |        96 |        26 |        67 |       189 |
## -----------------|-----------|-----------|-----------|-----------|
##
##
```

This table is highly misleading. For example, one cannot compare the 10 black women who smoked to the 12 "other" women who smoked. The 10 are out of 26, but the 12 are out of 67. That's why we need percentages.

As the explanatory variable is in the columns, we turn on column percentages using the `prop.c` option of `CrossTable`.

```
CrossTable(smoke_race$smoke, smoke_race$race,
           prop.r = FALSE, prop.c = TRUE,
           prop.t = FALSE, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Col Total |
## |-------------------------|
##
##
## Total Observations in Table:  189
##
##
##                  | smoke_race$race
## smoke_race$smoke |     White |     Black |     Other | Row Total |
## -----------------|-----------|-----------|-----------|-----------|
##              Yes |        52 |        10 |        12 |        74 |
##                  |     0.542 |     0.385 |     0.179 |           |
## -----------------|-----------|-----------|-----------|-----------|
##               No |        44 |        16 |        55 |       115 |
##                  |     0.458 |     0.615 |     0.821 |           |
## -----------------|-----------|-----------|-----------|-----------|
##     Column Total |        96 |        26 |        67 |       189 |
##                  |     0.508 |     0.138 |     0.354 |           |
## -----------------|-----------|-----------|-----------|-----------|
##
##
```

Now we can see that each column adds up to 100%. (Ignore the percentages in the last "Column Total" row.) In other words, each racial group is now on equal footing, and only the distribution of smokers within each group matters.

Exercise

What percentage of black women smoked during pregnancy? What percentage of "other" women smoked during pregnancy?

ANSWER

Please write up your answer here.

Does race appear to be associated with the likelihood of smoking during pregnancy for the women in this data set? Or are these variables independent?

ANSWER

Please write up your answer here.

Your turn

Choose two categorical variables of interest from the `birthwt` data set. (Choose at least one variable other than `race` or `smoke`.) Turn them into factor variables with meaningful labels. Create a new data frame containing both variables. Identify one as explanatory and one as response. Then create a contingency table with column percentages. Comment on the association (or independence) of the two variables.

ANSWER

```
# Add code here to convert one or more variables to factor variables
# and make a data frame.
```

```
# Add code here to create a contingency table with column percentages.
```

Please write up your answer here.

Conclusion

We use frequency tables to summarize a single categorical variable. Both raw counts and percentages can be useful.

We use contingency tables to summarize two categorical variables. Unless groups are of equal size, raw counts can be incredibly misleading here. As long as your explanatory variable appears as the column variable in the table, you should include column percentages to be able to compare the distributions of percentages across groups. If the percentages are roughly the same, the variables are more likely to be independent, whereas if the percentages are different, there may be an association between the variables.