Lab 4: Sweet Tooth

Chi-squared tests of proportions

Your name here

Date here

General information

This lab is due February 17 by 11:59 pm. **Upload the R Markdown document and the knitted pdf to Canvas.** The lab is worth 10 points (all questions worth 1 point unless otherwise noted). You are welcome and encouraged to talk with classmates and ask for help. However, each student should turn in their own lab assignment and all answers, including all code, needs to be solely your own.

Objective

In this lab you will learn how to implement chi-squared and goodness-of-fit tests using data that last year's class generated. You will also get further practice in identifying which statistical tests are appropriate, as well as plotting data.

Overview

This lab will start with a few examples to review the implementation of these tests in R. Then you will apply these tests to the data from last year's class.

Example 1: χ^2 goodness-of-fit tests

The χ^2 goodness-of-fit test is a specific type of goodness-of-fit test that compares a distribution of observed counts to a distribution of expected counts under some probability model. The data in this example come from example 8.1-1 in the book. This study examined whether babies were equally likely to be born on all 7 days of the week. (As you run through this example, think about which experiment from today is similar in structure to this experiment, and make sure you input and run your data in the same way as this example does).

Step 1: enter in the observed data and null proportions

The first step is to create a vector of observed counts. These observations come from the book example and are the total number of babies born on each day of the week. The observed data should always be formatted as *counts* rather than proportions.

```
birthDayObserved <- c(14, 26, 34, 21, 27, 38, 20) # observed counts
```

Next, I'll create a vector of expected *probabilities* under the null hypothesis. We need a null proportion (or an expected proportion) for each category. We might start by assuming that each day in the week has the same probability. In this case we have 7 categories since there are 7 days of the week. The null hypothesis in is that babies are equally likely to be born on any day of the week. Therefore our null hypothesis might be that 1/7 of babies are born on Monday, 1/7 on Tuesday, and so on. Here we might then use `Null.proportions<-rep(1/7, 7)`. In this case our theoretical/null distribution is a uniform distribution, but it doesn't always need to be a uniform distribution.

In fact, the textbook notes that in 2016 (when the data were collected) there were actually 52 Sundays, Mondays, Tuesdays, Wednesdays, and Thursdays, but 53 Fridays and Saturdays. You might note that this all sums to 366 - 2016 was a leap year!

So in this case we can calculate the null probability of a baby being born on each day of the week as 52/366 for Sundays through Thursdays and 53/366 for Fridays and Saturdays. Note that all of these Null Proportions must sum to one.

```
Null.Proportions <- c(52/366, 52/366, 52/366, 52/366, 52/366, 53/366, 53/366)
Null.Proportions
```

```
## [1] 0.1420765 0.1420765 0.1420765 0.1420765 0.1420765 0.1448087 0.1448087
```

```
sum(Null.Proportions)
```

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

If our distribution was not a uniform distribution (say, we expected more babies born on the weekends for some reason) we'd need to give the null proportion for each category, making sure they sum to one. So, for example: `Null.proportions <- c(0.2, 0.15, 0.1, 0.1, 0.1, 0.15, 0.2)`.

Remember that your null proportions/probabilities need to be between zero and one, and the sum of all of the proportions should equal 1.

It is crucial that the order of categories (e.g. Monday, Tuesday, Wednesday...) in the Observed and Null datasets is the same. When you run your own data, make sure this is the case.

Step 2: plot the data (bar plot)

A good rule of thumb is to visualize your data in a plot before running a statistical analysis. Bar charts are a good way to display proportional data when you have > 2 categories, as we do in this case. Here, I run through an example that plots observed and expected frequencies side-by-side.

First, you need to calculate the expected *counts* if the null hypothesis were true. To get expected counts (`birthDayExpected`), multiply the null proportion for each category by the total number of observations, summed across all categories (180 babies). Since each day of the week has a different null proportion then the expected number of births on each day will differ.

```
birthDayExpected <- Null.Proportions * sum(birthDayObserved)
birthDayExpected
```

```
## [1] 25.57377 25.57377 25.57377 25.57377 25.57377 26.06557 26.06557
```

To make plotting easier, we'll create a table of our observed and expected observations, using the function `rbind`, which combines the two vectors together as different rows in a table.

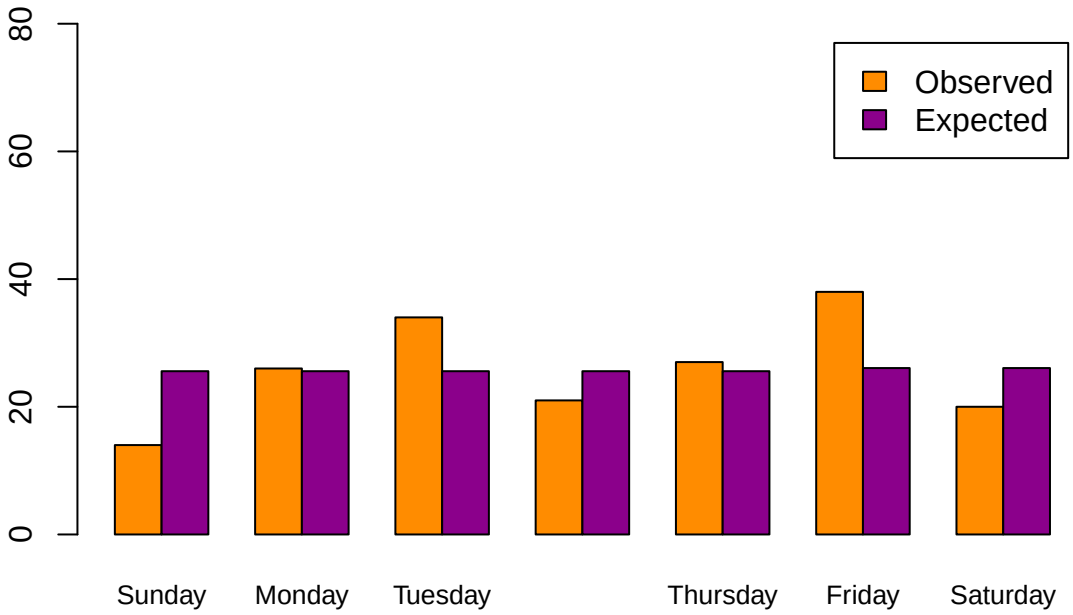
```
# combine data; row 1 is observed, row 2 is expected
birthData <- rbind(birthDayObserved, birthDayExpected)
# give the table column names
colnames(birthData)<- c("Sunday","Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")
birthData
```

```
##                       Sunday   Monday  Tuesday Wednesday Thursday   Friday
## birthDayObserved 14.00000 26.00000 34.00000  21.00000 27.00000 38.00000
## birthDayExpected 25.57377 25.57377 25.57377  25.57377 25.57377 26.06557
##                   Saturday
## birthDayObserved 20.00000
## birthDayExpected 26.06557
```

Pay close attention to the order in which you bind your data. In this case, I used `rbind()` which combines vectors as separate rows, rather than separate columns. Row 1 is observed and row 2 is expected, and each row goes in order of days of the week.

Next, use the function `barplot` to plot data. Bar plots are useful ways to visualize frequency data.

```
colors<-c("darkorange", "darkmagenta")
barplot(height = birthData, beside=TRUE, cex.names = 0.8, ylim = c(0,80),
        legend = c("Observed", "Expected"), col=colors)
```



R will first plot the first row of data (in this case Observed), then will plot the second row of data (Expected), so in this case, observed counts will be plotted in dark orange and expected counts in dark magenta. I've made sure that my legend has the names in the right order as well.

The command `beside = TRUE` tells R that the data should be grouped; the first element of each vector (Sunday counts) go next to each other, then the second elements, and so on. R plots the bars in the order they appear in the birthData table.

Step 3: running the statistical test

The goodness-of-fit test calculates a test statistic, χ^2, which measures the difference between observed and expected counts. The larger the difference, the higher the test statistic, and the lower the P-value.

To run a goodness-of-fit test, we use the function `chisq.test`. This function will run goodness-of-fit AND contingency tests.

For the goodness-of-fit test, we need to fill out a few arguments. The `x =` argument gives your vector of observed counts. The `p =` argument gives a vector of null *probabilities* (not expected counts), so be careful there! *I will repeat. When we calculate a χ^2 test by hand we need the expected number of counts. In R, what we need is the expected proportions.* There need to be the same number of entries in the `p =` vector as there

are in the `x =` vector, and the number of elements must be equal to the number of categories (in this example, 7).

```
chisq.test(x = birthDayObserved, p = Null.Proportions)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  birthDayObserved
## X-squared = 15.795, df = 6, p-value = 0.0149
```

The output of this tells us that there is a 1.5% chance of observing our birth distribution OR birth distributions even further from the null, if the null hypothesis is true. Since it is < 0.05, we reject the null hypothesis. Babies do NOT have an equal chance of being born on each day of the week. Our test statistic is 15.8.

Example 2: χ^2 contingency tests

χ^2 contingency tests are a sub-class of goodness-of-fit tests. Instead of comparing an observed distribution to a null or theoretical probability distribution, we compare two or more distributions which differ in some factor level. Here we are asking whether the sample distributions are different from one another, rather than whether they are different from some theoretical distribution. Another way of thinking about this is that χ^2 contingency tests for an association between two or more categorical variables. If your response variable is counts in different categories, and your explanatory variable is categorical, you will likely be running a contingency test. In contingency tests, there may be $>= 2$ categories in both the response and explanatory variables. Let's walk through an example.

From example 9.4 in the book, researchers studied the rates of predation of fish with differing infection levels. Researchers wanted to know if fish were more likely to be eaten by birds if they were infected by trematodes.

Fish in this study were grouped into 3 categories, uninfected, lightly infected, and highly infected (explanatory variable). Researchers marked each fish's fate: whether that fish was eaten by birds or not (response variable).

Therefore, the researchers are interested in whether or not their response variable (fish fate) is independent of explanatory variable (infection). If the two variables are independent (the null hypothesis), fish are equally likely to be eaten no matter what their infection status is. If the two variables are not independent (the alternative hypothesis), fish in each of the infection groups do NOT have equal chances of being selected.

Step 1: Read in the data

The data from this example are in the file fish_data.csv. Read in the data and take a look at the structure. Note that this data gives the response of each individual fit, rather than counts (or frequencies) of each group.

Step 2: Create a summary table for plotting

To plot the χ^2 data, we need a contingency table, which gives the frequencies or counts in each combination of categories. Since we have 3 explanatory variable categories (uninfected, lightly, highly) and 2 response variable categories (eaten or not eaten), we want a table with 6 cells. So, we need to sum the observations in each category. You could do this by hand, or you could use a handy function called `table`:

```
fishTable <- table(fish$infection, fish$fate)
```

```
## Error in table(fish$infection, fish$fate): object 'fish' not found
```

```
addmargins(fishTable)
```

```
## Error in addmargins(fishTable): object 'fishTable' not found
```

The `table` function sums all of the observations for a specific combination of fate and infection, and gives row and column totals. The first argument you give the `table()` function will be your row names, the second will be your column names. In this case, the explanatory categories are rows, and response categories are columns. However, contingency tables can be made in the opposite format; with explanatory categories as columns. It doesn't change the math! The `addmargins` function is a useful function that adds row and column sums to a given table.

Note that this format of data is very different from the raw data!

Step 3: Plot the data (mosaic plot)

By now you've seen mosaic plots. They are useful ways of visualizing data where you are looking at categorical response variables and categorical explanatory variables.

```
mosaicplot( fishTable, col = c("firebrick", "pink"), cex.axis = 1,
            sub = "Infection level", ylab = "Relative frequency",
            main = "Infection status and fish fate")
```

```
## Error in mosaicplot(fishTable, col = c("firebrick", "pink"), cex.axis = 1, : object 'fishTable' not
```

Recall that if Infection level and predation are independent, then the proportion of each of the three bars (for infection) should have the same proportion eaten (i.e. pink versus firebrick colors).

So they look different, but we need to actually test how likely these sorts of differences are when the null hypothesis is true.

Step 4: Run the contingency test

Since contingency tests are simply one type of goodness-of-fit test and also use the χ^2 test statistic, we use the same function to run contingency tests:

```
FishChiTest <- chisq.test(y = fish$fate, x = fish$infection, correct = FALSE)
```

```
## Error in is.data.frame(x): object 'fish' not found
```

```
FishChiTest
```

```
## Error in eval(expr, envir, enclos): object 'FishChiTest' not found
```

Important: the arguments are different than they were in the first example (babies). Instead of a vector of expected probabilities (p), we give the `chisq.test` the names of the two categorical variables for which we are testing independence (x and y). Remember, we want to know if fish fate and fish infection status are independent. An important detail to note is that we are running this test using the two columns from our original dataframe, `fish` instead of using the contingency table `fishTable` (which is for plotting only). We've also saved the results of our contingency test as a variable, `FishChiTest`. This is not necessary but allows us to look up the results of our test at a later date.

In this case, we have an extremely low P-value. 7.124e-16 is equivalent to 7.124 x 10ˆ-16 (the former is referred to as E-notation). Our test statistic is large: 69.76. So, there is an extremely low chance of observing as big of a difference in fish fate (or even bigger differences) between infection categories if there really were no link between infection level and fish fate. So, we reject the null hypothesis. Our data indicate there is a link between infection level and chance of bird predation. In other words, fish fate *depends* on infection level. We don't know without further testing which level of infection differs from which other level; it may be that only one group is different or that all three are different from one another.

The `chisq.test` function calculates expected frequencies in order to compute the test statistic. If you want to see these expected frequencies you can add $expected to the variable-name of your saved statistical test. This is why it's nice to save the test as an object in R.

```
FishChiTest$expected
```

```
## Error in eval(expr, envir, enclos): object 'FishChiTest' not found
```

Recall that one of the assumptions of a χ^2 contingency test are that there are no expected frequencies less than one and that no more than 20% of expected frequencies are less than five. If this were to have happened in the above example, we could have combined some categories using the following code:

```
# These first few lines utilize the familiar which() function
fish$infection2 <- "yes"
```

```
## Error in fish$infection2 <- "yes": object 'fish' not found
```

```
uninfected_indices <- which(fish$infection == "uninfected")
```

```
## Error in which(fish$infection == "uninfected"): object 'fish' not found
```

```
fish$infection2[uninfected_indices] = "no"
```

```
## Error in fish$infection2[uninfected_indices] = "no": object 'fish' not found
```

```
# Or, use this one-liner to do the same thing using
fish$infection2<-ifelse(fish$infection == "uninfected", "no", "yes")
```

```
## Error in ifelse(fish$infection == "uninfected", "no", "yes"): object 'fish' not found
```

```
# Then analyze as...
FishChiTest <- chisq.test(y = fish$fate, x = fish$infection2, correct = FALSE)
```

```
## Error in is.data.frame(x): object 'fish' not found
```

```
FishChiTest
```

```
## Error in eval(expr, envir, enclos): object 'FishChiTest' not found
```

```
FishChiTest$expected
```

```
## Error in eval(expr, envir, enclos): object 'FishChiTest' not found
```

In the above example this wasn't needed but in other cases where expected frequencies are too low we might need to combine categories.

Great! You now know how to run a χ^2 goodness-of-fit test, as well as a more specific type of goodness-of-fit test, called a χ^2 contingency test using R's built-in functions. And recall, you have run the binomial test in R last week. That means you are prepared to analyze the data from last year's class!

Class Data

Since we aren't able to meet in person this year we won't be able to get together to sample sweets. So instead we will use the data that the 2018 and 2019 classes collected. These data are posted to Canvas. Each experiment uses a separate .csv file. There were 2 experiments that are described in more detail in the Handout, but here is an overview:

1) Are Jelly Belly flavors really that different?

Experiment: each student tried one Jelly bean and had to guess which of 6 flavors it was. We wanted to know if students can identify flavors better than chance.

Data: the guesses (correct = 1; incorrect = 0) of each student in the class is in the file "JellyBelly.csv".

2) Can soda drinkers tell Coke and Pepsi apart?

Experiment: each student tried an unknown soda flavor and guesses which one is Coca-Cola. Their guess is recorded as correct or incorrect. The class is split up into 3 categories: frequent, occasional, and non-soda drinkers. We want to know whether the proportion of correct guesses differs among these three categories. In other words: are soda drinkers better at distinguishing the correct brand?

Data: the total number of correct and incorrect guesses of each group are in the file "CokevsPepsi.csv".

Experiment 1: Jelly Belly Taste Test

Use the class' data to see if students can discriminate between Jelly Belly flavors better than chance. See the summaries at the beginning of this lab, or the background document on Canvas.

Question 1: 1 point What is the null hypothesis being tested here? What is the alternative hypothesis?

Null hypothesis:

Alternative hypothesis:

Question 2: 1 point Run the appropriate statistical test. Include code for reading in the class' data from a csv. Hint: there are two ways to approach this data; one will give an exact result, and the other an approximation. Either approach is fine here. Report the p-value and test statistic.

```
# your code here
```

Your p-value and test statistic here

Question 3: 2 points Interpret the results of your statistical test (be specific about what the P-value actually means). Explain what it reveals about the initial question.

Your answer here

Experiment 2: Pepsi vs. Coke

Use the class' data to evaluate whether taste ability differs between frequent, occasional, and non-soda-drinkers. See the summaries at the beginning of this lab, or the background document on Canvas.

Question 4: 1 point What is the null and alternative hypothesis being tested here?

Null hypothesis:

Alternative hypothesis:

Question 5: 2 points Run the appropriate statistical test. Include code for reading in the class' data from a csv. Report the p-value and test statistic.

```
# your code here
```

Your p-value and test statistic here

Question 6: 2 points Interpret the results of your statistical test and explain what it reveals about the initial question.

Your answer here

Plotting data

Question 7: 1 point For experiment2 (Cola), create a plot to illustrate the data. Think about what type of plot (mosaic or paired bar) is appropriate and follow closely that example above, substituting in your data for the example data.

Note: The default plot function will likely not produce a perfect plot. You will need to modify things like: labels, font size, colors, etc to make your plot more readable. This requires adding extra arguments to the plot command (demonstrated in above examples). Your plot will be graded on ease of interpretation (e.g. overlapping labels, axes with dollar signs in the titles, will not receive full credit.)

```
#your code here
```