Chapter 7
Comparing category sizes: Chi-squared tests and beyond

James Myers
2022/4/2 draft

1. Introduction

So far we've mainly looked at statistical methods for continuous-valued data, which is why most of the examples have come from psycholinguistics and phonetics, where lots of measurements are continuous valued. Nevertheless, there are many situations in linguistics where the data are not continuous, but instead are categorical (nominal) and so require a different kind of analysis. Many studies include accuracy measures, which are usually thought of as nominal: right versus wrong. In syntax, acceptability judgments are often assumed to be either/or: either a sentence is OK, or it's impossible. In phonology and morphology, data also usually involve a categorical distinction: either a word type exists in the language, or it doesn't. In sociolinguistics, data often involve the number of tokens in a corpus, which indicate how probable somebody is to say X given various factors (e.g., age and gender).

Sometimes categorical data can be treated as continuous by first transforming it. For example, as we've seen in earlier chapters, accuracy rates represent averages of sets of categorical data like {0, 1, 1, 0, 0, 0, 0, 1, 0, 1}, where 1 = correct and 0 = wrong; the accuracy rate of 40% is the same as the mean of .4. If each unit in such a set (e.g., an answer on a test) is independent of the others, then the distribution is **binomial**, and as we've also seen, the binomial distribution becomes normal with a large enough sample size. This is why you often see linguists and others analyzing accuracy rates with parametric statistics like linear regression or *t* tests, as long as the data aren't too skewed (which they would be, for example, if most of the responses are accurate, showing a **ceiling effect**: the study's task was too easy to be scientifically informative).

Often, however, this sort of transformation is inappropriate. For example, it's not at all obvious how a sociolinguistic study, with a categorical dependent measure and multiple independent variables, could possibly be turned into something parametric. Hence it's important to have special techniques for dealing specifically with categorical data (for an advanced overview of categorical data analysis, see Agresti, 2007).

In this chapter we look at a particular type of categorical data, where the dependent variable is **frequency** counts (0, 1, 2, 3, ...) and the independent variable(s) is/are also categorical (i.e., each variable represents two or more discrete categories, like nouns vs. verbs). Categorical independent variables are often called **factors**, and the categories that make them up are often called **levels** (particularly when discussing ANOVA). So we could have been using

these terms when we discussed *t* tests too. Here, however, the dependent variable is category sizes, which definitely isn't the case for *t* tests.

We actually looked at category sizes before, when we introduced the **binomial test**, which let us compute the *p* value for the null hypothesis that two sample sizes (frequencies) have some expected proportion (e.g., equal, as when flipping a coin). In this chapter we'll see how we can generalize this idea to situations comparing any number of sample sizes, even when the samples can be arranged into a two-way table, with the samples forming the table's **cells**; this is called a **contingency table**. The most familiar and traditional test of this type is the **chi-squared test**. The chi-squared test is named after yet another type of distribution, though it turns out to be related to others we've seen before. Because of its connection with an idealized distribution, it depends on the Central Limit Theorem to work properly, which means you need to have a sufficiently large sample. So we'll also learn some related **exact tests**, which can be used even for small samples.

2. Contingency tables

A **contingency table** (列聯表) cross-classifies the counts for all possible outcomes of a set of events, allowing us to see if the frequencies are distributed as "evenly" as the null hypothesis would expect. You might remember how Lilienfeld et al. (2010) discussed something that they call the Great Fourfold Table of Life, which summarizes all the possible ways an observation could match or mismatch a hypothesis about it:

		Hypothesis	
		Pattern	No pattern
Reality	Pattern	**Hit**	**Miss**
	No pattern	**False alarm**	**Correct rejection**

Remember how we pointed out that the human mind is naturally biased to look mainly for hits, mostly ignoring all the other cells. In a proper contingency table however, we have to pay attention to all of the cells, and make sure the cells divide up all of the possibilities completely without overlap (i.e., every possible outcome must fall into exactly one of the cells).

For example, consider the common claim that words in English starting with /sn/ tend to relate to the nose: *sneeze*, *snot*, *snore*, *sneer*. Crucially, to find out if this /sn/ element is really a kind of quasi-morpheme, we need to go beyond merely listing these **hits** (where /sn/ corresponds to the nose) and also list **false alarms** (where /sn/ doesn't refer to the nose, like *snow*), as well as **misses** (non-/sn/ words referring to the nose, like *nasal*) and **correct rejections** (all other types of words, like *banana*). Note that it's crucial to include words that don't contain /sn/ at all, in order to estimate the proportion of nose-related words in general. What we want to know if nose-ness and /sn/-ness tend to go together more often than chance.

To find out, we put these frequencies into a contingency table like Table 1, which counts *all* words of types A, B, C, D, *not* just the type A that inspired the hypothesis in the first place. We won't finish the analysis; see Drellishak (2007) if you're curious.

Table 1. Counting noses

	/sn/	no /sn/
Relates to nose	A	B
Doesn't relate to nose	C	D

What statistical tests like the chi-square test let you do, then, is compute the probability of the null hypothesis relating to such counts, that is, the probability that your observed distribution of frequencies (e.g., A, B, C, D) could have happened by randomly distributing the counts across the table (with the technical details to be explained in due time, of course).

It's always best to make pictures before we get to the numbers, and that starts with reviewing how to generate tables like Table 1 in the first place. Doing this in Excel takes a certain amount of manual labor, but R can do it automatically, and also has some useful graphs specially designed for plotting tables of frequencies.

We've already seen two of R's functions for creating tables from raw data: **table()** and **xtabs()** (for cross-classification tables). Both output the data arranged in the usual R style, with columns identifying the units, independent variables, and dependent variable. We've also seen that they differ in their syntax (as we'll review shortly), and they also turn out to differ in the kind of object that they create.

Let's try both of these functions out on the fake words in **vpat.txt**, to see whether there are any patterns in the combinations of first and second vowels. Before using the table functions, we first extract the first and second vowels into separate columns using the **substring()** function:

```
vpat = read.table("vpat.txt") # header=F, since there's no header this time
colnames(vpat) = "Words" # Change from R's default name "V1" (confusing here)
vpat$Vowel1 = substring(vpat$Words,2,2) # Second position character
vpat$Vowel2 = substring(vpat$Words,4,4) # Fourth position character
head(vpat) # Take a look at the structure of vpat
```

Now we can create a table counting all vowel combinations in two different ways: **xtabs()** creates a matrix object, while **table()** creates a contingency table object. They may look the same on the surface (e.g., when you type their names they show almost identical displays on the screen), but internally, R treats them differently, which will turn out to affect how you plot them and do chi-squared tests on them:

```
# xtabs() creates a matrix using formula syntax
VowelCombos.mat = xtabs(~Vowel1+Vowel2, data=vpat)
VowelCombos.mat # See what it looks like
```

```
        Vowel2
Vowel1   a  i  u
     a  25 18 25
     i   7  1  3
     u   6  7  8
```

```
# table() creates a contingency table from variables
VowelCombos.tab = table(vpat$Vowel1, vpat$Vowel2)
VowelCombos.tab # Looks almost the same (but looks can be deceiving)
```

```
   a  i  u
a 25 18 25
i  7  1  3
u  6  7  8
```

Let's look at this table more carefully. In Table 2, I've flipped it along the diagonal axis so the columns represent the first vowel frequencies and the rows represent the second vowel frequencies. Remember that R almost always orders rows before columns, which is why **xtabs()** and **table()** put the first vowel in the rows by default.

Table 2. Observed vowel frequencies

		First vowel			
		a	i	u	**Total**
Second vowel	a	25	7	6	38
	i	18	1	7	26
	u	25	3	8	36
	Total	68	11	21	100

You can do this matrix flipping in R by using the **t()** function (for "transpose"), and in Excel by using the "paste special" option that comes up when you click the right mouse button, and then selecting 轉置 (E).

```
t(VowelCombos.mat)
```

```
        Vowel1
Vowel2   a  i  u
     a  25  7  6
     i  18  1  7
     u  25  3  8
```

```
t(VowelCombos.tab)
```

```
   a  i  u
a 25  7  6
i 18  1  7
u 25  3  8
```

Note that Table 2 also adds up the row totals and column totals. Technically, these values are called the **marginals**, since they appear in the margins of the table. They're easy to compute in Excel (can you guess how?), and in R you can use the **apply()** function, which we used a few chapters back (try **?apply** if you don't remember; notice that one of its arguments is **MARGIN**):

```
VCM = t(VowelCombos.mat) # Transposed counts (as in Table 2)
VCM.marg = cbind(VCM,RowTotal= apply(VCM,1,sum)) # Add row totals
VCM.marg = rbind(VCM.marg,ColTotal= apply(VCM.marg,2,sum)) # Add col totals
VCM.marg # Take a look at it
```

```
         a  i  u RowTotal
a       25  7  6       38
i       18  1  7       26
u       25  3  8       36
ColTotal 68 11 21      100
```

As we've seen before, one way to plot a frequency table is with a bar plot, as in Figure 1:

```
barplot(VowelCombos.mat,        # table of values
  beside=T,                     # draw bars next to each other, not on top
  names.arg=c("V2=a","V2=i","V2=u"),          # the names at the bottom
  legend.text=c("V1=a","V1=i","V1=u"),        # the names in the legend box
  ylim = c(0,40),               # min & max y-axis (so legend doesn't cover bars...)
  ylab = "Counts"               # y-axis label     (... but it depends on your computer)
)
```

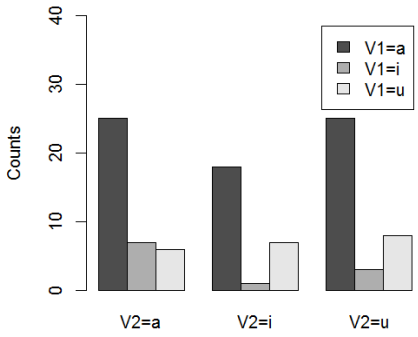

Figure 1. A bar plot of the vowel frequencies

But a bar plot is somewhat confusing to read for contingency values, since what we care about most here is not the specific frequencies, but how the two variables represented by the rows and columns **interact** with each other. In this case, for example, we are particularly interested in figuring out whether there's any relation between the first and second vowels. Interactions are not totally hidden in a bar plot, but they're not as clear as they could be (we'll discuss how to plot interactions a lot more when we get to ANOVA). This is because the bar plot doesn't show the relative proportions of the category sizes, and as you should remember by now, proportions represent probabilities, and probabilities tell you something about the difference between your observed data and random chance.

To plot the relative proportions of all of the cells, relative to each other as well as relative to their marginals, you can use something called a **mosaic plot** (named after the ancient square-tile art of mosaics: 鑲嵌藝術). This represents the total count as a large square that's divided up horizontally and vertically into little tiles representing each of the cells, proportional in size to each other and to the whole. You can see real-life examples of mosiac plots in Zuraw (2010), where they are used to study combinations of phonological properties, similar to what we're doing here with our fake data.

Excel doesn't have any tool for creating mosaic plots, but R does. We can create a mosaic plot from either a matrix or table object, but it's easier to do this using a matrix created by **xtabs()**, since as we saw above, this function preserves the row and column factor names, so they can appear in the plot. So Figure 2 was created using the first command below, not the second one.

mosaicplot(VowelCombos.mat, cex=1, main="VowelCombos") # cex for enlarged font
mosaicplot(VowelCombos.tab, cex=1, main="VowelCombos") # Not as nice

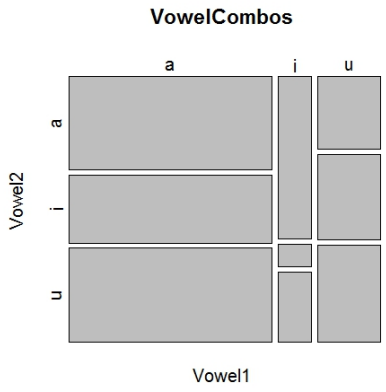

Figure 2. A mosaic plot of the observed vowel frequencies

You can make fancier mosaic plots using the **mosaic()** function in the **vcd** package ("Visualizing Categorical Data"; Friendly & Meyer, 2015) - try it!

```
library(vcd) # Yes, you have to download and install it first
mosaic(VowelCombos.mat, highlighting="Vowel1")
mosaic(VowelCombos.mat,highlighting="Vowel2",
  highlighting_fill=c("blue","white","red"))
```

Anyway, even just looking at the plain mosaic plot in Figure 2, it's now obvious that /a/ is consistently favored in the first syllable (the wide blocks on the left) but not in the second (roughly equal height blocks on the left and right edges). We can also see the hint of an interaction: /a/ seems to be disproportionately favored in the second syllable if the first syllable has /i/ (the tall block in the top center).

What would the plot have looked like if the vowels were distributed "evenly", while still preserving the actual marginals that we observed? If we were looking at just one variable, say just the first vowel categories, this would be easy to figure out: just divide the total number of first vowel tokens (here, 100, since we have 100 words) by the number of first vowel types (here, three, for /a/, /i/, /u/), so each cell would have an expected chance frequency of $100/3 \approx 33.3$. But in this case we have a two-way table, and it's not so obvious how to do this while preserving all of the marginal values. So let me just tell you the right answer: the evenly distributed table would look like Table 3 (later I'll explain how I did this magic):

Table 3. Expected vowel counts, if they were distributed evenly

		First vowel			
		a	i	u	**Total**
Second vowel	a	25.84	4.18	7.98	38
	i	17.68	2.86	5.46	26
	u	24.48	3.96	7.56	36
	Total	68	11	21	100

Notice that the marginals haven't changed; for example, in the first row, **=SUM(25.84,4.18,7.98)** is still 38 (try it!). It's like some kind weird sudoku (數獨) game. Note in particular that the marginal frequency for the first vowel /i/ remains 11, the lowest marginal frequency: /i/ still doesn't like to appear in the first syllable. This is possible because Table 3 shows **conditional** probabilities, that is, chance frequencies for the cells *given* the marginals, which are not distributed evenly.

Now let's make a mosaic plot of this. The result is in Figure 3: see how the mosaic "tiles" all line up evenly now, even though /i/ is still the least popular first vowel.

```
VowelCombos.exp = VowelCombos.mat # To preserve the column and row names
VowelCombos.exp [,] = (matrix(c(      25.84, 4.18, 7.98,
                                      17.68, 2.86, 5.46,
                                      24.48, 3.96, 7.56), ncol=3)) # See how I did this?
VowelCombos.exp # Take a look at our expected values (transposed from Table 3)
```

```
        Vowel2
Vowel1      a     i     u
     a  25.84 17.68 24.48
     i   4.18  2.86  3.96
     u   7.98  5.46  7.56
```

```
mosaicplot(VowelCombos.exp, cex=1, main="VowelCombos (Expected)")
```

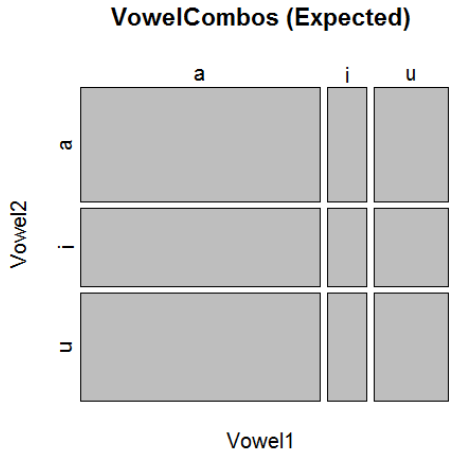

Figure 3. A mosaic plot of the expected evenly distributed vowel counts

3. Chi-squared tests

The mosaic plots in Figures 2 and 3 look a bit different, but are they different enough to count as statistically significant? That's the kind of question that the chi-squared test can answer.

The **chi-squared test** (also known as the **chi-square test**, I guess for phonological reasons) is quite old, older than the *t* test in fact, having been invented by Karl Pearson (the same guy who invented Pearson's *r* for correlation). It's named after yet another distribution called the **chi-squared distribution**, since it's symbolized by χ^2, with the Greek letter χ, which in ancient Greek represented an aspirated voiceless velar stop (IPA /k^h/) but in modern Greek represents a voiceless velar fricative (IPA /x/). The English name for this letter is spelled "chi", but following ancient Greek, it's pronounced /kai/ (using a modern English pronunciation for the vowel). In R, which displays only Roman letters in its outputs (unless you work hard to make it do otherwise), χ appears as "X".

Despite the mysterious name, the chi-squared (χ^2) distribution turns out to be closely related to the normal distribution (again!). The chi-square test is still a **nonparametric** test, though, since it doesn't depend on computing means or variances (the parameters of the normal distribution), which of course don't make sense for category sizes anyway.

The purpose of a chi-squared test is to compare the relative sizes of nominal categories, either comparing a row of observed categorical data with a specific null hypothesis in a **one-way** test (also known as the **chi-squared test for goodness of fit**), or testing a two-way contingency table (also known as the **chi-squared test for independence**).

As the names imply, the one-way and two-way chi-squared tests are testing different kinds of null hypotheses. In a one-way chi-squared test, there is only one factor, and the null

hypothesis is that the frequencies for all of the levels of this factor are the same, or that they fit some pre-specified frequencies. I haven't given any examples of this type yet, but I will soon.

In a two-way chi-squared test, there are two factors, and the null hypothesis is that the two factors do not interact, that is, that the proportions of counts within each factor aren't affected by the proportions in the other factor. This is the kind of analysis we were discussing in the examples above, when we looked at the contingency tables for the /sn/ hypothesis and the **vpat.txt** vowel combinations.

Note that a two-way chi-square does *not* test for the "main effects" of each of the factors, only for their interaction. So we could use it to test if the first and second vowels in **vpat.txt** interact (e.g., show assimilation or dissimilation), but to test if the first-vowel /i/ pattern by itself is significant, we would have to run a separate one-way chi-square test. That should be legal, since the two-way test takes the marginals as given, so the probabilities within the marginals are independent of the cell probabilities, allowing us to treat them as a separate data set that we can run a separate test on (no worries about Type I errors).

The basic idea behind both types of chi-squared tests is to compare the probabilities in the observed frequencies with those for the frequencies that are expected if they were evenly distributed. Let's try this out, starting with one-way chi-squared tests, first working through some examples in Excel and R, and then explaining the math behind them.

3.1 One-way chi-squared tests

Booboo is a baby Martian. She's in the two-word stage, and she can say 200 nouns, 100 verbs, and no other kinds of words. The null hypothesis is that she combines these words at random. You take a random sample of 100 of Booboo's utterances, and Table 4 shows the counts that you get (where "NV" means a sequence of a noun and a verb, and so on):

Table 4. Booboo's observed frequencies

Utterance type	NV	VN	NN	VV
Observed counts	24	21	43	12

Some careful thought (try it yourself) shows that according to the null hypothesis, we would not expect these four cells to have identical values (i.e., they won't all be 100/4 = 25), because there are twice as many nouns as verbs. The chance probability of choosing a noun is actually 200/(200+100) = 2/3 and the chance probability of choosing a verb is 1/3, and so by the multiplication rule of probability, the chance probability for the sequence NV is (2/3)(1/3) = 2/9. The rest of the probabilities are calculated with the same logic, resulting in Table 5.

Table 5. Booboo's expected chance probabilities

Utterance type	NV	VN	NN	VV
Expected probabilities	2/9	2/9	4/9	1/9

These probabilities translate into the following expected numbers of utterances, as a proportion of the total of 100, as in Table 6. Since this table is just for our own internal purposes, not for displaying in a paper or talk, I didn't round the values, so that the calculations I'll do next won't get affected.

Table 6. Booboo's expected chance frequencies

Utterance type	NV	VN	NN	VV
Expected counts	22.22...	22.22...	44.44...	11.11...

It should be obvious that these expected chance counts are extremely close to the observed counts, so we shouldn't be surprised to find out that there's no significant difference. But how can we actually calculate this?

3.1.1 One-way chi-squared tests using Excel

Excel doesn't have a built-in chi-square tool in the Analysis ToolPak, so if you use Excel, you have to do a lot of the work by hand. Fortunately, I've already done this work for you in the file **chi2.xlsx**, which also shows you how the math works (look at the cell functions I used). If you select the sheet called "one-way", you'll see that the observed and expected Booboo counts are already in there, but if you want to run a one-way test on different data, just clear the values in the cells with the thick black borders and fill them in with your values. Note that one of the things you have to do by hand is enter the values expected by chance, since only you know what they are (they may not be evenly distributed, just as they aren't in the Booboo example). Don't screw around with the values that aren't in the thick-bordered cells, or else you'll delete my lovely calculations.

The results of the chi-squared test are given in the bolded (but not boxed) report at the bottom of the "one-way" sheet:

chi2= 0.3275
chi2 (Yates)= N/A
df= 3
p= 0.954776156

You should be able to figure out what most of this means. The first value is the value of the **test statistic**, the number you use to determine if your sample is an outlier in the null hypothesis population. For the *t* test the test statistic was *t*, so can you guess what the test statistic is for the chi-squared test? Yes, you win: it's χ^2. Because it's a square, this value can never go below zero, and like most test statistics, higher values mean that you're further out along the distribution, where the area of the tail is smaller, and so the *p* value is lower. Since the χ^2 value has no upper limit, this value doesn't look very big.

I'll explain what "Yates" means later, but it's not applicable ("N/A") anyway.

Next we have *df*, for the degrees of freedom. Like the *t* distribution family, the χ^2 distribution also has a family, with different shapes depending on the degrees of freedom, which as usual are based in some way on your sample size and the particular type of analysis you are doing (e.g., remember that the unpaired and paired *t* tests use different *df* values). Here the *df* is 3. Why? Can you guess? Isn't *df* often sample size minus one? So what we do have here with four somethings? Not the total sample, which is made of 100 observations. But we do have four of something else: the number of categories (NV, VN, VV, NN). So you guessed it: for the one-way chi-squared test, $df = C - 1$, where C = number of categories (i.e., number of levels in your factor).

Finally, we get our *p* value. If you click on the cell with this *p* value, you'll see that I computed it with the help of Excel's function **=CHIDIST(χ^2, *df*)**, which computes the *p* value for a χ^2 distribution, based on the χ^2 test statistic and the *df*. Because I didn't use Yates's correction (which I'll explain later), you can get the same *p* value more directly with the function **=CHITEST(observed, expected)**, where the first argument is the set of observed counts and the second is the set of expected chance counts (try it!). But this method is not as useful, since as with all statistical tests, you need to report more than just the *p* value.

In any case, this *p* value is almost one. Thus, to nobody's surprise, the Booboo data do not show a pattern significantly different from chance. We can report this null result like so: $\chi^2(3) = 0.33$, $p = .95$, $n = 100$. Note the familiar syntax: *test_statistic_name*(*df*) = *test_statistic_value*, *p* = *p_value*.

Note also that we need to mention the actual sample size (*n*), since even though it doesn't affect the *df*, sample size still has a big effect on the chi-squared test. You can see that this is true simply by typing a zero at the end of all of Booboo's observed counts (i.e., multiplying them by 10). The **chi2.xlsx** file automatically updates the estimated counts, the test statistic, and the *p* value. Try it, and you'll have to modify your report like so: $\chi^2(3) = 3.275$, $p = .35$, $n = 1000$. It's still not significant, but it's looking better for Booboo (lower *p* value): once again, we can see that we get greater statistical power with larger samples, even for a tiny effect size like this.

3.1.2 One-way chi-squared tests in R

You get the same results for this one-way chi-squared test using R's **chisq.test()** function. Once still have to compute the expected Booboo probabilities by hand and enter them into the function. That it will give you a little text report:

```
booboo = c(24, 21, 43, 12) # Observed counts
chance = c(2/9, 2/9, 4/9, 1/9) # Expected probabilities for the Booboo example
chisq.test(booboo, p = chance)    # p = expected probabilities (changing the default)
```

```
        Chi-squared test for given probabilities

data:  booboo
X-squared = 0.3275, df = 3, p-value = 0.9548
```

If you type **?chisq.test**, you'll see that by default, **chisq.test()** computes the chance probabilities assuming that you want all cells to be equal (which we didn't use here, since it's not true for the Booboo case). Thus the default value this is **p = rep(1/length(x), length(x))**. For example, in our case, where **x = booboo**, the default expected probabilities would be **c(0.25, 0.25, 0.25, 0.25)**. Since we didn't want those we had to enter the proper expected probabilities by hand.

3.1.3 Chi-squared cautions

Even though the chi-squared test is simple to do, you still need to be careful.

First, it works on contingency tables containing count data. In this case, we only have a one-dimensional table, but the same logic will apply in two-way chi-squared tests too. Thus even though it's common to see tables in papers where the counts are supplemented with, or even replaced by, proportions, the values you plug into the chi-squared test must be the actual counts. In the Booboo example, the total number of observations was 100, so the counts look like percentages, but they are actually counts, and you have to use the counts. Can you see why? Right, because sample size (n) matters, not just the proportions.

Second, like most the tests we've seen so far, the chi-squared test assumes that all of your observations are independent. The only tests that we've seen so far that don't assume complete independence, namely correlation/regression and the paired t test, instead assume that your data come in known groupings (e.g., pairs), so that their non-independence can be factored out of the analysis. But in a chi-squared test, there should be no groupings at all. In the case of the Booboo data, for example, our calculations assume that we made those one hundred word counts separately from each other (e.g., it wasn't the case that some of the verbs were derived from the nouns or vice versa). Of course all of these values aren't fully independent, since they

all come from the same source (Booboo), but that's OK, since our sample of Booboo data is only intended to test hypotheses about Booboo, so the population is something like "all of Booboo's utterances", not "all Martian children's utterances".

Third, since the chi-squared test depends on one of those idealized distribution things, it also depends on using a sufficiently large sample size. In particular, the rule of thumb is that you shouldn't use a chi-squared test if any of the *expected* values is below 5 (some experts are stricter, and say that falling below 10 is already risky). Notice that this rule refers to the expected values, not your observed values. So if your null hypothesis is that two counts will be identical (say the number of nouns and verbs in some data set), and you observe that there are actually 998 nouns but only 2 verbs, that's OK, since the null hypothesis would give you the expected chance values of 500 nouns and 500 verbs. But if you observed 500 nouns and 500 verbs, but your null hypothesis (for some reason) was that you should get 998 nouns and 2 verbs by chance, then the chi-squared test would not give you reliable results, since 2 is less than 5.

Finally, the sample size issue also applies to the number of cells, though it is only relevant in the two-way chi-squared test. We'll come back to this when I finally explain Yates's correction.

3.1.4 Other linguistic applications of one-way chi-squared tests

Linguists have used one-way chi-squared tests in a variety of clever ways. Befitting its alternative name (chi-squared test for goodness of fit), Guy (1991) used it to test a hypothesis that he had about how language variation interfaces with formal grammar. He started by observing that the probability of deleting /t/ at the end of English words is lower if /t/ is a suffix (e.g., deletion is rarer in *passed* than in *past*). He then hypothesized that this follows from a formal model where the /t/-deletion rule applies at equal rates at the stem and word levels, so it has more chances to apply in *past* (both stem and word level) than in *passed* (only word level), due to the multiplication rule. That is, if the probability of /t/ being retained (not deleted) at any particular level is p_r, then the probability of it being retained on the surface if its the suffix *-ed* will simply be p_r, since this suffix is only added at this late level. By contrast, if /t/ is part of the stem morpheme, the probability of being retained all the way to the surface would be p_r^L where L is the number of levels you have to go through to get to the surface (by the multiplication rule); /t/ can only be retained on the surface if it's retained at all of the levels. Since p_r is a probability, it's less than 1, so multiplying it by itself just makes the overall probability lower, thus increasing the deletion rate.

This logic led Guy to make a specific numerical prediction about what the /t/ retention rates should be for various types of English words. He then compared the counts of deleted/retained /t/ predicted by his model with the actual counts using a one-way chi-squared

test, and found $p > .9$. This was *good* news for his hypothesis. Do you see why? It's because such a high p value gives him no reason to reject his null hypothesis, namely that the observed and theoretically expected values were the same. Goodness of fit tests are thus a rare example in traditional (non-Bayesian) statistics where your null hypothesis is the same as your theoretically interesting hypothesis. (R has a function specifically for computing goodness of fit for count data; check **?goodfit()** if you're curious.)

3.2 The two-way chi-squared test

Suppose you're a sociolinguist, and you've counted variations in the pronunciation of the /ŋ/ coda in Taiwan Mandarin as a function of the preceding vowel. Table 7 shows an example of data relevant to this kind of situation. This is a proper contingency table, because you're focusing on two rime inputs and two possible output classes, and thus you have divided up the data into all possible non-overlapping subtypes, so that the row and column marginals add up properly to the grand total.

As noted above, the numbers that you're analyzing must also be the actual *sizes* of the categories (i.e., the observed counts of items in each cell), not the *proportions* of the total, even if you also want to show your readers the percentages in another table, or want to plot the proportions in a mosaic plot.

Table 7. A contingency table for vowel-consonant interactions in Taiwan Mandarin

rime	/aŋ/	/iŋ/	totals
[n] tokens	57	112	169
[ŋ] tokens	190	11	201
totals	247	123	**370**

Speaking of mosaic plots, does the above data set look like it might be significant? There does seem to be a kind of interaction between the factors of vowel (columns) and consonant (rows), in that the numbers in the cells are bigger for the lower left cell (190) and the upper right cell (112), while the main diagonal values are smaller (57 and 11). But this impression might be confused by the differences in the sizes of the marginal values as well, so let's make a mosaic plot, as in Figure 4 (since we're just plotting this for ourselves, I won't bother making it look nice):

```
socdata = matrix(c(57, 112, 190, 11), ncol = 2)
mosaicplot(socdata)
```

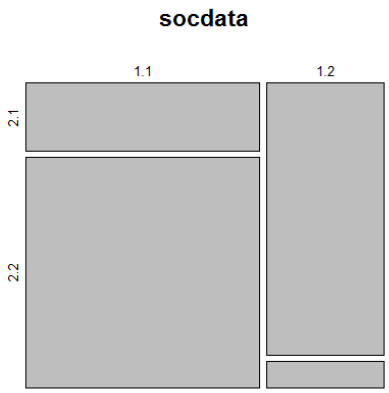

Figure 4. An ugly mosaic plot to get a feel for the sociolinguistic data

That looks pretty uneven to me; I'll bet it's statistically significant by a two-way chi-squared test. I'm especially confident about this because the total number of observations seems pretty high (n = 370), so the power of the test will be pretty strong, so such a big-looking effect seems more likely to come out significant.

3.2.1 Two-way chi-squared tests using Excel

Here goes, starting with the **chi2.xlsx** file. Take a look at the "two-way" sheet. Let's fill in the observed numbers ourselves; we can even rename the rows and columns. But because our null hypothesis is that the counts are "evenly" distributed (while keeping the marginals the same as what we observe), we can let Excel compute the expected chance counts by itself. Thus we only have to type in the values in the thick-bordered cells in the upper table.

According to **chi2.xlsx**, the expected chance values are as in Table 8. Note that the marginal row and column totals remain the same as for the observations, as does the grand total.

Table 8. The expected chance counts for vowel-consonant interactions in Taiwan Mandarin

rime	/aŋ/	/iŋ/	totals
[n] tokens (expected)	112.81892	56.181081	169
[ŋ] tokens (expected)	134.18108	66.818919	201
totals	247	123	**370**

Now we can make our intuitions about the vowel-consonant interaction a bit more concrete. For example, we can easily see now that the difference between [an] vs. [aŋ] as

expected by chance (about 113 vs. 134) is much smaller than what was actually observed (57 vs. 190): the chance distribution is much more even than the real observations.

The bottom of the Excel sheet reports the final results:

chi2= N/A
chi2 (Yates) = 150.1992327
df= 1
p= 1.56824E-34

The first thing you might notice is that this time we do need Yates's correction, and this is because each factor has only two levels (**chi2.xlsx** automatically determines this for you; click on the cells to see how I did this using a simple logic function). This means that Excel's built-in **=CHITEST()** function will give you the wrong *p* value, so I computed it the correct way here. The χ^2 value is very large, even with Yates's correction, so it's not surprising that the *p* value is very small ($1.6/10^{34}$); this would be true for any degrees of freedom, even one as small as the $df = 1$ value we have here. Where does this *df* value come from? Well, how many rows do we have? Two. How many columns? Also two. Any way to take two and two and get one, while subtracting stuff somewhere along the way? Give up? For the two-way chi-squared test, $df = (r - 1) \times (c - 1)$, where r = number of rows and c = number of columns. So here we have $(2-1) \times (2-1) = 1$.

So here's how we could report our results: $\chi^2(1) = 150.199$, $p < .001$, $n = 370$. Our intuitions are right: the observed pattern is extremely unlikely to have happened by chance.

3.2.2 Two-way chi-squared tests in R

R can also do this, of course. As with the Excel implementation, we don't have to specify our chance probabilities, since we want them to be evenly distributed, which R assumes by default. Note that R also applies Yates's correction automatically, and thus it gives the same results as my hand-written **chi2.xlsx** file.

```
chisq.test(socdata)
```

```
        Pearson's Chi-squared test with Yates's continuity correction

data:  socdata
X-squared = 150.2, df = 1, p-value < 2.2e-16
```

But maybe you like doing things the wrong way, or you hate that Yates guy. If so, you can turn off Yates's correction like so:

```
chisq.test(socdata, correct = F) # Ha ha! "Correct" is "false"!
```

This gives you a slightly higher chi-squared value of 152.93, so your *p* value is even smaller; not that it matters here, given how tiny our *p* value was already. The reason why turning off Yates's correction increases significance is that it's a correction to avoid Type I errors (a "false alarm", where you mistake chance for a real pattern), because, as the R output hints, the chi-squared test uses the continuous χ^2 distribution, so if you only have two cell frequencies along each dimension, those two discrete values give a particularly poor fit for a continuous distribution.

However, R doesn't seem to worry too much about Yates, as shown by an odd little quirk. Namely, it turns out that when our old friend, the **table()** function, creates a table object, one of the implicit properties of this object is a chi-squared value. You can see this value if you apply the **summary()** function to a table object. (The **summary()** function is like **plot()** in being a so-called **generic** function: the way it behaves depends on the argument you apply it to; summarizing a table shows you something very different from summarizing a linear model, for example.) So let's turn the matrix **socdata** into a table object using the **as.table()** function, and then put this inside **summary()**. This tells us a bunch of interesting things about our table, among which is the results of a two-way chi-squared test, but *without* the Yates's correction, for some reason.

summary(as.table(socdata))

```
Number of cases in table: 370
Number of factors: 2
Test for independence of all factors:
        Chisq = 152.93, df = 1, p-value = 3.975e-35
```

Just to drive this trivial point deep into the ground, we can also run two-way chi-squared tests on the **vpat.txt** data both ways (Yates's correction is irrelevant here, since the factors have three levels). Try it!

chisq.test(VowelCombos.mat) # X-squared = 4.3403, df = 4, p-value = 0.3619
summary(VowelCombos.tab) # Chisq = 4.34, df = 4, p-value = 0.3619

Notice that the vowel pattern turns out not to be statistically significant; the apparent asymmetry in the /i/ counts across different vowel contexts were too likely to have happened by chance alone. And look at those *df* values: do they make sense? Yes: we have three rows and three columns, so $df = (3\text{-}1) \times (3\text{-}1) = 2 \times 2 = 4$.

But why is R giving us a warning? It says: **Chi-squared approximation may be incorrect**. Now, this is just a "warning", not an "error", so we can decide to ignore it if we like. But why is R worried at all?

You can find out why by running the same analysis in **chi2.xlsx** and looking at the expected frequencies, or by looking at these frequencies in the object created by **chisq.test()**:

chisq.test(VowelCombos.mat)$expected

	Vowel2		
Vowel1	a	i	u
a	25.84	17.68	24.48
i	4.18	2.86	3.96
u	7.98	5.46	7.56

Note the chance frequencies expected for /i/ in first position are all less than 5. This cut-off point is used as a rule of thumb indicating that it's a bit risky to rely on the sample-size-sensitive chi-squared test for valid *p* values. I wouldn't worry so much in this case, though, since the *p* value is nowhere near ambiguous. That is, it's not right on the edge of .05, but instead is much higher, so even if the *p* value is slightly wrong, it's unlikely to go below .05 and change our conclusions. We can thus be pretty sure that the result is truly non-significant here. Later in the chapter, we'll confirm this by redoing the analysis using an exact test.

By the way, when you want to do a two-way chi-square but your data are in a data frame rather than a contingency table, you don't actually have to create the table first (using **table()** or **xtabs()** as we did above). It turns out that the function **chisq.test()** can do this job for you. For example, here's how we could analyze the vowel patterns in the **vpat** data frame.

chisq.test(vpat$Vowel1,vpat$Vowel2) # Same results (and warning) as above

Among the objects created by **chisq.test()** are the observed and expected matrices - try it! After all, you may want to give a table of the observed values in your research paper anyway, and this trick kind of saves you a step.

vpat.chsq = chisq.test(vpat$Vowel1,vpat$Vowel2)
vpat.chsq$observed
vpat.chsq$expected

3.2.3 Other linguistic applications of two-way chi-squared tests

As mentioned above, the two-way chi-squared test is also known as the chi-squared test for independence, since the null hypothesis is that the two factors don't interact at all. This makes it useful in many situations where we want to test for this kind of interaction. For example, one of the many quantitative methods for typological research reviewed by Cysouw (2005) uses two-way chi-squared tests to examine Greenberg-style implicational universals.

So if we want to know if languages with classifiers tend not to have grammatical gender and vice versa, we could count (independently sampled) languages of the relevant types, and enter their counts into a contingency table like that in Table 9. Then we could use a two-way chi-squared test to see if the two factors interact with each other, in particular that the counts for cells A and D are significantly lower than would be expected by chance.

Table 9. An abstract contingency table for testing a typological hypothesis

Counts of languages	Classifiers	No classifiers
Gender	A	B
No gender	C	D

Another common application of two-way chi-squared tests is as a test of **collocations** (e.g., Manning & Schütze, 1999) of word combinations in a corpus (e.g., "happy" and "birthday" often combine in English, since "happy birthday" is a set expression). If two words are quite common, it will also be quite common for them to appear near each other, just by chance alone; we don't want to count that as a collocation. Thus to test for collocations properly, we need to view our word combination in a wider context that takes all four logically possible contingencies into account, as reflected in Table 10:

Table 10. An abstract contingency table for testing a corpus collocation

		Word X	
Counts of word combinations		Present	Absent
Word Y	Present	A	B
	Absent	C	D

Gries (2005) uses a version of this logic for a psycholinguistic purpose, namely to look for syntactic priming (related/unrelated prime-target pairs) within a corpus (which, after all, reflects language production, and thus is psycholinguistic data), though because his frequencies are often quite small, he uses an exact test rather than the two-way chi-squared test.

3.3 More about the math of chi-squared tests

How does all of this work? It's quite simple, really. The chi-squared test statistic is calculated as follows, where f_o = the observed frequency for each cell and f_e = the expected frequency for that cell (i.e., the frequency expected according to your hypothesis, or by chance), all summed up. This works whether you're doing a one-way or a two-way chi-squared test.

Chi-squared test statistic: $\chi^2 = \sum \frac{(f_o - f_e)^2}{f_e}$

Look at that formula carefully. First of all, we can see that if $f_o = f_e$, where the observed and expected frequencies are identical, then the chi-squared value will be zero, which obviously ain't gonna be statistically significant.

More generally, χ^2 is a sum of the differences (squared as usual) between the observed and expected frequencies, treated as a proportion of the expected values, since the expected values represent the null hypothesis. This is rather similar to the *z* test formula (which gives rise to the formulas for the various *t* tests), since the numerator (at the top of the ratio) represents the difference between your observations and the null hypothesis (cf. $M - \mu$ for the *z* and *t* tests), and the denominator (at the bottom of the ratio) represents something about the null distribution (*SE* in the case of the *z* and *t* tests, f_e here).

One difference from the *z* test formula is that the numerator is squared, and in fact, the chi-squared distribution is related to the normal distribution in a square-ish kind of way, sort of like how the *F* distribution is related to the *t* distribution, which is also related to the normal distribution. Putting these points together, $F(df_1,df_2)$ gets closer and closer to $\chi^2(df_1)$ as df_2 gets closer and closer to infinity.

This is why χ^2 distributions even look kind of like *F* distributions, since neither can go below zero, giving them an infinitely long tail on the right, and a mode that moves toward the right too as *df* gets larger, as shown in Figure 5. This family was plotted using the density (**d...**) member of the **...chisq()** family of R functions:

```
plot(function(x) {dchisq(x, df=3)}, xlim=c(0,20), ylab = "", xlab = "")
plot(function(x) {dchisq(x, df=5)}, xlim=c(0,20), add=T, lty=2)
plot(function(x) {dchisq(x, df=10)}, xlim=c(0,20), add=T, lty=3)
legend("topright", lty=c(1,2,3), legend=c("df=3", "df=5", "df=10"))
```

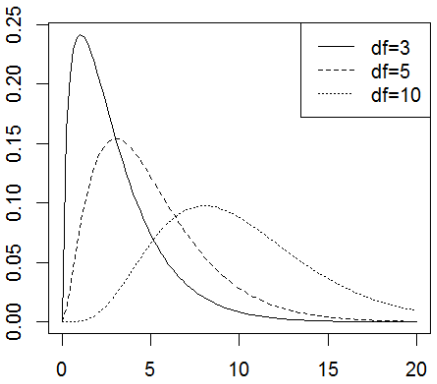

Figure 5. Some chi-squared distributions

After your computer calculates the χ^2 value for your particular data, you can then do the usual: find that point on the appropriate χ^2 distribution given your *df*, and that will mark off an area in the right tail that represents the *p* value. Unlike the *t* and *F* tests, it only makes sense to get a one-tailed *p* value here. Do you see why? It's because the χ^2 value is calculated as the sum of the squared **deviances** $(f_o\text{-}f_e)^2$ over the expected frequencies f_e, so higher values always indicate a greater deviation from the expectation. The higher this value, the smaller the area in the right tail; the left tail never comes into it because you can't get a smaller deviation than no deviation.

You can compute the (one-tailed, right-sided) *p* values "by hand" using Excel's function **=CHIDIST(χ^2, *df*)** (which gives you the tail to the right) or in R, using **pchisq(χ^2, *df*, lower.tail = F)** or **1-pchisq(χ^2, *df*)**, since by default, R's **pchisq()** function gives you the cumulative probability, starting from the left (just as it does with all distributions, even though it doesn't make sense here).

In order to use the above formula, however, you need to have the expected frequencies (f_e). When using the one-way chi-squared test to test the fit of your observations with a stipulated set of expected values (as we did with the Booboo analysis), you, the human, simply tell your computer those expected values, whatever you want them to be. When using the one-way chi-squared test to compare with evenly distributed counts, you simply divide the number of counts by the number of cells, as we've seen.

For a two-way chi-squared test, though, computing evenly distributed chance frequencies by hand is more annoying. You want the marginals to stay the same for both the observed and estimated tables, but somehow you have to do this for both row totals and column totals. Fortunately, there's a clever formula for this, as shown below, where f_e = expected frequency in a cell, f_c = column total for that cell, f_r = row total for that cell, and n = grand total (i.e., the sample size). This is the formula used by **chi2.xlsx** in the "two-way" sheet.

Estimated frequencies for two-way chi-squared test: $f_e = \frac{f_c f_r}{n}$

As for computing *df*, it's equally simple, but note that unlike the parametric tests we've seen, these formulas are *not* based on the number of observations, but on the number of cells:

one-way chi-squared *df*: $df = C - 1$ [C = number of cells]
two-way chi-squared *df*: $df = (R - 1)(C - 1)$ [C = number of columns, R = number of rows]

By the way, the deviances, which the **chi2.xlsx** file shows in their own table, provide useful information by themselves, since they reflect how deviant each individual cell is, giving us some sense of where the chi-squared value comes from. That is, if the chi-squared test is

significant, the cell with the largest deviance corresponds to the cell in our data that is "most responsible" for this significant result. We'll return to the concept of deviances again much later, when we learn how to apply regression techniques to categorical data.

Finally, I know you're dying to know: what exactly is **Yates's correction**? Well, it was invented by a British statistician named Frank Yates (1902-1994). He suggested that for small tables, as in a 2×2 two-way chi-squared test, you should subtract a tiny amount from the deviances, thereby making them smaller, as if your observations were actually more like chance. The reason is that when you have a contingency table of this small size, then $df = 1$, which is the most skewed member of the entire chi-squared family, and its tail is super skinny. Exactly how skinny is too skinny appears to be a matter of taste, as we've already seen, given that in R, **summary(table())** ignores Yates's correction entirely. In any case, in **chi2.xlsx** and in R's **chisq.test()**, the value of 1/2 (0.5) is subtracted to punish you for that skinny tail, so your *p* value ends up a bit higher as Yates claimed that it should. So the corrected chi-squared test statistic looks like so:

Chi-squared test statistic (Yates's correction): $\chi^2 = \sum \frac{(f_o - f_e - 0.5)^2}{f_e}$

3.4 The two-way chi-squared test as a nonparametric alternative to *t* tests

The chi-squared test has also sorts of clever applications. As just one example, consider the situation where we want to do something like an unpaired *t* test on two unrelated samples, but we want to compare the medians rather than the means, maybe because our distributions are highly skewed. In case you forgot what a median is, it's the midway point in your distribution (or the mean of the two midway points, if you have an even number of observations). How could we possibly do this median-oriented version of the mean-oriented unpaired *t* test?

Well, we can do it with something called the **median test**. Under the null hypothesis, both samples come from the same population, with a single common median, so each separate sample should be divided in half by this common median (i.e., the median for all the items collected from both samples). Think about this for a moment, and you'll see that this means that you can divide up your observed data as in Table 11. Now we just apply a two-way chi-squared test (yes, with Yates's correction), to see if the cell counts A, B, C, D are distributed evenly.

Table 11. Counting data points for the median test

	sample 1	sample 2
number of items above common median	A	B
number of items below common median	C	D

As another clever application, what can we do if we want to compare sample medians when the data come in pairs (i.e., a median-oriented version of the mean-oriented paired *t* test)? Once again the chi-squared distribution comes into play. The relevant test is called the **sign test**, or **McNemar's test** (named after the American psychologist Quinn McNemar [1901-1986]). Let me illustrate how it works with a syntax judgment experiment with pairs of matched sentences that you hypothesize to differ in acceptability. That is, for every supposedly grammatical sentence in the test set, there is a supposedly ungrammatical sentence that is otherwise identical to it (e.g., "You are here" vs. "You is here").

The results of such an experiment can be summarized as in the contingency table in Table 12, where A and D count the sentence pairs that received the same judgment, and B and C count sentence pairs that received different judgments, either in favor of the hypothesis (B) or against it (C):

Table 12. Pairs of sentence judgments

		Judgment for grammatical sentence	
		unacceptable	acceptable
Judgment for ungrammatical sentence	unacceptable	A	**B**
	acceptable	**C**	D

To get the test statistic for McNemar's test, you use the frequencies in just the two cells along the so-called **antidiagonal**, namely B and C, which represent the two crucial outcomes for testing our hypothesis, since B is the predicted outcome, and C is the reverse of this. The outcomes in A and D are useless for testing your hypothesis, since they represent pairs where both judgments are the same, so they basically provide no information.

It turns out that if we put the values of B and C into the formula below, the distribution is roughly normal, so we end up with a one-sample *z* test:

$$z = \frac{B - C}{\sqrt{B + C}}$$

But consistent with the idea that all statistical tests are related, if we square both sides of this formula, as below, we also get a test statistic that forms a close fit to the χ^2 distribution, with $df = 1$.

Test statistic for McNemar's test: $\chi^2(1) = \frac{(B-C)^2}{B+C}$

You may have noticed that our contingency table is 2 × 2 again, so - you guessed it - Yates's correction is recommended, so we should subtract something. For some reason, by convention what you subtract here is 1 rather than 1/2:

Test statistic for McNemar's test (Yates's correction): $\chi^2(1) = \frac{(|B-C|-1)^2}{B+C}$

Now you know enough to write your own Excel file or R function for computing the McNemar test. But I won't work out an actual example with these formulas, since as we'll see shortly, exact tests are easier to use and work even better.

4. Exact tests for contingency tables

Though chi-squared tests and McNemar's test are nonparametric and designed for nominal data, they are still **asymptotic** (漸近的) tests: they require a sufficiently large sample for our null hypothesis to be well modeled by the idealized χ^2 distribution (remember that the Central Limit Theorem depends on samples being large enough). That's why the expected cell size for a chi-squared test shouldn't go below 5 (or 10). Sometimes, however, we are forced to deal with very small samples. For example, maybe we're doing fieldwork on an endangered language with few speakers, who may be too old to provide a lot of utterances. Or we're doing a phonological or morphological analysis of a dead language, and the dictionary just doesn't have very many words of the type we're interested in.

In such cases, we want **exact tests**: tests that compute the *p* value as a proportion of the *actual* number of outcomes like what was observed divided by the *actual* total number of possible outcomes (not just an estimate of this proportion). We've already seen a few examples of these in earlier chapters, including the binomial test and the related method of **resampling**, where we get as close as we can to the exact *p* values by trying hundreds or thousands of samples or **permutations** (in case the actual values needed for the probability ratio are too big to count completely).

Let's look at some examples of these kinds of tests for contingency tables.

4.1 Fisher's exact test

The most famous exact test is **Fisher's exact test** (yes, *that* Fisher again). It is essentially the exact version of a two-way chi-square test, but it uses permutations instead of the χ^2 distribution.

Fisher (1935) was such a genius that he invented it almost as a joke, more to illustrate the logic of hypothesis testing than to propose an actual statistical test. This is where Salsburg's (2001) book gets its strange title from, by the way. One of Fisher's colleagues (the "lady" of Salsburg's title) claimed that she could taste the difference between milk tea when the tea was added to the cup first versus when the milk was added first (you can tell these guys were British). To test her, Fisher gave her a random series of eight cups of milk tea, four with the tea added first and four with the milk added first, and she had to guess which was which. As it happened, she correctly identified the mixture in all eight tries (though Fisher, 1935, doesn't mention this, maybe out of embarrassment; the true story was recorded by other people who were there). This forms the contingency table of observed frequencies in Table 13 (note the letter names in the cells, which we'll need a moment).

Table 13. Fisher's milk tea experiment

	Tea really added first	Milk really added first	Total
Lady says tea added first	4 (A)	0 (B)	4
Lady says milk added first	0 (C)	4 (D)	4
Total	4	4	8

Note that this is a situation where we're not allowed to use a chi-squared test, since the sample size is too small: the cell values expected by chance would all be 2, below the cut-off of 5. Yet, as Fisher showed, this is a statistically significant result. How did he conclude this?

He started by noting that even with exact tests like the one he was inventing, sample size is not totally irrelevant, since you can't test significance if your sample is so tiny that no matter happens it is too likely to be due to chance. For example, if the lady had been tested on only one cup, then she would have a 50/50 chance of guessing the right order of tea and milk, so if she guess right, $p = .5 > .05$, which ain't significant, according to the theory of p values that Fisher was also busy inventing at the time. As Fisher shows (see his paper), you need to have at least four pairs of cups to give the lady any chance of beating chance.

Here's a sketch of what he says next. Since the lady knows ahead of time that there are a total of eight cups, with four cups of each type, she only has to choose four cups correctly (the other four follow automatically, as the complement of her four main choices). This knowledge is encoded in the marginal values (the row and column totals), which remain the same no matter how many correct and wrong guesses she makes. There's only one way to get all four guesses

right, but if she makes a mistake, she could do so by missing any one of the four tea-first cups (and therefore also miss one of the four milk-first cups); the choices of the two wrong cups are independent, so by the multiplication rule of probability, this makes 16 = 4×4 ways to make one mistake. The logic continues in this way for making two mistakes and so on, and since all of these possibilities are complementary, we add them all up (by the addition rule of probability) to get 70 possible contingency tables, only one of which is all right (see Fisher, 1935, for the details).

The one-tailed chance probability here is thus $p = 1/70 = .01428571$. A two-tailed p value would also make sense, since guessing *all* of the cups *wrong* would also be impressive, like some kind of anti-power. But technically only the one-tailed p value is "exact", since we're computing actual probabilities based on our actual situation, not for the opposite of what happened. So Fisher reported the one-tailed value, and this is clearly statistically significant ($p < .05$).

The general formula for this analysis is as follows, where $n! = 1\times2\times3\times\ldots\times n$ (the factorial: 階乗), which you can compute with R's **factorial(n)** function or Excel's **=FACT(n)** function.

One-tailed p value for Fisher's exact test: $$p = \frac{(A+B)!(C+D)!(A+C)!(B+D)!}{A!B!C!D!(A+B+C+D)!}$$

```
A=4; B=0; C=0; D=4
p = (factorial(A+B)*factorial(C+D)*factorial(A+C)*factorial(B+D))/
    (factorial(A)*factorial(B)*factorial(C)*factorial(D)*factorial(A+B+C+D))
p # 0.01428571 again
```

R has a built-in function for this, called **fisher.test()**, which gives two-tailed p values by default:

```
tea = matrix(c(4,0,0,4), ncol = 2, nrow = 2)
fisher.test(tea) # two-tailed p = 0.02857
fisher.test(tea, alternative="greater") # one-tailed p = 0.01429
```

Computing the two-tailed p value involves computing the exact probabilities twice, once for our actual observations and then again for the opposite. By luck, these probabilities happen to be the same in the tea-lady story, but this isn't true of Fisher's test in general.

As you may know, the factorial function can generate extremely huge numbers; even 10! = 3,628,800! (The second exclamation point is for surprise.) This computational issue made Fisher's exact test relatively impractical before computers were invented, since you have to divide one huge number by another huge number accurately enough to get a proportion between 0 and 1.

But this book is supposed to be about linguistics, so here's another example. In a formal semantic experiment, Schmitz and Schröder (2002) asked 23 speakers to judge one type of

statement as true or false, and 24 other speakers to judge a different type of statement as true or false (so they used a between-group design, which isn't the best choice, but whatever). They wanted to see if one type of sentence was more likely to be accepted as true than the other type. This produced the contingency table for their observed frequencies shown in Table 14.

Table 14. True-false semantic contingency table.

	Type 1	Type 2
True	14	8
False	9	16

The first test you might think of trying is the two-way chi-squared test, which of course should use Yates's correction because it's a 2 × 2 table. But the results aren't significant:

semdata = matrix(c(14, 8, 9, 16), ncol = 2)
chisq.test(semdata) # X-squared = 2.5562, df = 1, p-value = 0.1099

But maybe Yates's correction makes this chi-squared test too conservative. That is, while subtracting that 0.5 value helps us avoid Type I errors, it also increases the risk of Type II errors (misses, i.e. missing a true pattern due to a null result). So Schmitz and Schröder decided to use a one-tailed Fisher's exact test instead, to compute the objectively most precise *p* value, not just an estimated *p* value depending on the chi-squared distribution and the particular value subtracted in Yates's correction. And luckily for them, the results did come out a bit closer to being statistically significant.

semdata = matrix(c(14, 8, 9, 16), ncol = 2, nrow = 2)
fisher.test(semdata, alternative="greater") # one-tailed p = 0.0545 (marginal)
fisher.test(semdata) # two-tailed p value = 0.08198: still better than chi-square

I'm not sure this is a good example for us to follow, though, since a chi-squared test would be fine here; it kind of looks like these authors were ***p*-hacking**, or playing around with statistics until they got a *p* value low enough to please them, which is, of course, an unethical research practice (Wasserstein & Lazar, 2016)!

While we're exploring exact replacements for the asymptotic chi-squared test, why don't we try rerunning the analysis that gave us a warning before, due to the expected cell sizes going below 5?

chisq.test(VowelCombos.mat) # X-squared = 4.3403, df = 4, p-value = 0.3619

Let's do Fisher's exact test so we don't have to worry about restrictions like this. Running the following code gives us a *p* value similar to what we got with the chi-squared test, but without the warning:

fisher.test(VowelCombos.mat)

```
        Fisher's Exact Test for Count Data

data:  VowelCombos.mat
p-value = 0.4088
alternative hypothesis: two.sided
```

4.2 The exact McNemar test

There's an exact version of McNemar's test too, where the contingency table represents data that come in pairs. It turns out to be extremely simple because it's actually just a special application of the **binomial test** that we've already used before. As you may (or may not) recall, you can use the binomial test whenever your data involve a set of binary events (i.e., with only two possible outcomes), where each event is independent of all the others and the outcomes occur with a fixed probability. Under the null hypothesis, such a data set would be generated like flipping a coin (with fixed probability of 1/2).

Let's first see how the binomial test can be related to ordinary (non-paired) contingency tables. Suppose you notice that in some language, there are eight words in some lexical class (e.g., all nouns for edible roots), seven of which have some property (e.g., feminine gender), while the remaining one word in this class does not. This data set can be summarized as in Table 15, where the first row shows the observations in something that looks like a one-way chi-squared table and the second row shows what we would expect by chance, if the word types were evenly distributed.

Table 15. Regular and exceptional words

	Patterned	Exceptional
Observed:	7	1
Expected:	4	4

This is a pretty tiny data set, and in fact, a one-way chi-squared would be problematic, since the expected cell sizes go below 5. So if we do the analysis using **chisq.test()**, the *p* value we get is not reliable, and R also gives us a warning:

chisq.test(c(7,1)) # p = .03389

Luckily, in this case we can calculate the exact *p* value for the null hypothesis, directly in terms of the chance probability of getting seven "heads" (H) out of eight "coin flips" of heads and "tails" (T), which is defined in terms of the area of the binomial distribution for $n = 8$. Since we've decided ahead of time what counts as regular and what counts as exceptional, we want to know the probability of getting *seven or more* heads, which is the number of getting exactly seven (=8: THHHHHHH, HTHHHHHH, ...) plus the number of getting exactly eight (=1: HHHHHHHH), divided by the total number of all possible sets of eight binary outcomes ($=2^8 = 256$, by the multiplication rule for independent probabilities), giving us (8+1)/256 = .03515625. That's the one-tailed *p* value for the null hypothesis. If we want, we can multiply this by two to get the two-tailed *p*, but remember that for exact tests, only the one-tailed *p* is *exact* (reflecting our actual proportion), so that's what we'll use below.

Of course, we don't have to do it "by hand" like this, since we can just use the binomial distribution functions of Excel or R to calculate it for us. The following formulas give us the same one-tailed *p* values for regular vs. exceptional outcomes:

Excel: **BINOMDIST(Number_Exceptional, Total, 0.5, TRUE)**
R: **pbinom(Number_Exceptional, Total, 0.5) # prob for each event = 1/2**

So, to analyze our null hypothesis in R, we could just do this, getting a significant one-tailed *p* value. Note how close the *p* value is to what we got with **chisq.test()**, but this one is more believable (the two-tailed *p* value would be double this, since the binomial distribution is symmetrical):

```
pbinom(1, 8, 0.5) # p = .03515625 again (one-tailed)
```

All very useful, but there's more. The binomial test becomes the **exact McNemar test** when the binary events are actually *pairs* of binary events, ignoring the cases where the events are the same within a pair, just as in the ordinary McNemar test, because those cases don't provide any information.

For example, returning to the paired-sentence judgment experiment from earlier, suppose the results come out as in Table 16:

Table 16. Pairs of sentence judgments

(Sentences grouped in pairs of grammatical + ungrammatical)		Judgment for grammatical sentence	
		unacceptable	acceptable
Judgment for ungrammatical sentence	unacceptable	5 (A)	15 (B)
	acceptable	3 (C)	7 (D)

Just as in the ordinary McNemar test, we only pay attention to cells B and C, where the two judgments in each pair are different. Let's call the pairs in B the "regular" ones (since the grammatical sentence is judged as acceptable and the ungrammatical sentence is judged as unacceptable), and those in C the "exceptional" ones (since they show the reversed judgments). Now we just plug these into the binomial distribution as before, and compute our one-tailed *p* value for getting at least as many "regular" judgment pairs by chance as we actually observed (or double this, for the two-tailed *p* value):

pbinom(3, 3+15, 0.5) # one-tailed p = .003768921

As usual, exact tests still require a bare minimum of data to test significance. So if we set our alpha level to .05 as usual, then it's impossible to reject a one-tailed null hypothesis if you have four or fewer key data points (i.e., B + C < 5). Even if all four go the predicted way, the probability of this happening by chance must be over .05: there just aren't enough observations to run any statistical test at all!

pbinom(0, 4, 0.5) # one-tailed p = .0625 > .05: not enough data to run the test!

4.3 Inventing your own exact tests

Once you get the idea of exact tests through permutations, it's possible to invent new statistical methods for all sorts of contingency tables.

For example, in Myers, Huang, and Tsay (2007), my coauthors (real statisticians) and I (not a real statistician) show how to compute the *p* values for what we call a "minimalist" factorial experiment, involving two binary factors and their interaction, with binary outputs. With larger sample sizes and continuous outputs, we could use a two-way independent-measures ANOVA (as we'll see in the next chapter), but if the outputs are binary (and thus not normally distributed) and the samples are too small, we need an exact categorical method. There wasn't any such test in the literature, so we made one up.

The logic is similar to that for Fisher's exact test. Namely, if we know the counts for each cell of the contingency table, this tells us something about the marginals and the grand total, so we just test all possible contingency tables with the same marginals to see how probable our observed table is in this set. To make the computations easier, these tables are not generated directly, but via permutations of the "observations" that would produce them, as in Fisher's exact test.

A linguistic application is given in Myers (2009, p. 433), using example sentences from Li (1998). Li claims that number phrases like 三個人 do not permit the kind of local binding of 自己 shown in sentence *a* in Table 17 (pay attention to the little *i*'s, marking coreference). In other words, she predicts an interaction between two factors: Number × Local. Suppose we

give each of the following four sentences to four different speakers (for a total of sixteen speakers), in a between-group design (like the chi-squared test or Fisher's test, our test assumes that all data points are independent). We ask each speaker to judge each sentence as "good" (seems acceptable) or "bad" (seems unacceptable). Let's say we can't test more for some reason, so we only have this small sample of 16 data points to deal with. So we count up the number of "good" responses for each sentence, as shown in Table 17:

Table 17. Sixteen independently collected sentence judgments.

	Num	Local	Sentence	Good	Bad
a	+	+	張三知道三個人$_i$一定搬得動自己$_i$的鋼琴。	0	4
b	+	-	張三$_i$知道三個人一定搬得動自己$_i$的鋼琴。	4	0
c	-	+	張三知道李四$_i$一定搬得動自己$_i$的鋼琴。	4	0
d	-	-	張三$_i$知道李四一定搬得動自己$_i$的鋼琴。	3	1

The number of "good" judgments above trend in the direction of Li's prediction, with more "good"s for sentences *b*, *c*, *d* than for sentence *a*, but is this trend statistically significant? It's not obvious, is it?

Myers (2009) points to a file defining an R function, based on Myers et al. (2007), that can answer this question. There's a link to this file **ExactFunctionsRCode.txt** on the class webpage. You can copy/paste it into R, or access it directly in R using the **source()** function, which loads in R code (without displaying it on the screen). Yes, R can download and run files from the web!

source("http://personal.ccu.edu.tw/~lngmyers/ExactFunctionsRCode.txt")

The function we want here is called **small.exp()**, which uses another function called **ext.fisher()** (for extended Fisher) that runs the Myers et al. (2007) algorithm. You can't search for help on either function in R, but if you type the function name (without parentheses), you can see what its inputs and outputs are. In the case of our minimalist experiment, we can get the *p* values like so. The commands below just recode the above data as vectors (in real life, the data would be collected in this format in the first place), using 1 = "good" vs. 0 = "bad" for the dependent variable **Judg**, and effect coding (-1 vs. 1) for the independent variables **Num** and **Local**.

```
Speaker = 1:16
Judg=c(0,0,0,0,  1,1,1,1,  1,1,1,1,  0,1,1,1) # Four conditions
Num=c(rep(1,8),rep(-1,8)) # First [+Num], then [-Num]
Local=rep(c(rep(1,4),rep(-1,4)),2) # [+Local] then [-Local] for each level of [Num]
data1 = data.frame(Speaker,Judg,Num,Local)
data1 # Take a look at it!
```

Once the variables are defined, the next line runs the statistical analysis for us, and gives some rather ugly (but useful) text output:

small.exp(data1$Judg, data1$Num, data1$Local)

Factor1	Factor2	Factor1:Factor2
1.0000	1.0000	0.0588

The three numbers are the two-tailed *p* values for the first factor (Num), the second factor (Local), and their interaction (R uses the symbol ":" to indicate the interaction between two factors). Thus there seems to be a marginal ($.1 > p > .05$) interaction between the two factors, though neither factor is significant by itself. So now we have something concrete and mathematically justified to say to our readers, instead of just showing them Table 17.

As Myers (2009) shows, the same function can also analyze experiments where instead of giving each sentence to a separate person, we give each person all four sentence types. As we saw when we discussed the unpaired and paired *t* tests, this kind of within-group has greater statistical power than between-group designs, so it's better to use this kind of design if you can.

The statistical logic for this kind of experimental design starts by noting that each participant can show 16 possible response patterns to the four sentences (so like flipping a coin four times, there are $2^4 = 16$ possible response patterns). Five of these patterns show more "good"s than "bad"s and five show more "bad"s than "good"s: an equal number for each type. This is true for each of the three comparisons we're testing (i.e., the two factors plus their interaction), and these comparisons are independent of each other. How do I know that they are independent? Because the two factors are completely uncorrelated: imagine treating the plus and minus signs in the Num and Local columns in Table 17 as +1 and -1 respectively: **cor(c(1,1,-1,-1), c(1,-1,1,-1))** is zero (try it!).

So we can just run the exact McNemar test here, comparing the number of more-"goods" quartets versus more-"bads" response quartets, separately on each of the three comparisons (first factor, second factor, and their interaction).

To see how this works, suppose we test the same four sentences in a new experiment, this time giving each sentence quartet to only six different participants (instead of the 16 needed for the between-group design), and suppose that all six say "bad" to sentence *a* but all say "good" to sentences *b*, *c*, *d*. This again looks like it supports Li's prediction, but is it statistically significant?

Here's how we'd do the analysis using the **small.exp()** function, again starting by just setting up the data frame:

```
Speaker= rep(1:6,4) # i.e. 1-6 for each of the four conditions
Judg=c(0,0,0,0,0,0,  1,1,1,1,1,1,  1,1,1,1,1,1,  1,1,1,1,1,1) # Four conditions
Num=c(rep(1,12),rep(-1,12))
Local=rep(c(rep(1,6),rep(-1,6)),2)
data2 = data.frame(Speaker,Judg,Num,Local)
data2 # Take a look at it!
```

Now we're ready to run the analysis. The new argument here is **group**, which tells the **small.exp()** function to group the data by Speaker. The output text is formatted the same way as before (though it looks a bit different because I'm a bad programmer):

```
small.exp(data2$Judg, data2$Num, data2$Local, group=Speaker)
```

```
        Factor1   Factor2   Factor1:Factor2
p=       0.0313    0.0313            0.0313
```

This is a different experiment from before, so the results are different. This time we learn that not only is there a significant interaction between Num and Local, as Li predicts, but also each of the two factors is itself significant. Since the speakers in the second experiment all accept sentences *b*, *c*, *d* (unlike the first experiment), it seems that they also favor [-Num] sentences (as in two of those three sentences, namely *c* and *d*) and favor [-Local] sentences (as in two of those three sentences, namely *b* and *d*). This is fake data, so your own judgments might disagree, but I think it's kind of cool that we can test three separate hypotheses (about Num, about Local, and about Num × Local), using just one function.

Get used to this idea, since it's going to take a starring role in ANOVA, which we are finally going to explain in the next chapter!

5. Conclusions

This chapter was mainly about chi-squared tests, which are used to do statistical tests on the cell counts (frequencies) in a contingency table, which is a table that divides up all of the observations in some sample completely and without overlap. The goal of these statistical tests is to compare your observed cell counts with a table with the grand total and same row and column totals (called marginals), but where the counts are distributed more evenly (the usual null hypothesis), or distributed in some prespecified way (as in a goodness of fit test). A one-way chi-squared test looks at one-dimensional tables, with some single factor divided up into a set of categories (levels), and a two-way chi-squared test looks at two-dimensional tables, with two factors. Since the marginals are fixed, a two-way chi-squared test cannot test if each factor is significant by itself (i.e., whether the row totals or column totals are distributed more evenly than by chance), but instead can only test if there is an interaction between the two factors (and thus it is also sometimes called a test of independence of the two factors). Chi-

squared tests use the chi-squared distribution, which you compute with the help of degrees of freedom (related to the levels of your factor or factors, that is, the number of cells or rows and columns), a test statistic (χ^2), and the area under the tail, so that you can finally compute the *p* value. As an asymptotic (non-exact) test, you have to make sure your sample size is reasonably big, and if you only have a two-by-two two-way test, you should use Yates's correction. None of these restrictions apply to exact tests like the binomial test or Fisher's exact test, which compute the *p* value directly using probability theory, but these tests still assume that your data points are independently sampled. If your data come in pairs, you can use the McNemar test or its exact version.

References

Agresti, A. (2007). *An introduction to categorical data analysis*, second edition. Wiley-Interscience.

Cysouw, M. (2005). Quantitative methods in typology. In R. Kohler, G. Altmann, & R. G. Piotrowski (Eds.) *Quantitative Linguistik: Ein internationales Handbuch* [Quantitative linguistics: An international handbook], pp. 554-578. Berlin: Walter de Gruyter.

Drellishak, S. (2007). Statistical techniques for detecting and validating phonesthemes. University of Washington ms.

Fisher, R. A. (1935). The mathematics of a lady tasting tea. Section of *The design of experiments*, Oliver & Boyd, reprinted and retitled 1956 in Newman, J. R. (Ed.) *The world of mathematics*. Simon and Shuster. Reprinted 1988 by Tempus Books (pp. 1487-1495).

Friendly, M., & Meyer, D. (2015). *Discrete data analysis with R: Visualization and modeling techniques for categorical and count data*. CRC Press.

Gries, S. Th. (2005). Syntactic priming: A corpus-based approach. *Journal of Psycholinguistic Research, 34*(4), 365-399.

Guy, G. R. (1991). Explanation in variable phonology: An exponential model of morphological constraints. *Language Variation and Change, 3*, 1-22.

Li, Y.-H.A. (1998). Argument determiner phrases and number phrases. *Linguistic Inquiry, 29* (4), 693-702.

Lilienfeld, S. O., Lynn, S. J., Ruscio, J., & Beyerstein, B. L. (2010). *50 great myths of popular psychology: Shattering widespread misconceptions about human behavior*. Wiley-Blackwell.

Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, MA: MIT press.

Myers, J. (2009). The design and analysis of small-scale syntactic judgment experiments. *Lingua, 119*, 425-444.

Myers, J., Huang, S.-F., & Tsay, J. (2007). Exact conditional inference for two-way randomized Bernoulli experiments. *Journal of Statistical Software, 21*, Code Snippet 1, 2007-09-02.

Salsburg, D. (2001). *The lady tasting tea: How statistics revolutionized science in the twentieth century*. New York: W. H. Freeman & Company.

Schmitz, H.-C., & Schröder, B. (2002). On focus and VP-deletion. *Snippets, 5*, 16-17.

Wasserstein, R. L., & Lazar, N. A. (2016). The ASA's statement on p-values: context, process, and purpose. *The American Statistician, 70*(2), 129-133.

Zuraw, K. (2010). A model of lexical variation and the grammar with application to Tagalog nasal substitution. *Natural Language & Linguistic Theory, 28*(2), 417-472.