16.6: ANOVA As a Linear Model

One of the most important things to understand about ANOVA and regression is that they're basically the same thing. On the surface of it, you wouldn't think that this is true: after all, the way that I've described them so far suggests that ANOVA is primarily concerned with testing for group differences, and regression is primarily concerned with understanding the correlations between variables. And as far as it goes, that's perfectly true. But when you look under the hood, so to speak, the underlying mechanics of ANOVA and regression are awfully similar. In fact, if you think about it, you've already seen evidence of this. ANOVA and regression both rely heavily on sums of squares (SS), both make use of F tests, and so on. Looking back, it's hard to escape the feeling that Chapters 14 and 15 were a bit repetitive.

The reason for this is that ANOVA and regression are both kinds of ***linear models***. In the case of regression, this is kind of obvious. The regression equation that we use to define the relationship between predictors and outcomes *is* the equation for a straight line, so it's quite obviously a linear model. And if that wasn't a big enough clue, the simple fact that the command to run a regression is `lm()` is kind of a hint too. When we use an R formula like `outcome ~ predictor1 + predictor2` what we're really working with is the somewhat uglier linear model:

$$Y_p = b_1 X_{1p} + b_2 X_{2p} + b_0 + \epsilon_p$$

where Y_p is the outcome value for the p-th observation (e.g., p-th person), X_{1p} is the value of the first predictor for the p-th observation, X_{2p} is the value of the second predictor for the p-th observation, the b_1, b_2 and b_0 terms are our regression coefficients, and ϵ_p is the p-th residual. If we ignore the residuals ϵ_p and just focus on the regression line itself, we get the following formula:

$$\hat{Y}_p = b_1 X_{1p} + b_2 X_{2p} + b_0$$

where $\hat{Y}_p$ is the value of Y that the regression line predicts for person p, as opposed to the actually-observed value Y_p. The thing that isn't immediately obvious is that we can write ANOVA as a linear model as well. However, it's actually pretty straightforward to do this. Let's start with a really simple example: rewriting a 2×2 factorial ANOVA as a linear model.

16.6.1 Some data

To make things concrete, let's suppose that our outcome variable is the `grade` that a student receives in my class, a ratio-scale variable corresponding to a mark from 0% to 100%. There are two predictor variables of interest: whether or not the student turned up to lectures (the `attend` variable), and whether or not the student actually read the textbook (the `reading` variable). We'll say that `attend = 1` if the student attended class, and `attend = 0` if they did not. Similarly, we'll say that `reading = 1` if the student read the textbook, and `reading = 0` if they did not.

Okay, so far that's simple enough. The next thing we need to do is to wrap some maths around this (sorry!). For the purposes of this example, let Y_p denote the `grade` of the p-th student in the class. This is not quite the same notation that we used earlier in this chapter: previously, we've used the notation Y_{rci} to refer to the i-th person in the r-th group for predictor 1 (the row factor) and the c-th group for predictor 2 (the column factor). This extended notation was really handy for describing how the SS values are calculated, but it's a pain in the current context, so I'll switch notation here. Now, the Yp notation is visually simpler than Y_{rci}, but it has the shortcoming that it doesn't actually keep track of the group memberships! That is, if I told you that $Y_{0,0,3}=35$, you'd immediately know that we're talking about a student (the 3rd such student, in fact) who didn't attend the lectures (i.e., `attend = 0`) and didn't read the textbook (i.e. `reading = 0`), and who ended up failing the class (`grade = 35`). But if I tell you that $Y_p=35$ all you know is that the p-th student didn't get a good grade. We've lost some key information here. Of course, it doesn't take a lot of thought to figure out how to fix this: what we'll do instead is introduce two new variables X_{1p} and X_{2p} that keep track of this information. In the case of our hypothetical student, we know that $X_{1p}=0$ (i.e., `attend = 0`) and $X_{2p}=0$ (i.e., `reading = 0`). So the data might look like this:

```
knitr::kable(tibble::tribble(
       ~V1,   ~V2,     ~V3,     ~V4,
       "1",  "90",     "1",     "1",
       "2",  "87",     "1",     "1",
       "3",  "75",     "0",     "1",
       "4",  "60",     "1",     "0",
       "5",  "35",     "0",     "0",
       "6",  "50",     "0",     "0",
       "7",  "65",     "1",     "0",
       "8",  "70",     "0",     "1"),
col.names= c("person $p$", "grade $Y_p$", "attendance $X_{1p}$", "reading $X_{2p}$"),
```

person p	grade Y_p	attendance X_{1p}	reading X_{2p}
5	35	0	0
6	50	0	0
4	60	1	0
7	65	1	0
8	70	0	1
3	75	0	1
2	87	1	1
1	90	1	1

This isn't anything particularly special, of course: it's exactly the format in which we expect to see our data! In other words, if your data have been stored as a data frame in R then you're probably expecting to see something that looks like the `rtfm.1` data frame:

```
load("./rbook-master/data/rtfm.rdata")
rtfm.1
```

```
##   grade attend reading
## 1    90      1       1
## 2    87      1       1
## 3    75      0       1
## 4    60      1       0
## 5    35      0       0
## 6    50      0       0
## 7    65      1       0
## 8    70      0       1
```

Well, sort of. I suspect that a few readers are probably frowning a little at this point. Earlier on in the book I emphasised the importance of converting nominal scale variables such as `attend` and `reading` to factors, rather than encoding them as numeric variables. The `rtfm.1` data frame doesn't do this, but the `rtfm.2` data frame does, and so you might instead be expecting to see data like this:

```
rtfm.2
```

```
##   grade attend reading
## 1    90    yes     yes
## 2    87    yes     yes
## 3    75     no     yes
## 4    60    yes      no
## 5    35     no      no
## 6    50     no      no
## 7    65    yes      no
## 8    70     no     yes
```

However, for the purposes of this section it's important that we be able to switch back and forth between these two different ways of thinking about the data. After all, our goal in this section is to look at some of the mathematics that underpins ANOVA, and if we want to do that we need to be able to see the numerical representation of the data (in `rtfm.1`) as well as the more meaningful factor representation (in `rtfm.2`). In any case, we can use the `xtabs()` function to confirm that this data set corresponds to a balanced design

```
xtabs( ~ attend + reading, rtfm.2 )
```

```
##       reading
## attend no yes
##    no   2   2
##    yes  2   2
```

For each possible combination of the `attend` and `reading` variables, we have exactly two students. If we're interested in calculating the mean `grade` for each of these cells, we can use the `aggregate()` function:

```
aggregate( grade ~ attend + reading, rtfm.2, mean )
```

```
##   attend reading grade
## 1     no      no  42.5
## 2    yes      no  62.5
## 3     no     yes  72.5
## 4    yes     yes  88.5
```

Looking at this table, one gets the strong impression that reading the text and attending the class both matter a lot.

16.6.2 ANOVA with binary factors as a regression model

Okay, let's get back to talking about the mathematics. We now have our data expressed in terms of three numeric variables: the continuous variable Y, and the two binary variables X_1 and X_2. What I want to you to recognise is that our 2$$2 factorial ANOVA is *exactly* equivalent to the regression model

$$Y_p = b_1 X_{1p} + b_2 X_{2p} + b_0 + \epsilon_p$$

This is, of course, the exact same equation that I used earlier to describe a two-predictor regression model! The only difference is that X_1 and X_2 are now *binary* variables (i.e., values can only be 0 or 1), whereas in a regression analysis we expect that X_1 and X_2 will be continuous. There's a couple of ways I could try to convince you of this. One possibility would be to do a lengthy mathematical exercise, proving that the two are identical. However, I'm going to go out on a limb and guess that most of the readership of this book will find that to be annoying rather than helpful. Instead, I'll explain the basic ideas, and then rely on R to show that that ANOVA analyses and regression analyses aren't just similar, they're identical for all intents and purposes.[239] Let's start by running this as an ANOVA. To do this, we'll use the `rtfm.2` data frame, since that's the one in which I did the proper thing of coding `attend` and `reading` as factors, and I'll use the `aov()` function to do the analysis. Here's what we get…

```
anova.model <- aov( grade ~ attend + reading, data = rtfm.2 )
 summary( anova.model )
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## attend       1    648     648   21.60 0.00559 **
## reading      1   1568    1568   52.27 0.00079 ***
## Residuals    5    150      30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, by reading the key numbers off the ANOVA table and the table of means that we presented earlier, we can see that the students obtained a higher grade if they attended class ($F_{1,5}$=26.1,p=.0056) and if they read the textbook ($F_{1,5}$=52.3,p=.0008). Let's make a note of those p-values and those F statistics.

```
library(effects)
 Effect( c("attend","reading"), anova.model )
```

```
##
##  attend*reading effect
##       reading
## attend   no  yes
##    no  43.5 71.5
##    yes 61.5 89.5
```

Now let's think about the same analysis from a linear regression perspective. In the `rtfm.1` data set, we have encoded `attend` and `reading` as if they were numeric predictors. In this case, this is perfectly acceptable. There really is a sense in which a student who turns up to class (i.e. `attend = 1`) has in fact done "more attendance" than a student who does not (i.e. `attend = 0`). So it's not at all unreasonable to include it as a predictor in a regression model. It's a little unusual, because the predictor only takes on two possible values, but it doesn't violate any of the assumptions of linear regression. And it's easy to interpret. If the regression coefficient for `attend` is greater than 0, it means that students that attend lectures get higher grades; if it's less than zero, then students attending lectures get lower grades. The same is true for our `reading` variable.

Wait a second… *why* is this true? It's something that is intuitively obvious to everyone who has taken a few stats classes and is comfortable with the maths, but it *isn't* clear to everyone else at first pass. To see why this is true, it helps to look closely at a few specific students. Let's start by considering the 6th and 7th students in our data set (i.e. p=6 and p=7). Neither one has read the textbook, so in both cases we can set `reading = 0` . Or, to say the same thing in our mathematical notation, we observe $X_{2,6}$=0 and $X_{2,7}$=0. However, student number 7 did turn up to lectures (i.e., `attend = 1` , $X_{1,7}$=1) whereas student number 6 did not (i.e., `attend = 0` , $X_{1,6}$=0). Now let's look at what happens when we insert these numbers into the general formula for our regression line. For student number 6, the regression predicts that

$$\begin{aligned} Y_6 &= b_1 X_{1,6} + b_2 X_{2,6} + b_0 \\ &= (b_1 \times 0) + (b_2 \times 0) + b_0 \\ &= b_0 \end{aligned}$$

So we're expecting that this student will obtain a grade corresponding to the value of the intercept term b0. What about student 7? This time, when we insert the numbers into the formula for the regression line, we obtain the following:

$$\begin{aligned} Y_6 &= b_1 X_{1,7} + b_2 X_{2,7} + b_0 \\ &= (b_1 \times 1) + (b_2 \times 0) + b_0 \\ &= b_1 + b_0 \end{aligned}$$

Because this student attended class, the predicted grade is equal to the intercept term b_0 *plus* the coefficient associated with the `attend` variable, b_1. So, if b_1 is greater than zero, we're expecting that the students who turn up to lectures will get higher grades than those students who don't. If this coefficient is negative, we're expecting the opposite: students who turn up at class end

up performing much worse. In fact, we can push this a little bit further. What about student number 1, who turned up to class ($X_{1,1}$=1) *and* read the textbook ($X_{2,1}$=1)? If we plug these numbers into the regression, we get

$$\begin{aligned} Y_6 &= b_1 X_{1,1} + b_2 X_{2,1} + b_0 \\ &= (b_1 \times 1) + (b_2 \times 1) + b_0 \\ &= b_1 + b_2 + b_0 \end{aligned}$$

So if we assume that attending class helps you get a good grade (i.e., b1>0) and if we assume that reading the textbook also helps you get a good grade (i.e., b_2>0), then our expectation is that student 1 will get a grade that that is higher than student 6 and student 7.

And at this point, you won't be at all suprised to learn that the regression model predicts that student 3, who read the book but didn't attend lectures, will obtain a grade of b_2+b_0. I won't bore you with yet another regression formula. Instead, what I'll do is show you the following table of *expected grades*:

```
knitr::kable(tibble::tribble(
               ~V1,           ~V2,   ~V3,

   "attended - no","$b_0$","$b_0 + b_2$",
  "attended - yes", "$b_0 + b_1$", "$b_0 + b_1 + b_2$"),
  col.names = c("","read textbook? no", "read textbook? yes"))
```

	read textbook? no	read textbook? yes
attended - no	b_0	b_0+b_2
attended - yes	b_0+b_1	b_0+b_1+b_2

As you can see, the intercept term b_0 acts like a kind of "baseline" grade that you would expect from those students who don't take the time to attend class or read the textbook. Similarly, b_1 represents the boost that you're expected to get if you come to class, and b_2 represents the boost that comes from reading the textbook. In fact, if this were an ANOVA you might very well want to characterise b_1 as the main effect of attendance, and b_2 as the main effect of reading! In fact, for a simple 2×2 ANOVA that's *exactly* how it plays out.

Okay, now that we're really starting to see why ANOVA and regression are basically the same thing, let's actually run our regression using the `rtfm.1` data and the `lm()` function to convince ourselves that this is really true. Running the regression in the usual way gives us the following output:[240]

```
regression.model <- lm( grade ~ attend + reading, data = rtfm.1 )
summary( regression.model )
```

```
##
## Call:
## lm(formula = grade ~ attend + reading, data = rtfm.1)
##
## Residuals:
##    1    2    3    4    5    6    7    8
##  0.5 -2.5  3.5 -1.5 -8.5  6.5  3.5 -1.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43.500      3.354  12.969 4.86e-05 ***
## attend        18.000      3.873   4.648  0.00559 **
## reading       28.000      3.873   7.230  0.00079 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.477 on 5 degrees of freedom
## Multiple R-squared:  0.9366, Adjusted R-squared:  0.9112
## F-statistic: 36.93 on 2 and 5 DF,  p-value: 0.001012
```

There's a few interesting things to note here. Firstly, notice that the intercept term is 43.5, which is close to the "group" mean of 42.5 observed for those two students who didn't read the text or attend class. Moreover, it's *identical* to the predicted group mean that we pulled out of our ANOVA using the `Effects()` function! Secondly, notice that we have the regression coefficient of b_1=18.0 for the attendance variable, suggesting that those students that attended class scored 18% higher than those who didn't. So our expectation would be that those students who turned up to class but didn't read the textbook would obtain a grade of b_0+b_1, which is equal to 43.5+18.0=61.5. Again, this is similar to the observed group mean of 62.5, and identical to the expected group mean that we pulled from our ANOVA. You can verify for yourself that the same thing happens when we look at the students that read the textbook.

Actually, we can push a little further in establishing the equivalence of our ANOVA and our regression. Look at the p-values associated with the `attend` variable and the `reading` variable in the regression output. They're identical to the ones we encountered earlier when running the ANOVA. This might seem a little surprising, since the test used when running our regression model calculates a t-statistic and the ANOVA calculates an F-statistic. However, if you can remember all the way back to Chapter 9, I mentioned that there's a relationship between the t-distribution and the F-distribution: if you have some quantity that is distributed according to a t-distribution with k degrees of freedom and you square it, then this new squared quantity follows an F-distribution whose degrees of freedom are 1 and k. We can check this with respect to the t statistics in our regression model. For the `attend` variable we get a t value of 4.648. If we square this number we end up with 21.604, which is identical to the corresponding F statistic in our ANOVA.

Finally, one last thing you should know. Because R understands the fact that ANOVA and regression are both examples of linear models, it lets you extract the classic ANOVA table from your regression model using the `anova()` function. All you have to do is this:

```
anova( regression.model )
```

```
## Analysis of Variance Table
##
## Response: grade
##           Df Sum Sq Mean Sq F value    Pr(>F)
## attend     1    648     648  21.600 0.0055943 **
## reading    1   1568    1568  52.267 0.0007899 ***
## Residuals  5    150      30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

16.6.3 Changing the baseline category

At this point, you're probably convinced that the ANOVA and the regression are actually identical to each other. So there's one last thing I should show you. What happens if I use the data from `rtfm.2` to run the regression? In `rtfm.2`, we coded the `attend` and `reading` variables as factors rather than as numeric variables. Does this matter? It turns out that it doesn't. The only differences are superficial:

```
regression.model.2 <- lm( grade ~ attend + reading, data = rtfm.2 )
 summary( regression.model.2 )
```

```
##
## Call:
## lm(formula = grade ~ attend + reading, data = rtfm.2)
##
## Residuals:
##    1    2    3    4    5    6    7    8
##  0.5 -2.5  3.5 -1.5 -8.5  6.5  3.5 -1.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   43.500      3.354  12.969 4.86e-05 ***
## attendyes     18.000      3.873   4.648  0.00559 **
## readingyes    28.000      3.873   7.230  0.00079 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.477 on 5 degrees of freedom
## Multiple R-squared:  0.9366, Adjusted R-squared:  0.9112
## F-statistic: 36.93 on 2 and 5 DF,  p-value: 0.001012
```

The only thing that is different is that R labels the two variables differently: the output now refers to `attendyes` and `readingyes`. You can probably guess what this means. When R refers to `readingyes` it's trying to indicate that it is assuming that "yes = 1" and "no = 0". This is important. Suppose we wanted to say that "yes = 0" and "no = 1". We could still run this as a regression model, but now all of our coefficients will go in the opposite direction, because the effect of `readingno` would be referring to the consequences of *not* reading the textbook. To show you how this works, we can use the `relevel()` function in R to change which level of the `reading` variable is set to "0". Here's how it works. First, let's get R to print out the `reading` factor as it currently stands:

```
rtfm.2$reading
```

```
## [1] yes yes yes no  no  no  no  yes
## Levels: no yes
```

Notice that order in which R prints out the levels is “no” and then “yes”. Now let’s apply the `relevel()` function:

```
 relevel( x = rtfm.2$reading, ref = "yes" )
```

```
## [1] yes yes yes no  no  no  no  yes
## Levels: yes no
```

R now lists “yes” before “no”. This means that R will now treat “yes” as the “reference” level (sometimes called the baseline level) when you include it in an ANOVA. So let’s now create a new data frame with our factors recoded…

```
rtfm.3 <- rtfm.2                                          # copy the old data frame
 rtfm.3$reading <- relevel( rtfm.2$reading, ref="yes" )  # re-level the reading facto
 rtfm.3$attend <- relevel( rtfm.2$attend, ref="yes" )    # re-level the attend factor
```

Finally, let’s re-run our regression, this time using the re-coded data:

```
regression.model.3 <- lm( grade ~ attend + reading, data = rtfm.3 )
 summary( regression.model.3 )
```

```
##
## Call:
## lm(formula = grade ~ attend + reading, data = rtfm.3)
##
## Residuals:
##    1    2    3    4    5    6    7    8
##  0.5 -2.5  3.5 -1.5 -8.5  6.5  3.5 -1.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   89.500      3.354  26.684 1.38e-06 ***
## attendno     -18.000      3.873  -4.648  0.00559 **
## readingno    -28.000      3.873  -7.230  0.00079 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.477 on 5 degrees of freedom
## Multiple R-squared:  0.9366, Adjusted R-squared:  0.9112
## F-statistic: 36.93 on 2 and 5 DF,  p-value: 0.001012
```

As you can see, there are now a few changes. Most obviously, the `attendno` and `readingno` effects are both negative, though they’re the same magnitude as before: if you *don’t* read the textbook, for instance, you should expect your grade to drop by 28% relative to someone who did. The t-statistics have reversed sign too. The p-values remain the same, of course. The intercept has changed too. In our original regression, the baseline corresponded to a student who didn’t attend class and didn’t read the textbook, so we got a value of 43.5 as the expected baseline grade. However, now that we’ve recoded our variables, the baseline corresponds to a student who has read the textbook and did attend class, and for that student we would expect a grade of 89.5.

16.6.4 encode non binary factors as contrasts

At this point, I've shown you how we can view a 2×2 ANOVA into a linear model. And it's pretty easy to see how this generalises to a 2×2×2 ANOVA or a 2×2×2×2 ANOVA… it's the same thing, really: you just add a new binary variable for each of your factors. Where it begins to get trickier is when we consider factors that have more than two levels. Consider, for instance, the 3×2 ANOVA that we ran earlier in this chapter using the `clin.trial` data. How can we convert the three-level `drug` factor into a numerical form that is appropriate for a regression?

The answer to this question is pretty simple, actually. All we have to do is realise that a three-level factor can be redescribed as *two* binary variables. Suppose, for instance, I were to create a new binary variable called `druganxifree` . Whenever the `drug` variable is equal to `"anxifree"` we set `druganxifree = 1` . Otherwise, we set `druganxifree = 0` . This variable sets up a ***contrast***, in this case between anxifree and the other two drugs. By itself, of course, the `druganxifree` contrast isn't enough to fully capture all of the information in our `drug` variable. We need a second contrast, one that allows us to distinguish between joyzepam and the placebo. To do this, we can create a second binary contrast, called `drugjoyzepam` , which equals 1 if the drug is joyzepam, and 0 if it is not. Taken together, these two contrasts allows us to perfectly discriminate between all three possible drugs. The table below illustrates this:

```
knitr::kable(tibble::tribble(
           ~V1,              ~V2,              ~V3,
   "`placebo`",              "0",              "0",
  "`anxifree`",              "1",              "0",
  "`joyzepam`",              "0",              "1"
  ), col.names = c(      "`drug`", "`druganxifree`", "`drugjoyzepam`"))
```

drug	druganxifree	drugjoyzepam
placebo	0	0
anxifree	1	0
joyzepam	0	1

If the drug administered to a patient is a placebo, then both of the two contrast variables will equal 0. If the drug is Anxifree, then the `druganxifree` variable will equal 1, and `drugjoyzepam` will be 0. The reverse is true for Joyzepam: `drugjoyzepam` is 1, and `druganxifree` is 0.

Creating contrast variables manually is not too difficult to do using basic R commands. For example, here's how we would create the `druganxifree` variable:

```
druganxifree <- as.numeric( clin.trial$drug == "anxifree" )
 druganxifree
```

```
##  [1] 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0
```

The `clin.trial$drug == "anxifree"` part of the command returns a logical vector that has a value of `TRUE` if the drug is Anxifree, and a value of `FALSE` if the drug is Joyzepam or the placebo. The `as.numeric()` function just converts `TRUE` to 1 and `FALSE` to 0. Obviously, this command creates the `druganxifree` variable inside the workspace. If you wanted to add it to the `clin.trial` data frame, you'd use a commmand like this instead:

```
clin.trial$druganxifree <- as.numeric( clin.trial$drug == "anxifree" )
```

You could then repeat this for the other contrasts that you wanted to use. However, it's kind of tedious to do this over and over again for every single contrast that you want to create. To make it a little easier, the `lsr` package contains a simple function called `expandFactors()` that will convert every factor in a data frame into a set of contrast variables.[241] We can use it to

create a new data frame, `clin.trial.2` that contains the same data as `clin.trial`, but with the two factors represented in terms of the contrast variables:

```
clin.trial.2 <- expandFactors( clin.trial )
```

```
##    (Intercept) druganxifree drugjoyzepam therapyCBT mood.gain druganxifree
## 1            1            0            0          0       0.5            0
## 2            1            0            0          0       0.3            0
## 3            1            0            0          0       0.1            0
## 4            1            1            0          0       0.6            1
## 5            1            1            0          0       0.4            1
## 6            1            1            0          0       0.2            1
## 7            1            0            1          0       1.4            0
## 8            1            0            1          0       1.7            0
## 9            1            0            1          0       1.3            0
## 10           1            0            0          1       0.6            0
## 11           1            0            0          1       0.9            0
## 12           1            0            0          1       0.3            0
## 13           1            1            0          1       1.1            1
## 14           1            1            0          1       0.8            1
## 15           1            1            0          1       1.2            1
## 16           1            0            1          1       1.8            0
## 17           1            0            1          1       1.3            0
## 18           1            0            1          1       1.4            0
## attr(,"assign")
## [1] 0 1 1 2 3 4
## attr(,"contrasts")
## attr(,"contrasts")$drug
## [1] "contr.treatment"
##
## attr(,"contrasts")$therapy
## [1] "contr.treatment"
```

```
clin.trial.2
```

```
##    druganxifree drugjoyzepam therapyCBT mood.gain druganxifree
## 1             0            0          0       0.5            0
## 2             0            0          0       0.3            0
## 3             0            0          0       0.1            0
## 4             1            0          0       0.6            1
## 5             1            0          0       0.4            1
## 6             1            0          0       0.2            1
## 7             0            1          0       1.4            0
## 8             0            1          0       1.7            0
## 9             0            1          0       1.3            0
## 10            0            0          1       0.6            0
## 11            0            0          1       0.9            0
## 12            0            0          1       0.3            0
## 13            1            0          1       1.1            1
## 14            1            0          1       0.8            1
## 15            1            0          1       1.2            1
## 16            0            1          1       1.8            0
## 17            0            1          1       1.3            0
## 18            0            1          1       1.4            0
```

It's not as pretty as the original `clin.trial` data, but it's definitely saying the same thing. We have now recoded our three-level factor in terms of two binary variables, and we've already seen that ANOVA and regression behave the same way for binary variables. However, there are some additional complexities that arise in this case, which we'll discuss in the next section.

16.6.5 equivalence between ANOVA and regression for non-binary factors

Now we have two different versions of the same data set: our original data frame `clin.trial` in which the `drug` variable is expressed as a single three-level factor, and the expanded data set `clin.trial.2` in which it is expanded into two binary contrasts. Once again, the thing that we want to demonstrate is that our original 3×2 factorial ANOVA is equivalent to a regression model applied to the contrast variables. Let's start by re-running the ANOVA:

```
drug.anova <- aov( mood.gain ~ drug + therapy, clin.trial )
summary( drug.anova )
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## drug         2  3.453  1.7267  26.149 1.87e-05 ***
## therapy      1  0.467  0.4672   7.076   0.0187 *
## Residuals   14  0.924  0.0660
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obviously, there's no surprises here. That's the exact same ANOVA that we ran earlier, except for the fact that I've arbitrarily decided to rename the output variable as `drug.anova` for some stupid reason.[242] Next, let's run a regression, using `druganxifree`, `drugjoyzepam` and `therapyCBT` as the predictors. Here's what we get:

```
drug.regression <- lm( mood.gain ~ druganxifree + drugjoyzepam + therapyCBT, clin.tri
summary( drug.regression )
```

```
##
## Call:
## lm(formula = mood.gain ~ druganxifree + drugjoyzepam + therapyCBT,
##     data = clin.trial.2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3556 -0.1806  0.0000  0.1972  0.3778
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.2889     0.1211   2.385   0.0318 *
## druganxifree   0.2667     0.1484   1.797   0.0939 .
## drugjoyzepam   1.0333     0.1484   6.965  6.6e-06 ***
## therapyCBT     0.3222     0.1211   2.660   0.0187 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.257 on 14 degrees of freedom
## Multiple R-squared:  0.8092, Adjusted R-squared:  0.7683
## F-statistic: 19.79 on 3 and 14 DF,  p-value: 2.64e-05
```

Hm. This isn't the same output that we got last time. Not surprisingly, the regression output prints out the results for each of the three predictors separately, just like it did every other time we used `lm()`. On the one hand, we can see that the p-value for the `therapyCBT` variable is exactly the same as the one for the `therapy` factor in our original ANOVA, so we can be reassured that the regression model is doing the same thing as the ANOVA did. On the other hand, this regression model is testing the `druganxifree` contrast and the `drugjoyzepam` contrast *separately*, as if they were two completely unrelated variables. It's not surprising of course, because the poor `lm()` function has no way of knowing that `drugjoyzepam` and `druganxifree` are actually the two different contrasts that we used to encode our three-level `drug` factor. As far as it knows, `drugjoyzepam` and `druganxifree` are no more related to one another than `drugjoyzepam` and `therapyCBT`. However, you and I know better. At this stage we're not at all interested in determining whether these two contrasts are individually significant. We just want to know if there's an "overall" effect of drug. That is, what *we* want R to do is to run some kind of "omnibus" test, one in which the two "drug-related" contrasts are lumped together for the purpose of the test. Sound familiar? This is *exactly* the situation that we discussed in Section 16.5, and it is precisely this situation that the F-test is built to handle. All we need to do is specify our null model, which in this case would include the `therapyCBT` predictor, and omit both of the drug-related variables, and then run it through the `anova()` function:

```
nodrug.regression <- lm( mood.gain ~ therapyCBT, clin.trial.2 )
 anova( nodrug.regression, drug.regression )
```

```
## Analysis of Variance Table
##
## Model 1: mood.gain ~ therapyCBT
## Model 2: mood.gain ~ druganxifree + drugjoyzepam + therapyCBT
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     16 4.3778
## 2     14 0.9244  2    3.4533 26.149 1.872e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ah, that's better. Our F-statistic is 26.1, the degrees of freedom are 2 and 14, and the p-value is 0.000019. The numbers are identical to the ones we obtained for the main effect of `drug` in our original ANOVA. Once again, we see that ANOVA and regression are essentially the same: they are both linear models, and the underlying statistical machinery for ANOVA is identical to the machinery used in regression. The importance of this fact should not be understated. Throughout the rest of this chapter we're going to rely heavily on this idea.

16.6.6 Degrees of freedom as parameter counting!

At long last, I can finally give a definition of degrees of freedom that I am happy with. Degrees of freedom are defined in terms of the number of parameters that have to be estimated in a model. For a regression model or an ANOVA, the number of parameters corresponds to the number of regression coefficients (i.e. b-values), including the intercept. Keeping in mind that any F-test is always a comparison between two models, the first df is the difference in the number of parameters. For example, model comparison above, the null model (`mood.gain ~ therapyCBT`) has two parameters: there's one regression coefficient for the `therapyCBT` variable, and a second one for the intercept. The alternative model (`mood.gain ~ druganxifree + drugjoyzepam + therapyCBT`) has four parameters: one regression coefficient for each of the three contrasts, and one more for the intercept. So the degrees of freedom associated with the *difference* between these two models is df_1=4−2=2.

What about the case when there doesn't seem to *be* a null model? For instance, you might be thinking of the F-test that appears at the very bottom of the regression output. I originally described that as a test of the regression model as a whole. However, that is still a comparison between two models. The null model is the trivial model that only includes an intercept, which is written as `outcome ~ 1` in R, and the alternative model is the full regression model. The null model in this case contains 1 regression coefficient, for the intercept term. The alternative model contains K+1 regression coefficients, one for each of the K predictor variables and one more for the intercept. So the df value that you see in this F test is equal to df_1=K+1−1=K.

What about the second df value that appears in the F-test? This always refers to the degrees of freedom associated with the residuals. It is possible to think of this in terms of parameters too, but in a slightly counterintuitive way. Think of it like this: suppose that the total number of observations across the study as a whole is N. If you wanted to *perfectly* describe each of these N values, you need to do so using, well… N numbers. When you build a regression model, what you're really doing is specifying some of the numbers need to perfectly describe the data. If your model has K predictors and an intercept, then you've specified K+1 numbers. So, without bothering to figure out exactly *how* this would be done, how many *more* numbers do you think are going to be needed to transform a K+1 parameter regression model into a perfect redescription of the raw data? If you found yourself thinking that (K+1)+(N−K−1)=N, and so the answer would have to be N−K−1, well done! That's exactly right: in principle you can imagine an absurdly complicated regression model that includes a parameter for every single data point, and it would of course provide a perfect description of the data. This model would contain N parameters in total, but we're interested in the difference between the number of parameters required to describe this full model (i.e. N) and the number of parameters used by the simpler regression model that you're actually interested in (i.e., K+1), and so the second degrees of freedom in the F test is df2=N−K−1, where K is the number of predictors (in a regression model) or the number of contrasts (in an ANOVA). In the example I gave above, there are N=18 observations in the data set, and K+1=4 regression coefficients associated with the ANOVA model, so the degrees of freedom for the residuals is df_2=18−4=14.

16.6.7 postscript

There's one last thing I want to mention in this section. In the previous example, I used the `aov()` function to run an ANOVA using the `clin.trial` data which codes `drug` variable as a single factor. I also used the `lm()` function to run a regression using the `clin.trial` data in which we have two separate contrasts describing the drug. However, it's also possible to use the `lm()` function on the the original data. That is, you could use a command like this:

```
drug.lm <- lm( mood.gain ~ drug + therapy, clin.trial )
```

The fact that `drug` is a three-level factor does not matter. As long as the `drug` variable has been declared to be a factor, R will automatically translate it into two binary contrast variables, and will perform the appropriate analysis. After all, as I've been saying throughout this section, ANOVA and regression are both linear models, and `lm()` is the function that handles linear models. In fact, the `aov()` function doesn't actually do very much of the work when you run an ANOVA using it: internally, R just passes all the hard work straight to `lm()` . However, I want to emphasise again that it is *critical* that your factor variables are

declared as such. If `drug` were declared to be a numeric variable, then R would be happy to treat it as one. After all, it might be that `drug` refers to the number of drugs that one has taken in the past, or something that is genuinely numeric. R won't second guess you here. It assumes your factors are factors and your numbers are numbers. Don't make the mistake of encoding your factors as numbers, or R will run the wrong analysis. This is *not* a flaw in R: it is *your* responsibility as the analyst to make sure you're specifying the right model for your data. Software really can't be trusted with this sort of thing.

Okay, warnings aside, it's actually kind of neat to run your ANOVA using the `lm()` function in the way I did above. Because you've called the `lm()` function, the `summary()` that R pulls out is formatted like a regression. To save space I won't show you the output here, but you can easily verify this by typing

```
summary( drug.lm )
```

```
##
## Call:
## lm(formula = mood.gain ~ drug + therapy, data = clin.trial)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.3556 -0.1806  0.0000  0.1972  0.3778
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.2889     0.1211   2.385   0.0318 *
## druganxifree   0.2667     0.1484   1.797   0.0939 .
## drugjoyzepam   1.0333     0.1484   6.965  6.6e-06 ***
## therapyCBT     0.3222     0.1211   2.660   0.0187 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.257 on 14 degrees of freedom
## Multiple R-squared:  0.8092, Adjusted R-squared:  0.7683
## F-statistic: 19.79 on 3 and 14 DF,  p-value: 2.64e-05
```

However, because the `drug` and `therapy` variables were both factors, the `anova()` function actually knows which contrasts to group together for the purposes of running the F-tests, so you can extract the classic ANOVA table. Again, I won't reproduce the output here since it's identical to the ANOVA table I showed at the start of the section, but it's worth trying the following command

```
anova( drug.lm )
```

```
## Analysis of Variance Table
##
## Response: mood.gain
##           Df Sum Sq Mean Sq F value    Pr(>F)
## drug       2 3.4533 1.72667 26.1490 1.872e-05 ***
## therapy    1 0.4672 0.46722  7.0757   0.01866 *
## Residuals 14 0.9244 0.06603
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

just to see for yourself. However, this behaviour of the `anova()` function only occurs when the predictor variables are factors. If we try a command like `anova( drug.regression )`, the output will continue to treate `druganxifree` and

`drugjoyzepam` as if they were two distinct binary factors. This is because in the `drug.regression` model we included all the contrasts as "raw" variables, so R had no idea which ones belonged together. However, when we ran the `drug.lm` model, we gave R the original factor variables, so it does know which contrasts go together. The behaviour of the `anova()` function reflects that.
