13. Comparing two means

In the previous chapter we covered the situation when your outcome variable is nominal scale and your predictor variable[1] is also nominal scale. Lots of real world situations have that character, and so you'll find that chi-square tests in particular are quite widely used. However, you're much more likely to find yourself in a situation where your outcome variable is interval scale or higher, and what you're interested in is whether the average value of the outcome variable is higher in one group or another. For instance, a psychologist might want to know if anxiety levels are higher among parents than non-parents, or if working memory capacity is reduced by listening to music (relative to not listening to music). In a medical context, we might want to know if a new drug increases or decreases blood pressure. An agricultural scientist might want to know whether adding phosphorus to Australian native plants will kill them.[2] In all these situations, our outcome variable is a fairly continuous, interval or ratio scale variable; and our predictor is a binary "grouping" variable. In other words, we want to compare the means of the two groups.

The standard answer to the problem of comparing means is to use a t-test, of which there are several varieties depending on exactly what question you want to solve. As a consequence, the majority of this chapter focuses on different types of t-test: one sample t-tests are discussed in Section 13.2, independent samples t-tests are discussed in Sections 13.3 and 13.4, and paired samples t-tests are discussed in Section 13.5. After that, we'll talk a bit about Cohen's d, which is the standard measure of effect size for a t-test (Section 13.8). The later sections of the chapter focus on the assumptions of the t-tests, and possible remedies if they are violated. However, before discussing any of these useful things, we'll start with a discussion of the z-test.

13.1 The one-sample z-test

In this section I'll describe one of the most useless tests in all of statistics: the **z-test**. Seriously – this test is almost never used in real life. Its only real purpose is that, when teaching statistics, it's a very convenient stepping stone along the way towards the t-test, which is probably the most (over)used tool in all statistics.

[1]We won't cover multiple predictors until Chapter 15

[2]Informal experimentation in my garden suggests that yes, it does. Australian natives are adapted to low phosphorus levels relative to everywhere else on Earth, apparently, so if you've bought a house with a bunch of exotics and you want to plant natives, don't follow my example: keep them separate. Nutrients to European plants are poison to Australian ones. There's probably a joke in that, but I can't figure out what it is.

13.1.1 The inference problem that the test addresses

To introduce the idea behind the z-test, let's use a simple example. A friend of mine, Dr Zeppo, grades his introductory statistics class on a curve. Let's suppose that the average grade in his class is 67.5, and the standard deviation is 9.5. Of his many hundreds of students, it turns out that 20 of them also take psychology classes. Out of curiosity, I find myself wondering: do the psychology students tend to get the same grades as everyone else (i.e., mean 67.5) or do they tend to score higher or lower? He emails me the `zeppo.Rdata` file, which I use to pull up the `grades` of those students,

```
> load( "zeppo.Rdata" )
> print( grades )
 [1] 50 60 60 64 66 66 67 69 70 74 76 76 77 79 79 79 81 82 82 89
```

and calculate the mean:

```
> mean( grades )
[1] 72.3
```

Hm. It *might* be that the psychology students are scoring a bit higher than normal: that sample mean of $\bar{X} = 72.3$ is a fair bit higher than the hypothesised population mean of $\mu = 67.5$, but on the other hand, a sample size of $N = 20$ isn't all that big. Maybe it's pure chance.

To answer the question, it helps to be able to write down what it is that I think I know. Firstly, I know that the sample mean is $\bar{X} = 72.3$. If I'm willing to assume that the psychology students have the same standard deviation as the rest of the class then I can say that the population standard deviation is $\sigma = 9.5$. I'll also assume that since Dr Zeppo is grading to a curve, the psychology student grades are normally distributed.

Next, it helps to be clear about what I want to learn from the data. In this case, my research hypothesis relates to the *population* mean μ for the psychology student grades, which is unknown. Specifically, I want to know if $\mu = 67.5$ or not. Given that this is what I know, can we devise a hypothesis test to solve our problem? The data, along with the hypothesised distribution from which they are thought to arise, are shown in Figure 13.1. Not entirely obvious what the right answer is, is it? For this, we are going to need some statistics.

13.1.2 Constructing the hypothesis test

The first step in constructing a hypothesis test is to be clear about what the null and alternative hypotheses are. This isn't too hard to do. Our null hypothesis, H_0, is that the true population mean μ for psychology student grades is 67.5%; and our alternative hypothesis is that the population mean *isn't* 67.5%. If we write this in mathematical notation, these hypotheses become,

$$\begin{array}{ll} H_0: & \mu = 67.5 \\ H_1: & \mu \neq 67.5 \end{array}$$

though to be honest this notation doesn't add much to our understanding of the problem, it's just a compact way of writing down what we're trying to learn from the data. The null hypotheses H_0 and the alternative hypothesis H_1 for our test are both illustrated in Figure 13.2. In addition to providing us with these hypotheses, the scenario outlined above provides us with a fair amount of background knowledge that might be useful. Specifically, there are two special pieces of information that we can add:

1. The psychology grades are normally distributed.

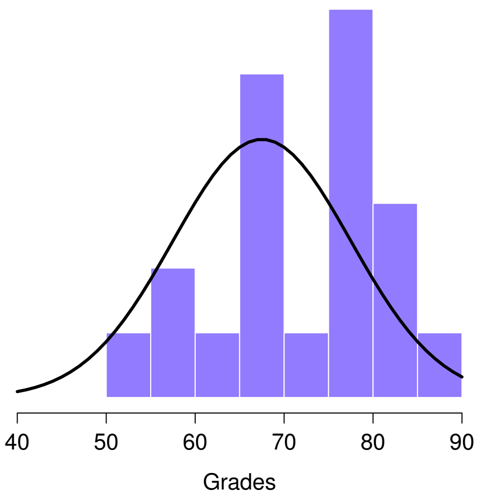


Figure 13.1: The theoretical distribution (solid line) from which the psychology student grades (grey bars) are supposed to have been generated.

2. The true standard deviation of these scores σ is known to be 9.5.

For the moment, we'll act as if these are absolutely trustworthy facts. In real life, this kind of absolutely trustworthy background knowledge doesn't exist, and so if we want to rely on these facts we'll just have make the *assumption* that these things are true. However, since these assumptions may or may not be warranted, we might need to check them. For now though, we'll keep things simple.

The next step is to figure out what we would be a good choice for a diagnostic test statistic; something that would help us discriminate between H_0 and H_1. Given that the hypotheses all refer to the population mean μ, you'd feel pretty confident that the sample mean $\bar{X}$ would be a pretty useful place to start. What we could do, is look at the difference between the sample mean $\bar{X}$ and the value that the null hypothesis predicts for the population mean. In our example, that would mean we calculate $\bar{X} - 67.5$. More generally, if we let μ_0 refer to the value that the null hypothesis claims is our population mean, then we'd want to calculate

$$\bar{X} - \mu_0$$

If this quantity equals or is very close to 0, things are looking good for the null hypothesis. If this quantity is a long way away from 0, then it's looking less likely that the null hypothesis is worth retaining. But how far away from zero should it be for us to reject H_0?

To figure that out, we need to be a bit more sneaky, and we'll need to rely on those two pieces of background knowledge that I wrote down previously, namely that the raw data are normally distributed, and we know the value of the population standard deviation σ. If the null hypothesis is actually true, and the true mean is μ_0, then these facts together mean that we know the complete population distribution of the data: a normal distribution with mean μ_0 and standard deviation σ. Adopting the notation from Section 9.5, a statistician might write this as:

$$X \sim \text{Normal}(\mu_0, \sigma^2)$$

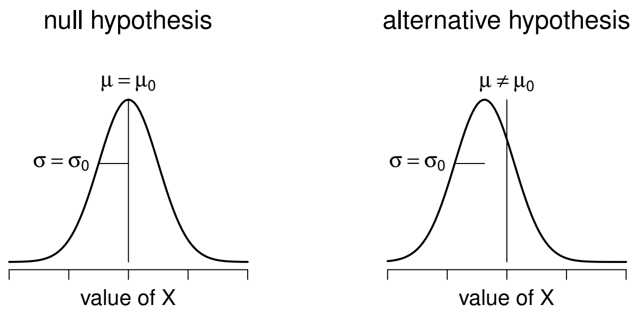


Figure 13.2: Graphical illustration of the null and alternative hypotheses assumed by the one sample z-test (the two sided version, that is). The null and alternative hypotheses both assume that the population distribution is normal, and additionally assumes that the population standard deviation is known (fixed at some value σ_0). The null hypothesis (left) is that the population mean μ is equal to some specified value μ_0. The alternative hypothesis is that the population mean differs from this value, $\mu \neq \mu_0$.

....

Okay, if that's true, then what can we say about the distribution of $\bar{X}$? Well, as we discussed earlier (see Section 10.3.3), the sampling distribution of the mean $\bar{X}$ is also normal, and has mean μ. But the standard deviation of this sampling distribution $\text{SE}(\bar{X})$, which is called the *standard error of the mean*, is

$$\text{SE}(\bar{X}) = \frac{\sigma}{\sqrt{N}}$$

In other words, if the null hypothesis is true then the sampling distribution of the mean can be written as follows:

$$\bar{X} \sim \text{Normal}(\mu_0, \text{SE}(\bar{X}))$$

Now comes the trick. What we can do is convert the sample mean $\bar{X}$ into a standard score (Section 5.6). This is conventionally written as z, but for now I'm going to refer to it as $z_{\bar{X}}$. (The reason for using this expanded notation is to help you remember that we're calculating standardised version of a sample mean, *not* a standardised version of a single observation, which is what a z-score usually refers to). When we do so, the z-score for our sample mean is

$$z_{\bar{X}} = \frac{\bar{X} - \mu_0}{\text{SE}(\bar{X})}$$

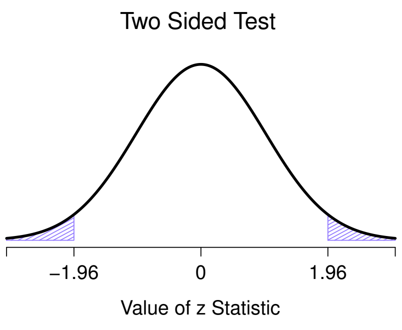
or, equivalently

$$z_{\bar{X}} = \frac{\bar{X} - \mu_0}{\sigma / \sqrt{N}}$$

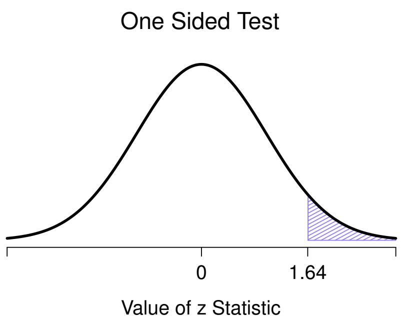
This z-score is our test statistic. The nice thing about using this as our test statistic is that like all z-scores, it has a standard normal distribution:

$$z_{\bar{X}} \sim \text{Normal}(0, 1)$$

(again, see Section 5.6 if you've forgotten why this is true). In other words, regardless of what scale the original data are on, the z-statistic iteself always has the same interpretation: it's equal to the number of standard errors that separate the observed sample mean $\bar{X}$ from the population mean μ_0 predicted by



(a)



(b)

Figure 13.3: Rejection regions for the two-sided z-test (panel a) and the one-sided z-test (panel b).

the null hypothesis. Better yet, regardless of what the population parameters for the raw scores actually are, the 5% critical regions for z-test are always the same, as illustrated in Figure 13.3. And what this meant, way back in the days where people did all their statistics by hand, is that someone could publish a table like this:

	critical z value	
desired α level	two-sided test	one-sided test
.1	1.644854	1.281552
.05	1.959964	1.644854
.01	2.575829	2.326348
.001	3.290527	3.090232

which in turn meant that researchers could calculate their z-statistic by hand, and then look up the critical value in a text book. That was an incredibly handy thing to be able to do back then, but it's kind of unnecessary these days, since it's trivially easy to do it with software like R.

13.1.3 A worked example using R

Now, as I mentioned earlier, the z-test is almost never used in practice. It's so rarely used in real life that the basic installation of R doesn't have a built in function for it. However, the test is so incredibly simple that it's really easy to do one manually. Let's go back to the data from Dr Zeppo's class. Having loaded the `grades` data, the first thing I need to do is calculate the sample mean:

```
> sample.mean <- mean( grades )
> print( sample.mean )
[1] 72.3
```

Then, I create variables corresponding to known population standard deviation ($\sigma = 9.5$), and the value of the population mean that the null hypothesis specifies ($\mu_0 = 67.5$):

```
> mu.null <- 67.5
> sd.true <- 9.5
```

Let's also create a variable for the sample size. We could count up the number of observations ourselves, and type `N <- 20` at the command prompt, but counting is tedious and repetitive. Let's get R to do the tedious repetitive bit by using the `length()` function, which tells us how many elements there are in a vector:

```
> N <- length( grades )
> print( N )
[1] 20
```

Next, let's calculate the (true) standard error of the mean:

```
> sem.true <- sd.true / sqrt(N)
> print(sem.true)
[1] 2.124265
```

And finally, we calculate our z-score:

```
> z.score <- (sample.mean - mu.null) / sem.true
> print( z.score )
[1] 2.259606
```

At this point, we would traditionally look up the value 2.26 in our table of critical values. Our original hypothesis was two-sided (we didn't really have any theory about whether psych students would be better or worse at statistics than other students) so our hypothesis test is two-sided (or two-tailed) also. Looking at the little table that I showed earlier, we can see that 2.26 is bigger than the critical value of 1.96 that would be required to be significant at $\alpha = .05$, but smaller than the value of 2.58 that would be required to be significant at a level of $\alpha = .01$. Therefore, we can conclude that we have a significant effect, which we might write up by saying something like this:

> With a mean grade of 73.2 in the sample of psychology students, and assuming a true population standard deviation of 9.5, we can conclude that the psychology students have significantly different statistics scores to the class average ($z = 2.26$, $N = 20$, $p < .05$).

However, what if want an exact p-value? Well, back in the day, the tables of critical values were huge, and so you could look up your actual z-value, and find the smallest value of α for which your data would be significant (which, as discussed earlier, is the very definition of a p-value). However, looking things up in books is tedious, and typing things into computers is awesome. So let's do it using R instead. Now, notice that the α level of a z-test (or any other test, for that matter) defines the total area "under the curve" for the critical region, right? That is, if we set $\alpha = .05$ for a two-sided test, then the critical region is set up such that the area under the curve for the critical region is .05. And, for the z-test, the critical value of 1.96 is chosen that way because the area in the lower tail (i.e., below -1.96) is exactly .025 and the area under the upper tail (i.e., above 1.96) is exactly .025. So, since our observed z-statistic is 2.26, why not calculate the area under the curve below -2.26 or above 2.26? In R we can calculate this using the `pnorm()` function. For the upper tail:

```
> upper.area <- pnorm( q = z.score, lower.tail = FALSE )
> print( upper.area )
[1] 0.01192287
```

The `lower.tail = FALSE` is me telling R to calculate the area under the curve from 2.26 *and upwards*. If I'd told it that `lower.tail = TRUE`, then R would calculate the area from 2.26 *and below*, and it would give me an answer 0.9880771. Alternatively, to calculate the area from -2.26 and below, we get

```
> lower.area <- pnorm( q = -z.score, lower.tail = TRUE )
> print( lower.area )
[1] 0.01192287
```

Thus we get our p-value:

```
> p.value <- lower.area + upper.area
> print( p.value )
[1] 0.02384574
```

13.1.4 Assumptions of the z-test

As I've said before, all statistical tests make assumptions. Some tests make reasonable assumptions, while other tests do not. The test I've just described – the one sample z-test – makes three basic assumptions. These are:

- *Normality*. As usually described, the z-test assumes that the true population distribution is normal.[3] is often pretty reasonable, and not only that, it's an assumption that we can check if we feel worried about it (see Section 13.9).
- *Independence*. The second assumption of the test is that the observations in your data set are not correlated with each other, or related to each other in some funny way. This isn't as easy to check statistically: it relies a bit on good experimetal design. An obvious (and stupid) example of something that violates this assumption is a data set where you "copy" the same observation over and over again in your data file: so you end up with a massive "sample size", consisting of only one genuine observation. More realistically, you have to ask yourself if it's really plausible to imagine that each observation is a completely random sample from the population that you're interested in. In practice, this assumption is never met; but we try our best to design studies that minimise the problems of correlated data.
- *Known standard deviation*. The third assumption of the z-test is that the true standard deviation of the population is known to the researcher. This is just stupid. In no real world data analysis problem do you know the standard deviation σ of some population, but are completely ignorant about the mean μ. In other words, this assumption is *always* wrong.

In view of the stupidity of assuming that σ is known, let's see if we can live without it. This takes us out of the dreary domain of the z-test, and into the magical kingdom of the t-test, with unicorns and fairies and leprechauns, and um...

13.2 The one-sample t-test

After some thought, I decided that it might not be safe to assume that the psychology student grades necessarily have the same standard deviation as the other students in Dr Zeppo's class. After all, if I'm

[3] Actually this is too strong. Strictly speaking the z test only requires that the sampling distribution of the mean be normally distributed; if the population is normal then it necessarily follows that the sampling distribution of the mean is also normal. However, as we saw when talking about the central limit theorem, it's quite possible (even commonplace) for the sampling distribution to be normal even if the population distribution itself is non-normal. However, in light of the sheer ridiculousness of the assumption that the true standard deviation is known, there really isn't much point in going into details on this front!

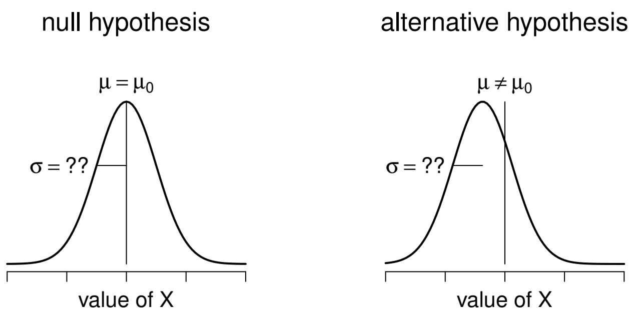


Figure 13.4: Graphical illustration of the null and alternative hypotheses assumed by the (two sided) one sample t-test. Note the similarity to the z-test (Figure 13.2). The null hypothesis is that the population mean μ is equal to some specified value μ_0, and the alternative hypothesis is that it is not. Like the z-test, we assume that the data are normally distributed; but we do not assume that the population standard deviation σ is known in advance.

. . . .

entertaining the hypothesis that they don't have the same mean, then why should I believe that they absolutely have the same standard deviation? In view of this, I should really stop assuming that I know the true value of σ. This violates the assumptions of my z-test, so in one sense I'm back to square one. However, it's not like I'm completely bereft of options. After all, I've still got my raw data, and those raw data give me an *estimate* of the population standard deviation:

```
> sd( grades )
[1] 9.520615
```

In other words, while I can't say that I know that $\sigma = 9.5$, I *can* say that $\hat{\sigma} = 9.52$.

Okay, cool. The obvious thing that you might think to do is run a z-test, but using the estimated standard deviation of 9.52 instead of relying on my assumption that the true standard deviation is 9.5. So, we could just type this new number into R and out would come the answer. And you probably wouldn't be surprised to hear that this would still give us a significant result. This approach is close, but it's not *quite* correct. Because we are now relying on an *estimate* of the population standard deviation, we need to make some adjustment for the fact that we have some uncertainty about what the true population standard deviation actually is. Maybe our data are just a fluke . . . maybe the true population standard deviation is 11, for instance. But if that were actually true, and we ran the z-test assuming $\sigma = 11$, then the result would end up being *non-significant*. That's a problem, and it's one we're going to have to address.

13.2.1 Introducing the t-test

This ambiguity is annoying, and it was resolved in 1908 by a guy called William Sealy Gosset (Student, 1908), who was working as a chemist for the Guinness brewery at the time (see J. F. Box, 1987). Because Guinness took a dim view of its employees publishing statistical analysis (apparently they felt it was a trade secret), he published the work under the pseudonym "A Student", and to this day, the full name of the t-test is actually **Student's t-test**. The key thing that Gosset figured out is how we should

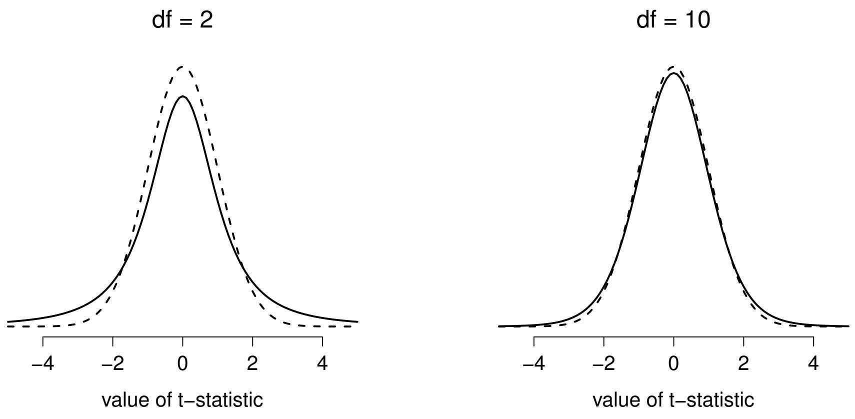


Figure 13.5: The t distribution with 2 degrees of freedom (left) and 10 degrees of freedom (right), with a standard normal distribution (i.e., mean 0 and std dev 1) plotted as dotted lines for comparison purposes. Notice that the t distribution has heavier tails (higher kurtosis) than the normal distribution; this effect is quite exaggerated when the degrees of freedom are very small, but negligible for larger values. In other words, for large df the t distribution is essentially identical to a normal distribution.

....

accommodate the fact that we aren't completely sure what the true standard deviation is.[4] The answer is that it subtly changes the sampling distribution. In the t-test, our test statistic (now called a t-statistic) is calculated in exactly the same way I mentioned above. If our null hypothesis is that the true mean is μ, but our sample has mean $\bar{X}$ and our estimate of the population standard deviation is $\hat{\sigma}$, then our t statistic is:

$$t = \frac{\bar{X} - \mu}{\hat{\sigma}/\sqrt{N}}$$

The only thing that has changed in the equation is that instead of using the known true value σ, we use the estimate $\hat{\sigma}$. And if this estimate has been constructed from N observations, then the sampling distribution turns into a t-distribution with $N - 1$ **degrees of freedom** (df). The t distribution is very similar to the normal distribution, but has "heavier" tails, as discussed earlier in Section 9.6 and illustrated in Figure 13.5. Notice, though, that as df gets larger, the t-distribution starts to look identical to the standard normal distribution. This is as it should be: if you have a sample size of $N = 70,000,000$ then your "estimate" of the standard deviation would be pretty much perfect, right? So, you should expect that for large N, the t-test would behave exactly the same way as a z-test. And that's exactly what happens!

13.2.2 Doing the test in R

As you might expect, the mechanics of the t-test are almost identical to the mechanics of the z-test. So there's not much point in going through the tedious exercise of showing you how to do the calculations using low level commands: it's pretty much identical to the calculations that we did earlier, except that we use the estimated standard deviation (i.e., something like `se.est <- sd(grades)`), and then we test our hypothesis using the t distribution rather than the normal distribution (i.e. we use `pt()` rather than

[4]Well, sort of. As I understand the history, Gosset only provided a partial solution: the general solution to the problem was provided by Sir Ronald Fisher.

`pnorm()`. And so instead of going through the calculations in tedious detail for a second time, I'll jump straight to showing you how *t*-tests are actually done in practice.

The situation with *t*-tests is very similar to the one we encountered with chi-squared tests in Chapter 12. R comes with one function called `t.test()` that is very flexible (it can run lots of different kinds of *t*-tests) and is somewhat terse (the output is quite compressed). Later on in the chapter I'll show you how to use the `t.test()` function (Section 13.7), but to start out with I'm going to rely on some simpler functions in the `lsr` package. Just like last time, what I've done is written a few simpler functions, each of which does only one thing. So, if you want to run a one-sample *t*-test, use the `oneSampleTTest()` function! It's pretty straightforward to use: all you need to do is specify `x`, the variable containing the data, and `mu`, the true population mean according to the null hypothesis. Assuming that you've already loaded the `lsr` package, all you need to type is this:

```
> oneSampleTTest( x=grades, mu=67.5 )
```

Easy enough. Now lets go through the output. Just like we saw in the last chapter, I've written the functions so that the output is pretty verbose. It tries to describe in a lot of detail what its actually done:

```
   One sample t-test

Data variable:   grades

Descriptive statistics:
            grades
   mean     72.300
   std dev.  9.521

Hypotheses:
   null:        population mean equals 67.5
   alternative: population mean not equal to 67.5

Test results:
   t-statistic:  2.255
   degrees of freedom:  19
   p-value:  0.036

Other information:
   two-sided 95% confidence interval:  [67.844, 76.756]
   estimated effect size (Cohen's d):  0.504
```

Reading this output from top to bottom, you can see it's trying to lead you through the data analysis process. The first two lines tell you what kind of test was run and what data were used. It then gives you some basic information about the sample: specifically, the sample mean and standard deviation of the data. It then moves towards the inferential statistics part. It starts by telling you what the null and alternative hypotheses were, and then it reports the results of the test: the *t*-statistic, the degrees of freedom, and the *p*-value. Finally, it reports two other things you might care about: the confidence interval for the mean, and a measure of effect size (we'll talk more about effect sizes later).

So that seems straightforward enough. Now what do we *do* with this output? Well, since we're pretending that we actually care about my toy example, we're overjoyed to discover that the result is statistically significant (i.e. p value below .05). We could report the result by saying something like this:

> With a mean grade of 72.3, the psychology students scored slightly higher than the average grade of 67.5 ($t(19) = 2.25$, $p < .05$); the 95% confidence interval is [67.8, 76.8].

where $t(19)$ is shorthand notation for a *t*-statistic that has 19 degrees of freedom. That said, it's often the case that people don't report the confidence interval, or do so using a much more compressed form than I've done here. For instance, it's not uncommon to see the confidence interval included as part of the stat block, like this:

$$t(19) = 2.25,\ p < .05,\ \mathrm{CI}_{95} = [67.8, 76.8]$$

With that much jargon crammed into half a line, you know it must be really smart.[5]

13.2.3 Assumptions of the one sample t-test

Okay, so what assumptions does the one-sample t-test make? Well, since the t-test is basically a z-test with the assumption of known standard deviation removed, you shouldn't be surprised to see that it makes the same assumptions as the z-test, minus the one about the known standard deviation. That is

- *Normality*. We're still assuming that the the population distribution is normal[6], and as noted earlier, there are standard tools that you can use to check to see if this assumption is met (Section 13.9), and other tests you can do in it's place if this assumption is violated (Section 13.10).
- *Independence*. Once again, we have to assume that the observations in our sample are generated independently of one another. See the earlier discussion about the z-test for specifics (Section 13.1.4).

Overall, these two assumptions aren't terribly unreasonable, and as a consequence the one-sample t-test is pretty widely used in practice as a way of comparing a sample mean against a hypothesised population mean.

13.3 The independent samples t-test (Student test)

Although the one sample t-test has its uses, it's not the most typical example of a t-test[7]. A much more common situation arises when you've got two different groups of observations. In psychology, this tends to correspond to two different groups of participants, where each group corresponds to a different condition in your study. For each person in the study, you measure some outcome variable of interest, and the research question that you're asking is whether or not the two groups have the same population mean. This is the situation that the independent samples t-test is designed for.

[5] More seriously, I tend to think the reverse is true: I get very suspicious of technical reports that fill their results sections with nothing except the numbers. It might just be that I'm an arrogant jerk, but I often feel like an author that makes no attempt to explain and interpret their analysis to the reader either doesn't understand it themselves, or is being a bit lazy. Your readers are smart, but not infinitely patient. Don't annoy them if you can help it.

[6] A technical comment... in the same way that we can weaken the assumptions of the z-test so that we're only talking about the sampling distribution, we *can* weaken the t test assumptions so that we don't have to assume normality of the population. However, for the t-test, it's trickier to do this. As before, we can replace the assumption of population normality with an assumption that the sampling distribution of $\bar{X}$ is normal. However, remember that we're also relying on a sample estimate of the standard deviation; and so we also require the sampling distribution of $\hat{\sigma}$ to be chi-square. That makes things nastier, and this version is rarely used in practice: fortunately, if the population is normal, then both of these two assumptions are met.

[7] Although it is the simplest, which is why I started with it.

13.3.1 The data

Suppose we have 33 students taking Dr Harpo's statistics lectures, and Dr Harpo doesn't grade to a curve. Actually, Dr Harpo's grading is a bit of a mystery, so we don't really know anything about what the average grade is for the class as a whole. There are two tutors for the class, Anastasia and Bernadette. There are $N_1 = 15$ students in Anastasia's tutorials, and $N_2 = 18$ in Bernadette's tutorials. The research question I'm interested in is whether Anastasia or Bernadette is a better tutor, or if it doesn't make much of a difference. Dr Harpo emails me the course grades, in the `harpo.Rdata` file. As usual, I'll load the file and have a look at what variables it contains:

```
> load( "harpo.Rdata" )
> who(TRUE)
   -- Name --    -- Class --    -- Size --
   harpo         data.frame     33 x 2
    $grade       numeric        33
    $tutor       factor         33
```

As we can see, there's a single data frame with two variables, `grade` and `tutor`. The `grade` variable is a numeric vector, containing the grades for all $N = 33$ students taking Dr Harpo's class; the `tutor` variable is a factor that indicates who each student's tutor was. The first six observations in this data set are shown below:

```
> head( harpo )
  grade      tutor
1    65  Anastasia
2    72 Bernadette
3    66 Bernadette
4    74  Anastasia
5    73  Anastasia
6    71 Bernadette
```

We can calculate means and standard deviations, using the `mean()` and `sd()` functions. Rather than show the R output, here's a nice little summary table:

	mean	std dev	N
Anastasia's students	74.53	9.00	15
Bernadette's students	69.06	5.77	18

To give you a more detailed sense of what's going on here, I've plotted histograms showing the distribution of grades for both tutors (Figure 13.6), as well as a simpler plot showing the means and corresponding confidence intervals for both groups of students (Figure 13.7).

13.3.2 Introducing the test

The **independent samples *t*-test** comes in two different forms, Student's and Welch's. The original Student *t*-test – which is the one I'll describe in this section – is the simpler of the two, but relies on much more restrictive assumptions than the Welch *t*-test. Assuming for the moment that you want to run a two-sided test, the goal is to determine whether two "independent samples" of data are drawn from populations with the same mean (the null hypothesis) or different means (the alternative hypothesis). When we say "independent" samples, what we really mean here is that there's no special relationship between observations in the two samples. This probably doesn't make a lot of sense right now, but it

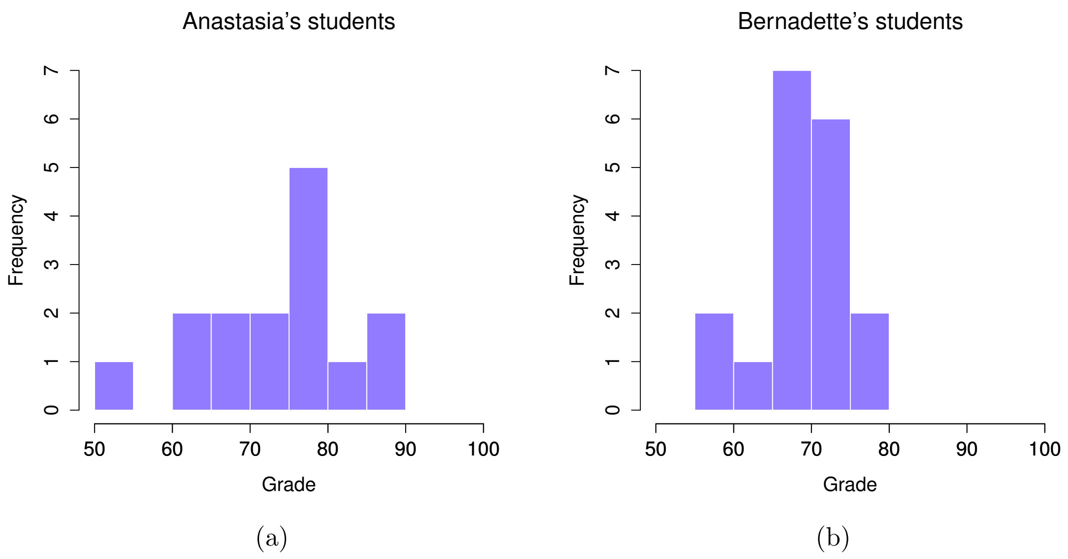


Figure 13.6: Histograms showing the overall distribution of grades for students in Anastasia's class (panel a) and in Bernadette's class (panel b). Inspection of these histograms suggests that the students in Anastasia's class may be getting slightly better grades on average, though they also seem a little more variable.

....

will be clearer when we come to talk about the paired samples *t*-test later on. For now, let's just point out that if we have an experimental design where participants are randomly allocated to one of two groups, and we want to compare the two groups' mean performance on some outcome measure, then an independent samples *t*-test (rather than a paired samples *t*-test) is what we're after.

Okay, so let's let μ_1 denote the true population mean for group 1 (e.g., Anastasia's students), and μ_2 will be the true population mean for group 2 (e.g., Bernadette's students),[8] and as usual we'll let $\bar{X}_1$ and $\bar{X}_2$ denote the observed sample means for both of these groups. Our null hypothesis states that the two population means are identical ($\mu_1 = \mu_2$) and the alternative to this is that they are not ($\mu_1 \neq \mu_2$). Written in mathematical-ese, this is...

$$\begin{array}{ll} H_0: & \mu_1 = \mu_2 \\ H_1: & \mu_1 \neq \mu_2 \end{array}$$

To construct a hypothesis test that handles this scenario, we start by noting that if the null hypothesis is true, then the difference between the population means is *exactly* zero, $\mu_1 - \mu_2 = 0$ As a consequence, a diagnostic test statistic will be based on the difference between the two sample means. Because if the null hypothesis is true, then we'd expect $\bar{X}_1 - \bar{X}_2$ to be *pretty close* to zero. However, just like we saw with our one-sample tests (i.e., the one-sample *z*-test and the one-sample *t*-test) we have to be precise

[8]A funny question almost always pops up at this point: what the heck *is* the population being referred to in this case? Is it the set of students actually taking Dr Harpo's class (all 33 of them)? The set of people who might take the class (an unknown number) of them? Or something else? Does it matter which of these we pick? It's traditional in an introductory behavioural stats class to mumble a lot at this point, but since I get asked this question every year by my students, I'll give a brief answer. Technically yes, it does matter: if you change your definition of what the "real world" population actually is, then the sampling distribution of your observed mean $\bar{X}$ changes too. The *t*-test relies on an assumption that the observations are sampled at random from an infinitely large population; and to the extent that real life isn't like that, then the *t*-test can be wrong. In practice, however, this isn't usually a big deal: even though the assumption is almost always wrong, it doesn't lead to a lot of pathological behaviour from the test, so we tend to just ignore it.

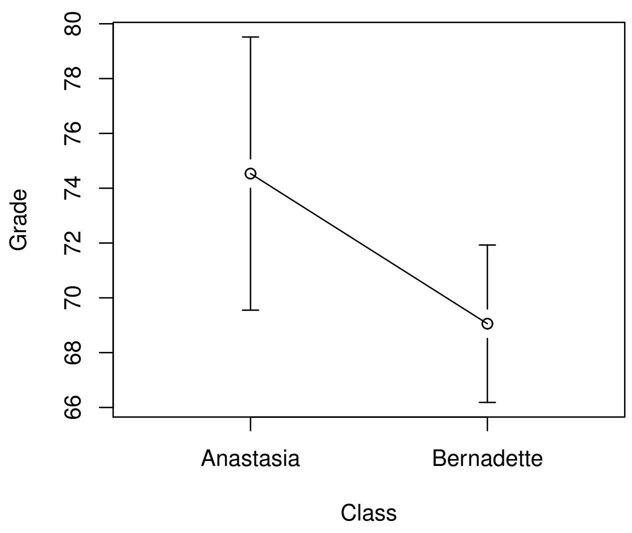


Figure 13.7: Plots showing the mean grade for the students in Anastasia's and Bernadette's tutorials. Error bars depict 95% confidence intervals around the mean. On the basis of visual inspection, it does look like there's a real difference between the groups, though it's hard to say for sure.

....

about exactly *how close* to zero this difference should be. And the solution to the problem is more or less the same one: we calculate a standard error estimate (SE), just like last time, and then divide the difference between means by this estimate. So our **t-statistic** will be of the form

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\mathrm{SE}}$$

We just need to figure out what this standard error estimate actually is. This is a bit trickier than was the case for either of the two tests we've looked at so far, so we need to go through it a lot more carefully to understand how it works.

13.3.3 A "pooled estimate" of the standard deviation

In the original "Student t-test", we make the assumption that the two groups have the same population standard deviation: that is, regardless of whether the population means are the same, we assume that the population standard deviations are identical, $\sigma_1 = \sigma_2$. Since we're assuming that the two standard deviations are the same, we drop the subscripts and refer to both of them as σ. How should we estimate this? How should we construct a single estimate of a standard deviation when we have two samples? The answer is, basically, we average them. Well, sort of. Actually, what we do is take a *weighed* average of the *variance* estimates, which we use as our **pooled estimate of the variance**. The weight assigned to each sample is equal to the number of observations in that sample, minus 1. Mathematically, we can write this as

$$\begin{aligned} w_1 &= N_1 - 1 \\ w_2 &= N_2 - 1 \end{aligned}$$

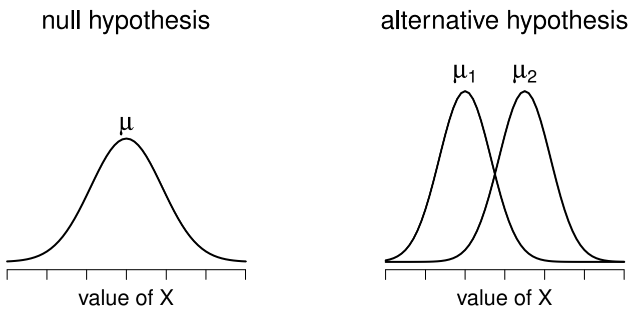


Figure 13.8: Graphical illustration of the null and alternative hypotheses assumed by the Student t-test. The null hypothesis assumes that both groups have the same mean μ, whereas the alternative assumes that they have different means μ_1 and μ_2. Notice that it is assumed that the population distributions are normal, and that, although the alternative hypothesis allows the group to have different means, it assumes they have the same standard deviation.

....

Now that we've assigned weights to each sample, we calculate the pooled estimate of the variance by taking the weighted average of the two variance estimates, $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$

$$\hat{\sigma}_p^2 = \frac{w_1 \hat{\sigma}_1^2 + w_2 \hat{\sigma}_2^2}{w_1 + w_2}$$

Finally, we convert the pooled variance estimate to a pooled standard deviation estimate, by taking the square root. This gives us the following formula for $\hat{\sigma}_p$,

$$\hat{\sigma}_p = \sqrt{\frac{w_1 \hat{\sigma}_1^2 + w_2 \hat{\sigma}_2^2}{w_1 + w_2}}$$

And if you mentally substitute $w_1 = N_1 - 1$ and $w_2 = N_2 - 1$ into this equation you get a very ugly looking formula; a very ugly formula that actually seems to be the "standard" way of describing the pooled standard deviation estimate. It's not my favourite way of thinking about pooled standard deviations, however.[9]

13.3.4 The same pooled estimate, described differently

I prefer to think about it like this. Our data set actually corresponds to a set of N observations, which are sorted into two groups. So let's use the notation X_{ik} to refer to the grade received by the i-th student in the k-th tutorial group: that is, X_{11} is the grade received by the first student in Anastasia's class, X_{21} is her second student, and so on. And we have two separate group means $\bar{X}_1$ and $\bar{X}_2$, which we could "generically" refer to using the notation $\bar{X}_k$, i.e., the mean grade for the k-th tutorial group. So far, so good. Now, since every single student falls into one of the two tutorials, and so we can describe their deviation from the group mean as the difference

$$X_{ik} - \bar{X}_k$$

[9]Yes, I have a "favourite" way of thinking about pooled standard deviation estimates. So what?

So why not just use these deviations (i.e., the extent to which each student's grade differs from the mean grade in their tutorial?) Remember, a variance is just the average of a bunch of squared deviations, so let's do that. Mathematically, we could write it like this:

$$\frac{\sum_{ik} \left(X_{ik} - \bar{X}_k\right)^2}{N}$$

where the notation "$\sum_{ik}$" is a lazy way of saying "calculate a sum by looking at all students in all tutorials", since each "ik" corresponds to one student.[10] But, as we saw in Chapter 10, calculating the variance by dividing by N produces a biased estimate of the population variance. And previously, we needed to divide by $N-1$ to fix this. However, as I mentioned at the time, the reason why this bias exists is because the variance estimate relies on the sample mean; and to the extent that the sample mean isn't equal to the population mean, it can systematically bias our estimate of the variance. But this time we're relying on *two* sample means! Does this mean that we've got more bias? Yes, yes it does. And does this mean we now need to divide by $N-2$ instead of $N-1$, in order to calculate our pooled variance estimate? Why, yes...

$$\hat{\sigma}_p^2 = \frac{\sum_{ik} \left(X_{ik} - \bar{X}_k\right)^2}{N-2}$$

Oh, and if you take the square root of this then you get $\hat{\sigma}_p$, the pooled standard deviation estimate. In other words, the pooled standard deviation calculation is nothing special: it's not terribly different to the regular standard deviation calculation.

13.3.5 Completing the test

Regardless of which way you want to think about it, we now have our pooled estimate of the standard deviation. From now on, I'll drop the silly p subscript, and just refer to this estimate as $\hat{\sigma}$. Great. Let's now go back to thinking about the bloody hypothesis test, shall we? Our whole reason for calculating this pooled estimate was that we knew it would be helpful when calculating our *standard error* estimate. But, standard error of *what*? In the one-sample t-test, it was the standard error of the sample mean, $\mathrm{SE}(\bar{X})$, and since $\mathrm{SE}(\bar{X}) = \sigma/\sqrt{N}$ that's what the denominator of our t-statistic looked like. This time around, however, we have *two* sample means. And what we're interested in, specifically, is the the difference between the two $\bar{X}_1 - \bar{X}_2$. As a consequence, the standard error that we need to divide by is in fact the **standard error of the difference** between means. As long as the two variables really do have the same standard deviation, then our estimate for the standard error is

$$\mathrm{SE}(\bar{X}_1 - \bar{X}_2) = \hat{\sigma}\sqrt{\frac{1}{N_1} + \frac{1}{N_2}}$$

and our t-statistic is therefore

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\mathrm{SE}(\bar{X}_1 - \bar{X}_2)}$$

Just as we saw with our one-sample test, the sampling distribution of this t-statistic is a t-distribution (shocking, isn't it?) as long as the null hypothesis is true, and all of the assumptions of the test are met. The degrees of freedom, however, is slightly different. As usual, we can think of the degrees of freedom to be equal to the number of data points minus the number of constraints. In this case, we have N observations (N_1 in sample 1, and N_2 in sample 2), and 2 constraints (the sample means). So the total

[10] A more correct notation will be introduced in Chapter 14.

degrees of freedom for this test are $N - 2$.

13.3.6 Doing the test in R

Not surprisingly, you can run an independent samples t-test using the `t.test()` function (Section 13.7), but once again I'm going to start with a somewhat simpler function in the `lsr` package. That function is unimaginatively called `independentSamplesTTest()`. First, recall that our data look like this:

```
> head( harpo )
  grade      tutor
1    65  Anastasia
2    72 Bernadette
3    66 Bernadette
4    74  Anastasia
5    73  Anastasia
6    71 Bernadette
```

The outcome variable for our test is the student `grade`, and the groups are defined in terms of the `tutor` for each class. So you probably won't be too surprised to see that we're going to describe the test that we want in terms of an R formula that reads like this `grade ~ tutor`. The specific command that we need is:

```
> independentSamplesTTest(
      formula = grade ~ tutor,  # formula specifying outcome and group variables
      data = harpo,             # data frame that contains the variables
      var.equal = TRUE          # assume that the two groups have the same variance
  )
```

The first two arguments should be familiar to you. The first one is the formula that tells R what variables to use and the second one tells R the name of the data frame that stores those variables. The third argument is not so obvious. By saying `var.equal = TRUE`, what we're really doing is telling R to use the *Student* independent samples t-test. More on this later. For now, lets ignore that bit and look at the output:

```
   Student's independent samples t-test

Outcome variable:   grade
Grouping variable:  tutor

Descriptive statistics:
            Anastasia Bernadette
   mean        74.533     69.056
   std dev.     8.999      5.775

Hypotheses:
   null:        population means equal for both groups
   alternative: different population means in each group

Test results:
   t-statistic:  2.115
   degrees of freedom:  31
   p-value:  0.043

Other information:
   two-sided 95% confidence interval:  [0.197, 10.759]
   estimated effect size (Cohen's d):  0.74
```

The output has a very familiar form. First, it tells you what test was run, and it tells you the names of the variables that you used. The second part of the output reports the sample means and standard deviations for both groups (i.e., both tutorial groups). The third section of the output states the null hypothesis and the alternative hypothesis in a fairly explicit form. It then reports the test results: just like last time, the test results consist of a t-statistic, the degrees of freedom, and the p-value. The final section reports two things: it gives you a confidence interval, and an effect size. I'll talk about effect sizes later. The confidence interval, however, I should talk about now.

It's pretty important to be clear on what this confidence interval actually refers to: it is a confidence interval for the *difference* between the group means. In our example, Anastasia's students had an average grade of 74.5, and Bernadette's students had an average grade of 69.1, so the difference between the two sample means is 5.4. But of course the difference between population means might be bigger or smaller than this. The confidence interval reported by the `independentSamplesTTest()` function tells you that there's a 95% chance that the true difference between means lies between 0.2 and 10.8.

In any case, the difference between the two groups is significant (just barely), so we might write up the result using text like this:

> The mean grade in Anastasia's class was 74.5% (std dev = 9.0), whereas the mean in Bernadette's class was 69.1% (std dev = 5.8). A Student's independent samples t-test showed that this 5.4% difference was significant ($t(31) = 2.1$, $p < .05$, $CI_{95} = [0.2, 10.8]$, $d = .74$), suggesting that a genuine difference in learning outcomes has occurred.

Notice that I've included the confidence interval and the effect size in the stat block. People don't always do this. At a bare minimum, you'd expect to see the t-statistic, the degrees of freedom and the p value. So you should include something like this at a minimum: $t(31) = 2.1$, $p < .05$. If statisticians had their way, everyone would also report the confidence interval and probably the effect size measure too, because they are useful things to know. But real life doesn't always work the way statisticians want it to: you should make a judgment based on whether you think it will help your readers, and (if you're writing a scientific paper) the editorial standard for the journal in question. Some journals expect you to report effect sizes, others don't. Within some scientific communities it is standard practice to report confidence intervals, in other it is not. You'll need to figure out what your audience expects. But, just for the sake of clarity, if you're taking my class: my default position is that it's usually worth includng the effect size, but don't worry about the confidence interval unless the assignment asks you to or implies that you should.

13.3.7 Positive and negative t values

Before moving on to talk about the assumptions of the t-test, there's one additional point I want to make about the use of t-tests in practice. The first one relates to the sign of the t-statistic (that is, whether it is a positive number or a negative one). One very common worry that students have when they start running their first t-test is that they often end up with negative values for the t-statistic, and don't know how to interpret it. In fact, it's not at all uncommon for two people working independently to end up with R outputs that are almost identical, except that one person has a negative t values and the other one has a positive t value. Assuming that you're running a two-sided test, then the p-values will be identical. On closer inspection, the students will notice that the confidence intervals also have the opposite signs. This is perfectly okay: whenever this happens, what you'll find is that the two versions of the R output arise from slightly different ways of running the t-test. What's happening here is very simple. The t-statistic that R is calculating here is always of the form

$$t = \frac{(\text{mean 1}) - (\text{mean 2})}{(\text{SE})}$$

If "mean 1" is larger than "mean 2" the t statistic will be positive, whereas if "mean 2" is larger then the t statistic will be negative. Similarly, the confidence interval that R reports is the confidence interval for the difference "(mean 1) minus (mean 2)", which will be the reverse of what you'd get if you were calculating the confidence interval for the difference "(mean 2) minus (mean 1)".

Okay, that's pretty straightforward when you think about it, but now consider our t-test comparing Anastasia's class to Bernadette's class. Which one should we call "mean 1" and which one should we call "mean 2". It's arbitrary. However, you really do need to designate one of them as "mean 1" and the other one as "mean 2". Not surprisingly, the way that R handles this is also pretty arbitrary. In earlier versions of the book I used to try to explain it, but after a while I gave up, because it's not really all that important, and to be honest I can never remember myself. Whenever I get a significant t-test result, and I want to figure out which mean is the larger one, I don't try to figure it out by looking at the t-statistic. Why would I bother doing that? It's foolish. It's easier just look at the actual group means, since the R output actually shows them!

Here's the important thing. Because it really doesn't matter what R printed out, I usually try to *report* the t-statistic in such a way that the numbers match up with the text. Here's what I mean... suppose that what I want to write in my report is "Anastasia's class had higher grades than Bernadette's class". The phrasing here implies that Anastasia's group comes first, so it makes sense to report the t-statistic as if Anastasia's class corresponded to group 1. If so, I would write

Anastasia's class had $\underline{\text{higher}}$ grades than Bernadette's class ($t(31) = 2.1, p = .04$).

(I wouldn't actually underline the word "higher" in real life, I'm just doing it to emphasise the point that "higher" corresponds to positive t values). On the other hand, suppose the phrasing I wanted to use has Bernadette's class listed first. If so, it makes more sense to treat her class as group 1, and if so, the write up looks like this:

Bernadette's class had $\underline{\text{lower}}$ grades than Anastasia's class ($t(31) = -2.1, p = .04$).

Because I'm talking about one group having "lower" scores this time around, it is more sensible to use the negative form of the t-statistic. It just makes it read more cleanly.

One last thing: please note that you *can't* do this for other types of test statistics. It works for t-tests, but it wouldn't be meaningful for chi-square testsm F-tests or indeed for most of the tests I talk about in this book. So don't overgeneralise this advice! I'm really just talking about t-tests here and nothing else!

13.3.8 Assumptions of the test

As always, our hypothesis test relies on some assumptions. So what are they? For the Student t-test there are three assumptions, some of which we saw previously in the context of the one sample t-test (see Section 13.2.3):

- *Normality.* Like the one-sample t-test, it is assumed that the data are normally distributed. Specifically, we assume that both groups are normally distributed. In Section 13.9 we'll discuss how to test for normality, and in Section 13.10 we'll discuss possible solutions.
- *Independence.* Once again, it is assumed that the observations are independently sampled. In the context of the Student test this has two aspects to it. Firstly, we assume that the observations within each sample are independent of one another (exactly the same as for the one-sample test). However, we also assume that there are no cross-sample dependencies. If, for instance, it turns out that you included some participants in both experimental conditions of your study (e.g., by

accidentally allowing the same person to sign up to different conditions), then there are some cross sample dependencies that you'd need to take into account.

- *Homogeneity of variance* (also called "homoscedasticity"). The third assumption is that the population standard deviation is the same in both groups. You can test this assumption using the Levene test, which I'll talk about later on in the book (Section 14.7). However, there's a very simple remedy for this assumption, which I'll talk about in the next section.

13.4 The independent samples t-test (Welch test)

The biggest problem with using the Student test in practice is the third assumption listed in the previous section: it assumes that both groups have the same standard deviation. This is rarely true in real life: if two samples don't have the same means, why should we expect them to have the same standard deviation? There's really no reason to expect this assumption to be true. We'll talk a little bit about how you can check this assumption later on because it does crop up in a few different places, not just the t-test. But right now I'll talk about a different form of the t-test (Welch, 1947) that does not rely on this assumption. A graphical illustration of what the **Welch t test** assumes about the data is shown in Figure 13.9, to provide a contrast with the Student test version in Figure 13.8. I'll admit it's a bit odd to talk about the cure before talking about the diagnosis, but as it happens the Welch test is the default t-test in R, so this is probably the best place to discuss it.

The Welch test is very similar to the Student test. For example, the t-statistic that we use in the Welch test is calculated in much the same way as it is for the Student test. That is, we take the difference between the sample means, and then divide it by some estimate of the standard error of that difference:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\text{SE}(\bar{X}_1 - \bar{X}_2)}$$

The main difference is that the standard error calculations are different. If the two populations have different standard deviations, then it's a complete nonsense to try to calculate a pooled standard deviation estimate, because you're averaging apples and oranges.[11] But you can still estimate the standard error of the difference between sample means; it just ends up looking different:

$$\text{SE}(\bar{X}_1 - \bar{X}_2) = \sqrt{\frac{\hat{\sigma}_1^2}{N_1} + \frac{\hat{\sigma}_2^2}{N_2}}$$

The reason why it's calculated this way is beyond the scope of this book. What matters for our purposes is that the t-statistic that comes out of the Welch test is actually somewhat different to the one that comes from the Student test.

The second difference between Welch and Student is that the degrees of freedom are calculated in a very different way. In the Welch test, the "degrees of freedom " doesn't have to be a whole number any more, and it doesn't correspond all that closely to the "number of data points minus the number of constraints" heuristic that I've been using up to this point. The degrees of freedom are, in fact...

$$\text{df} = \frac{(\hat{\sigma}_1^2/N_1 + \hat{\sigma}_2^2/N_2)^2}{(\hat{\sigma}_1^2/N_1)^2/(N_1 - 1) + (\hat{\sigma}_2^2/N_2)^2/(N_2 - 1)}$$

[11] Well, I guess you can average apples and oranges, and what you end up with is a delicious fruit smoothie. But no one really thinks that a fruit smoothie is a very good way to describe the original fruits, do they?

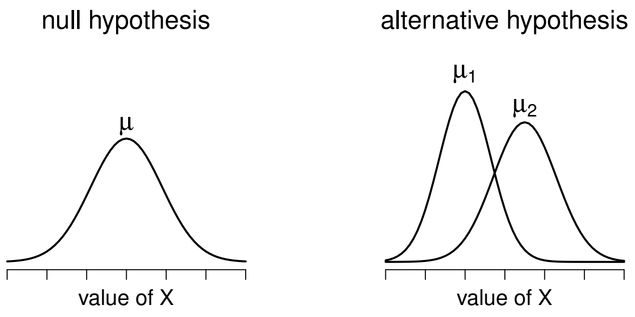


Figure 13.9: Graphical illustration of the null and alternative hypotheses assumed by the Welch *t*-test. Like the Student test (Figure 13.8) we assume that both samples are drawn from a normal population; but the alternative hypothesis no longer requires the two populations to have equal variance.

....

... which is all pretty straightforward and obvious, right? Well, perhaps not. It doesn't really matter for out purposes. What matters is that you'll see that the "df" value that pops out of a Welch test tends to be a little bit smaller than the one used for the Student test, and it doesn't have to be a whole number.

13.4.1 Doing the test in R

To run a Welch test in R is pretty easy. All you have to do is not bother telling R to assume equal variances. That is, you take the command we used to run a Student's *t*-test and drop the `var.equal = TRUE` bit. So the command for a Welch test becomes:

```
> independentSamplesTTest(
      formula = grade ~ tutor,  # formula specifying outcome and group variables
      data = harpo              # data frame that contains the variables
  )
```

Not too difficult, right? Not surprisingly, the output has exactly the same format as it did last time too:

```
   Welch's independent samples t-test

Outcome variable:   grade
Grouping variable:  tutor

Descriptive statistics:
            Anastasia Bernadette
   mean        74.533     69.056
   std dev.     8.999      5.775

Hypotheses:
   null:        population means equal for both groups
   alternative: different population means in each group

Test results:
   t-statistic:  2.034
```

```
      degrees of freedom:  23.025
      p-value:  0.054

   Other information:
      two-sided 95% confidence interval:  [-0.092, 11.048]
      estimated effect size (Cohen's d):  0.724
```

The very first line is different, because it's telling you that its run a Welch test rather than a Student test, and of course all the numbers are a bit different. But I hope that the interpretation of this output should be fairly obvious. You read the output in the same way that you would for the Student test. You've got your descriptive statistics, the hypotheses, the test results and some other information. So that's all pretty easy.

Except, except... our result isn't significant anymore. When we ran the Student test, we did get a significant effect; but the Welch test on the same data set is not ($t(23.03) = 2.03$, $p = .054$). What does this mean? Should we panic? Is the sky burning? Probably not. The fact that one test is significant and the other isn't doesn't itself mean very much, especially since I kind of rigged the data so that this would happen. As a general rule, it's not a good idea to go out of your way to try to interpret or explain the difference between a p-value of .049 and a p-value of .051. If this sort of thing happens in real life, the *difference* in these p-values is almost certainly due to chance. What does matter is that you take a little bit of care in thinking about what test you use. The Student test and the Welch test have different strengths and weaknesses. If the two populations really do have equal variances, then the Student test is slightly more powerful (lower Type II error rate) than the Welch test. However, if they *don't* have the same variances, then the assumptions of the Student test are violated and you may not be able to trust it: you might end up with a higher Type I error rate. So it's a trade off. However, in real life, I tend to prefer the Welch test; because almost no-one *actually* believes that the population variances are identical.

13.4.2 Assumptions of the test

The assumptions of the Welch test are very similar to those made by the Student t-test (see Section 13.3.8), except that the Welch test does not assume homogeneity of variance. This leaves only the assumption of normality, and the assumption of independence. The specifics of these assumptions are the same for the Welch test as for the Student test.

13.5 The paired-samples t-test

Regardless of whether we're talking about the Student test or the Welch test, an independent samples t-test is intended to be used in a situation where you have two samples that are, well, independent of one another. This situation arises naturally when participants are assigned randomly to one of two experimental conditions, but it provides a very poor approximation to other sorts of research designs. In particular, a repeated measures design – in which each participant is measured (with respect to the same outcome variable) in both experimental conditions – is not suited for analysis using independent samples t-tests. For example, we might be interested in whether listening to music reduces people's working memory capacity. To that end, we could measure each person's working memory capacity in two conditions: with music, and without music. In an experimental design such as this one,[12] each participant

[12]This design is very similar to the one in Section 12.8 that motivated the McNemar test. This should be no surprise. Both are standard repeated measures designs involving two measurements. The only difference is that this time our outcome variable is interval scale (working memory capacity) rather than a binary, nominal scale variable (a yes-or-no question).

appears in *both* groups. This requires us to approach the problem in a different way; by using the **paired samples t-test**.

13.5.1 The data

The data set that we'll use this time comes from Dr Chico's class.[13] In her class, students take two major tests, one early in the semester and one later in the semester. To hear her tell it, she runs a very hard class, one that most students find very challenging; but she argues that by setting hard assessments, students are encouraged to work harder. Her theory is that the first test is a bit of a "wake up call" for students: when they realise how hard her class really is, they'll work harder for the second test and get a better mark. Is she right? To test this, let's have a look at the `chico.Rdata` file:

```
> load( "chico.Rdata" )
> who(TRUE)
   -- Name --        -- Class --    -- Size --
   chico             data.frame     20 x 3
    $id              factor         20
    $grade_test1     numeric        20
    $grade_test2     numeric        20
```

The data frame `chico` contains three variables: an `id` variable that identifies each student in the class, the `grade_test1` variable that records the student grade for the first test, and the `grade_test2` variable that has the grades for the second test. Here's the first six students:

```
> head( chico )
        id grade_test1 grade_test2
1 student1        42.9        44.6
2 student2        51.8        54.0
3 student3        71.7        72.3
4 student4        51.6        53.4
5 student5        63.5        63.8
6 student6        58.0        59.3
```

At a glance, it does seem like the class is a hard one (most grades are between 50% and 60%), but it does look like there's an improvement from the first test to the second one. If we take a quick look at the descriptive statistics

```
> library( psych )
> describe( chico )
            var  n  mean   sd median trimmed  mad  min  max range  skew kurtosis   se
id*           1 20 10.50 5.92   10.5   10.50 7.41  1.0 20.0  19.0  0.00    -1.20 1.32
grade_test1   2 20 56.98 6.62   57.7   56.92 7.71 42.9 71.7  28.8  0.05     0.28 1.48
grade_test2   3 20 58.38 6.41   59.7   58.35 6.45 44.6 72.3  27.7 -0.05     0.23 1.43
```

we see that this impression seems to be supported. Across all 20 students[14] the mean grade for the first test is 57%, but this rises to 58% for the second test. Although, given that the standard deviations are 6.6% and 6.4% respectively, it's starting to feel like maybe the improvement is just illusory; maybe just random variation. This impression is reinforced when you see the means and confidence intervals plotted in Figure 13.10a. If we were to rely on this plot alone, we'd come to the same conclusion that

[13] At this point we have Drs Harpo, Chico and Zeppo. No prizes for guessing who Dr Groucho is.

[14] This is obviously a class being taught at a very small or very expensive university, or else is a postgraduate class. *I've* never taught an intro stats class with less than 350 students.

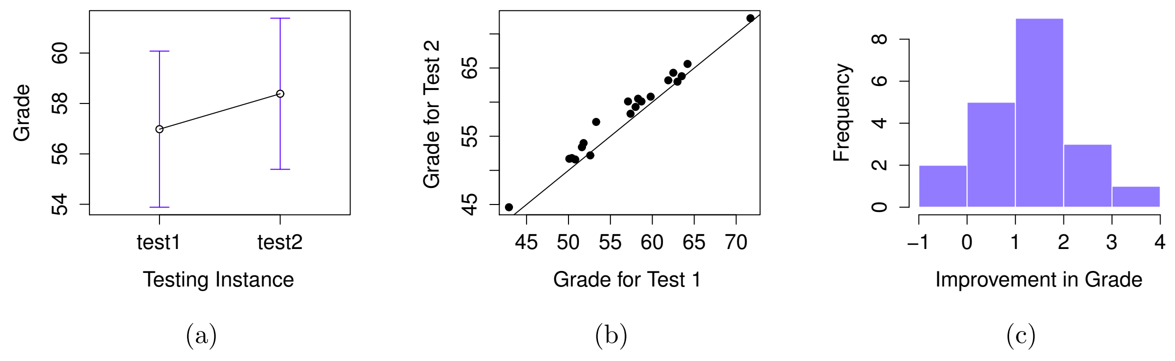


Figure 13.10: Mean grade for test 1 and test 2, with associated 95% confidence intervals (panel a). Scatterplot showing the individual grades for test 1 and test 2 (panel b). Histogram showing the improvement made by each student in Dr Chico's class (panel c). In panel c, notice that almost the entire distribution is above zero: the vast majority of students did improve their performance from the first test to the second one

....

we got from looking at the descriptive statistics that the `describe()` function produced. Looking at how wide those confidence intervals are, we'd be tempted to think that the apparent improvement in student performance is pure chance.

Nevertheless, this impression is wrong. To see why, take a look at the scatterplot of the grades for test 1 against the grades for test 2. shown in Figure 13.10b. In this plot, each dot corresponds to the two grades for a given student: if their grade for test 1 (x co-ordinate) equals their grade for test 2 (y co-ordinate), then the dot falls on the line. Points falling above the line are the students that performed better on the second test. Critically, almost all of the data points fall above the diagonal line: almost all of the students *do* seem to have improved their grade, if only by a small amount. This suggests that we should be looking at the *improvement* made by each student from one test to the next, and treating that as our raw data. To do this, we'll need to create a new variable for the `improvement` that each student makes, and add it to the `chico` data frame. The easiest way to do this is as follows:

```
> chico$improvement <- chico$grade_test2 - chico$grade_test1
```

Notice that I assigned the output to a variable called `chico$improvement`. That has the effect of creating a new variable called `improvement` inside the `chico` data frame. So now when I look at the `chico` data frame, I get an output that looks like this:

```
> head( chico )
        id grade_test1 grade_test2 improvement
1 student1        42.9        44.6         1.7
2 student2        51.8        54.0         2.2
3 student3        71.7        72.3         0.6
4 student4        51.6        53.4         1.8
5 student5        63.5        63.8         0.3
6 student6        58.0        59.3         1.3
```

Now that we've created and stored this `improvement` variable, we can draw a histogram showing the distribution of these improvement scores (using the `hist()` function), shown in Figure 13.10c. When we look at histogram, it's very clear that there *is* a real improvement here. The vast majority of the students scored higher on the test 2 than on test 1, reflected in the fact that almost the entire histogram is above zero. In fact, if we use `ciMean()` to compute a confidence interval for the population mean of this new variable,

```
> ciMean( x = chico$improvement )
     2.5%     97.5%
0.9508686 1.8591314
```

we see that it is 95% certain that the true (population-wide) average improvement would lie between 0.95% and 1.86%. So you can see, qualitatively, what's going on: there is a real "within student" improvement (everyone improves by about 1%), but it is very small when set against the quite large "between student" differences (student grades vary by about 20% or so).

13.5.2 What is the paired samples t-test?

In light of the previous exploration, let's think about how to construct an appropriate t test. One possibility would be to try to run an independent samples t-test using `grade_test1` and `grade_test2` as the variables of interest. However, this is clearly the wrong thing to do: the independent samples t-test assumes that there is no particular relationship between the two samples. Yet clearly that's not true in this case, because of the repeated measures structure to the data. To use the language that I introduced in the last section, if we were to try to do an independent samples t-test, we would be conflating the **within subject** differences (which is what we're interested in testing) with the **between subject** variability (which we are not).

The solution to the problem is obvious, I hope, since we already did all the hard work in the previous section. Instead of running an independent samples t-test on `grade_test1` and `grade_test2`, we run a *one-sample* t-test on the within-subject difference variable, `improvement`. To formalise this slightly, if X_{i1} is the score that the i-th participant obtained on the first variable, and X_{i2} is the score that the same person obtained on the second one, then the difference score is:

$$D_i = X_{i1} - X_{i2}$$

Notice that the difference scores is *variable 1 minus variable 2* and not the other way around, so if we want improvement to correspond to a positive valued difference, we actually want "test 2" to be our "variable 1". Equally, we would say that $\mu_D = \mu_1 - \mu_2$ is the population mean for this difference variable. So, to convert this to a hypothesis test, our null hypothesis is that this mean difference is zero; the alternative hypothesis is that it is not:

$$\begin{array}{ll} H_0: & \mu_D = 0 \\ H_1: & \mu_D \neq 0 \end{array}$$

(this is assuming we're talking about a two-sided test here). This is more or less identical to the way we described the hypotheses for the one-sample t-test: the only difference is that the specific value that the null hypothesis predicts is 0. And so our t-statistic is defined in more or less the same way too. If we let $\bar{D}$ denote the mean of the difference scores, then

$$t = \frac{\bar{D}}{\text{SE}(\bar{D})}$$

which is

$$t = \frac{\bar{D}}{\hat{\sigma}_D / \sqrt{N}}$$

where $\hat{\sigma}_D$ is the standard deviation of the difference scores. Since this is just an ordinary, one-sample t-test, with nothing special about it, the degrees of freedom are still $N - 1$. And that's it: the paired samples t-test really isn't a new test at all: it's a one-sample t-test, but applied to the difference between two variables. It's actually very simple; the only reason it merits a discussion as long as the one we've just gone through is that you need to be able to recognise *when* a paired samples test is appropriate, and to understand *why* it's better than an independent samples t test.

13.5.3 Doing the test in R, part 1

How do you do a paired samples t-test in R. One possibility is to follow the process I outlined above: create a "difference" variable and then run a one sample t-test on that. Since we've already created a variable called `chico$improvement`, let's do that:

```
> oneSampleTTest( chico$improvement, mu=0 )

   One sample t-test

Data variable:   chico$improvement

Descriptive statistics:
            improvement
   mean           1.405
   std dev.       0.970

Hypotheses:
   null:        population mean equals 0
   alternative: population mean not equal to 0

Test results:
   t-statistic:  6.475
   degrees of freedom:  19
   p-value:  <.001

Other information:
   two-sided 95% confidence interval:  [0.951, 1.859]
   estimated effect size (Cohen's d):  1.448
```

The output here is (obviously) formatted exactly the same was as it was the last time we used the `oneSampleTTest()` function (Section 13.2), and it confirms our intuition. There's an average improvement of 1.4% from test 1 to test 2, and this is significantly different from 0 ($t(19) = 6.48, p < .001$).

However, suppose you're lazy and you don't want to go to all the effort of creating a new variable. Or perhaps you just want to keep the difference between one-sample and paired-samples tests clear in your head. If so, you can use the `pairedSamplesTTest()` function, also in the `lsr` package. Let's assume that your data organised like they are in the `chico` data frame, where there are two separate variables, one for each measurement. The way to run the test is to input a *one-sided* formula, just like you did when running a test of association using the `associationTest()` function in Chapter 12. For the `chico` data frame, the formula that you need would be `~ grade_time2 + grade_time1`. As usual, you'll also need to input the name of the data frame too. So the command just looks like this:

```
> pairedSamplesTTest(
     formula = ~ grade_test2 + grade_test1, # one-sided formula listing the two variables
     data = chico                           # data frame containing the two variables
  )
```

The output is is shown below. The numbers are identical to those that come from the one sample test, which of course they have to be given that the paired samples t-test is just a one sample test under the hood. However, the output is a bit more detailed:

```
   Paired samples t-test

Variables:  grade_test2 , grade_test1

Descriptive statistics: 
            grade_test2 grade_test1 difference
   mean          58.385      56.980      1.405
   std dev.       6.406       6.616      0.970

Hypotheses: 
   null:        population means equal for both measurements
   alternative: different population means for each measurement

Test results: 
   t-statistic:  6.475 
   degrees of freedom:  19 
   p-value:  <.001 

Other information: 
   two-sided 95% confidence interval:  [0.951, 1.859] 
   estimated effect size (Cohen's d):  1.448
```

This time around the descriptive statistics block shows you the means and standard deviations for the original variables, as well as for the difference variable (notice that it always defines the difference as the first listed variable mines the second listed one). The null hypothesis and the alternative hypothesis are now framed in terms of the original variables rather than the difference score, but you should keep in mind that in a paired samples test it's still the difference score being tested. The statistical information at the bottom about the test result is of course the same as before.

13.5.4 Doing the test in R, part 2

The paired samples t-test is a little different from the other t-tests, because it is used in repeated measures designs. For the `chico` data, every student is "measured" twice, once for the first test, and again for the second test. Back in Section 7.7 I talked about the fact that repeated measures data can be expressed in two standard ways, known as *wide form* and *long form*. The `chico` data frame is in wide form: every row corresponds to a unique *person*. I've shown you the data in that form first because that's the form that you're most used to seeing, and it's also the format that you're most likely to receive data in. However, the majority of tools in R for dealing with repeated measures data expect to receive data in long form. The paired samples t-test is a bit of an exception that way.

As you make the transition from a novice user to an advanced one, you're going to have to get comfortable with long form data, and switching between the two forms. To that end, I want to show you how to apply the `pairedSamplesTTest()` function to long form data. First, let's use the `wideToLong()` function to create a long form version of the `chico` data frame. If you've forgotten how the `wideToLong()` function works, it might be worth your while quickly re-reading Section 7.7. Assuming that you've done so, or that you're already comfortable with data reshaping, I'll use it to create a new data frame called `chico2`:

```
> chico2 <- wideToLong( chico, within="time" )
> head( chico2 )
```

```
        id  time grade
1 student1 test1  42.9
2 student2 test1  51.8
3 student3 test1  71.7
4 student4 test1  51.6
5 student5 test1  63.5
6 student6 test1  58.0
```

As you can see, this has created a new data frame containing three variables: an `id` variable indicating which person provided the data, a `time` variable indicating which test the data refers to (i.e., test 1 or test 2), and a `grade` variable that records what score the person got on that test. Notice that this data frame is in long form: every row corresponds to a unique *measurement*. Because every person provides two observations (test 1 and test 2), there are two rows for every person. To see this a little more clearly, I'll use the `sortFrame()` function to sort the rows of `chico2` by `id` variable (see Section 7.6.3).

```
> chico2 <- sortFrame( chico2, id )
> head( chico2 )
          id  time grade
1   student1 test1  42.9
21  student1 test2  44.6
10 student10 test1  61.9
30 student10 test2  63.2
11 student11 test1  50.4
31 student11 test2  51.8
```

As you can see, there are two rows for "student1": one showing their grade on the first test, the other showing their grade on the second test.[15]

Okay, suppose that we were given the `chico2` data frame to analyse. How would we run our paired samples t-test now? One possibility would be to use the `longToWide()` function (Section 7.7) to force the data back into wide form, and do the same thing that we did previously. But that's sort of defeating the point, and besides, there's an easier way. Let's think about what how the `chico2` data frame is structured: there are three variables here, and they all matter. The outcome measure is stored as the `grade`, and we effectively have two "groups" of measurements (test 1 and test 2) that are defined by the `time` points at which a test is given. Finally, because we want to keep track of which measurements should be paired together, we need to know which student obtained each grade, which is what the `id` variable gives us. So, when your data are presented to you in long form, we would want specify a *two-sided* formula and a data frame, in the same way that we do for an independent samples t-test: the formula specifies the outcome variable and the groups, so in this case it would be `grade ~ time`, and the data frame is `chico2`. However, we also need to tell it the id variable, which in this case is boringly called `id`. So our command is:

```
> pairedSamplesTTest(
     formula = grade ~ time,  # two sided formula: outcome ~ group
     data = chico2,           # data frame
     id = "id"                # name of the id variable
  )
```

Note that the name of the id variable is `"id"` and not `id`. Note that the `id` variable must be a factor. As of the current writing, you do need to include the quote marks, because the `pairedSamplesTTest()` function is expecting a *character string* that specifies the name of a variable. If I ever find the time I'll try to relax this constraint. Anyway, here's the output that you get:

[15] The `sortFrame()` function sorts factor variables like `id` in alphabetical order, which is why it jumps from "student1" to "student10"

```
   Paired samples t-test

Outcome variable:   grade
Grouping variable:  time
ID variable:        id

Descriptive statistics:
             test1  test2
   mean     56.980 58.385
   std dev.  6.616  6.406

Hypotheses:
   null:        population means equal for both measurements
   alternative: different population means for each measurement

Test results:
   t-statistic:  -6.475
   degrees of freedom:  19
   p-value:  <.001

Other information:
   two-sided 95% confidence interval:  [-1.859, -0.951]
   estimated effect size (Cohen's d):  1.448
```

As you can see, it's a bit more detailed than the output from `oneSampleTTest()`. It gives you the descriptive statistics for the original variables, states the null hypothesis in a fashion that is a bit more appropriate for a repeated measures design, and then reports all the nuts and bolts from the hypothesis test itself. Not surprisingly the numbers the same as the ones that we saw last time.

One final comment about the `pairedSamplesTTest()` function. One of the reasons I designed it to be able handle long form and wide form data is that I want you to be get comfortable thinking about repeated measures data in both formats, and also to become familiar with the different ways in which R functions tend to specify models and tests for repeated measures data. With that last point in mind, I want to highlight a slightly different way of thinking about what the paired samples *t*-test is doing. There's a sense in which what you're really trying to do is look at how the outcome variable (`grade`) is related to the grouping variable (`time`), after taking account of the fact that there are individual differences between people (`id`). So there's a sense in which `id` is actually a *second* predictor: you're trying to predict the `grade` on the basis of the `time` and the `id`. With that in mind, the `pairedSamplesTTest()` function lets you specify a formula like this one

```
grade ~ time + (id)
```

This formula tells R everything it needs to know: the variable on the left (`grade`) is the outcome variable, the bracketed term on the right (`id`) is the id variable, and the other term on the right is the grouping variable (`time`). If you specify your formula that way, then you only need to specify the `formula` and the `data` frame, and so you can get away with using a command as simple as this one:

```
> pairedSamplesTTest(
     formula = grade ~ time + (id),
     data = chico2
  )
```

or you can drop the argument names and just do this:

```
> pairedSamplesTTest( grade ~ time + (id), chico2 )
```

These commands will produce the same output as the last one, I personally find this format a lot more elegant. That being said, the main reason for allowing you to write your formulas that way is that they're quite similar to the way that mixed models (fancy pants repeated measures analyses) are specified in the `lme4` package. This book doesn't talk about mixed models (yet!), but if you go on to learn more statistics you'll find them pretty hard to avoid, so I've tried to lay a little bit of the groundwork here.

13.6 One sided tests

When introducing the theory of null hypothesis tests, I mentioned that there are some situations when it's appropriate to specify a *one-sided* test (see Section 11.4.3). So far, all of the t-tests have been two-sided tests. For instance, when we specified a one sample t-test for the grades in Dr Zeppo's class, the null hypothesis was that the true mean was 67.5%. The alternative hypothesis was that the true mean was greater than *or* less than 67.5%. Suppose we were only interested in finding out if the true mean is greater than 67.5%, and have no interest whatsoever in testing to find out if the true mean is lower than 67.5%. If so, our null hypothesis would be that the true mean is 67.5% or less, and the alternative hypothesis would be that the true mean is greater than 67.5%. The `oneSampleTTest()` function lets you do this, by specifying the `one.sided` argument. If you set `one.sided="greater"`, it means that you're testing to see if the true mean is larger than `mu`. If you set `one.sided="less"`, then you're testing to see if the true mean is smaller than `mu`. Here's how it would work for Dr Zeppo's class:

```
> oneSampleTTest( x=grades, mu=67.5, one.sided="greater" )

   One sample t-test

Data variable:   grades

Descriptive statistics:
            grades
   mean     72.300
   std dev.  9.521

Hypotheses:
   null:        population mean less than or equal to 67.5
   alternative: population mean greater than 67.5

Test results:
   t-statistic:  2.255
   degrees of freedom:  19
   p-value:  0.018

Other information:
   one-sided 95% confidence interval:  [68.619, Inf]
   estimated effect size (Cohen's d):  0.504
```

Notice that there are a few changes from the output that we saw last time. Most important is the fact that the null and alternative hypotheses have changed, to reflect the different test. The second thing to note is that, although the t-statistic and degrees of freedom have not changed, the p-value has. This is because the one-sided test has a different rejection region from the two-sided test. If you've forgotten why this is and what it means, you may find it helpful to read back over Chapter 11, and Section 11.4.3 in particular. The third thing to note is that the confidence interval is different too: it now reports a "one-sided" confidence interval rather than a two-sided one. In a two-sided confidence interval, we're trying to find numbers a and b such that we're 95% confident that the true mean lies *between* a and b. In a one-sided confidence interval, we're trying to find a single number a such that we're 95% confident that the true mean is *greater than* a (or less than a if you set `one.sided="less"`).

So that's how to do a one-sided one sample t-test. However, all versions of the t-test can be one-sided. For an independent samples t test, you could have a one-sided test if you're only interestd in testing to see if group A has *higher* scores than group B, but have no interest in finding out if group B has higher scores than group A. Let's suppose that, for Dr Harpo's class, you wanted to see if Anastasia's students had higher grades than Bernadette's. The `independentSamplesTTest()` function lets you do this, again by specifying the `one.sided` argument. However, this time around you need to specify the name of the group that you're expecting to have the higher score. In our case, we'd write `one.sided = "Anastasia"`. So the command would be:

```
> independentSamplesTTest(
    formula = grade ~ tutor,
    data = harpo,
    one.sided = "Anastasia"
  )

   Welch's independent samples t-test

Outcome variable:   grade
Grouping variable:  tutor

Descriptive statistics:
            Anastasia Bernadette
   mean        74.533     69.056
   std dev.     8.999      5.775

Hypotheses:
   null:        population means are equal, or smaller for group 'Anastasia'
   alternative: population mean is larger for group 'Anastasia'

Test results:
   t-statistic:  2.034
   degrees of freedom:  23.025
   p-value:  0.027

Other information:
   one-sided 95% confidence interval:  [0.863, Inf]
   estimated effect size (Cohen's d):  0.724
```

Again, the output changes in a predictable way. The definition of the null and alternative hypotheses has changed, the p-value has changed, and it now reports a one-sided confidence interval rather than a two-sided one.

What about the paired samples t-test? Suppose we wanted to test the hypothesis that grades go *up* from test 1 to test 2 in Dr Zeppo's class, and are not prepared to consider the idea that the grades go down. Again, we can use the `one.sided` argument to specify the one-sided test, and it works the same way it does for the independent samples t-test. You need to specify the name of the group whose scores are expected to be larger under the alternative hypothesis. If your data are in wide form, as they are in the `chico` data frame, you'd use this command:

```
> pairedSamplesTTest(
     formula = ~ grade_test2 + grade_test1,
     data = chico,
     one.sided = "grade_test2"
  )

   Paired samples t-test

Variables:  grade_test2 , grade_test1

Descriptive statistics:
            grade_test2 grade_test1 difference
   mean          58.385      56.980      1.405
   std dev.       6.406       6.616      0.970

Hypotheses:
   null:        population means are equal, or smaller for measurement 'grade_test2'
   alternative: population mean is larger for measurement 'grade_test2'

Test results:
   t-statistic:  6.475
   degrees of freedom:  19
   p-value:  <.001

Other information:
   one-sided 95% confidence interval:  [1.03, Inf]
   estimated effect size (Cohen's d):  1.448
```

Yet again, the output changes in a predictable way. The hypotheses have changed, the p-value has changed, and the confidence interval is now one-sided. If your data are in long form, as they are in the `chico2` data frame, it still works the same way. Either of the following commands would work,

```
> pairedSamplesTTest(
    formula = grade ~ time,
    data = chico2,
    id = "id",
    one.sided = "test2"
  )

> pairedSamplesTTest(
    formula = grade ~ time + (id),
    data = chico2,
    one.sided = "test2"
  )
```

and would produce the same answer as the output shown above.

13.7 Using the t.test() function

In this chapter, we've talked about three different kinds of t-test: the one sample test, the independent samples test (Student's and Welch's), and the paired samples test. In order to run these different tests, I've shown you three different functions: `oneSampleTTest()`, `independentSamplesTTest()` and `pairedSamplesTTest()`. I wrote these as three different functions for two reasons. Firstly, I thought it made sense to have separate functions for each test, in order to help make it clear to beginners that there *are* different tests. Secondly, I wanted to show you some functions that produced "verbose" output, to help you see what hypotheses are being tested and so on.

However, once you've started to become familiar with *t*-tests and with using R, you might find it easier to use the `t.test()` function. It's one function, but it can run all four of the different *t*-tests that we've talked about. Here's how it works. Firstly, suppose you want to run a one sample *t*-test. To run the test on the `grades` data from Dr Zeppo's class (Section 13.2), we'd use a command like this:

```
> t.test( x = grades, mu = 67.5 )
```

The input is the same as for the `oneSampleTTest()`: we specify the sample data using the argument `x`, and the value against which it is to be tested using the argument `mu`. The output is a lot more compressed:

```
        One Sample t-test

data:  grades
t = 2.2547, df = 19, p-value = 0.03615
alternative hypothesis: true mean is not equal to 67.5
95 percent confidence interval:
 67.84422 76.75578
sample estimates:
mean of x
     72.3
```

As you can see, it still has all the information you need. It tells you what type of test it ran and the data it tested it on. It gives you the *t*-statistic, the degrees of freedom and the *p*-value. And so on. There's nothing wrong with this output, but in my experience it can be a little confusing when you're just starting to learn statistics, because it's a little disorganised. Once you know what you're looking at though, it's pretty easy to read off the relevant information.

What about independent samples *t*-tests? As it happens, the `t.test()` function can be used in much the same way as the `independentSamplesTTest()` function, by specifying a formula, a data frame, and using `var.equal` to indicate whether you want a Student test or a Welch test. If you want to run the Welch test from Section 13.4, then you'd use this command:

```
> t.test( formula = grade ~ tutor, data = harpo )
```

The output looks like this:

```
        Welch Two Sample t-test

data:  grade by tutor
t = 2.0342, df = 23.025, p-value = 0.05361
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.09249349 11.04804904
sample estimates:
 mean in group Anastasia mean in group Bernadette
               74.53333                69.05556
```

If you want to do the Student test, it's exactly the same except that you need to add an additional argument indicating that `var.equal = TRUE`. This is no different to how it worked in the `independentSamplesTTest()` function.

Finally, we come to the paired samples *t*-test. Somewhat surprisingly, given that most R functions for dealing with repeated measures data require data to be in long form, the `t.test()` function isn't really set up to handle data in long form. Instead it expects to be given two separate variables, `x` and `y`, and you need to specify `paired=TRUE`. And on top of that, you'd better make sure that the first element of `x` and the first element of `y` actually correspond to the same person! Because it doesn't ask for an "id" variable. I don't know why. So, in order to run the paired samples t test on the data from Dr Chico's class, we'd use this command:

```
> t.test( x = chico$grade_test2,   # variable 1 is the "test2" scores
+         y = chico$grade_test1,   # variable 2 is the "test1" scores
+         paired = TRUE           # paired test
+ )
```

and the output is

```
            Paired t-test

data:  chico$grade_test2 and chico$grade_test1
t = 6.4754, df = 19, p-value = 3.321e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.9508686 1.8591314
sample estimates:
mean of the differences
                  1.405
```

Yet again, these are the same numbers that we saw in Section 13.5. Feel free to check.

13.8 Effect size

The most commonly used measure of effect size for a t-test is **Cohen's d** (Cohen, 1988). It's a very simple measure in principle, with quite a few wrinkles when you start digging into the details. Cohen himself defined it primarily in the context of an independent samples t-test, specifically the Student test. In that context, a natural way of defining the effect size is to divide the difference between the means by an estimate of the standard deviation. In other words, we're looking to calculate *something* along the lines of this:

$$d = \frac{(\text{mean } 1) - (\text{mean } 2)}{\text{std dev}}$$

and he suggested a rough guide for interpreting d in Table 13.1. You'd think that this would be pretty unambiguous, but it's not; largely because Cohen wasn't too specific on what he thought should be used as the measure of the standard deviation (in his defence, he was trying to make a broader point in his book, not nitpick about tiny details). As discussed by McGrath and Meyer (2006), there are several different version in common usage, and each author tends to adopt slightly different notation. For the sake of simplicity (as opposed to accuracy) I'll use d to refer to any statistic that you calculate from the sample, and use δ to refer to a theoretical population effect. Obviously, that does mean that there are several different things all called d. The `cohensD()` function in the `lsr` package uses the `method` argument to distinguish between them, so that's what I'll do in the text.

My suspicion is that the only time that you would want Cohen's d is when you're running a t-test, and if you're using the `oneSampleTTest`, `independentSamplesTTest` and `pairedSamplesTTest()` functions to run your t-tests, then you don't need to learn any new commands, because they automatically produce

Table 13.1: A (very) rough guide to interpreting Cohen's d. My personal recommendation is to not use these blindly. The d statistic has a natural interpretation in and of itself: it redescribes the different in means as the number of standard deviations that separates those means. So it's generally a good idea to think about what that means in practical terms. In some contexts a "small" effect could be of big practical importance. In other situations a "large" effect may not be all that interesting.

d-value	rough interpretation
about 0.2	"small" effect
about 0.5	"moderate" effect
about 0.8	"large" effect

....

an estimate of Cohen's d as part of the output. However, if you're using `t.test()` then you'll need to use the `cohensD()` function (also in the `lsr` package) to do the calculations.

13.8.1 Cohen's d from one sample

The simplest situation to consider is the one corresponding to a one-sample t-test. In this case, the one sample mean $\bar{X}$ and one (hypothesised) population mean μ_o to compare it to. Not only that, there's really only one sensible way to estimate the population standard deviation: we just use our usual estimate $\hat{\sigma}$. Therefore, we end up with the following as the only way to calculate d,

$$d = \frac{\bar{X} - \mu_0}{\hat{\sigma}}$$

When writing the `cohensD()` function, I've made some attempt to make it work in a similar way to `t.test()`. As a consequence, `cohensD()` can calculate your effect size regardless of which type of t-test you performed. If what you want is a measure of Cohen's d to accompany a one-sample t-test, there's only two arguments that you need to care about. These are:

- `x`. A numeric vector containing the sample data.
- `mu`. The mean against which the mean of `x` is compared (default value is `mu = 0`).

We don't need to specify what `method` to use, because there's only one version of d that makes sense in this context. So, in order to compute an effect size for the data from Dr Zeppo's class (Section 13.2), we'd type something like this:

```
> cohensD( x = grades,    # data are stored in the grades vector
+          mu = 67.5      # compare students to a mean of 67.5
+ )
[1] 0.5041691
```

and, just so that you can see that there's nothing fancy going on, the command below shows you how to calculate it if there weren't no fancypants `cohensD()` function available:

```
> ( mean(grades) - 67.5 ) / sd(grades)
[1] 0.5041691
```

Yep, same number. Overall, then, the psychology students in Dr Zeppo's class are achieving grades (mean = 72.3%) that are about .5 standard deviations higher than the level that you'd expect (67.5%) if they were performing at the same level as other students. Judged against Cohen's rough guide, this is a moderate effect size.

13.8.2 Cohen's d from a Student t test

The majority of discussions of Cohen's d focus on a situation that is analogous to Student's independent samples t test, and it's in this context that the story becomes messier, since there are several different versions of d that you might want to use in this situation, and you can use the `method` argument to the `cohensD()` function to pick the one you want. To understand why there are multiple versions of d, it helps to take the time to write down a formula that corresponds to the true population effect size δ. It's pretty straightforward,

$$\delta = \frac{\mu_1 - \mu_2}{\sigma}$$

where, as usual, μ_1 and μ_2 are the population means corresponding to group 1 and group 2 respectively, and σ is the standard deviation (the same for both populations). The obvious way to estimate δ is to do exactly the same thing that we did in the t-test itself: use the sample means as the top line, and a pooled standard deviation estimate for the bottom line:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{\hat{\sigma}_p}$$

where $\hat{\sigma}_p$ is the exact same pooled standard deviation measure that appears in the t-test. This is the most commonly used version of Cohen's d when applied to the outcome of a Student t-test ,and is sometimes referred to as Hedges' g statistic (Hedges, 1981). It corresponds to `method = "pooled"` in the `cohensD()` function, and it's the default.

However, there are other possibilities, which I'll briefly describe. Firstly, you may have reason to want to use only one of the two groups as the basis for calculating the standard deviation. This approach (often called Glass' Δ) only makes most sense when you have good reason to treat one of the two groups as a purer reflection of "natural variation" than the other. This can happen if, for instance, one of the two groups is a control group. If that's what you want, then use `method = "x.sd"` or `method = "y.sd"` when using `cohensD()`. Secondly, recall that in the usual calculation of the pooled standard deviation we divide by $N-2$ to correct for the bias in the sample variance; in one version of Cohen's d this correction is omitted. Instead, we divide by N. This version (`method = "raw"`) makes sense primarily when you're trying to calculate the effect size in the sample; rather than estimating an effect size in the population. Finally, there is a version based on Hedges and Olkin (1985), who point out there is a small bias in the usual (pooled) estimation for Cohen's d. Thus they introduce a small correction (`method = "corrected"`), by multiplying the usual value of d by $(N-3)/(N-2.25)$.

In any case, ignoring all those variations that you could make use of if you wanted, let's have a look at how to calculate the default version. In particular, suppose we look at the data from Dr Harpo's class (the `harpo` data frame). The command that we want to use is very similar to the relevant `t.test()` command, but also specifies a `method`

```
> cohensD( formula = grade ~ tutor,  # outcome ~ group
+          data = harpo,             # data frame
+          method = "pooled"         # which version to calculate?
+ )
[1] 0.7395614
```

This is the version of Cohen's d that gets reported by the `independentSamplesTTest()` function whenever it runs a Student t-test.

13.8.3 Cohen's d from a Welch test

Suppose the situation you're in is more like the Welch test: you still have two independent samples, but you no longer believe that the corresponding populations have equal variances. When this happens, we have to redefine what we mean by the population effect size. I'll refer to this new measure as δ', so as to keep it distinct from the measure δ which we defined previously. What Cohen (1988) suggests is that we could define our new population effect size by averaging the two population variances. What this means is that we get:

$$\delta' = \frac{\mu_1 - \mu_2}{\sigma'}$$

where

$$\sigma' = \sqrt{\frac{{\sigma_1}^2 + {\sigma_2}^2}{2}}$$

This seems quite reasonable, but notice that none of the measures that we've discussed so far are attempting to estimate this new quantity. It might just be my own ignorance of the topic, but I'm only aware of one version of Cohen's d that actually estimates the unequal-variance effect size δ' rather than the equal-variance effect size δ. All we do to calculate d for this version (`method = "unequal"`) is substitute the sample means $\bar{X}_1$ and $\bar{X}_2$ and the corrected sample standard deviations $\hat{\sigma}_1$ and $\hat{\sigma}_2$ into the equation for δ'. This gives us the following equation for d,

$$d = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{\hat{\sigma}_1^2 + \hat{\sigma}_2^2}{2}}}$$

as our estimate of the effect size. There's nothing particularly difficult about calculating this version in R, since all we have to do is change the `method` argument:

```
> cohensD( formula = grade ~ tutor,
+          data = harpo,
+          method = "unequal"
+ )
[1] 0.7244995
```

his is the version of Cohen's d that gets reported by the `independentSamplesTTest()` function whenever it runs a Welch t-test.

13.8.4 Cohen's d from a paired-samples test

Finally, what should we do for a paired samples t-test? In this case, the answer depends on what it is you're trying to do. *If* you want to measure your effect sizes relative to the distribution of difference scores, the measure of d that you calculate is just (`method = "paired"`)

$$d = \frac{\bar{D}}{\hat{\sigma}_D}$$

where $\hat{\sigma}_D$ is the estimate of the standard deviation of the differences. The calculation here is pretty straightforward

```
> cohensD( x = chico$grade_test2,
+          y = chico$grade_test1,
+          method = "paired"
+ )
[1] 1.447952
```

This is the version of Cohen's d that gets reported by the `pairedSamplesTTest()` function. The only wrinkle is figuring out whether this is the measure you want or not. To the extent that you care about the practical consequences of your research, you often want to measure the effect size relative to the *original* variables, not the *difference* scores (e.g., the 1% improvement in Dr Chico's class is pretty small when measured against the amount of between-student variation in grades), in which case you use the same versions of Cohen's d that you would use for a Student or Welch test. For instance, when we do that for Dr Chico's class,

```
> cohensD( x = chico$grade_test2,
+          y = chico$grade_test1,
+          method = "pooled"
+ )
[1] 0.2157646
```

what we see is that the overall effect size is quite small, when assessed on the scale of the original variables.

13.9 Checking the normality of a sample

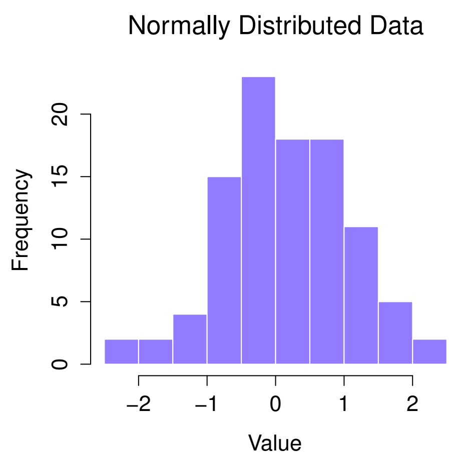
All of the tests that we have discussed so far in this chapter have assumed that the data are normally distributed. This assumption is often quite reasonable, because the central limit theorem (Section 10.3.3) does tend to ensure that many real world quantities are normally distributed: any time that you suspect that your variable is *actually* an average of lots of different things, there's a pretty good chance that it will be normally distributed; or at least close enough to normal that you can get away with using t-tests. However, life doesn't come with guarantees; and besides, there are lots of ways in which you can end up with variables that are highly non-normal. For example, any time you think that your variable is actually the minimum of lots of different things, there's a very good chance it will end up quite skewed. In psychology, response time (RT) data is a good example of this. If you suppose that there are lots of things that could trigger a response from a human participant, then the actual response will occur the first time one of these trigger events occurs.[16] This means that RT data are systematically non-normal. Okay, so if normality is assumed by all the tests, and is mostly but not always satisfied (at least approximately) by real world data, how can we check the normality of a sample? In this section I discuss two methods: QQ plots, and the Shapiro-Wilk test.

13.9.1 QQ plots

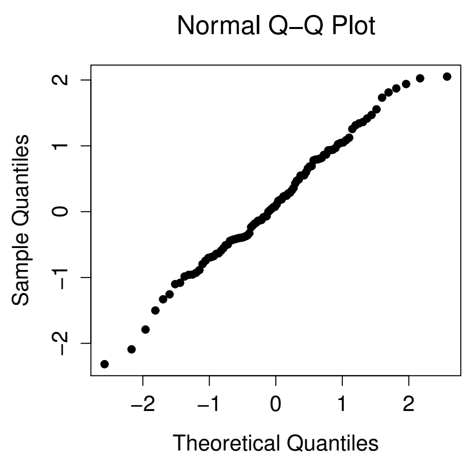
One way to check whether a sample violates the normality assumption is to draw a **"quantile-quantile" plot** (QQ plot). This allows you to visually check whether you're seeing any systematic violations. In a QQ plot, each observation is plotted as a single dot. The x co-ordinate is the theoretical quantile that the observation should fall in, if the data were normally distributed (with mean and variance estimated from the sample) and on the y co-ordinate is the actual quantile of the data within the sample. If the data are normal, the dots should form a straight line. For instance, lets see what happens if we generate data by sampling from a normal distribution, and then drawing a QQ plot using the R function `qqnorm()`. The `qqnorm()` function has a few arguments, but the only one we really need to care about here is `y`, a vector specifying the data whose normality we're interested in checking. Here's the R commands:

```
> normal.data <- rnorm( n = 100 )  # generate N = 100 normally distributed numbers
> hist( x = normal.data )          # draw a histogram of these numbers
> qqnorm( y = normal.data )        # draw the QQ plot
```

[16] This is a massive oversimplification.



(a)



(b)

Figure 13.11: Histogram (panel a) and normal QQ plot (panel b) of `normal.data`, a normally distributed sample with 100 observations. The Shapiro-Wilk statistic associated with these data is $W = .99$, indicating that no significant departures from normality were detected ($p = .73$).

And the results are shown in Figure 13.11. As you can see, these data form a pretty straight line; which is no surprise given that we sampled them from a normal distribution! In contrast, have a look at the two data sets shown in Figure 13.12. The top panels show the histogram and a QQ plot for a data set that is highly skewed: the QQ plot curves upwards. The lower panels show the same plots for a heavy tailed (i.e., high kurtosis) data set: in this case, the QQ plot flattens in the middle and curves sharply at either end.

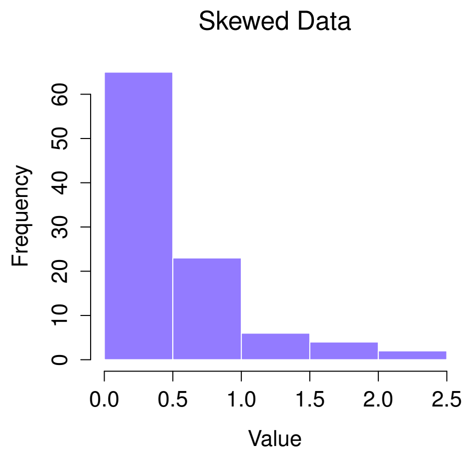
13.9.2 Shapiro-Wilk tests

Although QQ plots provide a nice way to informally check the normality of your data, sometimes you'll want to do something a bit more formal. And when that moment comes, the **Shapiro-Wilk test** (Shapiro & Wilk, 1965) is probably what you're looking for.[17] As you'd expect, the null hypothesis being tested is that a set of N observations is normally distributed. The test statistic that it calculates is conventionally denoted as W, and it's calculated as follows. First, we sort the observations in order of increasing size, and let X_1 be the smallest value in the sample, X_2 be the second smallest and so on. Then the value of W is given by

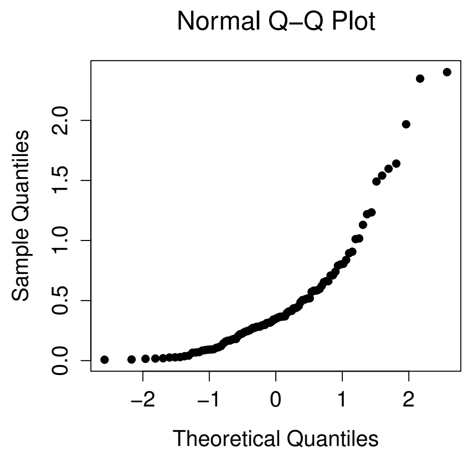
$$W = \frac{\left(\sum_{i=1}^N a_i X_i \right)^2}{\sum_{i=1}^N (X_i - \bar{X})^2}$$

where $\bar{X}$ is the mean of the observations, and the a_i values are ... mumble, mumble ... something complicated that is a bit beyond the scope of an introductory text.

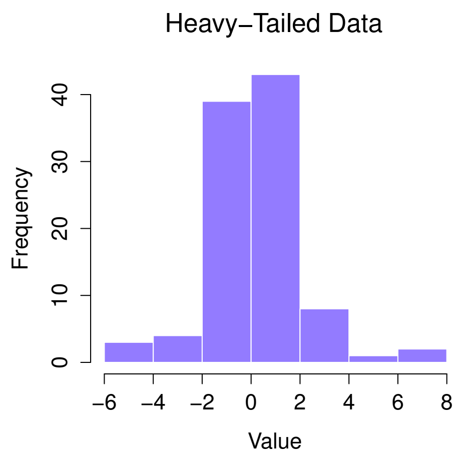
[17]Either that, or the Kolmogorov-Smirnov test, which is probably more traditional than the Shapiro-Wilk, though most things I've read seem to suggest Shapiro-Wilk is the better test of normality; although Kolomogorov-Smirnov is a general purpose test of distributional equivalence, so it can be adapted to handle other kinds of distribution tests; in R it's implemented via the `ks.test()` function.



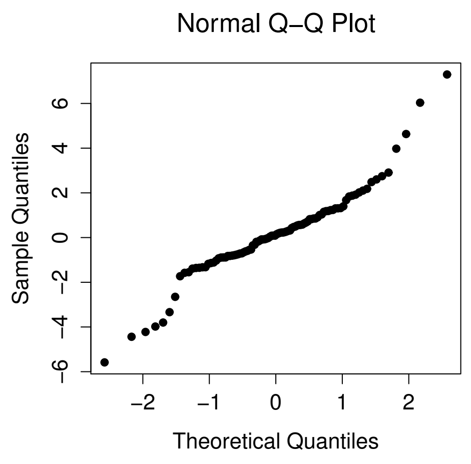
(a)



(b)



(c)



(d)

Figure 13.12: In the top row, a histogram (panel a) and normal QQ plot (panel b) of the 100 observations in a `skewed.data` set. The skewness of the data here is 1.94, and is reflected in a QQ plot that curves upwards. As a consequence, the Shapiro-Wilk statistic is $W = .80$, reflecting a significant departure from normality ($p < .001$). The bottom row shows the same plots for a heavy tailed data set, again consisting of 100 observations. In this case, the heavy tails in the data produce a high kurtosis (2.80), and cause the QQ plot to flatten in the middle, and curve away sharply on either side. The resulting Shapiro-Wilk statistic is $W = .93$, again reflecting significant non-normality ($p < .001$).

....

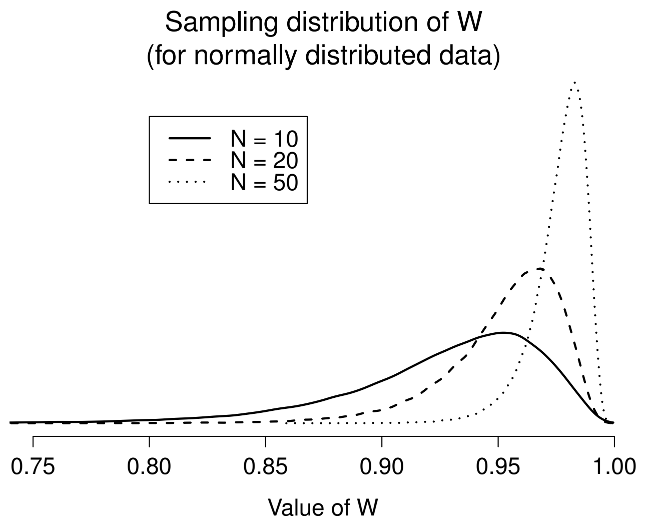


Figure 13.13: Sampling distribution of the Shapiro-Wilk W statistic, under the null hypothesis that the data are normally distributed, for samples of size 10, 20 and 50. Note that *small* values of W indicate departure from normality.

Because it's a little hard to explain the maths behind the W statistic, a better idea is to give a broad brush description of how it behaves. Unlike most of the test statistics that we'll encounter in this book, it's actually *small* values of W that indicated departure from normality. The W statistic has a maximum value of 1, which arises when the data look "perfectly normal". The smaller the value of W, the less normal the data are. However, the sampling distribution for W – which is not one of the standard ones that I discussed in Chapter 9 and is in fact a complete pain in the arse to work with – does depend on the sample size N. To give you a feel for what these sampling distributions look like, I've plotted three of them in Figure 13.13. Notice that, as the sample size starts to get large, the sampling distribution becomes very tightly clumped up near $W = 1$, and as a consequence, for larger samples W doesn't have to be very much smaller than 1 in order for the test to be significant.

To run the test in R, we use the `shapiro.test()` function. It has only a single argument `x`, which is a numeric vector containing the data whose normality needs to be tested. For example, when we apply this function to our `normal.data`, we get the following:

```
> shapiro.test( x = normal.data )

        Shapiro-Wilk normality test

data:  normal.data
W = 0.9903, p-value = 0.6862
```

So, not surprisingly, we have no evidence that these data depart from normality. When reporting the results for a Shapiro-Wilk test, you should (as usual) make sure to include the test statistic W and the p value, though given that the sampling distribution depends so heavily on N it would probably be a politeness to include N as well.

13.10 Testing non-normal data with Wilcoxon tests

Okay, suppose your data turn out to be pretty substantially non-normal, but you still want to run something like a t-test? This situation occurs a lot in real life: for the AFL winning margins data, for instance, the Shapiro-Wilk test made it very clear that the normality assumption is violated. This is the situation where you want to use Wilcoxon tests.

Like the t-test, the Wilcoxon test comes in two forms, one-sample and two-sample, and they're used in more or less the exact same situations as the corresponding t-tests. Unlike the t-test, the Wilcoxon test doesn't assume normality, which is nice. In fact, they don't make any assumptions about what kind of distribution is involved: in statistical jargon, this makes them **nonparametric tests**. While avoiding the normality assumption is nice, there's a drawback: the Wilcoxon test is usually less powerful than the t-test (i.e., higher Type II error rate). I won't discuss the Wilcoxon tests in as much detail as the t-tests, but I'll give you a brief overview.

13.10.1 Two sample Wilcoxon test

I'll start by describing the **two sample Wilcoxon test** (also known as the Mann-Whitney test), since it's actually simpler than the one sample version. Suppose we're looking at the scores of 10 people on some test. Since my imagination has now failed me completely, let's pretend it's a "test of awesomeness", and there are two groups of people, "A" and "B". I'm curious to know which group is more awesome. The data are included in the file `awesome.Rdata`, and like many of the data sets I've been using, it contains only a single data frame, in this case called `awesome`. Here's the data:

```
> load("awesome.Rdata")
> print( awesome )
   scores group
1     6.4     A
2    10.7     A
3    11.9     A
4     7.3     A
5    10.0     A
6    14.5     B
7    10.4     B
8    12.9     B
9    11.7     B
10   13.0     B
```

As long as there are no ties (i.e., people with the exact same awesomeness score), then the test that we want to do is surprisingly simple. All we have to do is construct a table that compares every observation in group A against every observation in group B. Whenever the group A datum is larger, we place a check mark in the table:

		group B				
		14.5	10.4	12.4	11.7	13.0
	6.4	.	.	.	.	.
	10.7	.	✓	.	.	.
group A	11.9	.	✓	.	✓	.
	7.3	.	.	.	.	.
	10.0	.	.	.	.	.

We then count up the number of checkmarks. This is our test statistic, W.[18] The actual sampling distribution for W is somewhat complicated, and I'll skip the details. For our purposes, it's sufficient to note that the interpretation of W is qualitatively the same as the interpretation of t or z. That is, if we want a two-sided test, then we reject the null hypothesis when W is very large or very small; but if we have a directional (i.e., one-sided) hypothesis, then we only use one or the other.

The structure of the `wilcox.test()` function should feel very familiar to you by now. When you have your data organised in terms of an outcome variable and a grouping variable, then you use the `formula` and `data` arguments, so your command looks like this:

```
> wilcox.test( formula = scores ~ group, data = awesome)

        Wilcoxon rank sum test

data:  scores by group
W = 3, p-value = 0.05556
alternative hypothesis: true location shift is not equal to 0
```

Just like we saw with the `t.test()` function, there is an `alternative` argument that you can use to switch between two-sided tests and one-sided tests, plus a few other arguments that we don't need to worry too much about at an introductory level. Similarly, the `wilcox.test()` function allows you to use the `x` and `y` arguments when you have your data stored separately for each group. For instance, suppose we use the data from the `awesome2.Rdata` file:

```
> load( "awesome2.Rdata" )
> score.A
[1]  6.4 10.7 11.9  7.3 10.0
> score.B
[1] 14.5 10.4 12.9 11.7 13.0
```

When your data are organised like this, then you would use a command like this:

```
> wilcox.test( x = score.A, y = score.B )
```

The output that R produces is pretty much the same as last time.

13.10.2 One sample Wilcoxon test

What about the **one sample Wilcoxon test** (or equivalently, the paired samples Wilcoxon test)? Suppose I'm interested in finding out whether taking a statistics class has any effect on the happiness of students. Here's my data:

```
> load( "happy.Rdata" )
> print( happiness )
   before after change
1      30     6    -24
2      43    29    -14
3      21    11    -10
4      24    31      7
5      23    17     -6
6      40     2    -38
```

[18] Actually, there are two different versions of the test statistic; they differ from each other by a constant value. The version that I've described is the one that R calculates.

```
7       29    31      2
8       56    21    -35
9       38     8    -30
10      16    21      5
```

What I've measured here is the happiness of each student `before` taking the class and `after` taking the class; the `change` score is the difference between the two. Just like we saw with the t-test, there's no fundamental difference between doing a paired-samples test using `before` and `after`, versus doing a one-sample test using the `change` scores. As before, the simplest way to think about the test is to construct a tabulation. The way to do it this time is to take those change scores that are positive valued, and tabulate them against all the complete sample. What you end up with is a table that looks like this:

		all differences									
		-24	-14	-10	7	-6	-38	2	-35	-30	5
positive differences	7	.	.	.	✓	✓	.	✓	.	.	✓
	2	.	.	.	.	.	.	✓	.	.	.
	5	.	.	.	.	.	.	✓	.	.	✓

Counting up the tick marks this time, we get a test statistic of $V = 7$. As before, if our test is two sided, then we reject the null hypothesis when V is very large or very small. As far of running it in R goes, it's pretty much what you'd expect. For the one-sample version, the command you would use is

```
> wilcox.test( x = happiness$change,
+              mu = 0
+ )
```

and the output that you get is

```
        Wilcoxon signed rank test

data:  happiness$change
V = 7, p-value = 0.03711
alternative hypothesis: true location is not equal to 0
```

As this shows, we have a significant effect. Evidently, taking a statistics class does have an effect on your happiness. Switching to a paired samples version of the test won't give us different answers, of course; but here's the command to do it:

```
> wilcox.test( x = happiness$after,
+              y = happiness$before,
+              paired = TRUE
+ )
```

13.11 Summary

- A one sample t-test is used to compare a single sample mean against a hypothesised value for the population mean. (Section 13.2)
- An independent samples t-test is used to compare the means of two groups, and tests the null hypothesis that they have the same mean. It comes in two forms: the Student test (Section 13.3) assumes that the groups have the same standard deviation, the Welch test (Section 13.4) does not.

- A paired samples t-test is used when you have two scores from each person, and you want to test the null hypothesis that the two scores have the same mean. It is equivalent to taking the difference between the two scores for each person, and then running a one sample t-test on the difference scores. (Section 13.5)
- Effect size calculations for the difference between means can be calculated via the Cohen's d statistic. (Section 13.8).
- You can check the normality of a sample using QQ plots and the Shapiro-Wilk test. (Section 13.9)
- If your data are non-normal, you can use Wilcoxon tests instead of t-tests. (Section 13.10)

14. Comparing several means (one-way ANOVA)

This chapter introduces one of the most widely used tools in statistics, known as "the analysis of variance", which is usually referred to as ANOVA. The basic technique was developed by Sir Ronald Fisher in the early 20th century, and it is to him that we owe the rather unfortunate terminology. The term ANOVA is a little misleading, in two respects. Firstly, although the name of the technique refers to variances, ANOVA is concerned with investigating differences in means. Secondly, there are several different things out there that are all referred to as ANOVAs, some of which have only a very tenuous connection to one another. Later on in the book we'll encounter a range of different ANOVA methods that apply in quite different situations, but for the purposes of this chapter we'll only consider the simplest form of ANOVA, in which we have several different groups of observations, and we're interested in finding out whether those groups differ in terms of some outcome variable of interest. This is the question that is addressed by a **one-way ANOVA**.

The structure of this chapter is as follows: In Section 14.1 I'll introduce a fictitious data set that we'll use as a running example throughout the chapter. After introducing the data, I'll describe the mechanics of how a one-way ANOVA actually works (Section 14.2) and then focus on how you can run one in R (Section 14.3). These two sections are the core of the chapter. The remainder of the chapter discusses a range of important topics that inevitably arise when running an ANOVA, namely how to calculate effect sizes (Section 14.4), post hoc tests and corrections for multiple comparisons (Section 14.5) and the assumptions that ANOVA relies upon (Section 14.6). We'll also talk about how to check those assumptions and some of the things you can do if the assumptions are violated (Sections 14.7 to 14.10). At the end of the chapter we'll talk a little about the relationship between ANOVA and other statistical tools (Section 14.11).

14.1 An illustrative data set

Suppose you've become involved in a clinical trial in which you are testing a new antidepressant drug called *Joyzepam*. In order to construct a fair test of the drug's effectiveness, the study involves three separate drugs to be administered. One is a placebo, and the other is an existing antidepressant / anti-anxiety drug called *Anxifree*. A collection of 18 participants with moderate to severe depression are recruited for your initial testing. Because the drugs are sometimes administered in conjunction with psychological therapy, your study includes 9 people undergoing cognitive behavioural therapy (CBT) and 9 who are not. Participants are randomly assigned (doubly blinded, of course) a treatment, such that there are 3 CBT people and 3 no-therapy people assigned to each of the 3 drugs. A psychologist assesses the mood of each person after a 3 month run with each drug: and the overall *improvement* in each person's mood is assessed on a scale ranging from -5 to $+5$.

With that as the study design, let's now look at what we've got in the data file:

```
> load( "clinicaltrial.Rdata" ) # load data
> who( expand = TRUE )   # inspect the workspace
   -- Name --     -- Class --    -- Size --
   clin.trial     data.frame     18 x 3
    $drug         factor         18
    $therapy      factor         18
    $mood.gain    numeric        18
```

So we have a single data frame called `clin.trial`, containing three variables; `drug`, `therapy` and `mood.gain`. Next, let's print the data frame to get a sense of what the data actually look like.

```
> print( clin.trial )
       drug    therapy mood.gain
1   placebo no.therapy       0.5
2   placebo no.therapy       0.3
3   placebo no.therapy       0.1
4  anxifree no.therapy       0.6
5  anxifree no.therapy       0.4
6  anxifree no.therapy       0.2
7  joyzepam no.therapy       1.4
8  joyzepam no.therapy       1.7
9  joyzepam no.therapy       1.3
10  placebo        CBT       0.6
11  placebo        CBT       0.9
12  placebo        CBT       0.3
13 anxifree        CBT       1.1
14 anxifree        CBT       0.8
15 anxifree        CBT       1.2
16 joyzepam        CBT       1.8
17 joyzepam        CBT       1.3
18 joyzepam        CBT       1.4
```

For the purposes of this chapter, what we're really interested in is the effect of `drug` on `mood.gain`. The first thing to do is calculate some descriptive statistics and draw some graphs. In Chapter 5 we discussed a variety of different functions that can be used for this purpose. For instance, we can use the `xtabs()` function to see how many people we have in each group:

```
> xtabs( ~drug, clin.trial )
drug
 placebo anxifree joyzepam
       6        6        6
```

Similarly, we can use the `aggregate()` function to calculate means and standard deviations for the `mood.gain` variable broken down by which `drug` was administered:

```
> aggregate( mood.gain ~ drug, clin.trial, mean )
      drug mood.gain
1  placebo 0.4500000
2 anxifree 0.7166667
3 joyzepam 1.4833333

> aggregate( mood.gain ~ drug, clin.trial, sd )
      drug mood.gain
```

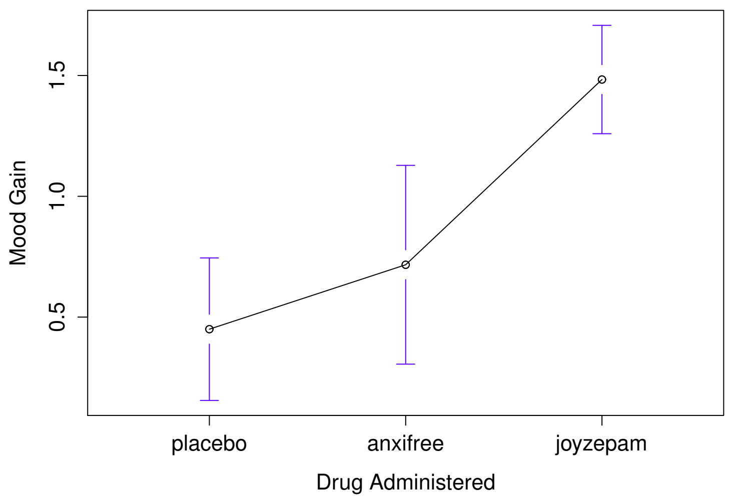


Figure 14.1: Average mood gain as a function of drug administered. Error bars depict 95% confidence intervals associated with each of the group means.

....

```
1  placebo 0.2810694
2 anxifree 0.3920034
3 joyzepam 0.2136976
```

Finally, we can use `plotmeans()` from the `gplots` package to produce a pretty picture.

```
> library(gplots)
> plotmeans(  formula = mood.gain ~ drug,  # plot mood.gain by drug
+             data = clin.trial,           # the data frame
+             xlab = "Drug Administered",  # x-axis label
+             ylab = "Mood Gain",          # y-axis label
+             n.label = FALSE              # don't display sample size
+ )
```

The results are shown in Figure 14.1, which plots the average mood gain for all three conditions; error bars show 95% confidence intervals. As the plot makes clear, there is a larger improvement in mood for participants in the Joyzepam group than for either the Anxifree group or the placebo group. The Anxifree group shows a larger mood gain than the control group, but the difference isn't as large.

The question that we want to answer is: are these difference "real", or are they just due to chance?

14.2 How ANOVA works

In order to answer the question posed by our clinical trial data, we're going to run a one-way ANOVA. As usual, I'm going to start by showing you how to do it the hard way, building the statistical tool from the ground up and showing you how you could do it in R if you didn't have access to any of the cool built-in ANOVA functions. And, as always, I hope you'll read it carefully, try to do it the long way once or twice to make sure you really understand how ANOVA works, and then – once you've grasped the concept – never *ever* do it this way again.

The experimental design that I described in the previous section strongly suggests that we're interested in comparing the average mood change for the three different drugs. In that sense, we're talking about an analysis similar to the *t*-test (Chapter 13), but involving more than two groups. If we let μ_P denote the population mean for the mood change induced by the placebo, and let μ_A and μ_J denote the corresponding means for our two drugs, Anxifree and Joyzepam, then the (somewhat pessimistic) null hypothesis that we want to test is that all three population means are identical: that is, *neither* of the two drugs is any more effective than a placebo. Mathematically, we write this null hypothesis like this:

$$H_0 \quad : \quad \text{it is true that } \mu_P = \mu_A = \mu_J$$

As a consequence, our alternative hypothesis is that at least one of the three different treatments is different from the others. It's a little trickier to write this mathematically, because (as we'll discuss) there are quite a few different ways in which the null hypothesis can be false. So for now we'll just write the alternative hypothesis like this:

$$H_1 \quad : \quad \text{it is } \underline{\text{not}} \text{ true that } \mu_P = \mu_A = \mu_J$$

This null hypothesis is a lot trickier to test than any of the ones we've seen previously. How shall we do it? A sensible guess would be to "do an ANOVA", since that's the title of the chapter, but it's not particularly clear why an "analysis of *variances*" will help us learn anything useful about the *means*. In fact, this is one of the biggest conceptual difficulties that people have when first encountering ANOVA. To see how this works, I find it most helpful to start by talking about variances. In fact, what I'm going to do is start by playing some mathematical games with the formula that describes the variance. That is, we'll start out by playing around with variances, and it will turn out that this gives us a useful tool for investigating means.

14.2.1 Two formulas for the variance of Y

Firstly, let's start by introducing some notation. We'll use G to refer to the total number of groups. For our data set, there are three drugs, so there are $G = 3$ groups. Next, we'll use N to refer to the total sample size: there are a total of $N = 18$ people in our data set. Similarly, let's use N_k to denote the number of people in the k-th group. In our fake clinical trial, the sample size is $N_k = 6$ for all three groups.[1] Finally, we'll use Y to denote the outcome variable: in our case, Y refers to mood change. Specifically, we'll use Y_{ik} to refer to the mood change experienced by the i-th member of the k-th group. Similarly, we'll use $\bar{Y}$ to be the average mood change, taken across all 18 people in the experiment, and $\bar{Y}_k$ to refer to the average mood change experienced by the 6 people in group k.

Excellent. Now that we've got our notation sorted out, we can start writing down formulas. To start with, let's recall the formula for the variance that we used in Section 5.2, way back in those kinder days when we were just doing descriptive statistics. The sample variance of Y is defined as follows:

$$\text{Var}(Y) = \frac{1}{N} \sum_{k=1}^{G} \sum_{i=1}^{N_k} \left(Y_{ik} - \bar{Y}\right)^2$$

[1] When all groups have the same number of observations, the experimental design is said to be "balanced". Balance isn't such a big deal for one-way ANOVA, which is the topic of this chapter. It becomes more important when you start doing more complicated ANOVAs.

This formula looks pretty much identical to the formula for the variance in Section 5.2. The only difference is that this time around I've got two summations here: I'm summing over groups (i.e., values for k) and over the people within the groups (i.e., values for i). This is purely a cosmetic detail: if I'd instead used the notation Y_p to refer to the value of the outcome variable for person p in the sample, then I'd only have a single summation. The only reason that we have a double summation here is that I've classified people into groups, and then assigned numbers to people within groups.

A concrete example might be useful here. Let's consider this table, in which we have a total of $N = 5$ people sorted into $G = 2$ groups. Arbitrarily, let's say that the "cool" people are group 1, and the "uncool" people are group 2, and it turns out that we have three cool people ($N_1 = 3$) and two uncool people ($N_2 = 2$).

name	person	group	group num.	index in group	grumpiness
	p		k	i	Y_{ik} or Y_p
Ann	1	cool	1	1	20
Ben	2	cool	1	2	55
Cat	3	cool	1	3	21
Dan	4	uncool	2	1	91
Egg	5	uncool	2	2	22

Notice that I've constructed two different labelling schemes here. We have a "person" variable p, so it would be perfectly sensible to refer to Y_p as the grumpiness of the p-th person in the sample. For instance, the table shows that Dan is the four so we'd say $p = 4$. So, when talking about the grumpiness Y of this "Dan" person, whoever he might be, we could refer to his grumpiness by saying that $Y_p = 91$, for person $p = 4$ that is. However, that's not the only way we could refer to Dan. As an alternative we could we could note that Dan belongs to the "uncool" group ($k = 2$), and is in fact the first person listed in the uncool group ($i = 1$). So it's equally valid to refer to Dan's grumpiness by saying that $Y_{ik} = 91$, where $k = 2$ and $i = 1$. In other words, each person p corresponds to a unique ik combination, and so the formula that I gave above is actually identical to our original formula for the variance, which would be

$$\mathrm{Var}(Y) = \frac{1}{N} \sum_{p=1}^{N} \left(Y_p - \bar{Y} \right)^2$$

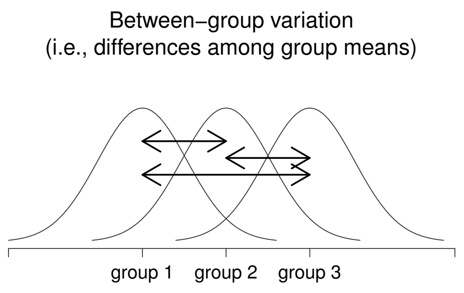
In both formulas, all we're doing is summing over all of the observations in the sample. Most of the time we would just use the simpler Y_p notation: the equation using Y_p is clearly the simpler of the two. However, when doing an ANOVA it's important to keep track of which participants belong in which groups, and we need to use the Y_{ik} notation to do this.

14.2.2 From variances to sums of squares

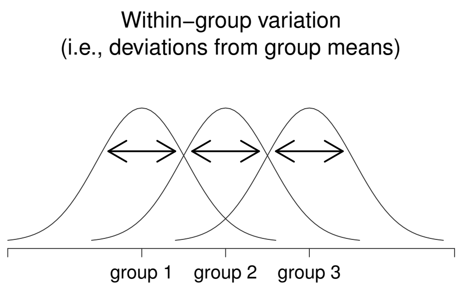
Okay, now that we've got a good grasp on how the variance is calculated, let's define something called the **total sum of squares**, which is denoted SS_{tot}. This is very simple: instead of averaging the squared deviations, which is what we do when calculating the variance, we just add them up. So the formula for the total sum of squares is almost identical to the formula for the variance:

$$\mathrm{SS}_{tot} = \sum_{k=1}^{G} \sum_{i=1}^{N_k} \left(Y_{ik} - \bar{Y} \right)^2$$

When we talk about analysing variances in the context of ANOVA, what we're really doing is working with the total sums of squares rather than the actual variance. One very nice thing about the total sum of squares is that we can break it up into two different kinds of variation. Firstly, we can talk about the



(a)



(b)

Figure 14.2: Graphical illustration of "between groups" variation (panel a) and "within groups" variation (panel b). On the left, the arrows show the differences in the group means; on the right, the arrows highlight the variability within each group.

....

within-group sum of squares, in which we look to see how different each individual person is from their own group mean:

$$\text{SS}_w = \sum_{k=1}^{G} \sum_{i=1}^{N_k} \left(Y_{ik} - \bar{Y}_k \right)^2$$

where $\bar{Y}_k$ is a group mean. In our example, $\bar{Y}_k$ would be the average mood change experienced by those people given the k-th drug. So, instead of comparing individuals to the average of all people in the experiment, we're only comparing them to those people in the the same group. As a consequence, you'd expect the value of SS_w to be smaller than the total sum of squares, because it's completely ignoring any group differences – that is, the fact that the drugs (if they work) will have different effects on people's moods.

Next, we can define a third notion of variation which captures *only* the differences between groups. We do this by looking at the differences between the group means $\bar{Y}_k$ and grand mean $\bar{Y}$. In order to quantify the extent of this variation, what we do is calculate the **between-group sum of squares**:

$$\begin{aligned} \text{SS}_b &= \sum_{k=1}^{G} \sum_{i=1}^{N_k} \left(\bar{Y}_k - \bar{Y} \right)^2 \\ &= \sum_{k=1}^{G} N_k \left(\bar{Y}_k - \bar{Y} \right)^2 \end{aligned}$$

It's not too difficult to show that the total variation among people in the experiment SS_{tot} is actually the sum of the differences between the groups SS_b and the variation inside the groups SS_w. That is:

$$\text{SS}_w + \text{SS}_b = \text{SS}_{tot}$$

Yay.

Okay, so what have we found out? We've discovered that the total variability associated with the outcome variable (SS_{tot}) can be mathematically carved up into the sum of "the variation due to the differences in the sample means for the different groups" (SS_b) plus "all the rest of the variation" (SS_w). How does that help me find out whether the groups have different population means? Um. Wait. Hold on a second... now that I think about it, this is *exactly* what we were looking for. If the null hypothesis is true, then you'd expect all the sample means to be pretty similar to each other, right? And that would imply that you'd expect SS_b to be really small, or at least you'd expect it to be a lot smaller than the "the variation associated with everything else", SS_w. Hm. I detect a hypothesis test coming on...

14.2.3 From sums of squares to the F-test

As we saw in the last section, the *qualitative* idea behind ANOVA is to compare the two sums of squares values SS_b and SS_w to each other: if the between-group variation is SS_b is large relative to the within-group variation SS_w then we have reason to suspect that the population means for the different groups aren't identical to each other. In order to convert this into a workable hypothesis test, there's a little bit of "fiddling around" needed. What I'll do is first show you *what* we do to calculate our test statistic – which is called an **F ratio** – and then try to give you a feel for *why* we do it this way.

In order to convert our SS values into an F-ratio, the first thing we need to calculate is the **degrees of freedom** associated with the SS_b and SS_w values. As usual, the degrees of freedom corresponds to the number of unique "data points" that contribute to a particular calculation, minus the number of "constraints" that they need to satisfy. For the within-groups variability, what we're calculating is the variation of the individual observations (N data points) around the group means (G constraints). In contrast, for the between groups variability, we're interested in the variation of the group means (G data points) around the grand mean (1 constraint). Therefore, the degrees of freedom here are:

$$\begin{aligned}\text{df}_b &= G - 1 \\ \text{df}_w &= N - G\end{aligned}$$

Okay, that seems simple enough. What we do next is convert our summed squares value into a "mean squares" value, which we do by dividing by the degrees of freedom:

$$\begin{aligned}\text{MS}_b &= \frac{\text{SS}_b}{\text{df}_b} \\ \text{MS}_w &= \frac{\text{SS}_w}{\text{df}_w}\end{aligned}$$

Finally, we calculate the F-ratio by dividing the between-groups MS by the within-groups MS:

$$F = \frac{\text{MS}_b}{\text{MS}_w}$$

At a very general level, the intuition behind the F statistic is straightforward: bigger values of F means that the between-groups variation is large, relative to the within-groups variation. As a consequence, the larger the value of F, the more evidence we have against the null hypothesis. But how large does F have to be in order to actually *reject* H_0? In order to understand this, you need a slightly deeper understanding of what ANOVA is and what the mean squares values actually are.

The next section discusses that in a bit of detail, but for readers that aren't interested in the details of what the test is actually measuring, I'll cut to the chase. In order to complete our hypothesis test, we need to know the sampling distribution for F if the null hypothesis is true. Not surprisingly, the sampling distribution for the F statistic under the null hypothesis is an F distribution. If you recall back to our discussion of the F distribution in Chapter 9, the F distribution has two parameters, corresponding to the two degrees of freedom involved: the first one df_1 is the between groups degrees of freedom df_b, and the second one df_2 is the within groups degrees of freedom df_w.

A summary of all the key quantities involved in a one-way ANOVA, including the formulas showing

Table 14.1: All of the key quantities involved in an ANOVA, organised into a "standard" ANOVA table. The formulas for all quantities (except the p-value, which has a very ugly formula and would be nightmarishly hard to calculate without a computer) are shown.

	df	sum of squares	mean squares	F-statistic	p-value
between groups	$\text{df}_b = G - 1$	$\text{SS}_b = \sum_{k=1}^{G} N_k(\bar{Y}_k - \bar{Y})^2$	$\text{MS}_b = \frac{\text{SS}_b}{\text{df}_b}$	$F = \frac{\text{MS}_b}{\text{MS}_w}$	[complicated]
within groups	$\text{df}_w = N - G$	$\text{SS}_w = \sum_{k=1}^{G}\sum_{i=1}^{N_k}(Y_{ik} - \bar{Y}_k)^2$	$\text{MS}_w = \frac{\text{SS}_w}{\text{df}_w}$	-	-

how they are calculated, is shown in Table 14.1.

14.2.4 The model for the data and the meaning of F

At a fundamental level, ANOVA is a competition between two different statistical models, H_0 and H_1. When I described the null and alternative hypotheses at the start of the section, I was a little imprecise about what these models actually are. I'll remedy that now, though you probably won't like me for doing so. If you recall, our null hypothesis was that all of the group means are identical to one another. If so, then a natural way to think about the outcome variable Y_{ik} is to describe individual scores in terms of a single population mean μ, plus the deviation from that population mean. This deviation is usually denoted ϵ_{ik} and is traditionally called the *error* or **residual** associated with that observation. Be careful though: just like we saw with the word "significant", the word "error" has a technical meaning in statistics that isn't quite the same as its everyday English definition. In everyday language, "error" implies a mistake of some kind; in statistics, it doesn't (or at least, not necessarily). With that in mind, the word "residual" is a better term than the word "error". In statistics, both words mean "leftover variability": that is, "stuff" that the model can't explain. In any case, here's what the null hypothesis looks like when we write it as a statistical model:

$$Y_{ik} = \mu + \epsilon_{ik}$$

where we make the *assumption* (discussed later) that the residual values ϵ_{ik} are normally distributed, with mean 0 and a standard deviation σ that is the same for all groups. To use the notation that we introduced in Chapter 9 we would write this assumption like this:

$$\epsilon_{ik} \sim \text{Normal}(0, \sigma^2)$$

What about the alternative hypothesis, H_1? The only difference between the null hypothesis and the alternative hypothesis is that we allow each group to have a different population mean. So, if we let μ_k denote the population mean for the k-th group in our experiment, then the statistical model corresponding to H_1 is:

$$Y_{ik} = \mu_k + \epsilon_{ik}$$

where, once again, we assume that the error terms are normally distributed with mean 0 and standard deviation σ. That is, the alternative hypothesis also assumes that $\epsilon \sim \text{Normal}(0, \sigma^2)$

Okay, now that we've described the statistical models underpinning H_0 and H_1 in more detail, it's now pretty straightforward to say what the mean square values are measuring, and what this means for the interpretation of F. I won't bore you with the proof of this, but it turns out that the within-groups mean square, MS_w, can be viewed as an estimator (in the technical sense: Chapter 10) of the error variance σ^2. The between-groups mean square MS_b is also an estimator; but what it estimates is the error variance *plus* a quantity that depends on the true differences among the group means. If we call this quantity Q, then we can see that the F-statistic is basically[2]

$$F = \frac{\hat{Q} + \hat{\sigma}^2}{\hat{\sigma}^2}$$

where the true value $Q = 0$ if the null hypothesis is true, and $Q > 0$ if the alternative hypothesis is true (e.g. Hays, 1994, ch. 10). Therefore, at a bare minimum *the F value must be larger than 1* to have any chance of rejecting the null hypothesis. Note that this *doesn't* mean that it's impossible to get an F-value less than 1. What it means is that, if the null hypothesis is true the sampling distribution of the F ratio has a mean of 1,[3] and so we need to see F-values larger than 1 in order to safely reject the null.

To be a bit more precise about the sampling distribution, notice that if the null hypothesis is true, both MS_b and MS_w are estimators of the variance of the residuals ϵ_{ik}. If those residuals are normally distributed, then you might suspect that the estimate of the variance of ϵ_{ik} is chi-square distributed... because (as discussed in Section 9.6) that's what a chi-square distribution *is*: it's what you get when you square a bunch of normally-distributed things and add them up. And since the F distribution is (again, by definition) what you get when you take the ratio between two things that are χ^2 distributed... we have our sampling distribution. Obviously, I'm glossing over a whole lot of stuff when I say this, but in broad terms, this really is where our sampling distribution comes from.

14.2.5 A worked example

The previous discussion was fairly abstract, and a little on the technical side, so I think that at this point it might be useful to see a worked example. For that, let's go back to the clinical trial data that I introduced at the start of the chapter. The descriptive statistics that we calculated at the beginning tell us our group means: an average mood gain of 0.45 for the placebo, 0.72 for Anxifree, and 1.48 for Joyzepam. With that in mind, let's party like it's 1899 [4] and start doing some pencil and paper calculations. I'll only do this for the first 5 observations, because it's not bloody 1899 and I'm very lazy. Let's start by calculating SS_w, the within-group sums of squares. First, let's draw up a nice table to help us with our calculations...

group	outcome
k	Y_{ik}
placebo	0.5
placebo	0.3
placebo	0.1
anxifree	0.6
anxifree	0.4

At this stage, the only thing I've included in the table is the raw data itself: that is, the grouping variable (i.e., `drug`) and outcome variable (i.e. `mood.gain`) for each person. Note that the outcome variable here corresponds to the Y_{ik} value in our equation previously. The next step in the calculation is to write down, for each person in the study, the corresponding group mean; that is, $\bar{Y}_k$. This is slightly repetitive, but not particularly difficult since we already calculated those group means when doing our descriptive

[2] In a later versions I'm intending to expand on this. But because I'm writing in a rush, and am already over my deadlines, I'll just briefly note that if you read ahead to Chapter 16 and look at how the "treatment effect" at level k of a factor is defined in terms of the α_k values (see Section 16.2), it turns out that Q refers to a weighted mean of the squared treatment effects, $Q = (\sum_{k=1}^{G} N_k \alpha_k^2)/(G-1)$.

[3] Or, if we want to be sticklers for accuracy, $1 + \frac{2}{df_2 - 2}$.

[4] Or, to be precise, party like "it's 1899 and we've got no friends and nothing better to do with our time than do some calculations that wouldn't have made any sense in 1899 because ANOVA didn't exist until about the 1920s".

statistics:

group k	outcome Y_{ik}	**group mean** $\bar{Y}_k$
placebo	0.5	**0.45**
placebo	0.3	**0.45**
placebo	0.1	**0.45**
anxifree	0.6	**0.72**
anxifree	0.4	**0.72**

Now that we've written those down, we need to calculate – again for every person – the deviation from the corresponding group mean. That is, we want to subtract $Y_{ik} - \bar{Y}_k$. After we've done that, we need to square everything. When we do that, here's what we get: .

group k	outcome Y_{ik}	group mean $\bar{Y}_k$	**dev. from group mean** $Y_{ik} - \bar{Y}_k$	**squared deviation** $(Y_{ik} - \bar{Y}_k)^2$
placebo	0.5	0.45	**0.05**	**0.0025**
placebo	0.3	0.45	**-0.15**	**0.0225**
placebo	0.1	0.45	**-0.35**	**0.1225**
anxifree	0.6	0.72	**-0.12**	**0.0136**
anxifree	0.4	0.72	**-0.32**	**0.1003**

The last step is equally straightforward. In order to calculate the within-group sum of squares, we just add up the squared deviations across all observations:

$$\begin{aligned} \text{SS}_w &= 0.0025 + 0.0225 + 0.1225 + 0.0136 + 0.1003 \\ &= 0.2614 \end{aligned}$$

Of course, if we actually wanted to get the *right* answer, we'd need to do this for all 18 observations in the data set, not just the first five. We could continue with the pencil and paper calculations if we wanted to, but it's pretty tedious. Alternatively, it's not too hard to get R to do it. Here's how:

```
> outcome <- clin.trial$mood.gain
> group <- clin.trial$drug
> gp.means <- tapply(outcome,group,mean)
> gp.means <- gp.means[group]
> dev.from.gp.means <- outcome - gp.means
> squared.devs <- dev.from.gp.means ^2
```

It might not be obvious from inspection what these commands are doing: as a general rule, the human brain seems to just shut down when faced with a big block of programming. However, I strongly suggest that – if you're like me and tend to find that the mere sight of this code makes you want to look away and see if there's any beer left in the fridge or a game of footy on the telly – you take a moment and look closely at these commands one at a time. Every single one of these commands is something you've seen before somewhere else in the book. There's nothing novel about them (though I'll have to admit that the `tapply()` function takes a while to get a handle on), so if you're not quite sure how these commands work, this might be a good time to try playing around with them yourself, to try to get a sense of what's happening. On the other hand, if this does seem to make sense, then you won't be all that surprised at what happens when I wrap these variables in a data frame, and print it out...

```
> Y <- data.frame( group, outcome, gp.means,
+                  dev.from.gp.means, squared.devs )
> print(Y, digits = 2)
      group outcome gp.means dev.from.gp.means squared.devs
1   placebo     0.5     0.45             0.050       0.0025
2   placebo     0.3     0.45            -0.150       0.0225
```

```
3   placebo     0.1     0.45             -0.350       0.1225
4  anxifree     0.6     0.72             -0.117       0.0136
5  anxifree     0.4     0.72             -0.317       0.1003

BLAH BLAH BLAH

16 joyzepam     1.8     1.48              0.317       0.1003
17 joyzepam     1.3     1.48             -0.183       0.0336
18 joyzepam     1.4     1.48             -0.083       0.0069
```

If you compare this output to the contents of the table I've been constructing by hand, you can see that R has done exactly the same calculations that I was doing, and much faster too. So, if we want to finish the calculations of the within-group sum of squares in R, we just ask for the `sum()` of the `squared.devs` variable:

```
> SSw <- sum( squared.devs )
> print( SSw )
[1] 1.39
```

Obviously, this isn't the same as what I calculated, because R used all 18 observations. But if I'd typed `sum( squared.devs[1:5] )` instead, it would have given the same answer that I got earlier.

Okay. Now that we've calculated the within groups variation, SS_w, it's time to turn our attention to the between-group sum of squares, SS_b. The calculations for this case are very similar. The main difference is that, instead of calculating the differences between an observation Y_{ik} and a group mean $\bar{Y}_k$ for all of the observations, we calculate the differences between the group means $\bar{Y}_k$ and the grand mean $\bar{Y}$ (in this case 0.88) for all of the groups...

group	group mean	grand mean	deviation	squared deviations
k	$\bar{Y}_k$	$\bar{Y}$	$\bar{Y}_k - \bar{Y}$	$(\bar{Y}_k - \bar{Y})^2$
placebo	0.45	0.88	-0.43	0.18
anxifree	0.72	0.88	-0.16	0.03
joyzepam	1.48	0.88	0.60	0.36

However, for the between group calculations we need to multiply each of these squared deviations by N_k, the number of observations in the group. We do this because every *observation* in the group (all N_k of them) is associated with a between group difference. So if there are six people in the placebo group, and the placebo group mean differs from the grand mean by 0.19, then the *total* between group variation associated with these six people is $6 \times 0.16 = 1.14$. So we have to extend our little table of calculations...

group	...	squared deviations	sample size	weighted squared dev
k	...	$(\bar{Y}_k - \bar{Y})^2$	N_k	$N_k(\bar{Y}_k - \bar{Y})^2$
placebo	...	0.18	6	1.11
anxifree	...	0.03	6	0.16
joyzepam	...	0.36	6	2.18

And so now our between group sum of squares is obtained by summing these "weighted squared deviations" over all three groups in the study:

$$\begin{aligned} \mathrm{SS}_b &= 1.11 + 0.16 + 2.18 \\ &= 3.45 \end{aligned}$$

As you can see, the between group calculations are a lot shorter, so you probably wouldn't usually want to bother using R as your calculator. However, if you *did* decide to do so, here's one way you could do it:

```
> gp.means <- tapply(outcome,group,mean)
> grand.mean <- mean(outcome)
> dev.from.grand.mean <- gp.means - grand.mean
> squared.devs <- dev.from.grand.mean ^2
> gp.sizes <- tapply(outcome,group,length)
> wt.squared.devs <- gp.sizes * squared.devs
```

Again, I won't actually try to explain this code line by line, but – just like last time – there's nothing in there that we haven't seen in several places elsewhere in the book, so I'll leave it as an exercise for you to make sure you understand it. Once again, we can dump all our variables into a data frame so that we can print it out as a nice table:

```
> Y <- data.frame( gp.means, grand.mean, dev.from.grand.mean,
+                  squared.devs, gp.sizes, wt.squared.devs )
> print(Y, digits = 2)
         gp.means grand.mean dev.from.grand.mean squared.devs gp.sizes wt.squared.devs
placebo      0.45       0.88               -0.43        0.188        6            1.13
anxifree     0.72       0.88               -0.17        0.028        6            0.17
joyzepam     1.48       0.88                0.60        0.360        6            2.16
```

Clearly, these are basically the same numbers that we got before. There are a few tiny differences, but that's only because the hand-calculated versions have some small errors caused by the fact that I rounded all my numbers to 2 decimal places at each step in the calculations, whereas R only does it at the end (obviously, Rs version is more accurate). Anyway, here's the R command showing the final step:

```
> SSb <- sum( wt.squared.devs )
> print( SSb )
[1] 3.453333
```

which is (ignoring the slight differences due to rounding error) the same answer that I got when doing things by hand.

Now that we've calculated our sums of squares values, SS_b and SS_w, the rest of the ANOVA is pretty painless. The next step is to calculate the degrees of freedom. Since we have $G = 3$ groups and $N = 18$ observations in total, our degrees of freedom can be calculated by simple subtraction:

$$\begin{array}{rcccl} \text{df}_b &=& G-1 &=& 2 \\ \text{df}_w &=& N-G &=& 15 \end{array}$$

Next, since we've now calculated the values for the sums of squares and the degrees of freedom, for both the within-groups variability and the between-groups variability, we can obtain the mean square values by dividing one by the other:

$$\begin{array}{rcccccl} \text{MS}_b &=& \frac{\text{SS}_b}{\text{df}_b} &=& \frac{3.45}{2} &=& 1.73 \\ \text{MS}_w &=& \frac{\text{SS}_w}{\text{df}_w} &=& \frac{1.39}{15} &=& 0.09 \end{array}$$

We're almost done. The mean square values can be used to calculate the F-value, which is the test statistic that we're interested in. We do this by dividing the between-groups MS value by the and within-groups MS value.

$$F = \frac{\text{MS}_b}{\text{MS}_w} = \frac{1.73}{0.09} = 18.6$$

Woohooo! This is terribly exciting, yes? Now that we have our test statistic, the last step is to find out whether the test itself gives us a significant result. As discussed in Chapter 11, what we really *ought* to do is choose an α level (i.e., acceptable Type I error rate) ahead of time, construct our rejection region, etc etc. But in practice it's just easier to directly calculate the p-value. Back in the "old days", what we'd do is open up a statistics textbook or something and flick to the back section which would actually have a huge lookup table... that's how we'd "compute" our p-value, because it's too much effort to do it any other way. However, since we have access to R, I'll use the `pf()` function to do it instead. Now, remember that I explained earlier that the F-test is always one sided? And that we only reject the null hypothesis for very large F-values? That means we're only interested in the *upper tail* of the F-distribution. The command that you'd use here would be this...

```
> pf( 18.6, df1 = 2, df2 = 15, lower.tail = FALSE)
[1] 8.6727e-05
```

Therefore, our p-value comes to 0.0000867, or 8.67×10^{-5} in scientific notation. So, unless we're being *extremely* conservative about our Type I error rate, we're pretty much guaranteed to reject the null hypothesis.

At this point, we're basically done. Having completed our calculations, it's traditional to organise all these numbers into an ANOVA table like the one in Table 14.1. For our clinical trial data, the ANOVA table would look like this:

	df	sum of squares	mean squares	F-statistic	p-value
between groups	2	3.45	1.73	18.6	8.67×10^{-5}
within groups	15	1.39	0.09	-	-

These days, you'll probably never have much reason to want to construct one of these tables yourself, but you will find that almost all statistical software (R included) tends to organise the output of an ANOVA into a table like this, so it's a good idea to get used to reading them. However, although the software will output a full ANOVA table, there's almost never a good reason to include the whole table in your write up. A pretty standard way of reporting this result would be to write something like this:

One-way ANOVA showed a significant effect of drug on mood gain ($F(2, 15) = 18.6, p < .001$).

Sigh. So much work for one short sentence.

14.3 Running an ANOVA in R

I'm pretty sure I know what you're thinking after reading the last section, *especially* if you followed my advice and tried typing all the commands in yourself.... doing the ANOVA calculations yourself *sucks*. There's quite a lot of calculations that we needed to do along the way, and it would be tedious to have to do this over and over again every time you wanted to do an ANOVA. One possible solution to the problem would be to take all these calculations and turn them into some R functions yourself. You'd still have to do a lot of typing, but at least you'd only have to do it the one time: once you've created the functions, you can reuse them over and over again. However, writing your own functions is a lot of work, so this is kind of a last resort. Besides, it's much better if someone else does all the work for you...

14.3.1 Using the `aov()` function to specify your ANOVA

To make life easier for you, R provides a function called `aov()`, which – obviously – is an acronym of "Analysis Of Variance".[5] If you type `?aov` and have a look at the help documentation, you'll see that there are several arguments to the `aov()` function, but the only two that we're interested in are `formula` and `data`. As we've seen in a few places previously, the `formula` argument is what you use to specify the outcome variable and the grouping variable, and the `data` argument is what you use to specify the data frame that stores these variables. In other words, to do the same ANOVA that I laboriously calculated in the previous section, I'd use a command like this:

```
> aov( formula = mood.gain ~ drug, data = clin.trial )
```

Actually, that's not *quite* the whole story, as you'll see as soon as you look at the output from this command, which I've hidden for the moment in order to avoid confusing you. Before we go into specifics, I should point out that either of these commands will do the same thing:

```
> aov( clin.trial$mood.gain ~ clin.trial$drug )
> aov( mood.gain ~ drug, clin.trial )
```

In the first command, I didn't specify a `data` set, and instead relied on the `$` operator to tell R how to find the variables. In the second command, I dropped the argument names, which is okay in this case because `formula` is the first argument to the `aov()` function, and `data` is the second one. Regardless of how I specify the ANOVA, I can assign the output of the `aov()` function to a variable, like this for example:

```
> my.anova <- aov( mood.gain ~ drug, clin.trial )
```

This is almost always a good thing to do, because there's *lots* of useful things that we can do with the `my.anova` variable. So let's assume that it's this last command that I used to specify the ANOVA that I'm trying to run, and as a consequence I have this `my.anova` variable sitting in my workspace, waiting for me to do something with it...

14.3.2 Understanding what the `aov()` function produces

Now that we've seen how to use the `aov()` function to create `my.anova` we'd better have a look at what this variable actually is. The first thing to do is to check to see what class of variable we've created, since it's kind of interesting in this case. When we do that...

```
> class( my.anova )
[1] "aov" "lm"
```

... we discover that `my.anova` actually has *two* classes! The first class tells us that it's an `aov` (analysis of variance) object, but the second tells us that it's *also* an `lm` (linear model) object. Later on, we'll see that this reflects a pretty deep statistical relationship between ANOVA and regression (Chapter 15) and it means that any function that exists in R for dealing with regressions can also be applied to `aov` objects, which is neat; but I'm getting ahead of myself. For now, I want to note that what we've created is an `aov` object, and to also make the point that `aov` objects are actually rather complicated beasts. I *won't* be trying to explain everything about them, since it's way beyond the scope of an introductory statistics subject, but to give you a tiny hint of some of the stuff that R stores inside an `aov` object, let's ask it to print out the `names()` of all the stored quantities...

[5] Actually, it *also* provides a function called `anova()`, but that works a bit differently, so let's just ignore it for now.

```
> names( my.anova )
 [1] "coefficients"  "residuals"     "effects"
 [4] "rank"          "fitted.values" "assign"
 [7] "qr"            "df.residual"   "contrasts"
[10] "xlevels"       "call"          "terms"
[13] "model"
```

As we go through the rest of the book, I hope that a few of these will become a little more obvious to you, but right now that's going to look pretty damned opaque. That's okay. You don't need to know any of the details about it right now, and most of it you don't need at all... what you *do* need to understand is that the `aov()` function does a lot of calculations for you, not just the basic ones that I outlined in the previous sections. What this means is that it's generally a good idea to create a variable like `my.anova` that stores the output of the `aov()` function... because later on, you can use `my.anova` as an input to lots of other functions: those other functions can pull out bits and pieces from the `aov` object, and calculate various other things that you might need.

Right then. The simplest thing you can do with an `aov` object is to `print()` it out. When we do that, it shows us a few of the key quantities of interest:

```
> print( my.anova )
Call:
   aov(formula = mood.gain ~ drug, data = clin.trial)

Terms:
                  drug Residuals
Sum of Squares  3.4533    1.3917
Deg. of Freedom      2        15

Residual standard error: 0.30459
Estimated effects may be unbalanced
```

Specificially, it prints out a reminder of the command that you used when you called `aov()` in the first place, shows you the sums of squares values, the degrees of freedom, and a couple of other quantities that we're not really interested in right now. Notice, however, that R doesn't use the names "between-group" and "within-group". Instead, it tries to assign more meaningful names: in our particular example, the *between groups* variance corresponds to the effect that the `drug` has on the outcome variable; and the *within groups* variance is corresponds to the "leftover" variability, so it calls that the *residuals*. If we compare these numbers to the numbers that I calculated by hand in Section 14.2.5, you can see that they're identical... the between groups sums of squares is $\text{SS}_b = 3.45$, the within groups sums of squares is $\text{SS}_w = 1.39$, and the degrees of freedom are 2 and 15 repectively.

14.3.3 Running the hypothesis tests for the ANOVA

Okay, so we've verified that `my.anova` seems to be storing a bunch of the numbers that we're looking for, but the `print()` function didn't quite give us the output that we really wanted. Where's the F-value? The p-value? These are the most important numbers in our hypothesis test, but the `print()` function doesn't provide them. To get those numbers, we need to use a different function. Instead of asking R to `print()` out the `aov` object, we should have asked for a `summary()` of it.[6] When we do that...

```
> summary( my.anova )
            Df Sum Sq Mean Sq F value   Pr(>F)
```

[6]It's worth noting that you can get the same result by using the command `anova( my.anova )`.

```
drug          2   3.45   1.727    18.6 8.6e-05 ***
Residuals    15   1.39   0.093
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

... we get all of the key numbers that we calculated earlier. We get the sums of squares, the degrees of freedom, the mean squares, the F-statistic, and the p-value itself. These are all identical to the numbers that we calculated ourselves when doing it the long and tedious way, and it's even organised into the same kind of ANOVA table that I showed in Table 14.1, and then filled out by hand in Section 14.2.5. The only things that are even slightly different is that some of the row and column names are a bit different.

14.4 Effect size

There's a few different ways you could measure the effect size in an ANOVA, but the most commonly used measures are η^2 (**eta squared**) and partial η^2. For a one way analysis of variance they're identical to each other, so for the moment I'll just explain η^2. The definition of η^2 is actually really simple:

$$\eta^2 = \frac{\text{SS}_b}{\text{SS}_{tot}}$$

That's all it is. So when I look at the ANOVA table above, I see that $\text{SS}_b = 3.45$ and $\text{SS}_{tot} = 3.45 + 1.39 = 4.84$. Thus we get an η^2 value of

$$\eta^2 = \frac{3.45}{4.84} = 0.71$$

The interpretation of η^2 is equally straightforward: it refers to the proportion of the variability in the outcome variable (`mood.gain`) that can be explained in terms of the predictor (`drug`). A value of $\eta^2 = 0$ means that there is no relationship at all between the two, whereas a value of $\eta^2 = 1$ means that the relationship is perfect. Better yet, the η^2 value is very closely related to a squared correlation (i.e., r^2). So, if you're trying to figure out whether a particular value of η^2 is big or small, it's sometimes useful to remember that

$$\eta = \sqrt{\frac{\text{SS}_b}{\text{SS}_{tot}}}$$

can be interpreted as if it referred to the *magnitude* of a Pearson correlation. So in our drugs example, the η^2 value of .71 corresponds to an η value of $\sqrt{.71} = .84$. If we think about this as being equivalent to a correlation of about .84, we'd conclude that the relationship between `drug` and `mood.gain` is strong.

The core packages in R don't include any functions for calculating η^2. However, it's pretty straightforward to calculate it directly from the numbers in the ANOVA table. In fact, since I've already got the `SSw` and `SSb` variables lying around from my earlier calculations, I can do this:

```
> SStot <- SSb + SSw          # total sums of squares
> eta.squared <- SSb / SStot  # eta-squared value
> print( eta.squared )
[1] 0.71276
```

However, since it can be tedious to do this the long way (especially when we start running more complicated ANOVAs, such as those in Chapter 16) I've included an `etaSquared()` function in the `lsr` package which will do it for you. For now, the only argument you need to care about is `x`, which should be the `aov` object corresponding to your ANOVA. When we do this, what we get as output is this:

```
> etaSquared( x = my.anova )
        eta.sq eta.sq.part
drug 0.7127623   0.7127623
```

The output here shows two different numbers. The first one corresponds to the η^2 statistic, precisely as described above. The second one refers to "partial η^2", which is a somewhat different measure of effect size that I'll describe later. For the simple ANOVA that we've just run, they're the same number. But this won't always be true once we start running more complicated ANOVAs.

14.5 Multiple comparisons and post hoc tests

Any time you run an ANOVA with more than two groups, and you end up with a significant effect, the first thing you'll probably want to ask is which groups are actually different from one another. In our drugs example, our null hypothesis was that all three drugs (placebo, Anxifree and Joyzepam) have the exact same effect on mood. But if you think about it, the null hypothesis is actually claiming *three* different things all at once here. Specifically, it claims that:

- Your competitor's drug (Anxifree) is no better than a placebo (i.e., $\mu_A = \mu_P$)
- Your drug (Joyzepam) is no better than a placebo (i.e., $\mu_J = \mu_P$)
- Anxifree and Joyzepam are equally effective (i.e., $\mu_J = \mu_A$)

If any one of those three claims is false, then the null hypothesis is also false. So, now that we've rejected our null hypothesis, we're thinking that *at least* one of those things isn't true. But which ones? All three of these propositions are of interest: you certainly want to know if your new drug Joyzepam is better than a placebo, and it would be nice to know how well it stacks up against an existing commercial alternative (i.e., Anxifree). It would even be useful to check the performance of Anxifree against the placebo: even if Anxifree has already been extensively tested against placebos by other researchers, it can still be very useful to check that your study is producing similar results to earlier work.

When we characterise the null hypothesis in terms of these three distinct propositions, it becomes clear that there are eight possible "states of the world" that we need to distinguish between:

possibility:	is $\mu_P = \mu_A$?	is $\mu_P = \mu_J$?	is $\mu_A = \mu_J$?	which hypothesis?
1	✓	✓	✓	null
2	✓	✓		alternative
3	✓		✓	alternative
4	✓			alternative
5		✓	✓	alternative
6		✓		alternative
7			✓	alternative
8				alternative

By rejecting the null hypothesis, we've decided that we *don't* believe that #1 is the true state of the world. The next question to ask is, which of the other seven possibilities *do* we think is right? When faced with this situation, its usually helps to look at the data. For instance, if we look at the plots in Figure 14.1, it's tempting to conclude that Joyzepam is better than the placebo and better than Anxifree, but there's no real difference between Anxifree and the placebo. However, if we want to get a clearer answer about this, it might help to run some tests.

14.5.1 Running "pairwise" *t*-tests

How might we go about solving our problem? Given that we've got three separate pairs of means (placebo versus Anxifree, placebo versus Joyzepam, and Anxifree versus Joyzepam) to compare, what we could do is run three separate *t*-tests and see what happens. There's a couple of ways that we could do this. One method would be to construct new variables corresponding the groups you want to compare (e.g., `anxifree`, `placebo` and `joyzepam`), and then run a *t*-test on these new variables:

```
> t.test( anxifree, placebo, var.equal = TRUE )   # Student t-test
```

or, you could use the `subset` argument in the `t.test()` function to select only those observations corresponding to one of the two groups we're interested in:

```
> t.test( formula = mood.gain ~ drug,
+         data = clin.trial,
+         subset = drug %in% c("placebo","anxifree"),
+         var.equal = TRUE
+ )
```

See Chapter 7 if you've forgotten how the `%in%` operator works. Regardless of which version we do, R will print out the results of the *t*-test, though I haven't included that output here. If we go on to do this for all possible pairs of variables, we can look to see which (if any) pairs of groups are significantly different to each other. This "lots of *t*-tests idea" isn't a bad strategy, though as we'll see later on there are some problems with it. However, for the moment our bigger problem is that it's a *pain* to have to type in such a long command over and over again: for instance, if your experiment has 10 groups, then you have to run 45 *t*-tests. That's way too much typing.

To help keep the typing to a minimum, R provides a function called `pairwise.t.test()` that automatically runs all of the *t*-tests for you. There are three arguments that you need to specify, the outcome variable `x`, the group variable `g`, and the `p.adjust.method` argument, which "adjusts" the *p*-value in one way or another. I'll explain *p*-value adjustment in a moment, but for now we can just set `p.adjust.method = "none"` since we're not doing any adjustments. For our example, here's what we do:

```
> pairwise.t.test( x = clin.trial$mood.gain,   # outcome variable
+                  g = clin.trial$drug,        # grouping variable
+                  p.adjust.method = "none"    # which correction to use?
+ )

        Pairwise comparisons using t tests with pooled SD

data:  clin.trial$mood.gain and clin.trial$drug

         placebo anxifree
anxifree 0.15021 -
joyzepam 3e-05   0.00056

P value adjustment method: none
```

One thing that bugs me slightly about the `pairwise.t.test()` function is that you can't just give it an `aov` object, and have it produce this output. After all, I went to all that trouble earlier of getting R to create the `my.anova` variable and – as we saw in Section 14.3.2 – R has actually stored enough information inside it that I should just be able to get it to run all the pairwise tests using `my.anova` as an input. To that end, I've included a `posthocPairwiseT()` function in the `lsr` package that lets you do this. The idea behind this function is that you can just input the `aov` object itself,[7] and then get the pairwise tests as an output. As of the current writing, `posthocPairwiseT()` is actually just a simple way of calling `pairwise.t.test()` function, but you should be aware that I intend to make some changes to it later on. Here's an example:

```
> posthocPairwiseT( x = my.anova, p.adjust.method = "none" )

        Pairwise comparisons using t tests with pooled SD

data:  mood.gain and drug

         placebo anxifree
anxifree 0.15021 -
joyzepam 3e-05   0.00056

P value adjustment method: none
```

In later versions, I plan to add more functionality (e.g., adjusted confidence intervals), but for now I think it's at least kind of useful. To see why, let's suppose you've run your ANOVA and stored the results in `my.anova`, and you're happy using the Holm correction (the default method in `pairwise.t.test()`, which I'll explain this in a moment). In that case, all you have to do is type this:

```
> posthocPairwiseT( my.anova )
```

and R will output the test results. Much more convenient, I think.

14.5.2 Corrections for multiple testing

In the previous section I hinted that there's a problem with just running lots and lots of t-tests. The concern is that when running these analyses, what we're doing is going on a "fishing expedition": we're running lots and lots of tests without much theoretical guidance, in the hope that some of them come up significant. This kind of theory-free search for group differences is referred to as **post hoc analysis** ("post hoc" being Latin for "after this").[8]

It's okay to run post hoc analyses, but a lot of care is required. For instance, the analysis that I ran in the previous section is actually pretty dangerous: each *individual* t-test is designed to have a 5% Type I error rate (i.e., $\alpha = .05$), and I ran three of these tests. Imagine what would have happened if my ANOVA involved 10 different groups, and I had decided to run 45 "post hoc" t-tests to try to find out which ones were significantly different from each other, you'd expect 2 or 3 of them to come up significant *by chance alone*. As we saw in Chapter 11, the central organising principle behind null hypothesis testing is that we seek to control our Type I error rate, but now that I'm running lots of t-tests at once, in order to determine the source of my ANOVA results, my actual Type I error rate across this whole *family* of tests has gotten completely out of control.

The usual solution to this problem is to introduce an adjustment to the p-value, which aims to control the total error rate across the family of tests (see Shaffer, 1995). An adjustment of this form, which is usually (but not always) applied because one is doing post hoc analysis, is often referred to as a **correction for multiple comparisons**, though it is sometimes referred to as "simultaneous inference".

[7] I should point out that there are other functions in R for running multiple comparisons, and at least one of them works this way: the `TukeyHSD()` function takes an `aov` object as its input, and outputs Tukey's "honestly significant difference" tests. I talk about Tukey's HSD in Chapter 16.

[8] If you *do* have some theoretical basis for wanting to investigate some comparisons but not others, it's a different story. In those circumstances you're not really running "post hoc" analyses at all: you're making "planned comparisons". I do talk about this situation later in the book (Section 16.9), but for now I want to keep things simple.

In any case, there are quite a few different ways of doing this adjustment. I'll discuss a few of them in this section and in Section 16.8, but you should be aware that there are many other methods out there (see, e.g., Hsu, 1996).

14.5.3 Bonferroni corrections

The simplest of these adjustments is called the **Bonferroni correction** (Dunn, 1961), and it's very very simple indeed. Suppose that my post hoc analysis consists of m separate tests, and I want to ensure that the total probability of making *any* Type I errors at all is at most α.[9] If so, then the Bonferroni correction just says "multiply all your raw p-values by m". If we let p denote the original p-value, and let p'_j be the corrected value, then the Bonferroni correction tells that:

$$p' = m \times p$$

And therefore, if you're using the Bonferroni correction, you would reject the null hypothesis if $p' < \alpha$. The logic behind this correction is very straightforward. We're doing m different tests; so if we arrange it so that each test has a Type I error rate of at most α/m, then the *total* Type I error rate across these tests cannot be larger than α. That's pretty simple, so much so that in the original paper, the author writes:

> The method given here is so simple and so general that I am sure it must have been used before this. I do not find it, however, so can only conclude that perhaps its very simplicity has kept statisticians from realizing that it is a very good method in some situations (Dunn, 1961, pp 52-53)

To use the Bonferroni correction in R, you can use the `pairwise.t.test()` function,[10] making sure that you set `p.adjust.method = "bonferroni"`. Alternatively, since the whole reason why we're doing these pairwise tests in the first place is because we have an ANOVA that we're trying to understand, it's probably more convenient to use the `posthocPairwiseT()` function in the `lsr` package, since we can use `my.anova` as the input:

```
> posthocPairwiseT( my.anova, p.adjust.method = "bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  mood.gain and drug

         placebo anxifree
anxifree 0.4506  -
joyzepam 9.1e-05 0.0017

P value adjustment method: bonferroni
```

If we compare these three p-values to those that we saw in the previous section when we made no adjustment at all, it is clear that the only thing that R has done is multiply them by 3.

[9] It's worth noting in passing that not all adjustment methods try to do this. What I've described here is an approach for controlling "family wise Type I error rate". However, there are other post hoc tests seek to control the "false discovery rate", which is a somewhat different thing.

[10] There's also a function called `p.adjust()` in which you can input a vector of raw p-values, and it will output a vector of adjusted p-values. This can be handy sometimes. I should also note that more advanced users may wish to consider using some of the tools provided by the `multcomp` package.

14.5.4 Holm corrections

Although the Bonferroni correction is the simplest adjustment out there, it's not usually the best one to use. One method that is often used instead is the **Holm correction** (Holm, 1979). The idea behind the Holm correction is to pretend that you're doing the tests sequentially; starting with the smallest (raw) p-value and moving onto the largest one. For the j-th largest of the p-values, the adjustment is *either*

$$p'_j = j \times p_j$$

(i.e., the biggest p-value remains unchanged, the second biggest p-value is doubled, the third biggest p-value is tripled, and so on), *or*

$$p'_j = p'_{j+1}$$

whichever one is larger. This might sound a little confusing, so let's go through it a little more slowly. Here's what the Holm correction does. First, you sort all of your p-values in order, from smallest to largest. For the smallest p-value all you do is multiply it by m, and you're done. However, for all the other ones it's a two-stage process. For instance, when you move to the second smallest p value, you first multiply it by $m - 1$. If this produces a number that is bigger than the adjusted p-value that you got last time, then you keep it. But if it's smaller than the last one, then you copy the last p-value. To illustrate how this works, consider the table below, which shows the calculations of a Holm correction for a collection of five p-values:

raw p	rank j	$p \times j$	Holm p
.001	5	.005	.005
.005	4	.020	.020
.019	3	.057	.057
.022	2	.044	.057
.103	1	.103	.103

Hopefully that makes things clear.

Although it's a little harder to calculate, the Holm correction has some very nice properties: it's more powerful than Bonferroni (i.e., it has a lower Type II error rate), but – counterintuitive as it might seem – it has the *same* Type I error rate. As a consequence, in practice there's never any reason to use the simpler Bonferroni correction, since it is always outperformed by the slightly more elaborate Holm correction. Because of this, the Holm correction is the default one used by `pairwise.t.test()` and `posthocPairwiseT()`. To run the Holm correction in R, you could specify `p.adjust.method = "Holm"` if you wanted to, but since it's the default you can just to do this:

```
> posthocPairwiseT( my.anova )

         Pairwise comparisons using t tests with pooled SD

data:  mood.gain and drug

          placebo anxifree
anxifree 0.1502  -
joyzepam 9.1e-05 0.0011

P value adjustment method: holm
```

As you can see, the biggest p-value (corresponding to the comparison between Anxifree and the placebo) is unaltered: at a value of .15, it is exactly the same as the value we got originally when we applied no correction at all. In contrast, the smallest p-value (Joyzepam versus placebo) has been multiplied by three.

14.5.5 Writing up the post hoc test

Finally, having run the post hoc analysis to determine which groups are significantly different to one another, you might write up the result like this:

> Post hoc tests (using the Holm correction to adjust p) indicated that Joyzepam produced a significantly larger mood change than both Anxifree ($p = .001$) and the placebo ($p = 9.1 \times 10^{-5}$). We found no evidence that Anxifree performed better than the placebo ($p = .15$).

Or, if you don't like the idea of reporting exact p-values, then you'd change those numbers to $p < .01$, $p < .001$ and $p > .05$ respectively. Either way, the key thing is that you indicate that you used Holm's correction to adjust the p-values. And of course, I'm assuming that elsewhere in the write up you've included the relevant descriptive statistics (i.e., the group means and standard deviations), since these p-values on their own aren't terribly informative.

14.6 Assumptions of one-way ANOVA

Like any statistical test, analysis of variance relies on some assumptions about the data. There are three key assumptions that you need to be aware of: *normality*, *homogeneity of variance* and *independence*. If you remember back to Section 14.2.4 – which I hope you at least skimmed even if you didn't read the whole thing – I described the statistical models underpinning ANOVA, which I wrote down like this:

$$\begin{array}{lrcl} H_0: & Y_{ik} & = & \mu + \epsilon_{ik} \\ H_1: & Y_{ik} & = & \mu_k + \epsilon_{ik} \end{array}$$

In these equations μ refers to a single, grand population mean which is the same for all groups, and μ_k is the population mean for the k-th group. Up to this point we've been mostly interested in whether our data are best described in terms of a single grand mean (the null hypothesis) or in terms of different group-specific means (the alternative hypothesis). This makes sense, of course: that's actually the important research question! However, all of our testing procedures have – implicitly – relied on a specific assumption about the residuals, ϵ_{ik}, namely that

$$\epsilon_{ik} \sim \text{Normal}(0, \sigma^2)$$

None of the maths works properly without this bit. Or, to be precise, you can still do all the calculations, and you'll end up with an F-statistic, but you have no guarantee that this F-statistic actually measures what you think it's measuring, and so any conclusions that you might draw on the basis of the F test might be wrong.

So, how do we check whether this assumption about the residuals is accurate? Well, as I indicated above, there are three distinct claims buried in this one statement, and we'll consider them separately.

- **Normality**. The residuals are assumed to be normally distributed. As we saw in Section 13.9, we can assess this by looking at QQ plots or running a Shapiro-Wilk test. I'll talk about this in an ANOVA context in Section 14.9.
- **Homogeneity of variance**. Notice that we've only got the one value for the population standard deviation (i.e., σ), rather than allowing each group to have it's own value (i.e., σ_k). This is referred to as the homogeneity of variance (sometimes called homoscedasticity) assumption. ANOVA assumes that the population standard deviation is the same for all groups. We'll talk about this extensively in Section 14.7.

- **Independence**. The independence assumption is a little trickier. What it basically means is that, knowing one residual tells you nothing about any other residual. All of the ϵ_{ik} values are assumed to have been generated without any "regard for" or "relationship to" any of the other ones. There's not an obvious or simple way to test for this, but there are some situations that are clear violations of this: for instance, if you have a repeated-measures design, where each participant in your study appears in more than one condition, then independence doesn't hold; there's a special relationship between some observations... namely those that correspond to the same person! When that happens, you need to use something like repeated measures ANOVA. I don't currently talk about repeated measures ANOVA in this book, but it will be included in later versions.

14.6.1 How robust is ANOVA?

One question that people often want to know the answer to is the extent to which you can trust the results of an ANOVA if the assumptions are violated. Or, to use the technical language, how **robust** is ANOVA to violations of the assumptions. Due to deadline constraints I don't have the time to discuss this topic. This is a topic I'll cover in some detail in a later version of the book.

14.7 Checking the homogeneity of variance assumption

There's more than one way to skin a cat, as the saying goes, and more than one way to test the homogeneity of variance assumption, too (though for some reason no-one made a saying out of that). The most commonly used test for this that I've seen in the literature is the **Levene test** (Levene, 1960), and the closely related **Brown-Forsythe test** (Brown & Forsythe, 1974), both of which I'll describe here. Alternatively, you could use the Bartlett test, which is implemented in R via the `bartlett.test()` function, but I'll leave it as an exercise for the reader to go check that one out if you're interested.

Levene's test is shockingly simple. Suppose we have our outcome variable Y_{ik}. All we do is define a new variable, which I'll call Z_{ik}, corresponding to the absolute deviation from the group mean:

$$Z_{ik} = \left| Y_{ik} - \bar{Y}_k \right|$$

Okay, what good does this do us? Well, let's take a moment to think about what Z_{ik} actually is, and what we're trying to test. The value of Z_{ik} is a measure of how the i-th observation in the k-th group deviates from its group mean. And our null hypothesis is that all groups have the same variance; that is, the same overall deviations from the group means! So, the null hypothesis in a Levene's test is that the population means of Z are identical for all groups. Hm. So what we need now is a statistical test of the null hypothesis that all group means are identical. Where have we seen that before? Oh right, that's what ANOVA is... and so all that the Levene's test does is run an ANOVA on the new variable Z_{ik}.

What about the Brown-Forsythe test? Does that do anything particularly different? Nope. The only change from the Levene's test is that it constructs the transformed variable Z in a slightly different way, using deviations from the group *medians* rather than deviations from the group *means*. That is, for the Brown-Forsythe test,

$$Z_{ik} = |Y_{ik} - \text{median}_k(Y)|$$

where $\text{median}_k(Y)$ is the median for group k. Regardless of whether you're doing the standard Levene test or the Brown-Forsythe test, the test statistic – which is sometimes denoted F, but sometimes written as W – is calculated in exactly the same way that the F-statistic for the regular ANOVA is calculated, just using a Z_{ik} rather than Y_{ik}. With that in mind, let's just move on and look at how to run the test in R.

14.7.1 Running the Levene's test in R

Okay, so how do we run the Levene test? Obviously, since the Levene test is just an ANOVA, it would be easy enough to manually create the transformed variable Z_{ik} and then use the `aov()` function to run an ANOVA on that. However, that's the tedious way to do it. A better way to do run your Levene's test is to use the `leveneTest()` function, which is in the `car` package. As usual, we first load the package

```
> library( car )
```

and now that we have, we can run our Levene test. The main argument that you need to specify is `y`, but you can do this in lots of different ways. Probably the simplest way to do it is actually input the original `aov` object. Since I've got the `my.anova` variable stored from my original ANOVA, I can just do this:

```
> leveneTest( my.anova )
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2    1.47   0.26
      15
```

If we look at the output, we see that the test is non-significant ($F_{2,15} = 1.47, p = .26$), so it looks like the homogeneity of variance assumption is fine. Remember, although R reports the test statistic as an F-value, it could equally be called W, in which case you'd just write $W_{2,15} = 1.47$. Also, note the part of the output that says `center = median`. That's telling you that, by default, the `leveneTest()` function actually does the Brown-Forsythe test. If you want to use the mean instead, then you need to explicitly set the `center` argument, like this:

```
> leveneTest( y = my.anova, center = mean )
Levene's Test for Homogeneity of Variance (center = mean)
      Df F value Pr(>F)
group  2    1.45   0.27
      15
```

That being said, in most cases it's probably best to stick to the default value, since the Brown-Forsythe test is a bit more robust than the original Levene test.

14.7.2 Additional comments

Two more quick comments before I move onto a different topic. Firstly, as mentioned above, there are other ways of calling the `leveneTest()` function. Although the vast majority of situations that call for a Levene test involve checking the assumptions of an ANOVA (in which case you probably have a variable like `my.anova` lying around), sometimes you might find yourself wanting to specify the variables directly. Two different ways that you can do this are shown below:

```
> leveneTest(y = mood.gain ~ drug, data = clin.trial)   # y is a formula in this case
> leveneTest(y = clin.trial$mood.gain, group = clin.trial$drug)   # y is the outcome
```

Secondly, I did mention that it's possible to run a Levene test just using the `aov()` function. I don't want to waste a lot of space on this, but just in case some readers are interested in seeing how this is done, here's the code that creates the new variables and runs an ANOVA. If you are interested, feel free to run this to verify that it produces the same answers as the Levene test (i.e., with `center = mean`):

```
> Y <- clin.trial $ mood.gain    # the original outcome variable, Y
> G <- clin.trial $ drug         # the grouping variable, G
> gp.mean <- tapply(Y, G, mean)  # calculate group means
> Ybar <- gp.mean[G]             # group mean associated with each obs
> Z <- abs(Y - Ybar)             # the transformed variable, Z
> summary( aov(Z ~ G) )          # run the ANOVA
```

That said, I don't imagine that many people will care about this. Nevertheless, it's nice to know that you could do it this way if you wanted to. And for those of you who do try it, I think it helps to demystify the test a little bit when you can see – with your own eyes – the way in which Levene's test relates to ANOVA.

14.8 Removing the homogeneity of variance assumption

In our example, the homogeneity of variance assumption turned out to be a pretty safe one: the Levene test came back non-significant, so we probably don't need to worry. However, in real life we aren't always that lucky. How do we save our ANOVA when the homogeneity of variance assumption is violated? If you recall from our discussion of *t*-tests, we've seen this problem before. The Student *t*-test assumes equal variances, so the solution was to use the Welch *t*-test, which does not. In fact, Welch (1951) also showed how we can solve this problem for ANOVA too (the **Welch one-way test**). It's implemented in R using the `oneway.test()` function. The arguments that we'll need for our example are:

- `formula`. This is the model formula, which (as usual) needs to specify the outcome variable on the left hand side and the grouping variable on the right hand side: i.e., something like `outcome ~ group`.
- `data`. Specifies the data frame containing the variables.
- `var.equal`. If this is `FALSE` (the default) a Welch one-way test is run. If it is `TRUE` then it just runs a regular ANOVA.

The function also has a `subset` argument that lets you analyse only some of the observations and a `na.action` argument that tells it how to handle missing data, but these aren't necessary for our purposes. So, to run the Welch one-way ANOVA for our example, we would do this:

```
> oneway.test(mood.gain ~ drug, data = clin.trial)

        One-way analysis of means (not assuming equal variances)

data:  mood.gain and drug
F = 26.322, num df = 2.000, denom df = 9.493, p-value = 0.000134
```

To understand what's happening here, let's compare these numbers to what we got earlier in Section 14.3 when we ran our original ANOVA. To save you the trouble of flicking back, here are those numbers again, this time calculated by setting `var.equal = TRUE` for the `oneway.test()` function:

```
> oneway.test(mood.gain ~ drug, data = clin.trial, var.equal = TRUE)

        One-way analysis of means

data:  mood.gain and drug
F = 18.611, num df = 2, denom df = 15, p-value = 8.646e-05
```

Okay, so originally our ANOVA gave us the result $F(2, 15) = 18.6$, whereas the Welch one-way test gave us $F(2, 9.49) = 26.32$. In other words, the Welch test has reduced the within-groups degrees of freedom from 15 to 9.49, and the F-value has increased from 18.6 to 26.32.

14.9 Checking the normality assumption

Testing the normality assumption is relatively straightforward. We covered most of what you need to know in Section 13.9. The only thing we really need to know how to do is pull out the residuals (i.e., the ϵ_{ik} values) so that we can draw our QQ plot and run our Shapiro-Wilk test. First, let's extract the residuals. R provides a function called `residuals()` that will do this for us. If we pass our `my.anova` to this function, it will return the residuals. So let's do that:

```
> my.anova.residuals <- residuals( object = my.anova )   # extract the residuals
```

We can print them out too, though it's not exactly an edifying experience. In fact, given that I'm on the verge of putting *myself* to sleep just typing this, it might be a good idea to skip that step. Instead, let's draw some pictures and run ourselves a hypothesis test:

```
> hist( x = my.anova.residuals )           # plot a histogram (similar to Figure 14.3a)
> qqnorm( y = my.anova.residuals )         # draw a QQ plot (similar to Figure 14.3b)
> shapiro.test( x = my.anova.residuals )   # run Shapiro-Wilk test

        Shapiro-Wilk normality test

data:  my.anova.residuals
W = 0.9602, p-value = 0.6053
```

The histogram and QQ plot are both shown in Figure 14.3, and both look pretty normal to me.[11] This is supported by the results of our Shapiro-Wilk test ($W = .96$, $p = .61$) which finds no indication that normality is violated.

14.10 Removing the normality assumption

Now that we've seen how to check for normality, we are led naturally to ask what we can do to address violations of normality. In the context of a one-way ANOVA, the easiest solution is probably to switch to a non-parametric test (i.e., one that doesn't rely on any particular assumption about the kind of distribution involved). We've seen non-parametric tests before, in Chapter 13: when you only have two groups, the Wilcoxon test provides the non-parametric alternative that you need. When you've got three or more groups, you can use the **Kruskal-Wallis rank sum test** (Kruskal & Wallis, 1952). So that's the test we'll talk about next.

[11] Note that neither of these figures has been tidied up at all: if you want to create nicer looking graphs it's always a good idea to use the tools from Chapter 6 to help you draw cleaner looking images.

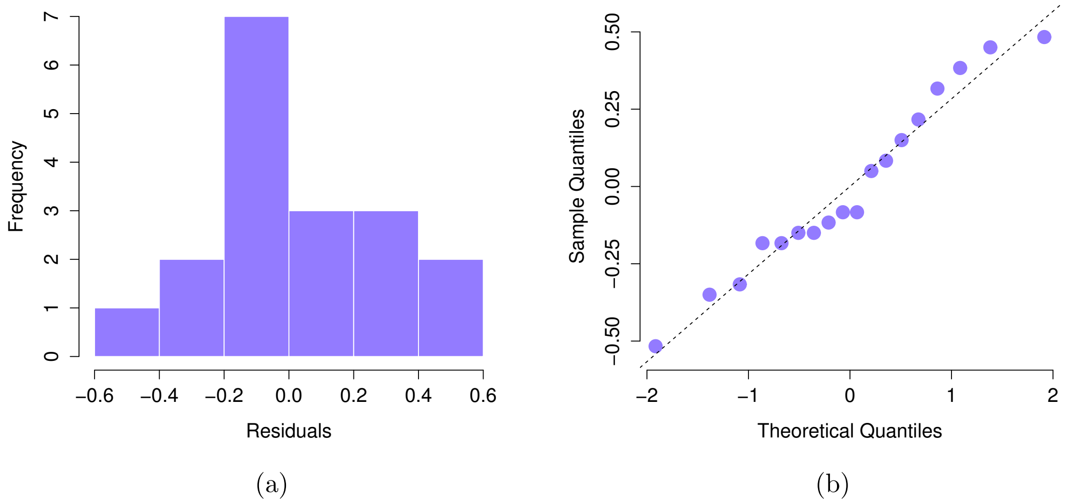


Figure 14.3: Histogram (panel a) and QQ plot (panel b) for the residuals of our ANOVA: both of these are in agreement with the Shapiro-Wilk test, and suggest that the residuals are normally distributed

14.10.1 The logic behind the Kruskal-Wallis test

The Kruskal-Wallis test is surprisingly similar to ANOVA, in some ways. In ANOVA, we started with Y_{ik}, the value of the outcome variable for the ith person in the kth group. For the Kruskal-Wallis test, what we'll do is rank order all of these Y_{ik} values, and conduct our analysis on the ranked data. So let's let R_{ik} refer to the ranking given to the ith member of the kth group. Now, let's calculate $\bar{R}_k$, the average rank given to observations in the kth group:

$$\bar{R}_k = \frac{1}{N_K} \sum_i R_{ik}$$

and let's also calculate $\bar{R}$, the grand mean rank:

$$\bar{R} = \frac{1}{N} \sum_i \sum_k R_{ik}$$

Now that we've done this, we can calculate the squared deviations from the grand mean rank $\bar{R}$. When we do this for the individual scores – i.e., if we calculate $(R_{ik} - \bar{R})^2$ – what we have is a "nonparametric" measure of how far the ik-th observation deviates from the grand mean rank. When we calculate the squared deviation of the group means from the grand means – i.e., if we calculate $(\bar{R}_k - \bar{R})^2$ – then what we have is a nonparametric measure of how much the *group* deviates from the grand mean rank. With this in mind, let's follow the same logic that we did with ANOVA, and define our *ranked* sums of squares measures in much the same way that we did earlier. First, we have our "total ranked sums of squares":

$$\mathrm{RSS}_{tot} = \sum_k \sum_i (R_{ik} - \bar{R})^2$$

and we can define the "between groups ranked sums of squares" like this:

$$\begin{array}{rcl} \mathrm{RSS}_b &=& \sum_k \sum_i (\bar{R}_k - \bar{R})^2 \\ &=& \sum_k N_k (\bar{R}_k - \bar{R})^2 \end{array}$$

So, if the null hypothesis is true and there are no true group differences at all, you'd expect the between group rank sums RSS_b to be very small, much smaller than the total rank sums RSS_{tot}. Qualitatively this is very much the same as what we found when we went about constructing the ANOVA F-statistic; but for technical reasons the Kruskal-Wallis test statistic, usually denoted K, is constructed in a slightly different way:

$$K = (N-1) \times \frac{\text{RSS}_b}{\text{RSS}_{tot}}$$

and, if the null hypothesis is true, then the sampling distribution of K is *approximately* chi-square with $G-1$ degrees of freedom (where G is the number of groups). The larger the value of K, the less consistent the data are with null hypothesis, so this is a one-sided test: we reject H_0 when K is sufficiently large.

14.10.2 Additional details

The description in the previous section illustrates the logic behind the Kruskal-Wallis test. At a conceptual level, this is the right way to think about how the test works. However, from a purely mathematical perspective it's needlessly complicated. I won't show you the derivation, but you can use a bit of algebraic jiggery-pokery[12] to show that the equation for K can be rewritten as

$$K = \frac{12}{N(N-1)} \sum_k N_k \bar{R}_k^2 - 3(N+1)$$

It's this last equation that you sometimes see given for K. This is way easier to calculate than the version I described in the previous section, it's just that it's totally meaningless to actual humans. It's probably best to think of K the way I described it earlier... as an analogue of ANOVA based on ranks. But keep in mind that the test statistic that gets calculated ends up with a rather different look to it than the one we used for our original ANOVA.

But wait, there's more! Dear lord, why is there always *more*? The story I've told so far is only actually true when there are no ties in the raw data. That is, if there are no two observations that have exactly the same value. If there *are* ties, then we have to introduce a correction factor to these calculations. At this point I'm assuming that even the most diligent reader has stopped caring (or at least formed the opinion that the tie-correction factor is something that doesn't require their immediate attention). So I'll very quickly tell you how it's calculated, and omit the tedious details about *why* it's done this way. Suppose we construct a frequency table for the raw data, and let f_j be the number of observations that have the j-th unique value. This might sound a bit abstract, so here's the R code showing a concrete example:

```
> f <- table( clin.trial$mood.gain )   # frequency table for mood gain
> print(f)    # we have some ties ...

0.1 0.2 0.3 0.4 0.5 0.6 0.8 0.9 1.1 1.2 1.3 1.4 1.7 1.8
  1   1   2   1   1   2   1   1   1   1   2   2   1   1
```

Looking at this table, notice that the third entry in the frequency table has a value of 2. Since this corresponds to a `mood.gain` of 0.3, this table is telling us that two people's mood increased by 0.3. More to the point, note that we can say that `f[3]` has a value of `2`. Or, in the mathematical notation I introduced above, this is telling us that $f_3 = 2$. Yay. So, now that we know this, the tie correction factor (TCF) is:

$$\text{TCF} = 1 - \frac{\sum_j {f_j}^3 - f_j}{N^3 - N}$$

[12] A technical term.

The tie-corrected value of the Kruskal-Wallis statistic obtained by dividing the value of K by this quantity: it is this tie-corrected version that R calculates. And at long last, we're actually finished with the theory of the Kruskal-Bloody-Wallis test. I'm sure you're all terribly relieved that I've cured you of the existential anxiety that naturally arises when you realise that you *don't* know how to calculate the tie-correction factor for the Kruskal-Wallis test. Right?

14.10.3 How to run the Kruskal-Wallis test in R

Despite the horror that we've gone through in trying to understand what the Kruskal-Wallis test actually does, it turns out that running the test is pretty painless, since R has a function called `kruskal.test()`. The function is pretty flexible, and allows you to input your data in a few different ways. Most of the time you'll have data like the `clin.trial` data set, in which you have your outcome variable `mood.gain`, and a grouping variable `drug`. If so, you can call the `kruskal.test()` function by specifying a formula, and a data frame:

```
> kruskal.test(mood.gain ~ drug, data = clin.trial)

        Kruskal-Wallis rank sum test

data:  mood.gain by drug
Kruskal-Wallis chi-squared = 12.076, df = 2, p-value = 0.002386
```

A second way of using the `kruskal.test()` function, which you probably won't have much reason to use, is to directly specify the outcome variable and the grouping variable as separate input arguments, `x` and `g`:

```
> kruskal.test(x = clin.trial$mood.gain, g = clin.trial$drug)
```

This isn't very interesting, since it's just plain easier to specify a formula. However, sometimes it can be useful to specify `x` as a list. What I mean is this. Suppose you actually had data as three separate variables, `placebo`, `anxifree` and `joyzepam`. If that's the format that your data are in, then it's convenient to know that you can bundle all three together as a list:

```
> mood.gain <- list( placebo, joyzepam, anxifree )
> kruskal.test( x = mood.gain )
```

And again, this would give you exactly the same results as the command we tried originally.

14.11 On the relationship between ANOVA and the Student t test

There's one last thing I want to point out before finishing. It's something that a lot of people find kind of surprising, but it's worth knowing about: an ANOVA with two groups is identical to the Student t-test. No, really. It's not just that they are similar, but they are actually equivalent in every meaningful way. I won't try to prove that this is always true, but I will show you a single concrete demonstration. Suppose that, instead of running an ANOVA on our `mood.gain ~ drug` model, let's instead do it using `therapy` as the predictor. If we run this ANOVA, here's what we get:

```
> summary( aov( mood.gain ~ therapy, data = clin.trial ))
            Df Sum Sq Mean Sq F value Pr(>F)
therapy      1   0.47   0.467    1.71   0.21
Residuals   16   4.38   0.274
```

Overall, it looks like there's no significant effect here at all but, as we'll see in Chapter 16 this is actually a misleading answer! In any case, it's irrelevant to our current goals: our interest here is in the F-statistic, which is $F(1,16) = 1.71$, and the p-value, which is .21. Since we only have two groups, I didn't actually need to resort to an ANOVA, I could have just decided to run a Student t-test. So let's see what happens when I do that:

```
> t.test( mood.gain ~ therapy, data = clin.trial, var.equal = TRUE )

        Two Sample t-test

data:  mood.gain by therapy
t = -1.3068, df = 16, p-value = 0.2098
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.84495  0.20051
sample estimates:
mean in group none  mean in group CBT
           0.72222            1.04444
```

Curiously, the p-values are identical: once again we obtain a value of $p = .21$. But what about the test statistic? Having run a t-test instead of an ANOVA, we get a somewhat different answer, namely $t(16) = -1.3068$. However, there is a fairly straightforward relationship here. If we square the t-statistic

```
> 1.3068 ^ 2
[1] 1.7077
```

we get the F-statistic from before.

14.12 Summary

There's a fair bit covered in this chapter, but there's still a lot missing. Most obviously, I haven't yet discussed any analog of the paired samples t-test for more than two groups. There is a way of doing this, known as *repeated measures ANOVA*, which will appear in a later version of this book. I also haven't discussed how to run an ANOVA when you are interested in more than one grouping variable, but that will be discussed in a lot of detail in Chapter 16. In terms of what we have discussed, the key topics were:

- The basic logic behind how ANOVA works (Section 14.2) and how to run one in R(Section 14.3).
- How to compute an effect size for an ANOVA (Section 14.4)
- Post hoc analysis and corrections for multiple testing (Section 14.5).
- The assumptions made by ANOVA (Section 14.6).
- How to check the homogeneity of variance assumption (Section 14.7) and what to do if it is violated (Section 14.8).
- How to check the normality assumption (Section 14.9) and what to do if it is violated (Section 14.10).

As with all of the chapters in this book, there are quite a few different sources that I've relied upon, but the one stand-out text that I've been most heavily influenced by is Sahai and Ageel (2000). It's not a good book for beginners, but it's an excellent book for more advanced readers who are interested in understanding the mathematics behind ANOVA.