Chapter 6
Comparing two continuous variables: *t* tests and beyond

James Myers
2022/3/22 draft

1. Introduction

Now that we've mastered the basics of parametric statistical tests, used for testing hypotheses about continuous distributions that are normal-ish, how can we make them more useful? After all, one-sample tests are useful to know about if you want to get a sense of how statistics works, but by themselves they're not very useful in real-life situations. Most of the time, we want to compare something with something else, where both of the somethings are somethings that we've actually observed, not an imaginary population. For example, in an experiment that gives you normal-ish continuous values (like phonetic measures or reaction times), good experimental design means that you have to have different conditions to compare, at least a **control** condition (the baseline condition).

So what you need to learn next is how to do a parametric test when you have two samples. That's what we'll do in this chapter, plus a bit more. The type of test we'll need for this comes from the ***t* test** family. Since statistical significance doesn't necessarily imply real-world "significance", we'll also see how to quantify the **effect size** of a *t* test, particularly via **confidence intervals**. We'll also briefly discuss how to do **nonparametric** two-sample tests that you might want to use when your data aren't normal at all.

Along the way, we'll learn that the specific type of *t* test you need to do depends on certain basic properties of your data: if your two samples are independent of each other, then you want to do an **unpaired *t* test**, but if your two samples come in correlated pairs, you do a **paired *t* test**. Moreover, if your two samples are independent of each other but also have different **variances** (i.e., the square of their standard deviations), then this violates an assumption of the most common type of unpaired *t* test, so you have use a more general (but less powerful) version that doesn't depend on this assumption. Everything is connected: deciding which version is better depends, in turn, on a **variance test** that will end up lying at the heart of running an ANOVA (where "VA" stands for variance), which turns out basically to be a generalization of the *t* test anyway.

2. *t* tests

It may seem that turning a one-sample *t* test into a two-sample *t* test is just as magical as turning one banana into two. But due to the hierarchical nature of mathematical logic, all *t* tests

are actually quite closely related. Later in this section we'll get into the mathematical tricks that make this work, but let's first start with some simple examples that we can run right away in Excel and R.

2.1 The unpaired *t* test

When is it appropriate to run an **unpaired *t* test** (非成對樣本 t 檢定)? As noted in the introduction, the key requirements are that your data are continuous, pretty much normally distributed, and divided into two independent samples. In case you forgot what "independent" means in statistics, here's a reminder: it has to do with probability. Two samples are independent if no value in one sample is predictable from any value in the other sample (just as you can't guess the color of a card simply by knowing that it's Q). In other words, the two samples aren't correlated, and in fact they may not even have the same sample size. This situation arises a lot in linguistics: you want to compare reaction times to a set of nouns with a set of unrelated verbs, or you want to compare accuracy rates produced by a bunch of female students with a bunch of male students.

As long as your samples are independent and have reasonably normal distributions, an unpaired *t* test should be helpful to test the null hypothesis that the two sample means are the same. Similar with the one-sample *t* test, you could instead decide to make your null hypothesis say that the difference between the two sample means (more precisely, between the means of the two populations that the null hypothesis says your means come from) is some specific number other than zero (e.g., $\mu_1 - \mu_2 = 235$ or whatever), or you could choose the directional hypothesis that one mean is equal to or bigger than the other (e.g., H_0: $\mu_1 \geq \mu_2$) and compute a one-tailed *p* value. But in real life, the null hypothesis is virtually always the simplest nondirectional hypothesis, namely that the means are identical (e.g., H_0: $\mu_1 = \mu_2$), giving you a two-tailed *p* value.

Of course, as with any **asymptotic test**, you'll get better results (i.e., have greater statistical **power**) with larger sample sizes, but as Gosset proved in his beer factory, the *t* test can handle smallish samples too. There are other technical issues about sample sizes and distribution parameters that affect the unpaired *t* test, but I'll save those for the mathematical sections below.

For now, though, let's skip all of the math and jump right into examples. If you are not too busy, please go get the file **t-test-samples.xls**, which has fake data cleverly designed to show the strengths and weaknesses of various kinds of *t* tests. Since we're doing an unpaired *t* test, we'll start with the first data set on the left (boys vs. girls), which involve two independent samples. Even though the sample sizes are the same, I hereby declare that these two sets of fake kids have no natural connection (e.g., they're not pairs of brothers and sisters). We want to know if the boys and girls show a significant difference in their mean scores.

2.1.1 The simplest kind of unpaired *t* test in Excel

To run an unpaired *t* test on these data in Excel, we need to use the Analysis ToolPak, so open it up, and look for the *t* tests (t 檢定). What you'll see is that there are three. For now, just trust me and choose this one: **t-Test: Two-Sample Assuming Equal Variances (t 檢定：兩個母體平均數差的檢定，假設變異數相等)**. Pretty soon I'll explain why variance matters, and show how to use the other two *t* test tools that Excel offers (though you can probably already guess that one of the other versions is for the paired *t* test).

Now we get a dialog box something like that shown in Figure 1 (this ancient image is from Excel 2003, but the content hasn't changed):

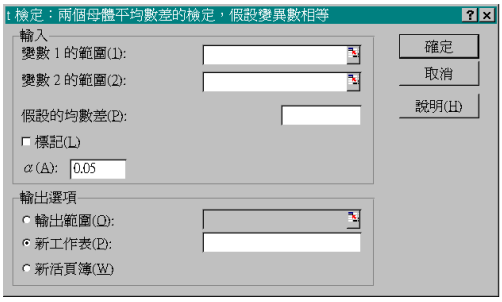

Figure 1. The basic unpaired *t* test dialog window in Excel

Then we select the two samples to go into the first two boxes, including the column labels ("Boys" and "Girls"). As usual with Analysis ToolPak tools, you need to select the precise ranges; unlike cell functions, you can't select the whole column, or else the TookPak tool will give you an error message. Since our null hypothesis is the usual nondirectional one, the "boring" population for the difference between the sample means is assumed to be zero. In formal terms, our null hypothesis is H_0:$\mu_1 = \mu_2$ (that the two population means associated with the two samples are identical), which is the same as H_0:$\mu_1 - \mu_2 = 0$. So we enter "0" into the next box, and I hope you remembered to select the labels that you typed in at the top of each sample, in which case you should click the 標記 (L) box, or else your results will be given in terms of「變數 1」(Variable 1) and「變數 2」(Variable 2), which would be pretty useless, now wouldn't it? You can just leave all the other defaults alone, including the alpha level (A) of .05 (I'll explain later why this can be useful).

Below is what you get if you did it right. You should be able to figure out most of this report, even if you don't know yet how all of it was calculated. There are the two sample means at the top, then their two variances (s^2: to confirm, use **=VAR()** or **=STDEV()^2**, or the slightly corrected sample functions **=VAR.S()** or **=STDEV.S()^2**), then their sample sizes, then "pooled" something (which we'll come back to later), then the sample mean difference assumed by the null hypothesis (zero), then the degrees of freedom (which we'll come back to

later too, though if you guess that this value is computed by adding up the two sample sizes and subtracting two, $df = n_1 + n_2 - 2$, then you are right), then the *t* value (computed in a way very similar to the one-sample *t* test, as we'll see soon). Next you get the one-tailed *p* value and the one-tailed **critical value** (which, as you should remember, is the value of *t* where the one-tailed $p = \alpha$, which we kept as the default .05), based on the directional null hypothesis that the first sample mean (Boys) is smaller or equal to the second sample mean (Girls) (Excel bases this direction on the order of the two columns). Then finally you get the two-tailed *p* value and critical value for the nondirectional test that we actually want to do here.

t 檢定：兩個母體平均數差的檢定，假設變異數相等

	Boys	Girls
平均數	4.511016	2.94041
變異數	1.710218	0.882993
觀察值個數	10	10
Pooled 變異數	1.296605	
假設的均數差	0	
自由度	18	
t 統計	3.08424	
P(T<=t) 單尾	0.003198	
臨界值：單尾	1.734063	
P(T<=t) 雙尾	0.006396	
臨界值：雙尾	2.100924	

How did we get that two-tailed *p* value from that *df* and *t* value? We got it exactly the same way we did for the one-sample *t* test: the two-tailed *p* value represents the total area in the two tails beyond the point marked by *t*, on both sides of zero, in the *t* distribution defined by that *df*. So to compute this using Excel cell functions, you can just use **=T.DIST(-ABS(*t*), *df*, TRUE)*2**, and then click on the relevant cells in the report table to get the *t* and *df* values. Try it!

So putting this all together (and ignoring the stuff we don't need), we could report this statistical analysis like so: "Boys (*M* 4.51, *SD* 1.31) were significantly different from girls (*M* 2.94, *SD* 0.94) by a two-tailed unpaired *t* test ($t(18) = 3.08$, $p < .05$)" or "$p < .01$" or "$p = .006$" (depending on the style preferred by the journal you're submitting this to). If you want to be even more precise, you could call this an unpaired *t* test assuming equal variance, but that's the most common kind of unpaired *t* test, and anyway, people will know which kind you used just from the *df* (as I'll explain shortly). That wasn't so hard, was it?

2.1.2 The simplest kind of unpaired *t* test in R

Of course we can do exactly the same thing in R. One way to get the same data into R would be to copy and paste the Boys and Girls columns from Excel into a text file, naming it, and then using **read.table()** or **read.delim()** to read it into R. But this method won't work in general, because R prefers data frames, which are unlike Excel sheets in that no cell can be empty. Instead, data frames are more of like matrixes, in that they're a full rectangle of values. The logic behind this preference is simple: in a typical data frame, each row represents everything we know about exactly one data point. In this case, each data point has two properties: the gender of the kid, and the kid's measurement score.

So it's better to rearrange the data the way R likes it, and in my experience, this is easier to do in Excel. In this case, I already did it for you: the file **BoysGirls.txt** contains the two fake data sets about boys and girls, which I decided to call Study 1 and Study 2, respectively, and I also gave the fake participants unique identification numbers.

Let's look at the first data set (as usual, make sure you see what my R code is doing):

```
bg = read.table("BoysGirls.txt",T)
study1 = subset(bg, bg$Study==1)
boys1 = study1$Measure[study1$Gender=="Boy"]
girls1 = study1$Measure[study1$Gender=="Girl"]
```

R uses the same function to run any kind of *t* test, namely **t.test()**. Yes, that's exactly the same one we've already used for one sample *t* tests. To run different kinds of *t* tests, you just use different values for the arguments (see what they are by checking help with **?t.test**, and looking in the **Arguments** section).

R deals with the problem of the two kinds of unpaired *t* tests by making the more general version the default (i.e., the one that doesn't care about variance). However, this general version is not the most commonly used version, nor is it the conceptually simplest version, so I'll introduce it a little bit later. So in order to imitate what we just did in Excel, we need to tell R to please use the more common, simpler version, by setting the **var.equal** argument to **TRUE**. Unlike Excel, R knows that the usual null hypothesis is H_0: $\mu_1 = \mu_2$, so it's set by default.

With that as background, here we go:

```
t.test(boys1, girls1, var.equal = T)
```

This produces the following text report. Look carefully, and you'll see the same crucial values that we got from Excel. We also get that confidence interval thing, which (you'll be happy to learn) I will finally explain in this chapter. Moreover, whereas in Excel we can click on the values in the output table to do other calculations (as you should have tried doing in the previous section with the *t* and *df* values to double-check the *p* value), in R we can pull out

specific values (if we want) using **$** (again, check help with **?t.test**, then look at **Values**). So we can write the same *t* test report in the standard format.

```
        Two Sample t-test

data:  boys1 and girls1
t = 3.0842, df = 18, p-value = 0.006396
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.5007406 2.6404700
sample estimates:
mean of x mean of y
 4.511016  2.940410
```

2.2 The paired *t* test

The unpaired *t* test is extremely useful in linguistic research, but it is only relevant if the two samples are **independent**. This was appropriate with our example above, since we used a **between-group** design to test separate groups of boys and girls. But in other studies, each value in one sample is actually paired with a value in the other sample, in that they represent pairs of values associated with the same unit, like the same word or person. This happens a lot in linguistic research, for example in a **within-group** (within-participants) design, where you get two measures from each participant (e.g., each person responds both to nouns and to verbs). In this kind of situation (and assuming your dependent variable is still continuous and basically normal), you are justified in doing a **paired *t* test**.

Within-group designs are *logically* more powerful than between-group designs, since they remove the **confound** between conditions and participants. For example, if you test one random group of participants on nouns and a second group on verbs, you wouldn't be able to be sure if any difference you find was truly caused by the noun-verb contrast, and not by some other unknown difference in the two groups of participants. With large enough random samples, the risk of this kind of confound should go down, but the risk never disappears entirely.

Within-group designs are also *statistically* more powerful than between-group designs. For example, say that for every participant in an experiment, noun reaction times (RTs) are exactly 10 ms faster than verb RTs, but the variance across participants is much higher than that, say 100 ms. If you did an unpaired *t* test, this 10 ms difference may be hidden in all the random noise, but with a *t* test conducted on the pairs, the consistency of the difference in each pair of data may make the interesting difference stand out.

2.2.1 The paired *t* test in Excel

You can see the effect of pairing by looking at the last two examples in **t-test-samples.xls**. I've faked the data so that in the first set, there's a relatively high correlation between the paired data, while in the second set, there's a relatively low correlation. You can confirm the correlations yourself using **=CORREL()**: you should get r = .89 and r = -.30, respectively. Note that the two sample means and standard deviations are identical across the two data sets, because in fact their values are exactly the same; the correlation difference was created by me cleverly randomizing the order of one sample in the latter data set.

Now see what happens if we do paired *t* tests on these two data sets using Excel's statistics tool 「**t 檢定：成對母體平均數差異檢定**」**(t-Test: Paired Two Sample for Means)**. Try it yourself (the procedure is very similar to what we did above for the unpaired *t* test). If you do it right, you could report the first set of results like so: "The mean score for nouns (*M* 4.51, *SD* 1.31) was significantly different from the mean score for verbs (*M* 2.94, *SD* 0.94) by a two-tailed paired *t* test ($t(9) = 7.71$, $p < .001$)". By contrast, the report for the second data set analysis might say: "The mean score for nouns (*M* 4.51, *SD* 1.31) was significantly different from the mean score for verbs (*M* 2.94, *SD* 0.94) by a two-tailed paired *t* test ($t(9) = 2.72$, $p < .05$)." We'll explain all these numbers soon, but as before, you might be able to guess that here, $df = n - 1$ (where n is the number of pairs, that is, the size of either sample, since both samples have to be the same size in order to form pairs).

Notice that the first fake paired data are exactly the same as the "boys vs. girls" data that we analyzed with an unpaired *t* test, and yet the results are quite different in terms of the *df*, *t* values, and *p* values ("Boys (*M* 4.51, *SD* 1.31) were significantly different from girls (*M* 2.94, *SD* 0.94) by a two-tailed unpaired *t* test ($t(18) = 3.08$, $p < .01$"). This is because the data in the first noun-verb study are correlated, the paired *t* test is able to give a more powerful result, giving a higher *t* value and lower *p* value ($t(9) = 7.71$, $p < .001$) than an unpaired *t* test would ($t(18) = 3.08$, $p < .01$).

In real life, you wouldn't choose between the two types of *t* tests based on the correlation; you would choose between them based on your real-world situation. If your data come from samples that are independent in the real world (like unrelated boys and girls), choose the unpaired test; if your data represent pairs of values (like the noun and verb responses where each pair comes from the same person), then choose the paired test. These fake examples are just meant to show the statistical logic behind this real-life logic.

2.2.2 The paired *t* test in R

To do a paired *t* test in R, we just use the same **t.test()** function as before, but now set the parameter **paired** to **TRUE**. We don't have to bother with **var.equal**, since that only applies

to unpaired *t* tests. Let's try it on the R-friendly version of the last two fake data sets in **NounsVerbs.txt**:

```
nv = read.table("NounsVerbs.txt",T)
study3 = subset(nv, nv$Study==3)
nouns3 = study3$Measure[study3$WordType=="Noun"]
verbs3 = study3$Measure[study3$WordType=="Verb"]
study4 = subset(nv, nv$Study==4)
nouns4 = study4$Measure[study4$WordType=="Noun"]
verbs4 = study4$Measure[study4$WordType=="Verb"]
t.test(nouns3,verbs3,paired=T) # nouns3[i] & verbs3[i] both from Participant i
t.test(nouns4,verbs4,paired=T) # compare results with Excel
```

Here are R's text reports for the more-correlated and less-correlated data sets, respectively:

```
        Paired t-test

data:  nouns3 and verbs3
t = 7.7077, df = 9, p-value = 2.976e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.109642 2.031569
sample estimates:
mean of the differences
                1.570605

        Paired t-test

data:  nouns4 and verbs4
t = 2.7228, df = 9, p-value = 0.0235
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.265729 2.875482
sample estimates:
mean of the differences
                1.570605
```

One difference you might notice from Excel's report tables is that R only gives us the mean of the differences, not the two sample means. This is because the null hypothesis tested by the paired *t* test is subtly different from that tested by the unpaired *t* test. Again, linguists should be well qualified to understand the difference, since it has to do with scope. Namely, while the unpaired *t* test tests the null hypothesis that the difference between the sample means is zero (or some other specific value), the paired *t* test tests the null hypothesis that the mean of the paired differences is zero (or some other specific value). That is, the unpaired *t* test tests H_0: difference(means) = 0, whereas the paired *t* test tests H_0: mean(differences) = 0.

To end this section, let's fake some new data where each pair shows a difference of 10 but each set has variance of 100 (i.e., *SD* = 10), then compare unpaired and paired *t* tests, just to show that the latter is statistically more powerful. Try it!

```
fakeA = rnorm(5,sd=10) # Your fake data will be different from mine
fakeB = fakeA + 10 # So mean(fakeB) = mean(fakeA) + 10, and sd = 10 (try it!)
cor(fakeA,fakeB) # Perfectly correlated
t.test(fakeA,fakeB,var.equal=T) # Unpaired t test: may not be significant
# Note: above WILL be significant around every 10-20 tries: can you guess why?
t.test(fakeA,fakeB,paired=T) # Paired t test: sd(D) = 0, so can't run the test
fakeB = jitter(fakeB) # jitter adds a small amount of noise
cor(fakeA,fakeB) # Highly correlated, but not perfectly
t.test(fakeA,fakeB,paired=T) # Paired t test: ridiculously significant!
```

So our fake examples show that the paired *t* test is more powerful if the data come in correlated pairs, and the unpaired *t* test is more powerful if the samples are independent. This is just as we would hope for two tests designed for precisely these two different situations. In actual practice, you can't switch between tests on the same data set, as I've been doing with these fake examples: that would be cheating, just as much as switching from two-tailed to one-tail *p* values would be. Instead, as I noted earlier, you have to decide which type of test to use ahead of time, based solely on the real-world situation.

2.3 More about the math of *t* tests

Are you satisfied with what you've learned so far in this chapter? I hope not! You shouldn't just assume that your computer is some kind of magic genie that automatically tells you the truth. You need to understand what it's doing in order to know whether the results it gives you are believable. Less philosophically, you also need to know enough about what the "right" statistics look like to be able to detect when you've accidentally done them wrong (e.g., by clicking the wrong button in Excel), or even to detect when a published paper made a statistical error (accidentally or for more evil reasons). So once again, we have to take a little swim in the lovely but mysterious sea of mathematics.

2.3.1 How the unpaired *t* test works

What an unpaired *t* test is actually asking about is what the observed difference in our sample means implies about the believability of the null hypothesis that this difference is zero. Since we have two samples, with means of M_1 and M_2, what we want to know is whether the difference between these two means is too large to have come from two populations with means μ_1 and μ_2 that are actually identical.

The key insight is that we can address this issue by taking one tiny step up the ladder of mathematical abstractness. Just as in the one-sample *t* test we want to know if our observed sample mean is an outlier in the null distribution of sample means, we can now think of our observed difference in the two sample means (i.e., M_1 - M_2) is just one point in an imaginary distribution of *all* the differences of *any* two sample means: the so-called **sampling distribution of the differences between means** (ugh, what a name). This is exactly the same trick we used before when we introduced the Central Limit Theorem, and because of this theorem, this new distribution also has the shape of the *t* distribution, so a variation on the one-sample *t* test can be used to check hypotheses about its mean. Wow!!!!

In practical terms, this insight tells us that we can take the one-sample *t* test formula that we used before, and replace the old one-sample mean *M* with our new *difference* between sample means M_1 - M_2, and likewise replace the single null population mean μ with the hypothesized *difference* between population means μ_1 - μ_2. Since your null hypothesis is almost always H_0:μ_1=μ_2 (i.e., that the two population means are the same), you can make it even simpler and just use 0 for μ_1 - μ_2.

So conceptually the formula for the unpaired *t* test (testing H_0:μ_1=μ_2) is as follows, where *SE* stands for **standard error**, which (as you remember) is the standard deviation of the sampling distribution. Thus this *t* formula is just a variation on our previous formula, which turn was based on the one-sample *z* formula, which was based on the ordinary *z* score (mean difference divided by standard deviation), which was just a way to standardize our actual values to fit into a universal distribution by shifting the center to zero and dividing away the spread. It's all connected!

Unpaired *t* test formula testing H_0:μ_1=μ_2: $$t = \frac{M_1 - M_2}{SE}$$

The only tricky part is how to compute *SE* here. Remember that for the one sample *t* test, the sampling distribution involved sample means, so the true value of *SE* is σ_M (the standard deviation of the sampling distribution of sample means), which can be estimated using $s/\sqrt{n}$. But now we're dealing the sampling distribution of the *differences between sample means*, so we have two sample sizes (n_1 and n_2) and two sample standard deviations (s_1 and s_2).

Fortunately, through the Magic of Mathematics (mainly probability theory and algebra), it turns out that the standard error we want is directly related to the estimated standard error for each sample: $s_1/\sqrt{n_1}$ and $s_2/\sqrt{n_2}$. The math is especially simple if the two samples are the same size: $n_1 = n_2$. It's even simpler if we also assume that the sample standard deviations s_1 and s_2 are both estimates of exactly the same population standard deviation, i.e., that $\sigma_1 = \sigma_2$. This is what Excel and R are talking about in the version of the unpaired *t* test that we introduced

above (with 假設變異數相等 and **var.equal=T**; remember that variance is just the square of standard deviation, so if the standard deviations are the same, so are the variances).

Starting with the simplest case, where the sample sizes are the same and we assume that the population variances are the same too, we can test the null hypothesis that H_0: $\mu_1 = \mu_2$ using the following formula.

Simplest version of unpaired *t* value: $t = \frac{M_1 - M_2}{\sqrt{\frac{{s_1}^2}{n_1} + \frac{{s_2}^2}{n_2}}}$ {H_0:$\mu_1 = \mu_2$; assume $n_1 = n_2$ & $\sigma_1 = \sigma_2$}

As a review, compare this to the explicit version of the one-sample *t* test formula (with *SE* spelled out as $s/\sqrt{n} = \sqrt{(s^2/n)}$), for H_0: $\mu_1 = 0$. Pretty similar, right?

One-sample *t* formula: $t = \frac{M}{SE} = \frac{M}{\frac{s}{\sqrt{n}}} = \frac{M}{\sqrt{\frac{s^2}{n}}}$

Since we're talking about the family of *t* distributions, to get the *p* value from this *t* value we also need to know the degrees of freedom defining the *t* distribution. So what is the appropriate *df*? It's not entirely arbitrary; remember that *df* stands for "degrees of freedom", so it's supposed to represent the number of values that aren't fixed by our sample, including the values that we have to guess or assume. How many values are we estimating here? Well, if in the one-sample *t* test we assumed one value ($\sigma = s$), by the same logic, here we should be assuming two: $\sigma_1 = s_1$, and $\sigma_2 = s_2$. That turns out to be the right logic: $df = (n_1 - 1) + (n_2 - 1) = n_1 + n_2 - 2$:

Degrees of freedom for unpaired t test: $df = n_1 + n_2 - 2$

Let's try out our newly learned knowledge on another data set. It's phonetics again, if that's OK with you. Let's start with Excel, where the file we want is **voweldurationsExcel.txt**. This contains (fake) durations in milliseconds for a random sample of productions of /i/ and /u/. The two samples are independent and the same size ($n_1 = n_2 = 24$), and for now we'll assume they have equal variance ($s_{/i/} = 48$ ms, $s_{/u/} = 41$ ms, which seem sort of close), so the above formula seems appropriate. (Note that in my reports here, I'm rounding the values to the closest millisecond, since the original durations also only had one-millisecond precision. For the real calculations in Excel and R I'll use the "raw" values, without any rounding.)

Are the means significantly different? Well, $M_{/i/} = 309$ ms and $M_{/u/} = 292$, which also seem pretty close (only 17 ms apart), especially given how wide the distributions are (as measured by their *SD*s). Let's see what an unpaired *t* test says.

Just to get a sense for how it works, let's do the calculations step by step. If the /i/ values are in A2:A25 and /u/ values are in B2:B25, we can make the following computations:

D2 (difference in means): **=AVERAGE(A2:A25) - AVERAGE(B2:B25)**
E2 (*SD* for /i/): **=STDEV(A2:A25) # or =STDEV.S(A2:A25)**
F2 (*SD* for /u/): **=STDEV(B2:B25) # ditto**
E3 (*n* for /i/): **=COUNT(A2:A25)**
F3 (*n* for /u/): **=COUNT(B2:B25)**
D4 (*SE*): **=SQRT(E2^2/E3 + F2^2/F3)**
E4 (*t* value): **=D2/D4**
D5 (df value): **=E3 + F3 - 2**
D6 (two-tailed *p* value): **=T.DIST(-ABS(E4), D5, TRUE)**

Unsurprisingly, it's not significant: $t(46) = 1.33$, $p = .19$.

You can do the same step-by-step computations in R, using **voweldurationsR.txt**, which has the same (fake) data formatted in a more R-like way (look inside to see):

```
vowels = read.delim("voweldurationsR.txt")
i = vowels$Duration[vowels$Vowel == "i"] # This saves space below
u = vowels$Duration[vowels$Vowel == "u"] # Do NOT attach(vowels) - see why?
meandif = mean(i) - mean(u)
sdi = sd(i)
sdu = sd(u)
ni = length(i) # Remember "length" counts because R treats variables like vectors
nu = length(u)
SE = sqrt(sdi^2/ni + sdu^2/nu)
tval = meandif/SE
dfval = ni + nu - 2
pval = 2*pt(-abs(tval), dfval)
pval # 0.1907439, same as before.
```

Feel free to compare these results with what you get with Excel's Analysis ToolPak or R's **t.test(var.equal = T)** functions. I'll wait.

What if your sample sizes aren't the same? The math gets a bit more complex, since for reasons beyond the scope of this book (ahem), using different sample sizes means that *SE* cannot be computed from separate s_1 and s_2 values, as in the above formula. Instead the individual variances s_1^2 and s_2^2 need to be merged into the **pooled** (整合) variance s_p^2, which represents our best estimate of $\sigma_1^2 = \sigma_2^2$. The simplest way to show how this works is to revive the notion of **sum of squares** (SS) that we last mentioned a few chapters ago, when introducing *SD* the first time:

Sum of squares for unpaired *t* test: $SS = \Sigma(x\text{-}M)^2$

Then we can express the variance of each sample using its *SS* like so (note that this formula is just the square of the sample standard deviation, where $df = n - 1$):

Sample variance: $s^2 = \frac{\sum(x-M)^2}{n-1} = \frac{SS}{df}$

The pooled variance in the unpaired *t* test below looks very similar to the above formula (and you should also be able to see that the overall *df* is still $n_1 + n_2 - 2$):

Pooled variance in unpaired *t* test: ${s_p}^2 = \frac{SS_1+SS_2}{df_1+df_2}$

Hm, pooled variance... where have we seen that term before...? Ah yes, in Excel's report for the unpaired *t* test! In the original Boys vs. Girls data set, Excel said the pooled variance ("Pooled 變異數") was around 1.2966. Let's see if we can recalculate this ourselves (I'll do it in R, but feel free to try it in Excel too):

```
# Reload everything, in case you lost it
bg = read.table("BoysGirls.txt",T)
study1 = subset(bg, bg$Study==1)
boys1 = study1$Measure[study1$Gender=="Boy"]
girls1 = study1$Measure[study1$Gender=="Girl"]
# Calculate pooled variance, step by step
SS.boy = sum((boys1-mean(boys1))^2)
SS.girl = sum((girls1-mean(girls1))^2)
df.boy = length(boys1) -1
df.girl = length(girls1) -1
var.pooled = (SS.boy + SS.girl)/(df.boy + df.girl) # Compare with the Excel report
```

So now all we do is plug this pooled variance into our unpaired *t* test formula, using ${s_p}^2$ to replace both ${s_1}^2$ and ${s_2}^2$:

More general version of unpaired *t* value: $t = \frac{M_1-M_2}{\sqrt{\frac{{s_p}^2}{n_1}+\frac{{s_p}^2}{n_2}}}$ {H_0: $\mu_1 = \mu_2$; assume $\sigma_1 = \sigma_2$}

Try it out!

```
M.boy = mean(boys1)
M.girl = mean(girls1)
n.boy = length(boys1)
n.girl = length(girls1)
```

```
SE = sqrt(var.pooled/n.boy + var.pooled/n.girl) # The denominator in the equation
tval = (M.boy - M.girl)/SE # Compare with Excel or R report
dfval = n.boy + n.girl - 2 # Compare with Excel or R report
pval = 2*pt(-abs(tval), dfval) # Compare with Excel or R report
```

2.3.2 Testing for variance differences

Of course, in this complex world of ours, there is no guarantee that two independent samples happen to come from populations with identical variance (they're independent, after all). If they don't, we'll need to generalize the *t* formula again. Before telling you how to do this, however, we first have a more fundamental question: How can we tell if our samples come from populations with different variances?

Note that the above formulas assume $\sigma_1^2 = \sigma_2^2$, but this is a claim about those abstract, unreachable, idealized populations, not the concrete samples that we can directly observe. Our observed s_1^2 and s_2^2 values are merely estimates of σ_1^2 and σ_2^2, so it is quite possible for the sample variance values to be different, but not different enough for us to reject the null hypothesis that H_0: $\sigma_1^2 = \sigma_2^2$.

What we need is a genius to come along to tell us how to test a new type of null hypothesis, one that relates to variances, not to means. Luckily for us, such a genius already came along a long time ago: that Fisher guy again. He invented something now called the ***F* test** (or **variance test**, in this particular use of it), in his honor (capital-F for "Fisher").

Just as *t* tests use *t* distributions, the *F* test uses another kind of distribution called the ***F* distribution**. In later chapters we'll see that a version of the *F* test is crucial for doing ANOVAs (remember "VA" means "variance"). That's why reports of an ANOVA always say something like "$F(1, 29) = ...$". So please rest assured that this section is not a waste of time; you'll definitely be seeing this *F* thing a lot later on.

The *F* distribution describes the ratios of two variances. This means that the *F* test has the simplest formula in all of statistics: just divide one variance by the other. When testing for the difference in variance between two samples, you divide them like this:

F formula comparing two sample variances: $$F = \frac{{s_1}^2}{{s_2}^2}$$

Since the number at the top of this ratio (i.e., the **numerator** 分子) is a square, and so is the number at the bottom of this ratio (i.e., the **denominator** 分母), *F* can never go below zero, but there's no upper limit (as when dividing a huge variance by a tiny one). Thus the family of *F* distributions is not symmetrical like the normal distribution or the family of *t* distributions, but is instead highly skewed, as I'll show you in a moment.

Like *t* distributions, your specific *F* distribution depends on the *df* (related to *n* = sample size). But since *F* is defined by two variances in two different places in the *F* formula, you need two *df* values, not just one. In the type of *F* test we're doing here, $df_1 = n_1 - 1$ and $df_2 = n_2 - 1$, just as for one-sample *t* tests:

Degrees of freedom for *F* test: $df_1 = n_1 - 1$ $df_2 = n_2 - 1$

R has the same type of function family for *F* distributions as for lots of other distributions, this time all with names ending in the lower-case letter "f" (cf. "t" for *t* distributions, "norm" for the normal distribution, "binom" for the binomial distribution, and "unif" for the uniform distribution). So *p* values can be computed with the **pf(F, df1, df2)** function, and we can make density plots with the help of the **df(F, df1, df2)** function (note: **df()** does *not* stand for degrees of freedom here!). Similarly, you can compute *p* values in Excel using **=FDIST(F, df1, df2)**.

Before looking at the *p* values more carefully, let's draw a picture of a few *F* distributions, as shown in Figure 2 (remember that line type **lty=1** is a solid line, **lty=2** is a dashed line, line width **lwd=1** is a thin line and **lwd=2** is a thick line). Notice how the tails, especially on the right, get thinner as the sample size (and thus *df*) goes up, which as with the *t* test, implies that it's easier to get a significant result (greater statistical power) in a larger sample (assuming the null hypothesis is false).

```
plot(function(x) {df(x, df1=5, df2=5)}, xlim=c(0,3), ylim=c(0,1), ylab = "", xlab = "")
plot(function(x) {df(x, df1=5, df2=10)}, xlim=c(0,3), add=T, lty=2)
plot(function(x) {df(x, df1=10, df2=5)}, xlim=c(0,3), add=T, lwd=2)
plot(function(x) {df(x, df1=10, df2=10)}, xlim=c(0,3), add=T, lty=2, lwd=2)
legend("topright", lty=c(1,2,1,2), lwd = c(1,1,2,2),
  legend=c("df1=5, df2=5", "df1=5, df2=10", "df1=10, df2=5", "df1=10, df2=10"))
```

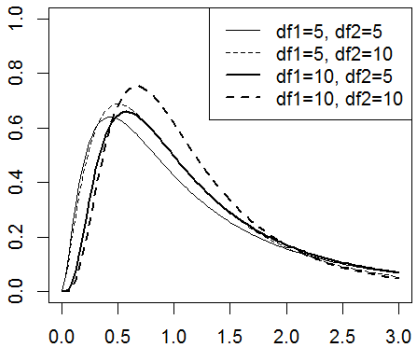

Figure 2. Some *F* distributions

Note also that as the sample sizes get bigger (i.e., with larger *df* values), the mode moves towards the right and the right tail gets thinner. You might think, then, that as the sample sizes

move to infinity, the *F* distribution will turn into the normal distribution, just like the *t* distribution does. You would be wrong, though, because the normal distribution has no lower end, while the *F* distribution can't go below zero; the mode of the *F* distribution also gets closer to 1, not 0 as for the standard normal distribution. However, as we'll discuss more when we get to ANOVA, the *F* distribution is indeed related to the normal distribution in another way. Namely, the *t* value that you compute in a *t* test is related to the *F* value that you compute in ANOVA, in that $t(df)^2 = F(1,df)$, and you'll get exactly the same *p* values too: *t* tests are just a special case of ANOVA!

How do we actually compute *p* values from *F* distributions? Confusingly, while the R command **pf()** is consistent with **pnorm()** and **pbinom()** in giving the area to the *left* of the *F* value (**cumulative probability**, remember?), the Excel function **=FDIST()** gives the area to the *right* of the *F* value, which is thus unlike Excel's other functions **=NORMSDIST()** and **=BINOMDIST()**. Thus **pf(F, df1, df2) = 1 - FDIST(F, df1, df2)**.

Does this asymmetry of the *F* distribution mean that we should use the one-tailed (right-sided) *p* value in an *F* test? Not at all. The asymmetry comes from how ratios work: in s_1^2/s_2^2, a big s_1^2 will spread *F* out infinitely to the right, but a big s_2^2 will squish *F* between 1 and 0. We can't know ahead of time whether s_1^2 or s_2^2 will be bigger, though, so the most reasonable null hypothesis is still the nondirectional H_0: $\sigma_1^2 = \sigma_2^2$. All the asymmetry means is that we cannot compute the two-tailed *p* value simply by doubling either of the one-tailed *p* values. Instead, we have to add the two *p* values together. I'll give an example of this shortly.

The easiest way to do an *F* test in Excel is to choose the appropriate tool in the data analysis toolbox (「**F 檢定：兩個常態母體變異數的檢定**」). When selecting the ranges for your two samples, Excel will assume that you want the first one to be at the top of the *F* ratio, and the second one to be at the bottom. Thus if the first sample has a larger variance than the latter, *F* will be higher than 1, and if it's very much higher, you'll get a *p* value representing the area of the tail to the right, which will thus be very small. If, however, the first sample has a much smaller variance than the second, you'll get a very tiny *F* value - but you'll still get the same small *p* value, since it's still measuring the tail on the right.

Try it on the first data set (boys and girls). If you did it right, you could report the results like so: $F(9, 9) = 1.94$, $p = .169$. Notice how you have to report both *df* values, in the order numerator (top of ratio) before denominator (bottom of ratio). I got an *F* value above 1, because I selected Boys as the first sample and Girls as the second sample, and the first has a higher variance than the second. If I select the samples the other way around, I get this result: $F(9, 9) = 0.52$, $p = .169$. Note that the two Fs are inverses: 1.94 = 1/0.52, and vice versa. So as I said, if you put the small variance on top, you get a lower *F* value (below 1), but the *p* value is exactly the same.

The latter analysis may confuse your readers, though, since in most statistical tests, higher test statistic values mean lower *p* values. We've already seen this for the binomial test, the *z*

test, the one-sample *t* test, and both the unpaired and paired *t* tests. So when running a variance test, I would recommend ordering your two samples so that you divide the larger variance by the smaller one. This way a higher *F* implies a lower *p*.

Let's make sure this *p* value is right. As noted above, Excel gives you the one-tailed (right-tailed) *p* value for the F distribution, as is made explicit in the statistics report table produced by the *F* test tool. Thus you can get exactly the same *p* value in this report table by using the **=FDIST(F, df1, df2)** function, where you select the report table cells that contain *F*, df_1, and df_2. Try it!

Anyway, we just ran an *F* test to check if the variances for Boys and Girls in the first data set are significantly different, and we got $p = .169 > .05$. What can we do with this information? Technically, nothing: it's a null result, and so it's possible that the null hypothesis is still actually false, but our test just wasn't powerful enough to reject it. In other words, maybe we have made a Type II error. But at least this null result tells us that we have no motivation to reject the null hypothesis about the equal variances, and so we may as well stick with the version of the unpaired *t* test that assumes this.

The situation is quite different for the second unpaired data set, also about boys and girls (the one we've ignored through the whole chapter so far). Doing an *F* test here, we get $F(9, 9) = 0.099$, $p = .001$, or with the samples turned around so "bigger *F*" implies "smaller *p*," which is more intuitive, we get $F(9, 9) = 10.03$, $p = .001$. (Again, note that $0.099 = 1/10.03$.) So in this case, we are justified in rejecting the null hypothesis that the two population variances are the same, and so maybe we should think about using a more general version of the unpaired *t* test that doesn't depend on this assumption. We'll learn how to do this in the next section.

R also has a built-in *F* test function for comparing variances, called **var.test()**. Like Excel's *F* test tool, it computes *F* by dividing the variance of the first sample by that of the second, so if you want big *F* to imply small *p*, you should make sure you put the higher-variance sample in first. For example, this is what I would do for the first data set, where Boys has higher variance than Girls (you should try it too):

```
var.test(boys1,girls1) # Search for "boys1" above to reload if you lost it
```

R's text report tells us the following: $F = 1.9368$, num(erator) $df = 9$, denom(inator) $df = 9$, $p = .339$. Unlike Excel's *F* test tool and the **=FTEST()** function, it doesn't give $p = .169$ because **var.test()**, like **t.test()**, gives you the two-tailed *p* by default (just as with **t.test()**, you can change the default tail setting by changing the alternative argument from the default value **"two.sided"** to **"less"** or **"greater"**, but we won't bother with that). Notice also that $.339 \neq 2\times.169$, since the *F* distribution isn't symmetrical. Instead, the *p* value computed by **var.test()** is the sum of the left tail for s^2_{small}/s^2_{big} plus the right tail for s^2_{big}/s^2_{small}:

```
Fval.bs = var(boys1)/var(girls1) # Big above small
Fval.sb = var(girls1)/var(boys1) # Small above big
F.left = pf(Fval.sb,9,9) # Area of left tail (small/big gives smaller F value)
F.right = 1-pf(Fval.bs,9,9) # Area of right tail (= 1 minus area to left of bigger F value)
F.left + F.right # There's our 0.3389936
```

Using **var.test()** on the second set of boy/girl data confirms their unlike variances:

```
study2 = subset(bg, bg$Study==2)
boys2 = study2$Measure[study1$Gender=="Boy"]
girls2 = study2$Measure[study1$Gender=="Girl"]
var.test(boys2,girls2) # F(9,9) = .1, p = .002
```

Now, since Excel gives you a one-tailed *p* value for the *F* variance test and R gives you a two-tailed *p* value, which should you use and report? Remember that the higher the *p* value, the less the risk of a Type I error (thinking there's a pattern when there really isn't), but the greater the risk of a Type II error (thinking there's no pattern when there really is). Meanwhile, however, remember that it's almost always better to use a two-tailed test, so nobody can accuse you of cheating. In this case, neither Type I or Type II errors are much to worry about, since even the ordinary unpaired *t* test gives pretty good results, even if its assumption of equal population variances is not true. For consistency, I would recommend reporting R's two-tailed *p* value for the *F* variance test, but if you decide to use Excel's test instead, just tell your readers that it's a one-tailed test.

2.3.4 Welch's *t* test (unpaired *t* test not assuming equal variances)

Now that we know that the second boy/girl data set involves samples with significantly different variances, how could we improve the unpaired *t* test to optimize its accuracy (balancing Type I and Type II error risk) in this kind of situation? An otherwise obscure statistician named B. L. Welch (no Wikipedia article!), worrying about this in the late 1940s, came up with an ugly but now standard solution. The simplest name for the solution is **Welch's *t* test**, but it's also called the **unpaired *t* test not assuming equal variances**, or the **heteroscedastic** (異方差的) ***t* test**. You may remember that "hetero" means "different" (as in "heterosexual"), but you probably forgot that "scedastic" means "dispersion". So heteroscedastic basically means "different variance". By contrast, the unpaired *t* tests we have doing so far are **homoscedastic**.

This may seem like a picky little issue, but we've already seen that homoscedasticity is also an assumption of Pearson's correlation and regression: to get valid *p* values, we have to assume that the variance remains constant all along the scatter plot. The issue will also return, in a somewhat different form, when we get to ANOVA.

Welch's discovery was that we can deal with heteroscedasticity in an unpaired *t* test by changing the formula for *df*, rather than the formula for *t*. In fact, the formula for *t* in this case is exactly the same as for the simplest version of the unpaired *t* test (see, e.g., Everitt & Hay, 1992, or even Wikipedia). That is, you don't use pooled variance, since you can't rely on the assumption that the population variances are the same.

Welch's trick is to reduce the size of the *df* value, so that you use a thicker-tailed *t* distribution, which indicates that you're less sure of where the population mean actually is. This is similar to the trick we used when we moved from the *z* test to the *t* test: the formulas are actually identical, but we shifted to a distribution with thicker tails, since we had to pretend that the population variance was the same as our sample variance. Now we're pretending that the difference between the two sample variances doesn't matter. In both cases, we "punish" ourselves for this false assumption by spreading out our uncertainty in thicker-tailed distributions, which raises the *p* values (lessening our risk of Type I errors).

The specific amount you reduce *df* depends not just on the sample sizes, but the sample variances as well, calculated using the so-called **Welch correction**. Somewhat annoyingly, this correction usually produces a weird-looking *df* that is not a whole integer. You can see the incredibly ugly formula in Everitt & Hay (1992, p. 60), or in Wikipedia's article on this test, but I'll spare your delicate eyes.

Excel computes Welch's *t* test when you choose the Analysis ToolPak tool 「**t 檢定：兩個母體平均數差的檢定，假設變異數不相等**」**(t-Test: Two-Sample Assuming Unequal Variances)**. Note that Excel's name has the wrong scope: Welch's *t* test does *not* assume that the variances are unequal, but instead assumes *nothing* about the variances at all. So more properly, Excel should call it 「t 檢定：兩個母體平均數差的檢定，不假設變異數相等」: it should be not(assume(equal variance)), rather than assume(not(equal variance)). Another thing to note about Excel's implementation of Welch's *t* test is that it rounds the weird *df* given by Welch's correction to the nearest integer.

Let's try it. Using the second set of unpaired data (which "failed" the *F* test), we get $t(11) = 4.91$, $p = .0005$. If we report this, we should explain that it's heteroscedastic so people understand that unexpected *df*. Since the sample sizes are $n_1 = n_2 = 10$, the ordinary unpaired *t* test should give us $df = n_1 + n_2 - 2 = 18$, so your readers may be surprised to see $df = 11$ here, unless you tell them why.

If we use an ordinary (homoscedastic) unpaired *t* test on this data set, we get $t(18) = 4.91$, $p = .0001$. Thus the *t* value is the same, but the *df* value is higher, which makes the *p* value slightly lower (in this particular case there's no practical effect, since it's still significant). Similarly, we could use a heteroscedastic test on the first boy/girl data set (which "passed" the *F* test). With a homoscedastic test we got $t(18) = 3.08$, $p = .006$, but with a heteroscedastic test we get $t(16) = 3.08$, $p = .007$. Again the *t* values are the same, but since the *df* is lowered by

Welch's correction, the *p* value is slightly higher (again it doesn't make any practical difference in this particular case).

Now let's see how R does it. Remember that unless you set **equal.var = T**, R's **t.test()** function makes no assumption about the variances, so it performs Welch's *t* test by default, and that's what it tells you it's doing in the text report. Note also that the weird *df* given by Welch's correction is *not* rounded to a whole integer in R. And finally, notice that it names Mr. Welch personally (poor guy, with no Wikipedia article).

t.test(boys2,girls2) # Use default setting (don't assume equal variance)

```
        Welch Two Sample t-test

data:  boys2 and girls2
t = 4.9078, df = 10.775, p-value = 0.0004948
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.8416934 2.2168454
sample estimates:
mean of x mean of y
 4.469680  2.940410
```

Just as in these fake examples, it usually doesn't matter which version of the unpaired *t* test you use, unless your *p* value happens to be very close to .05 on one side or the other. Moreover, as I've mentioned before, parametric tests are **robust**, which means that you can violate their assumptions and they'll still give you pretty reliable inferences (see Glass, Peckham, & Sanders, 1972; Rasch & Guiard, 2004). In the case of the ordinary unpaired *t* test, this is especially true if your sample sizes are large and/or pretty similar.

For generality, though, R's **t.test()** function makes Welch's *t* test the default. This makes it useful when you're checking your own data during the course of your research, without having to worry about possible troubles with variance. Nevertheless, Welch's *t* test is less powerful, since it ignores part of your data (i.e., it doesn't care if your samples really have the same variance or not), and it also gives you that weird *df* value, so I would recommend using the ordinary unpaired *t* test for reports. Only if your sample variances are very different might you want to please critics who might otherwise suspect that you purposely ignored the variance difference to get a significant result. Welch's *t* test has a lower risk of Type I errors, so if it's still significant that way, your critics might be mollified.

2.3.4 How the paired *t* test works

After all the math stress of the unpaired *t* tests and the variance test, here's some good news: the math for the paired *t* test is super simple! Why? Well, since the two samples are

paired, this time the null hypothesis really concerns the *differences between the pairs*. That is, you're not looking at mean(x_1) - mean(x_2), but rather at mean(x_1 - x_2), which we can symbolize as M_D (D for "difference"). The null hypothesis claims that the population associated with this sample of differences has some fixed "boring" mean, so it turns directly into a one-sample *t* test, like so:

Paired *t* test value: $t = \frac{M_D - (\mu_1 - \mu_2)}{SE} = \frac{M_D - \mu_D}{SE}$

As with the other *t* tests, the "boring" mean of the null hypothesis is usually taken to be zero (e.g., the null hypothesis is that noun and verb responses will be the same, so their differences in the null population is zero), making the null hypothesis, formally speaking, look like this: H_0:$\mu_D = 0$. So the usual version of the paired *t* test formula is like so:

Paired *t* test value (H_0:$\mu_D = 0$): $t = \frac{M_D - \mu_D}{SE} = \frac{M_D}{SE}$

Just as in the ordinary one-sample *t* test, we don't have to estimate anything except σ, so we only have to subtract 1 to get the *df*, so $df = n - 1$ (note that due to the pairing of the data, $n = n_1 = n_2$).

df for paired *t* test: $df = n - 1$

And the math just keeps getting easier: the estimated **standard error of the difference between paired sample means** is based on the standard deviation of the *differences*, so it can be estimated just as in the one-sample *t* test too:

Standard error for paired *t* test: $SE = \sqrt{\frac{{s_D}^2}{n}} = \frac{s_D}{\sqrt{n}}$

Let's check if this formula gives us the correct *t* value for the first noun-verb study:

```
# Start over, in case you lost the data
nv = read.table("NounsVerbs.txt",T)
study3 = subset(nv, nv$Study==3)
nouns3 = study3$Measure[study3$WordType=="Noun"]
verbs3 = study3$Measure[study3$WordType=="Verb"]
tval1 = t.test(nouns3,verbs3,paired=T)$statistic # This extracts just the t value
```

```
# Recalculate using above formula
Difs = nouns3-verbs3 # R knows to subtract vectors element by element
MD = mean(Difs)
SD = sd(Difs)
SE = SD/sqrt(length(nouns3))
tval2 = MD/SE
tval1 == tval2 # TRUE!
```

2.3.5 *t* tests and regression

Before we climb out of the sea of mathematics, there's one more crucial point to make, in fact the same point we made in the previous chapter, namely: everything is regression. Unpaired and paired *t* tests are actually just special cases of linear regression, and this connection is made explicit in R syntax.

Remember that R prefers data to be arranged so that each row is a separate data point, and each column describes something about this data point, including the dependent measure and the factor(s) defining it. So in the **BoysGirls.txt** file, the key values are in the dependent variable **Measure**, whereas the independent variable **Gender** defines whether the measure comes from a boy or a girl (each row is a different person); in **NounsGirls.txt**, the random variable **Participant** defines the pairs, the independent variable **WordType** defines whether the measure comes from a noun or a verb, and again **Measure** is the dependent variable. Although this format is different from some other statistics programs, there's a very logical reason for it: R prefers to treat your data as a kind of Y ~ X model, where Y is the dependent measure and X is/are the independent variable(s), possibly with random variables in there too.

In fact, if your data set is arranged the right way, the **t.test()** function lets you analyze it using the **formula** notation (with that ~ symbol), instead of listing the two values separately. So instead of **t.test(A, B)**, you can write something like **t.test(Y ~ X)**. Try it for our boy/girl and noun/verb data below, and you'll see that both syntactic forms give you identical results:

```
t.test(boys1, girls1) # Unpaired (Welch's) t test
t.test(Measure ~ Gender, data=study1) # Exactly the same thing
# The following only work if Participants are in same order for nouns and verbs:
t.test(nouns3, verbs3, paired=T) # Paired t test
t.test(Measure ~ WordType, data = study3, paired = T) # Exactly the same thing
```

If you ever need to do a one-sample *t* test (outside of this book's mathematical explanations, that is), you can use the **t.test()** function for that too. Note that **Y~1** is how R represents a model where there are no predictor variables: the **1** represents the regression intercept (which makes sense if you think about regression in terms of matrix mathematics, but forget about that). In the examples below the regression notation isn't doing anything useful, but this won't be the last time we see the 1-as-intercept trick in this book.

```
t.test(boys1) # One-sample t test for H0:mean(boys1) = 0
t.test(boys1 ~ 1) # Exactly the same thing
```

By the way, the fact that R can run a *t* test in this way kind of falsifies the title I gave this chapter ("Comparing two continuous variables"), since it's possible to see a *t* test as analyzing just *one* continuous dependent variable, with the "two" part coming from the categorical independent variable.

I mentioned in the last chapter that R uses the formula notation everywhere. In addition to **lm()**, **xtabs()**, and **t.test()**, the **boxplot()** function from a few chapters back also permits you to use the formula notation (try it!):

```
boxplot(Measure~Gender, data=study1) # Sure looks like it should be significant
```

This formula notation is not just an arbitrary convention: *t* tests are, in fact, just a special case of linear regression. In the case of unpaired (homoscedastic) *t* tests, the **t.test()** function gives exactly the same results as **cor.test()** or **lm()**, if you code your independent variable in terms of two numbers rather than in terms of character strings. For example, if you use 0 vs. 1 (e.g., Boy = 0, Girl = 1), this is called **dummy coding**:

```
study1$Girlness = 1*(study1$Gender == "Girl") # New variable: change F/T to 0/1
study1 # See?
```

Now we do an ordinary correlation predicting Measure from these 0s and 1s; this method is technically called a **point-biserial correlation**. Try out the following code, and note the similarities and differences in the results:

```
cor.test(study1$Girlness, study1$Measure) # Note the df, t and p values
summary(lm(Measure~Girlness,data=study1)) # Same df, t and p values
summary(lm(Measure~Gender,data=study1)) # Same: lm() treats factors as dummy
t.test(Measure~Girlness,data=study1,var.equal=T) # Same df, t and p values as above
t.test(Measure~Girlness,data=study1) # Welch's t test works differently
```

Only Welch's *t* test gives different *df* and *p* from Pearson's correlation and regression, since unlike these tests it's a heteroscedastic test. This shows that I wasn't lying when I said that correlation and linear regression really do assume homoscedacity (equal variance across the whole range of data).

What about the paired *t* test? The link with regression is more subtle, but it's still there, if only at a conceptual level. Namely, think in terms of the formula Y ~ X, where Y is the dependent variable (e.g., Measure in the noun-verb study) and the independent variable X uses dummy coding (e.g., 0 = Noun, 1 = Verb). Now imagine that the data are arranged in a kind of scatter plot, encoded as (x, y) points, so $(0, y_0)$ gives one of the values of Y when X = 0 (Noun)

and (1, y_1) is the paired value for when X = 1 (Verb). What's the slope between these two points? It's the change in Y divided by the change in X, right? But that's just y_1-y_0 divided by 1-0, which is just the difference y_1-y_0 itself. And this is just the differences between paired data points used in the paired *t* test. So the differences in a paired *t* test are actually slopes! This insight, in turn, means you can do a one-sample *t* test on all of these slopes to see if they are significantly different from a flat horizontal line showing no effect of *x* on *y* (remember that for a horizontal line, slope = 0). Putting it symbolically:

$$b \text{ (slope)} = \frac{\text{change in } y}{\text{change in } x} = \frac{y_1 - y_0}{1 - 0} = y_1 - y_0 = D \text{ (difference)}$$

Unfortunately, this approach is merely conceptual; linear regression can't actually compute a slope for just two points. Still, regression really is everywhere in statistics, and this link will provide us with several other useful tricks in later chapters.

3. Effect size for *t* tests

We've been computing a lot of *p* values in this chapter. However, as we know, statistical significance isn't the same as real-life significance: it's quite possible to get a low *p* value for a very tiny effect. For this reason, many researchers today (including the American Psychology Association and the many journals influenced by the APA) emphasize that **effect size** should also be measured and reported, in addition to *p* values.

3.1 The what and why of effect sizes

There are two very commonly used ways to measure effect size for *t* tests: a simple one for reports in the text, and a more complicated one that's most often used for graphs.

The simple way of measuring effect size is **Cohen's *d*** (named after American statistician Jacob Cohen, 1923-1998, and the letter "d", for "difference" maybe?). This uses the standard deviation as a measuring stick. If your effect is large relative to the variability in the population (as estimated from your sample and the specific statistical test you used), then this is a more meaningful result than if you had observed the same-sized effect in a more "noisy" situation.

Applying this logic, Cohen's *d* is calculated as shown below for three types of *t* tests; note that the particular choice of standard deviation measuring stick is based on the particular *t* value formula used in the particular *t* test. By convention, $d < .2$ is taken as indicating a small effect and $d > .8$ indicates a big effect.

Cohen's *d* for one sample *t* test: $d = \frac{M-\mu}{s}$

Cohen's *d* for unpaired (homoscedastic) *t* test: $d = \frac{M_1 - M_2}{s_p}$

Cohen's *d* for paired *t* test: $d = \frac{M_D}{s_D}$

While the base package in R doesn't have a built-in function for computing the above equations, you can write your own function(s) based on the above formulas, or use the function **cohensD()** in the package **lsr** (for "Learning Statistics with R"; Navarro, 2014). Try it, and you'll see that all of these analyses show quite large effect sizes (in fact, $d > 1$).

```
library(lsr) # After you've installed it
cohensD(boys1) # Cohen's d for one-sample t test assuming mu = 0
mean(boys1)/sd(boys1) # Same result as above (try confirming below results too)
cohensD(boys1,girls1) # Cohen's d for unpaired homoscedastic t test
cohensD(Measure~Gender,data=study1) # Same result as above
cohensD(boys2,girls2,method="unequal") # Cohen's d for Welch's t test
cohensD(nouns3,verbs3,method="paired") # Cohen's d for paired t test
```

The more complex way to measure effect size, most often used in plots, is motivated by limitations with statistical hypothesis testing (i.e., computing *p* values). The first we've already mentioned: statistical significance isn't the same as practical significance in the real world. But there are other limitations too (see, e.g., Loftus & Masson, 1994). One is that statistical software (and textbooks like mine) tend to give the impression that statistical hypothesis testing is a purely mechanical algorithm that somehow gives us direct access to the truth. Yet real science is messy, and is done by our messy human brains in discussions within a messy social world, so interpreting the mathematical results given by statistics is more effective if we have enough extra information to allow our messy brains to be flexible. Another limitation is that you can't make a graph of a *p* value, and as I've repeatedly emphasized, people have a more intuitive sense about pictures than about numbers.

Most of these specific problems relate to a more fundamental issue. Namely, hypothesis tests like *t* tests give **point estimates** (點估計), not just those magical *p* values, but also for population values like the mean: a single number is offered as the best guess. Actually, since the sample mean is merely an estimate of the population mean, we also need to know how reliable this estimate is, in a practical, flexible, and intuitive way. Reporting the sample standard deviation, standard error (*SE*), and/or Cohen's *d* may help express the variability and reliability of our point estimates, but by themselves, they are also just point estimates.

Putting all of this together, a standard response is to represent our confidence in a point estimate by putting the point within a range of values. This range is called a **confidence interval** (信賴區間), and the degree of confidence is expressed as a percentage. For example,

a 95% confidence interval represents a range of values where we can be 95% sure about the population mean (albeit in a somewhat confusing sense).

Confidence intervals? Yes, confidence intervals! I'm finally going to explain them! We've already seen Excel and R giving us information about confidence intervals in their outputs. So when you use Excel's regression (迴歸) tool, the coefficients table includes 「下限 95%」 and 「上限 95%」. Similarly, R's **cor.test()**, **t.test()**, and even **var.test()** functions all report the "95 percent confidence interval" around their point estimates (i.e., around Pearson's *r*, the mean difference, and the variance ratio, respectively).

Just as a smaller standard deviation means less variability, and a smaller standard error (*SE*) means less uncertainty about the population mean, so too the width of a 95% confidence interval shows how reliable our point estimate is. Specifically, a *narrower* 95% confidence interval provides *more* information than a wider 95% confidence interval. If this is not intuitive, think of this analogy: imagine hearing on the radio that "We are 95% certain that the escaped criminal is somewhere in Chiayi county" or that "We are 95% certain that the escaped criminal is somewhere in Chiayi city". Which report provides more information? The one with the smaller range (Chiayi city), right?

Now what if we compare a 95% confidence interval and a 99% confidence interval for exactly the same data set. Which should be smaller? The answer is that a 95% confidence interval will be smaller. This may seem counterintuitive, but it may make more sense if you now imagine a 100% confidence interval. How wide would that have to be, to be sure that the "true" point estimate is definitely within it? If this estimate is on an infinitely long scale, as statistics generally assumes (remember that the tails of the normal distribution go off into infinity on both ends), then the only way to be 100% confident is if the confidence interval itself is also infinitely long! By the same logic, a 99% confidence interval will be wider than a 95% confidence interval for the same data.

What do those percentages mean, exactly? It sounds like for a 95% confidence interval, the population mean has a 95% chance of being somewhere within it, right? Sadly, that is not what a confidence interval means, at least not in traditional (frequentist) statistics. Now pay close attention, and I'll try to explain why.

Remember that in traditional inferential statistics, your sample is seen as just one of many (usually an infinite number of) possible samples from the same population, each sample with its own mean. You have to understand a 95% confidence interval within this sampling framework. If we take the 95% confidence interval calculated from *our* sample, and put a copy of it around the mean of *every* sample from the population (without changing the size of the interval), then 95% of *these copies* should contain the population mean. You might be able to see what I mean if you study Figure 3 for a while.

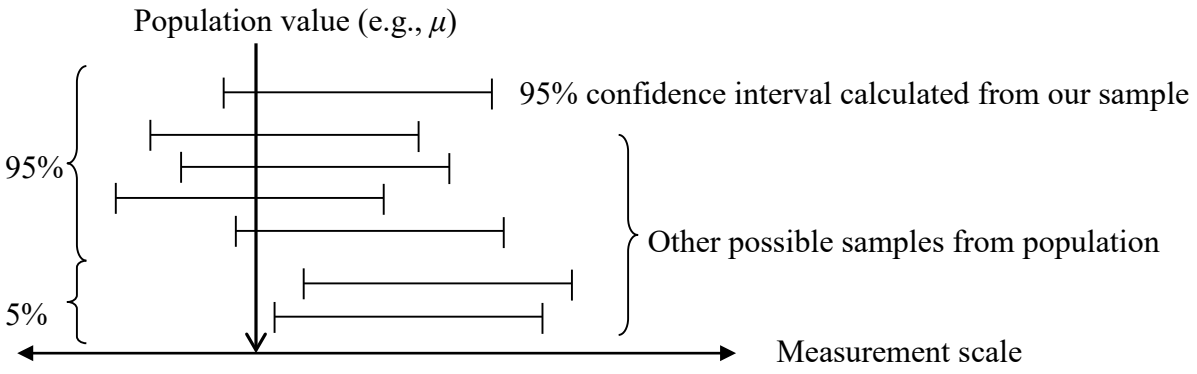

Figure 3. What a confidence interval really is

Everybody finds this very confusing, since the logic seems backwards. Here's what a statistician says about his reaction when he first learned this (Lee, 2004, p. xiii):

> When I first learned a little statistics, I felt confused, and others I spoke to confessed that they had similar feelings. [...] [T]he statement that a 95% confidence interval for an unknown parameter ran from -2 to +2 sounded as if the parameter lay in that interval with 95% probability and yet I was warned that all I could say was that if I carried out similar procedures time after time then the unknown parameters would lie in the confidence intervals I constructed 95% of the time....

Lee (2004) is a book on Bayesian statistics, and indeed, this is another situation where Bayesian logic is actually more intuitive, since it doesn't rely on the backwards logic of starting with the null population and then imagining that our observed sample was taken from it. As we'll see at the end of the book, the Bayesian version of the confidence interval (sometimes called the highest density interval) works very much the way we wish that the frequentist confidence interval worked, but doesn't.

3.2 More about the math of confidence intervals

Since Bayesian statistics has not yet taken over the world, we have to learn how to compute the traditional confidence interval. The size and location of a confidence interval is based on some specific statistical hypothesis test, so the numbers you need to compute it are exactly the same as used in a hypothesis test, namely the sample mean, the standard error, and a test statistic (e.g., *z* or *t* or *F*). This makes the underlying logic not too different from Cohen's *d*, except that now we're dealing with an interval instead of a point estimate.

If you want a 95% confidence interval, that's the same as setting the critical value (臨界値) $\alpha = .05$ (since .95 = 1 – .05). Similarly, a 99% confidence interval would set $\alpha = .01$. (As usual, we're talking about two-tailed tests here.) For example, for a 95% confidence interval

associated with a *z* test, we would use $z_{95\%crit} = 1.96$ (that's the point where the areas beyond both tails add up to .05, remember?). Now all we have to do is flip the numbers around in the formula. For example, for the one-sample *z* test, the test statistic is calculated this way (remember?):

z test statistic: $z = \frac{M-\mu}{SE} = \frac{M-\mu}{\sigma/\sqrt{n}}$

This time, however, instead of computing *z*, what we want is to estimate a range for μ. So we replace *z* with $z_{95\%crit}$ and solve for μ (we get the "±" because we're assuming a two-tailed test, where *M* might be bigger or smaller than μ).

z test confidence interval: $\mu = M \pm (z_{95\%crit})(SE) = M \pm (z_{95\%crit})(\sigma/\sqrt{n}) = M \pm (1.96)(\sigma/\sqrt{n})$

The formulas for confidence intervals all have basically the same form, regardless of the test they're based on. For example, here's how to compute the 95% confidence interval for a one-sample *t* test (as usual, you still must choose the right *t* distribution from the *t* family, here using $df = n - 1$).

One sample *t* test confidence interval: $\mu = M \pm (t(df)_{95\%crit})(SE) = M \pm (t(df)_{95\%crit}(s/\sqrt{n})$

Let's try computing a confidence interval for the one-sample *t* test. Remember your sister, who might be a Martian in disguise? Here are the key values again:

$\mu = 20$ ms VOT for Martians
$n = 16$ recordings of your sister
$df = n - 1 = 15$
$M = 22$ ms for your sister
$SD = 3$ ms for your sister

Using these numbers, we can compute the 95% confidence interval for the population mean estimated with our sample as follows:

$\mu = 22 \pm (t(15)_{95\%crit})(3/\sqrt{16}) = ...$

What's $t(15)_{95\%crit}$? Well, what *t* value on the $t(15)$ distribution defines two tails that together have an area of .05? We've often used the **p...** family of functions (e.g., **pt()**) to get area from points, but remember that we also have **q...** functions (e.g., **qt()**) to get points from

areas ("q" for "quantile"). So the *t* value on the *t*(15) distribution that defines an area .025 *to the left* is the following:

qt(p=0.05/2, df=15) # -2.13...

This is a negative number (do you remember why?), so the value we want to plug into the above formula is the absolute value (so we can apply ± to it):

abs(qt(p=0.05/2, df=15)) # 2.13....

If you think about, you should also be able to figure out why $t(15)_{95\%crit}$ is bigger than $z_{95\%crit} = 1.96$. If you can't, here's why: the tails of the *t* distributions are fatter than tails of the normal distribution (especially when *df* is small), so you need to move further out (both ways) to shrink the total area of the tails down to just .05. If the sample is big, though, the *t* distribution becomes normal:

abs(qt(p=0.05/2, df=1000)) # There's that 1.96...

Now that we have our critical value, we can finish the calculation as follows. Note how I cleverly exploited R's love of vectors to put a vector inside **qt()**, so I could calculate both ends of the confidence interval at the same time:

```
22 + qt(c(0.05/2,1-0.05/2),df=15)*(3/sqrt(16))
[1] 20.40141 23.59859
```

Did we do it right? Compare the above values with what you get automatically in the text report from **t.test()**:

```
set.seed(19483) # So you match my specific random sample (remember?)
sister = round(rnorm(n=16,mean=22,sd=3)) # Here, M = 22 & s = 3.03315
t.test(sister,mu=20) # H0: mu = 20
# 95 percent confidence interval reported by t.test:
#    20.38375 23.61625    ...Slightly different from above since M != 22 & s != 3
```

Notice that the Martian mean of 20 is outside of the range of 20.40 to 23.60. Since our analysis shows that 95% of the confidence intervals of this size will contain the true population mean, we are justified in saying that your sister's mean (22) is significantly different from the Martian mean (20).

Confidence intervals don't just repeat the information given by a *p* value, however. For one thing, the size of a confidence interval is not related just to *p*, but also to the sample size. For example, suppose John and Mary each perform *z* tests on unrelated hypotheses involving

distinct populations that both happen to have $\sigma = 10$, and both get a p value of .04 (just barely significant). Now suppose that John's sample size is 100, and Mary's is 10. This means that the width of John's 95% confidence interval is (2)(1.96)(10/√100) = 3.92 (do you see where I got this formula from?), while for Mary it's (2)(1.96)(10/√10) = 12.40, over three times wider. So in order for Mary's result to be significant, her sample mean must be much farther away from the hypothesized population mean than John's sample mean. This makes her effect size more impressive, even though the significance values are identical.

3.3 Plotting confidence intervals

The most common use for confidence intervals is in graphs. In fact, because of the relation between confidence intervals and hypothesis testing, a properly calculated confidence interval can visually express whether a specific statistical analysis is significant. In one sense, this is obviously very cool and useful. In another sense, though, it can be kind of a confusing mixture between two different functions of a graph (and different functions of statistics more generally): describing the data, versus drawing inferences from it. For this reason confidence intervals are easy to misuse in complex situations, and I will not discuss them in detail after this chapter. They're very widely used, however, so at least I hope this brief discussion will give you a sense of where they come from.

Confidence intervals are often used to make **error bars** (誤差線) in graphs. Error bars can also be used to express different kinds of information about variability, including sample standard deviation (variability within your sample) and standard error (variability of your sample as a reflection of the population), but in these uses, the bars do not express effect size or statistical significance directly. Here I'll focus on error bars for confidence intervals.

For example, Figure 4 shows the results of the Martian sister study. I represent the Martian mean (20 ms) by the bottom of the graph. The error bar associated with the sample mean doesn't cross the bottom of the graph, thereby expressing, visually, that your sister's VOT is significantly different by a two-tailed t test with $\alpha = .05$ (so $p < .05$).

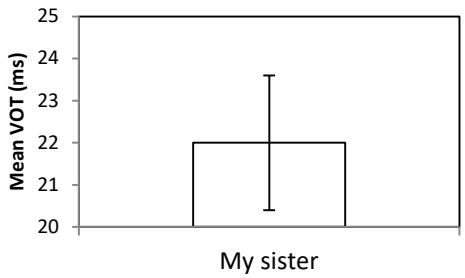

Figure 4. Visual evidence that your sister is not a Martian

How did I make this graph? I started by computing the sample mean, so I could plot the lonely white bar in the bar plot. Then I calculated the top and bottom of the confidence interval as explained above. Excel has a few other options for computing confidence intervals, though they're not easy to use. For example, you could try searching the internet for how to use the function **=CONFIDENCE_T()** (older version) or **=CONFIDENCE.T()** (newer version), or look into Excel's Descriptive Statistics tool in the Analysis ToolPak. But once you have the values of the endpoints of the confidence interval, you can use Excel's graphing tool to add an error bar (誤差線) and type in the values you want to define each end.

For some reason the R base package doesn't have a built-in function for plotting error bars; maybe the R creators share my discomfort with them (i.e., that they mix descriptive with interferential statistics). Fortunately, an internet search reveals that a guy named James Holland Jones (at http://monkeysuncle.stanford.edu/?p=485) has created a function for adding error bars using R's base graphic package. This gives you Figure 5.

```
# Here's the function
error.bar = function(x, y, upper, lower=upper, length=0.1,...){
  if(length(x) != length(y) | length(y) !=length(lower) | length(lower) != length(upper))
    stop("vectors must be same length")
    arrows(x,y+upper, x, y-lower, angle=90, code=3, length=length, ...)
}

# Now let's try it on your sister
conf95 = abs(qt(0.05/2,df=15))*( sd(sister)/sqrt(16)) # What you add/subtract
sister.plot = barplot(mean(sister),ylim=c(0,30),      # Create and name basic bar plot
  ylab= "Mean VOT (ms)", main="Sister vs. Martians")
error.bar(sister.plot, mean(sister), conf95) # Add error bar
abline(h=20, lty=2) # Dashed line for Martians, outside sister's 95% confidence interval
```

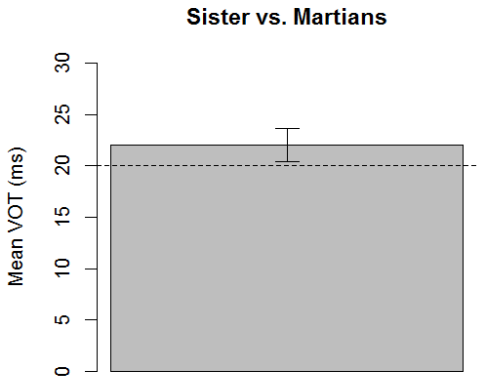

Figure 5. Another way to plot your sister

The graphing package **ggplot2** also has a built-in function for plotting error bars, yielding Figure 6:

```
sister.mean = data.frame(Speaker= "MySister",VOT=mean(sister)) # Use data frame!
library(ggplot2) # Remember that you have to have this package installed already
ggplot(sister.mean, aes(x=Speaker, y=VOT)) +
geom_bar(fill="white", color="black",    # White bars with black borders
  stat="identity") +       # To turn off default statistic, which is the count
geom_errorbar(aes(ymin=VOT-conf95, ymax=VOT+conf95), # Length of error bar
  width=0.2)       # Width of error bar
```

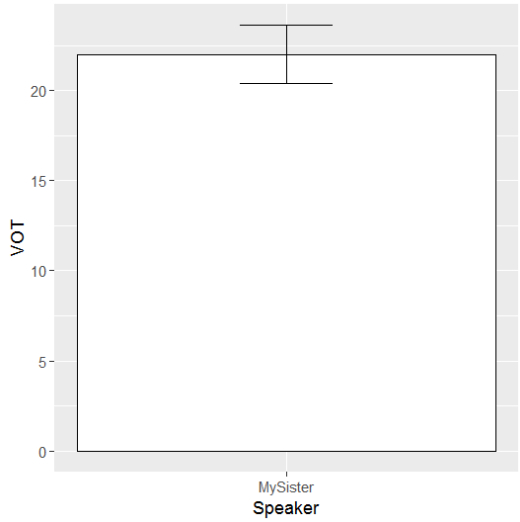

Figure 6. Yet another way to plot your sister

As noted above, the formulas for confidence intervals are derived from the formulas from the associated test. So you can calculate confidence intervals for any hypothesis test, including two-sample tests. For example, if the samples are independent, then we can use the following formula (based on the formula for the ordinary unpaired *t* test):

Unpaired *t* test confidence interval:

$$\mu_1 - \mu_2 = (M_1 - M_2) \pm (t(df)_{95\%crit})(SE)$$

$$= (M_1 - M_2) \pm (t(df)_{95\%crit})(s_{M1\text{-}M2})$$

$$= (M_1 - M_2) \pm (t(df)_{95\%crit}) \left(\sqrt{\frac{{s_p}^2}{n_1} + \frac{{s_p}^2}{n_2}} \right)$$

Unlike the one-sample *t* test confidence interval, for the unpaired *t* test confidence interval, Excel's *t* test tool gives all of the values that we need to calculate it. Namely, when we run t 檢定：兩個母體平均數差的檢定, we get ${s_p}^2$ as 「Pooled 變異數」, and $t(df)_{95\%crit}$ as 「臨界值：雙尾」, assuming that we kept α at the default 0.05 when running it. We can compute confidence intervals for paired *t* tests using the same logic, by switching around the paired *t* test formula.

But why bother using any of these formulas, when R gives us the confidence intervals for all of these *t* tests automatically? In order to add confidence intervals to a plot, we just take the upper and lower values of the **conf.int** object given by **t.test()** and find how long one half of it is, giving us the single value that we add/subtract to the means in the plot.

For example, let's plot the error bars for the first boy/girl data set, based on an unpaired *t* test, giving Figure 7:

```
t.res1 = t.test(boys1,girls1,var.equal=T) # Unpaired t test results
conf95.1 = (t.res1$conf.int[2]-t.res1$conf.int[1])/2 # Do you see how this works?
boy = mean(boys1)
girl = mean(girls1)
bg.plot = barplot(c(boy,girl), names.arg=c("Boys","Girls"), ylab = "Measure",
  ylim=c(0,6))    # Create and name basic bar plot
error.bar(bg.plot, c(boy,girl), c(conf95.1,conf95.1)) # Add error bars (same size)
```

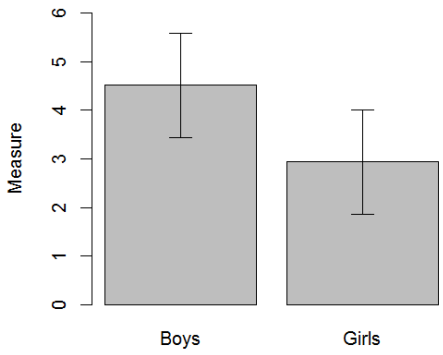

Figure 7. Plotting boys and girls

Note that neither mean falls within the error bar of the other category: this shows visually that $p < .05$. This is so even though the two error bars are the same size, since they both reflect a single statistical tests (unpaired *t* test), even though the variance actually differs somewhat across the two samples. Moreover, be very careful with this visual trick. First, overlapping confidence intervals don't say anything about significance: $p < .05$ only if each *mean* is outside the other 95% confidence interval. Second, this trick only works for between-group data (as in this unpaired *t* test), but not for within-group comparisons (as in a paired *t* tests). Third, sometimes error bars in published papers represent the standard error (*SE*) instead of confidence intervals, but the *SE* is merely used to calculate confidence interval (multiplied by the critical value), rather than being something that tells you the *p* value directly. But don't feel bad if you're still confused: Belia, Fidler, Williams, & Cumming (2005) found that many professional researchers were confused as well!

Now let's plot error bars for the third data set (nouns vs. verbs), based on a paired *t* test, where again $p < .05$, and so in Figure 8, the means of each sample do not fall within the confidence interval of the other sample. This figure also reflects visually that the effect size for this within-group design is greater (smaller 95% confidence intervals) than for the between-group design shown in Figure 7 (larger 95% confidence intervals). This is so despite the fact that actual numbers in these two fake data sets are exactly the same!

```
t.res3 = t.test(nouns3,verbs3,paired=T) # Paired t test results
conf95.3 = (t.res3$conf.int[2]-t.res3$conf.int[1])/2 # Same trick
N = mean(nouns3)
V = mean(verbs3)
nv.plot = barplot(c(N,V), names.arg=c("Nouns","Verbs"), ylab = "Measure",
  ylim=c(0,6))    # Create and name basic bar plot
error.bar(nv.plot, c(N,V), c(conf95.3,conf95.3)) # Add error bars (same size)
```

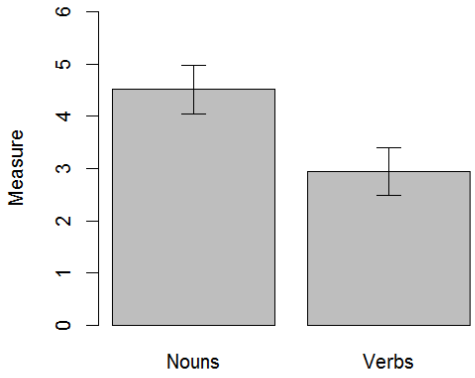

Figure 8. Plotting nouns and verbs.

Other R packages include functions not only for adding error bars to plots, but also for computing the confidence intervals that they're based on. One package for this measure of effect size is named, appropriately enough, **effects** (Fox, 2003); we'll see how to use it in a later chapter. Another package with error bar functions is **sciplot** (Morales, 2017).

4. Nonparametric tests

All of the above *t* tests assume that your samples come from a normally distributed population (though as we noted, they are relatively robust to violations of assumptions like this). So if your sample isn't at all normally distributed, to be safe you might want to try a **nonparametric** two sample test, that is, a test (like Spearman's ranked correlation) that makes no assumptions about distribution shape.

The **Mann-Whitney *U* test** (named after the coauthors of the late 1940s paper that proposed it, Henry Mann [1905-2000] and his student Donald Whitney; don't ask me where the capital-U comes from) is the standard nonparametric alternative to the unpaired *t* test. Like Spearman's ranked correlation, it relies solely on ranking, so it doesn't care about normality, which is good if your sample is skewed or ordinal, but for the same reason, it's also less powerful.

The heart of the *U* test is a comparison of how often the values in one sample outrank the values in the other sample. More precisely, the *U* value represents the number of times that individual ranks in the overall *lower* ranking group exceed individual ranks in the overall *higher* ranking group. So this test is a rarity among statistical tests: a *low U* value means that items in the lower-ranked sample are *lower* than items in the higher-ranked sample than would be expected by chance. Thus lower *U* means lower *p* too (greater significance).

There is also a nonparametric alternative to the paired *t* test, called the **Wilcoxon *T* test** (or **Wilcoxon signed-rank test**), named after a guy named Smith. No, sorry, he was actually named Frank Wilcoxon (1892-1965). This test is also based on ranks, so it has all the strengths and weaknesses of other ranked tests.

Like a paired *t* test, what you look at in the capital-*T* Wilcoxon test is the differences across the paired samples, in this case the differences of each pair's relative ranks. The idea is that you want to see if "most" of the differences are the same sign (i.e., "mostly" positive or "mostly" negative), which would show that there's a consistent difference between the samples. The *T* value then represents the sum of the ranks for the lower ranking set of difference scores (i.e., the "minus" scores vs. "plus" scores). If this sum is low, this shows that there is indeed a consistent difference between the samples (e.g., "mostly" positive differences), so the lower the *T* value, the less likely the difference between groups is due to chance. Thus as with *U*, the *lower* the *T* value, the *lower* the *p* value, backwards from the usual relation between test statistics and *p* values. Also like the Mann-Whitney *U* test, the Wilcoxon *T* test is an asymptotic test, so it still works better for larger samples. The Wilcoxon *T* test also assumes that the differences between the paired samples are symmetric (even if they're not necessarily normal).

Again, I'm not going to go through the detailed procedure here. If you want to do the Mann-Whitney test or the Wilcoxon test, you should use a computer program (unless your sample is quite small and you like to do arithmetic). In R, because the ranking algorithms are related, the command for both is the same: **wilcox.test()** (poor Mann and Whitney!). The syntax is otherwise kind of like that for **t.test()**: if you use the default argument values, you get the unpaired Mann-Whitney test, but if you set the **paired** argument to **TRUE**, you get the Wilcoxon test. You can even use the formula notation!

For example, compare the *p* values you get for our four data sets with the ones we got with the various types of *t* tests. Annoyingly, R's text output doesn't give you either *U* or *T* as test statistics, but rather the algebraically related *W* and *V* (respectively). I guess if you're going

to report this output, you're going to have to refer to those things instead. There are no degrees of freedom here either, but since sample size matters, you should report that too.

```
wilcox.test(boys1, girls1) # (two-tailed) Mann-Whitney U test
wilcox.test(Measure ~ Gender, data=study1) # The same, using formula notation
wilcox.test(nouns3, verbs3, paired=T) # (two-tailed) Wilcoxon T test
wilcox.test (Measure ~ WordType, data = study3, paired = T) # Exactly the same thing
```

As with Welch's *t* test (i.e., the heteroscedastic unpaired *t* test), it may not be crucial to worry too much about violations of the assumptions of parametric tests, since they are pretty robust (Glass et al., 1972). That is, you still get pretty reliable *p* values even if the tests' assumptions are violated (especially in larger samples). You can get a sense of this from the simulation below, where I randomly generate 10,000 couples of 20-element independent uniformly distributed samples (so they aren't normal), and then run both the ordinary unpaired *t* test and the Mann-Whitney *U* test on them, and finally count how often each test is significant ($p < .05$) and how often the two tests conflict in significance. When I ran this, I only found that both tests were significant around 5% of the time (which is proper, since these are totally random samples, so the null hypothesis is true), and they only disagreed about significance about 1% of the time (i.e., quite rarely). Try it!

```
t.sig = 0 # Counts how often t is sig (p < .05)
u.sig = 0 # Counts how often U is sig (p < .05)
conflict = 0 # Will count cases where the significances mismatch
for (i in 1:10000) {
  sample1 = runif(20)
  sample2 = runif(20) # So H0:mu1=mu2 should show p < .05 with 1/20 chance
  t.p = t.test(sample1,sample2,var.equal=T)$p.value
  u.p = wilcox.test(sample1,sample2)$p.value
  if (t.p < 0.05) {
    t.sig = t.sig + 1
  }
  if (u.p < 0.05) {
    u.sig = u.sig + 1
  }
  if ((t.p < 0.05 & u.p >= 0.05) | (t.p >= 0.05 & u.p < 0.05)) { # Remember "|" = "or"
    conflict = conflict + 1
  }
}
t.sig/10000 # Ideally should be 0.05 (since null hypothesis is true for random samples)
u.sig/10000 # Ditto
conflict/10000 # Is this low enough not to worry?
```

Just as the nonparametric Wilcoxon *T* test builds on the logic of the nonparametric Mann-Whitney test (and in the parametric world, both the unpaired and paired *t* tests are related to

the one-sample *t* test, which is related to Pearson's correlation coefficient *r* and the one-sample *z* test, and ANOVA is related to the *t* test too, etc etc), so it is possible to invent new nonparametric tests for special purposes - which people are still doing today! In a surprisingly recent paper, Parker, Vannest, Davis, and Sauber (2011) propose a solution to the challenge of analyzing the effect of an experimental manipulation (e.g., language therapy) on a single person, given that any difference in "before treatment" vs. "after treatment" is confounded with time (e.g., maybe the person just got better or worse naturally and the treatment didn't do anything). In their method, they combine a Mann-Whitney U test (for the before/after comparison) with Kendall's tau as a nonparametric correlation method (introduced in the previous chapter) in order to reduce the influence of mere time trends. The resulting measure is thus called **Tau-U** (see https://www.jepusto.com/what-is-tau-u/ for some R code). As an example of this method being used in clinical linguistics, see Liang (2020).

To summarize, then, nonparametric tests like the Mann-Whitney and Wilcoxon tests have (almost) no assumptions, so they are (almost) always "legal" to use, and thus are unlikely to commit a Type I error. However, precisely because they use less information than parametric tests (e.g., only the ranks in a ranked test, instead of the actual values), they are less powerful, and thus more likely to commit a Type II error. Meanwhile, parametric tests like the unpaired and paired *t* tests turn out to be relatively robust to violations of their assumptions anyway, so unless your distribution is very weirdly shaped or very small, you probably don't need to use nonparametric tests. Still, you sometimes see them being used precisely under those circumstances. For example, genuine linguistic uses of the Mann-Whitney test include Baars, Motley, & MacKay (1975) and Lee & Goldrick (2008).

5. Conclusions

This chapter was mainly about *t* tests, which are all related to each other, and in turn are related to the *z* test, ANOVA, correlation, and regression. We started with the unpaired *t* test, which is used to test the null hypothesis that the population means associated with two independent samples are the same (or differ by some fixed amount). Exactly how you compute the unpaired *t* test depends on whether the sample sizes and sample variances are the same, but you need to know which version you're using when you run an analysis in Excel or R, but the actual calculations are done by the computer. In a long side discussion, we learned how to test the null hypothesis that the variances of the populations associated with two independent samples are the same, using something called the *F* test, which will return to center stage when we get to ANOVA. Then we saw how the paired *t* test is used to test the null hypothesis that the difference between population means is zero for paired sample data; it is mathematically much simpler than the unpaired *t* test, but again we can just let Excel or R do the math for us. We also learned some useful but less crucial stuff, starting with how to go beyond *p* values to

estimate effect size as well, using Cohen's *d* and confidence intervals (even though they don't mean exactly what we wish they did). Finally we saw that if we (or our critics) are worried that the normality assumption of *t* tests is violated too severely, we have the option of using nonparametric rank-based tests like the Mann-Whitney *U* test (for unpaired samples) and the Wilcoxon *T* test (for paired samples).

References

Baars, B. J., Motley, M. T., & MacKay, D. G. (1975). Output editing for lexical status in artificially elicited slips of the tongue. *Journal of Verbal Learning and Verbal Behavior, 14*, 382-391.

Belia, S., Fidler, F., Williams, J., & Cumming, G. (2005). Researchers misunderstand confidence intervals and standard error bars. *Psychological Methods, 10* (4), 389-396.

Everitt, B. & Hay, D. (1992). *Talking about statistics: A psychologist's guide to design and analysis.* Edward Arnold.

Fox, J. (2003). Effect displays in R for generalised linear models. *Journal of Statistical Software, 8*(15), 1-27.

Glass, G. V., Peckham, P. D., & Sanders, J. R. (1972). Consequences of failure to meet assumptions underlying the fixed effects analysis of variance and covariance. *Review of Educational Research, 42*, 237-288.

Hogg, R. V., & Craig, A. T. (1995). *Introduction to mathematical statistics* (fifth edition). New Jersey: Prentice Hall.

Lee, P. M. (2004). *Bayesian statistics: An introduction* (third edition). Arnold.

Lee, Y., & Goldrick, M. (2008). The emergence of sub-syllabic representations. *Journal of Memory and Language, 59*(2), 155-168.

Liang, S. Y. (2020). *Language-literacy intervention through telepractice for school-age children: A single case design study*. Nashville, TN: Vanderbilt University doctoral dissertation.

Loftus, G. R., & Masson, M. E. J. (1994). Using confidence intervals in within-subject designs. *Psychonomic Bulletin & Review, 1*, 476-490.

Mansfield, E. (1986). *Basic statistics with applications*. W. W. Norton & Company.

Morales, M. (2017). sciplot: Scientific Graphing Functions for Factorial Designs. R package version 1.1-1. https://CRAN.R-project.org/package=sciplot

Navarro, D. (2014). *Learning statistics with R: A tutorial for psychology students and other beginners*. University of Adelaide ms. <https://open.umn.edu/opentextbooks/textbooks/559>

Parker, R. I., Vannest, K. J., Davis, J. L., & Sauber, S. B. (2011). Combining nonoverlap and trend for single-case research: Tau-U. *Behavior Therapy*, *42*(2), 284-299.

Rasch, D., & Guiard, V. (2004). The robustness of parametric statistical methods. *Psychology Science, 46* (2), 175-208.

Woods, A., Fletcher, P., & Hughes, A. (1986). *Statistics in language studies.* Cambridge, UK: Cambridge University Press.