16.10: Factorial ANOVA 3- Unbalanced Designs

Factorial ANOVA is a very handy thing to know about. It's been one of the standard tools used to analyse experimental data for many decades, and you'll find that you can't read more than two or three papers in psychology without running into an ANOVA in there somewhere. However, there's one huge difference between the ANOVAs that you'll see in a lot of real scientific articles and the ANOVA that I've just described: in real life, we're rarely lucky enough to have perfectly balanced designs. For one reason or another, it's typical to end up with more observations in some cells than in others. Or, to put it another way, we have an ***unbalanced design***.

Unbalanced designs need to be treated with a lot more care than balanced designs, and the statistical theory that underpins them is a lot messier. It might be a consequence of this messiness, or it might be a shortage of time, but my experience has been that undergraduate research methods classes in psychology have a nasty tendency to ignore this issue completely. A lot of stats textbooks tend to gloss over it too. The net result of this, I think, is that a lot of active researchers in the field don't actually know that there's several different "types" of unbalanced ANOVAs, and they produce quite different answers. In fact, reading the psychological literature, I'm kind of amazed at the fact that most people who report the results of an unbalanced factorial ANOVA don't actually give you enough details to reproduce the analysis: I secretly suspect that most people don't even realise that their statistical software package is making a whole lot of substantive data analysis decisions on their behalf. It's actually a little terrifying, when you think about it. So, if you want to avoid handing control of your data analysis to stupid software, read on…

16.10.1 coffee data

As usual, it will help us to work with some data. The `coffee.Rdata` file contains a hypothetical data set (the `coffee` data frame) that produces an unbalanced 3×2 ANOVA. Suppose we were interested in finding out whether or not the tendency of people to `babble` when they have too much coffee is purely an effect of the coffee itself, or whether there's some effect of the `milk` and `sugar` that people add to the coffee. Suppose we took 18 people, and gave them some coffee to drink. The amount of coffee / caffeine was held constant, and we varied whether or not milk was added: so `milk` is a binary factor with two levels, `"yes"` and `"no"` . We also varied the kind of sugar involved. The coffee might contain `"real"` sugar, or it might contain `"fake"` sugar (i.e., artificial sweetener), or it might contain `"none"` at all, so the `sugar` variable is a three level factor. Our outcome variable is a continuous variable that presumably refers to some psychologically sensible measure of the extent to which someone is "babbling". The details don't really matter for our purpose. To get a sense of what the data look like, we use the `some()` function in the `car` package. The `some()` function randomly picks a few of the observations in the data frame to print out, which is often very handy:

```
load("./rbook-master/data/coffee.rdata")
 some( coffee )
```

```
##    milk sugar babble
## 1   yes  real    4.6
## 5   yes  real    5.1
## 6    no  real    5.5
## 7   yes  none    3.9
## 9   yes  none    3.7
## 10   no  fake    5.6
## 11   no  fake    4.7
## 13   no  real    6.0
## 14   no  real    5.4
## 17   no  none    5.3
```

If we use the `aggregate()` function to quickly produce a table of means, we get a strong impression that there are differences between the groups:

```
aggregate( babble ~ milk + sugar, coffee, mean )
```

```
##   milk sugar babble
## 1  yes  none  3.700
## 2   no  none  5.550
## 3  yes  fake  5.800
## 4   no  fake  4.650
## 5  yes  real  5.100
## 6   no  real  5.875
```

This is especially true when we compare these means to the standard deviations for the `babble` variable, which you can calculate using `aggregate()` in much the same way. Across groups, this standard deviation varies from .14 to .71, which is fairly small relative to the differences in group means.[246] So far, it's looking like a straightforward factorial ANOVA, just like we did earlier. The problem arises when we check to see how many observations we have in each group:

```
xtabs( ~ milk + sugar, coffee )
```

```
##      sugar
## milk  none fake real
##   yes    3    2    3
##   no     2    4    4
```

This violates one of our original assumptions, namely that the number of people in each group is the same. We haven't really discussed how to handle this situation.

16.10.2 "Standard ANOVA" does not exist for unbalanced designs

Unbalanced designs lead us to the somewhat unsettling discovery that there isn't really any one thing that we might refer to as a standard ANOVA. In fact, it turns out that there are *three* fundamentally different ways[247] in which you might want to run an ANOVA in an unbalanced design. If you have a balanced design, all three versions produce identical results, with the sums of squares, F-values etc all conforming to the formulas that I gave at the start of the chapter. However, when your design is unbalanced they don't give the same answers. Furthermore, they are not all equally appropriate to every situation: some methods will be more appropriate to your situation than others. Given all this, it's important to understand what the different types of ANOVA are and how they differ from one another.

The first kind of ANOVA is conventionally referred to as ***Type I sum of squares***. I'm sure you can guess what they other two are called. The "sum of squares" part of the name was introduced by the SAS statistical software package, and has become standard nomenclature, but it's a bit misleading in some ways. I think the logic for referring to them as different types of sum of squares is that, when you look at the ANOVA tables that they produce, the key difference in the numbers is the SS values. The degrees of freedom don't change, the MS values are still defined as SS divided by df, etc. However, what the terminology gets wrong is that it hides the reason *why* the SS values are different from one another. To that end, it's a lot more helpful to think of the three different kinds of ANOVA as three different *hypothesis testing strategies*. These different strategies lead to different SS values, to be sure, but it's the strategy that is the important thing here, not the SS values themselves. Recall from Section 16.5 and 16.6 that any particular F-test is best thought of as a comparison between two linear models. So when you're looking at an ANOVA table, it helps to remember that each of those F-tests corresponds to a *pair* of models that are being compared. Of course, this leads naturally to the question of *which* pair of models is being compared. This is the fundamental difference between ANOVA Types I, II and III: each one corresponds to a different way of choosing the model pairs for the tests.

16.10.3 Type I sum of squares

The Type I method is sometimes referred to as the "sequential" sum of squares, because it involves a process of adding terms to the model one at a time. Consider the coffee data, for instance. Suppose we want to run the full 3×2 factorial ANOVA, including interaction terms. The full model, as we've discussed earlier, is expressed by the R formula `babble ~ sugar + milk + sugar:milk`, though we often shorten it by using the `sugar * milk` notation. The Type I strategy builds this model up sequentially, starting from the simplest possible model and gradually adding terms.

The simplest possible model for the data would be one in which neither milk nor sugar is assumed to have any effect on babbling. The only term that would be included in such a model is the intercept, and in R formula notation we would write it as `babble ~ 1`. This is our initial null hypothesis. The next simplest model for the data would be one in which only one of the two main effects is included. In the coffee data, there are two different possible choices here, because we could choose to add milk first or to add sugar first (pardon the pun). The order actually turns out to matter, as we'll see later, but for now let's just make a choice arbitrarily, and pick sugar. So the second model in our sequence of models is `babble ~ sugar`, and it forms the alternative hypothesis for our first test. We now have our first hypothesis test:

```
knitr::kable(tibble::tribble(
                    ~V1,                 ~V2,
         "Null model:",     "`babble ~ 1`",
  "Alternative model:", "`babble ~ sugar`"
  ), col.names= c("", ""))
```

Null model:	babble ~ 1
Alternative model:	babble ~ sugar

This comparison forms our hypothesis test of the main effect of `sugar`. The next step in our model building exercise it to add the other main effect term, so the next model in our sequence is `babble ~ sugar + milk`. The second hypothesis test is then formed by comparing the following pair of models:

```
knitr::kable(tibble::tribble(
                    ~V1,                 ~V2,
         "Null model:",     "`babble ~ sugar`",
  "Alternative model:", "`babble ~ sugar + milk`"
  ), col.names= c("", ""))
```

Null model:	babble ~ sugar
Alternative model:	babble ~ sugar + milk

This comparison forms our hypothesis test of the main effect of `milk`. In one sense, this approach is very elegant: the alternative hypothesis from the first test forms the null hypothesis for the second one. It is in this sense that the Type I method is strictly sequential. Every test builds directly on the results of the last one. However, in another sense it's very inelegant, because there's a strong asymmetry between the two tests. The test of the main effect of `sugar` (the first test) completely ignores `milk`, whereas the test of the main effect of `milk` (the second test) does take `sugar` into account. In any case, the fourth model in our sequence is now the full model, `babble ~ sugar + milk + sugar:milk`, and the corresponding hypothesis test is

```
knitr::kable(tibble::tribble(
                    ~V1,                 ~V2,
         "Null model:",     "`babble ~ sugar + milk`",
  "Alternative model:", "`babble ~ sugar + milk + sugar:milk`"
  ), col.names= c("", ""))
```

Null model:	babble ~ sugar + milk
Alternative model:	babble ~ sugar + milk + sugar:milk

Type I sum of squares is the default hypothesis testing method used by the `anova()` function, so it's easy to produce the results from a Type I analysis. We just type in the same commands that we always did. Since we've now reached the point that we don't need to hide the fact that ANOVA and regression are both linear models, I'll use the `lm()` function to run the analyses:

```
mod <- lm( babble ~ sugar + milk + sugar:milk, coffee )
anova( mod )
```

```
## Analysis of Variance Table
##
## Response: babble
##            Df Sum Sq Mean Sq F value   Pr(>F)
## sugar       2 3.5575 1.77876  6.7495 0.010863 *
## milk        1 0.9561 0.95611  3.6279 0.081061 .
## sugar:milk  2 5.9439 2.97193 11.2769 0.001754 **
## Residuals  12 3.1625 0.26354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Leaving aside for one moment the question of how this result should be interpreted, let's take note of the fact that our three p-values are .0109, .0811 and .0018 respectively. Next, let's see if we can replicate the analysis using tools that we're a little more familiar with. First, let's fit all four models:

```
mod.1 <- lm( babble ~ 1, coffee )
mod.2 <- lm( babble ~ sugar, coffee )
mod.3 <- lm( babble ~ sugar + milk, coffee )
mod.4 <- lm( babble ~ sugar + milk + sugar:milk, coffee )
```

To run the first hypothesis test comparing `mod.1` to `mod.2` we can use the command `anova(mod.1, mod.2)` in much the same way that we did in Section 16.5. Similarly, we can use the commands `anova(mod.2, mod.3)` and `anova(mod.3, mod.4)` and to run the second and third hypothesis tests. However, rather than run each of those commands separately, we can enter the full sequence of models like this:

```
anova( mod.1, mod.2, mod.3, mod.4 )
```

```
## Analysis of Variance Table
##
## Model 1: babble ~ 1
## Model 2: babble ~ sugar
## Model 3: babble ~ sugar + milk
## Model 4: babble ~ sugar + milk + sugar:milk
##   Res.Df     RSS Df Sum of Sq       F   Pr(>F)
## 1     17 13.6200
## 2     15 10.0625  2    3.5575  6.7495 0.010863 *
## 3     14  9.1064  1    0.9561  3.6279 0.081061 .
## 4     12  3.1625  2    5.9439 11.2769 0.001754 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This output is rather more verbose than the last one, but it's telling essentially the same story.[248]

The big problem with using Type I sum of squares is the fact that it really does depend on the order in which you enter the variables. Yet, in many situations the researcher has no reason to prefer one ordering over another. This is presumably the case for our milk and sugar problem. Should we add milk first, or sugar first? It feels exactly as arbitrary as a data analysis question as it does as a coffee-making question. There may in fact be some people with firm opinions about ordering, but it's hard to imagine a principled answer to the question. Yet, look what happens when we change the ordering:

```
mod <- lm( babble ~  milk + sugar + sugar:milk, coffee )
 anova( mod )
```

```
## Analysis of Variance Table
##
## Response: babble
##            Df Sum Sq Mean Sq F value   Pr(>F)
## milk        1 1.4440 1.44400  5.4792 0.037333 *
## sugar       2 3.0696 1.53482  5.8238 0.017075 *
## milk:sugar  2 5.9439 2.97193 11.2769 0.001754 **
## Residuals  12 3.1625 0.26354
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-values for both main effect terms have changed, and fairly dramatically. Among other things, the effect of `milk` has become significant (though one should avoid drawing any strong conclusions about this, as I've mentioned previously). Which of these two ANOVAs should one report? It's not immediately obvious.

When you look at the hypothesis tests that are used to define the "first" main effect and the "second" one, it's clear that they're qualitatively different from one another. In our initial example, we saw that the test for the main effect of `sugar` completely ignores `milk`, whereas the test of the main effect of `milk` does take `sugar` into account. As such, the Type I testing strategy really does treat the first main effect as if it had a kind of theoretical primacy over the second one. In my experience there is very rarely if ever any theoretically primacy of this kind that would justify treating any two main effects asymmetrically.

The consequence of all this is that Type I tests are very rarely of much interest, and so we should move on to discuss Type II tests and Type III tests. However, for the sake of completeness – on the off chance that you ever find yourself needing to run Type I tests – I'll comment briefly on how R determines the ordering of terms in a Type I test. The key principle in Type I sum of squares is that the hypothesis testing be sequential, with terms added one at a time. However, it does also imply that main effects be added first (e.g., factors `A`, `B`, `C` etc), followed by first order interaction terms (e.g., terms like `A:B` and `B:C`), then second order interactions (e.g., `A:B:C`) and so on. Within each "block" you can specify whatever order you like. So, for instance, if we specified our model using a command like this,

```
mod <- lm( outcome ~ A + B + C + B:C + A:B + A:C )
```

and then used `anova(mod)` to produce sequential hypothesis tests, what we'd see is that the main effect terms would be entered `A` then `B` and then `C`, but then the interactions would be entered in the order `B:C` first, then `A:B` and then finally `A:C`. Reordering the terms within each group will change the ordering, as we saw earlier. However, changing the order of terms across blocks has no effect. For instance, if we tried to move the interaction term `B:C` to the front, like this,

```
mod <- lm( outcome ~ B:C + A + B + C + A:B + A:C )
```

it would have no effect. R would still enter the terms in the same order as last time. If for some reason you really, really need an interaction term to be entered first, then you have to do it the long way, creating each model manually using a separate `lm()` command and then using a command like `anova(mod.1, mod.2, mod.3, mod.4)` to force R to enter them in the order that you want.

16.10.4 Type III sum of squares

Having just finished talking about Type I tests, you might think that the natural thing to do next would be to talk about Type II tests. However, I think it's actually a bit more natural to discuss Type III tests (which are simple) before talking about Type II tests (which are trickier). The basic idea behind Type III tests is extremely simple: regardless of which term you're trying to evaluate, run the F-test in which the alternative hypothesis corresponds to the full ANOVA model as specified by the user, and the null model just deletes that one term that you're testing. For instance, in the coffee example, in which our full model was

`babble ~ sugar + milk + sugar:milk`, the test for a main effect of `sugar` would correspond to a comparison between the following two models:

```
knitr::kable(tibble::tribble(
                    ~V1,                ~V2,
         "Null model:",     "`babble ~ milk + sugar:milk`",
  "Alternative model:", "`babble ~ sugar + milk + sugar:milk`"
  ), col.names= c("", ""))
```

Null model:	babble ~ milk + sugar:milk
Alternative model:	babble ~ sugar + milk + sugar:milk

Similarly the main effect of `milk` is evaluated by testing the full model against a null model that removes the `milk` term, like so:

```
knitr::kable(tibble::tribble(
                    ~V1,                ~V2,
         "Null model:",     "`babble ~ sugar + sugar:milk`",
  "Alternative model:", "`babble ~ sugar + milk + sugar:milk`"
  ), col.names= c("", ""))
```

Null model:	babble ~ sugar + sugar:milk
Alternative model:	babble ~ sugar + milk + sugar:milk

Finally, the interaction term `sugar:milk` is evaluated in exactly the same way. Once again, we test the full model against a null model that removes the `sugar:milk` interaction term, like so:

```
knitr::kable(tibble::tribble(
                    ~V1,                ~V2,
         "Null model:",     "`babble ~ sugar + milk`",
  "Alternative model:", "`babble ~ sugar + milk + sugar:milk`"
  ), col.names= c("", ""))
```

Null model:	babble ~ sugar + milk
Alternative model:	babble ~ sugar + milk + sugar:milk

The basic idea generalises to higher order ANOVAs. For instance, suppose that we were trying to run an ANOVA with three factors, `A`, `B` and `C`, and we wanted to consider all possible main effects and all possible interactions, including the three way interaction `A:B:C`. The table below shows you what the Type III tests look like for this situation:

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

                  "`A`", "`B + C + A:B + A:C + B:C + A:B:C`", "`A + B + C + A:B + A
                  "`B`", "`A + C + A:B + A:C + B:C + A:B:C`", "`A + B + C + A:B + A
                  "`C`", "`A + B + A:B + A:C + B:C + A:B:C`", "`A + B + C + A:B + A
                "`A:B`",   "`A + B + C + A:C + B:C + A:B:C`", "`A + B + C + A:B + A
                "`A:C`",   "`A + B + C + A:B + B:C + A:B:C`", "`A + B + C + A:B + A
                "`B:C`",   "`A + B + C + A:B + A:C + A:B:C`", "`A + B + C + A:B + A
              "`A:B:C`",     "`A + B + C + A:B + A:C + B:C`", "`A + B + C + A:B + A
  ), col.names = c(  "Term being tested is",     "Null model is `outcome ~ ...`",  "A
```

Term being tested is	Null model is `outcome ~ ...`	Alternative model is `outcome ~ ...`
`A`	`B + C + A:B + A:C + B:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`B`	`A + C + A:B + A:C + B:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`C`	`A + B + A:B + A:C + B:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`A:B`	`A + B + C + A:C + B:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`A:C`	`A + B + C + A:B + B:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`B:C`	`A + B + C + A:B + A:C + A:B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`
`A:B:C`	`A + B + C + A:B + A:C + B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`

As ugly as that table looks, it's pretty simple. In all cases, the alternative hypothesis corresponds to the full model, which contains three main effect terms (e.g. `A`), three first order interactions (e.g. `A:B`) and one second order interaction (i.e., `A:B:C`). The null model always contains 6 of thes 7 terms: and the missing one is the one whose significance we're trying to test.

At first pass, Type III tests seem like a nice idea. Firstly, we've removed the asymmetry that caused us to have problems when running Type I tests. And because we're now treating all terms the same way, the results of the hypothesis tests do not depend on the order in which we specify them. This is definitely a good thing. However, there is a big problem when interpreting the results of the tests, especially for main effect terms. Consider the coffee data. Suppose it turns out that the main effect of `milk` is not significant according to the Type III tests. What this is telling us is that `babble ~ sugar + sugar:milk` is a better model for the data than the full model. But what does that even *mean*? If the interaction term `sugar:milk` was also non-significant, we'd be tempted to conclude that the data are telling us that the only thing that matters is `sugar`. But suppose we have a significant interaction term, but a non-significant main effect of `milk`. In this case, are we to assume that there really is an "effect of sugar", an "interaction between milk and sugar", but no "effect of milk"? That seems crazy. The right answer simply *must* be that it's meaningless[249] to talk about the main effect if the interaction is significant. In general, this seems to be what most statisticians advise us to do, and I think that's the right advice. But if it really is meaningless to talk about non-significant main effects in the presence of a significant interaction, then it's not at all obvious why Type III tests should allow the null hypothesis to rely on a model that includes the interaction but omits one of the main effects that make it up. When characterised in this fashion, the null hypotheses really don't make much sense at all.

Later on, we'll see that Type III tests can be redeemed in some contexts, but I'd better show you how to actually compute a Type III ANOVA first. The `anova()` function in R does not directly support Type II tests or Type III tests. Technically, you *can* do it by

creating the various models that form the null and alternative hypotheses for each test, and then using `anova()` to compare the models to one another. I outlined the gist of how that would be done when talking about Type I tests, but speaking from first hand experience[250] I can tell you that it's very tedious. In practice, the `anova()` function is only used to produce Type I tests or to compare specific models of particular interest (see Section 16.5). If you want Type II or Type III tests you need to use the `Anova()` function in the `car` package. It's pretty easy to use, since there's a `type` argument that you specify. So, to return to our coffee example, our Type III tests are run as follows:

```
mod <- lm( babble ~ sugar * milk, coffee )
 Anova( mod, type=3 )
```

```
## Anova Table (Type III tests)
##
## Response: babble
##             Sum Sq Df F value   Pr(>F)
## (Intercept) 41.070  1 155.839 3.11e-08 ***
## sugar        5.880  2  11.156 0.001830 **
## milk         4.107  1  15.584 0.001936 **
## sugar:milk   5.944  2  11.277 0.001754 **
## Residuals    3.162 12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As you can see, I got lazy this time and used `sugar * milk` as a shorthand way of referring to `sugar + milk + sugar:milk`. The important point here is that this is just a regular ANOVA table, and we can see that our Type III tests are significant for all terms, even the intercept.

Except, as usual, it's not that simple. One of the perverse features of the Type III testing strategy is that the results turn out to depend on the *contrasts* that you use to encode your factors (see Section 16.7 if you've forgotten what the different types of contrasts are). The results that I presented in the ANOVA table above are based on the R default, which is treatment contrasts; and as we'll see later, this is usually a very poor choice if you want to run Type III tests. So let's see what happens if switch to Helmert contrasts:

```
my.contrasts <- list( milk = "contr.Helmert", sugar = "contr.Helmert" )
mod.H <- lm( babble ~ sugar * milk, coffee, contrasts = my.contrasts )
Anova( mod.H, type=3 )
```

```
## Anova Table (Type III tests)
##
## Response: babble
##             Sum Sq Df   F value    Pr(>F)
## (Intercept) 434.29  1 1647.8882 3.231e-14 ***
## sugar         2.13  2    4.0446  0.045426 *
## milk          1.00  1    3.8102  0.074672 .
## sugar:milk    5.94  2   11.2769  0.001754 **
## Residuals     3.16 12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Oh, that's not good at all. In the case of `milk` in particular, the p-value has changed from .002 to .07. This is a pretty substantial difference, and hopefully it gives you a sense of how important it is that you take care when using Type III tests.

Okay, so if the p-values that come out of Type III analyses are so sensitive to the choice of contrasts, does that mean that Type III tests are essentially arbitrary and not to be trusted? To some extent that's true, and when we turn to a discussion of Type II tests we'll see that Type II analyses avoid this arbitrariness entirely, but I think that's too strong a conclusion. Firstly, it's important to recognise that some choices of contrasts will always produce the same answers. Of particular importance is the fact that if the columns of our contrast matrix are all constrained to sum to zero, then the Type III analysis will always give the same answers. This means that you'll get the same answers if you use `contr.Helmert` or `contr.sum` or `contr.poly`, but different answers for `contr.treatment` or `contr.SAS`.

```
random.contrasts <- matrix( rnorm(6), 3, 2 )   # create a random matrix
 random.contrasts[, 1] <- random.contrasts[, 1] - mean( random.contrasts[, 1] ) # co
 random.contrasts[, 2] <- random.contrasts[, 2] - mean( random.contrasts[, 2] ) # co
 random.contrasts  # print it to check that we really have an arbitrary contrast mat
```

```
##           [,1]       [,2]
## [1,] -1.523759  0.6891740
## [2,] -0.334936  0.9999209
## [3,]  1.858695 -1.6890949
```

```
 contrasts( coffee$sugar ) <- random.contrasts # random contrasts for sugar
 contrasts( coffee$milk ) <- contr.Helmert(2)  # Helmert contrasts for the milk fact
 mod.R <- lm( babble ~ sugar * milk, coffee )  # R will use the contrasts that we as
 Anova( mod.R, type = 3 )
```

```
## Anova Table (Type III tests)
##
## Response: babble
##             Sum Sq Df   F value    Pr(>F)
## (Intercept) 434.29  1 1647.8882 3.231e-14 ***
## sugar         2.13  2    4.0446  0.045426 *
## milk          1.00  1    3.8102  0.074672 .
## sugar:milk    5.94  2   11.2769  0.001754 **
## Residuals     3.16 12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Yep, same answers.

16.10.5 Type II sum of squares

Okay, so we've seen Type I and III tests now, and both are pretty straightforward: Type I tests are performed by gradually adding terms one at a time, whereas Type III tests are performed by taking the full model and looking to see what happens when you remove each term. However, both have some serious flaws: Type I tests are dependent on the order in which you enter the terms, and Type III tests are dependent on how you code up your contrasts. Because of these flaws, neither one is easy to interpret. Type II tests are a little harder to describe, but they avoid both of these problems, and as a result they are a little easier to interpret.

Type II tests are broadly similar to Type III tests: start with a "full" model, and test a particular term by deleting it from that model. However, Type II tests are based on the ***marginality principle*** which states that you should not omit a lower order term from your model if there are any higher order ones that depend on it. So, for instance, if your model contains the interaction `A:B` (a 2nd order term), then it really ought to contain the main effects `A` and `B` (1st order terms). Similarly, if it contains a three way interaction term `A:B:C`, then the model must also include the main effects `A`, `B` and `C` as well as the simpler interactions `A:B`, `A:C` and `B:C`. Type III tests routinely violate the marginality principle. For instance, consider the test of the main

effect of `A` in the context of a three-way ANOVA that includes all possible interaction terms. According to Type III tests, our null and alternative models are:

```
knitr::kable(tibble::tribble(
                  ~V1,                                                  ~V2,
         "Null model:",     "`outcome ~ B + C + A:B + A:C + B:C + A:B:C`",
  "Alternative model:", "`outcome ~ A + B + C + A:B + A:C + B:C + A:B:C`"
  ), col.names = c("", ""))
```

Null model:	`outcome ~ B + C + A:B + A:C + B:C + A:B:C`
Alternative model:	`outcome ~ A + B + C + A:B + A:C + B:C + A:B:C`

Notice that the null hypothesis omits `A`, but includes `A:B`, `A:C` and `A:B:C` as part of the model. This, according to the Type II tests, is not a good choice of null hypothesis. What we should do instead, if we want to test the null hypothesis that `A` is not relevant to our `outcome`, is to specify the null hypothesis that is the most complicated model that does not rely on `A` in any form, even as an interaction. The alternative hypothesis corresponds to this null model plus a main effect term of `A`. This is a lot closer to what most people would intuitively think of as a "main effect of `A`", and it yields the following as our Type II test of the main effect of `A`.[251]

```
knitr::kable(tibble::tribble(
                  ~V1,                                          ~V2,
         "Null model:",     "`outcome ~ B + C + B:C`",
  "Alternative model:", "`outcome ~ A + B + C + B:C`"
  ), col.names = c("", ""))
```

Null model:	`outcome ~ B + C + B:C`
Alternative model:	`outcome ~ A + B + C + B:C`

Anyway, just to give you a sense of how the Type II tests play out, here's the full table of tests that would be applied in a three-way factorial ANOVA:

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

                  "`A`",                 "`B + C + B:C`",                    "`A +
                  "`B`",                 "`A + C + A:C`",                    "`A +
                  "`C`",                 "`A + B + A:B`",                    "`A +
                "`A:B`",       "`A + B + C + A:C + B:C`",         "`A + B + C + A:B
                "`A:C`",       "`A + B + C + A:B + B:C`",         "`A + B + C + A:B
                "`B:C`",       "`A + B + C + A:B + A:C`",         "`A + B + C + A:B
              "`A:B:C`", "`A + B + C + A:B + A:C + B:C`", "`A + B + C + A:B + A:C +
  ), col.names = c(  "Term being tested is", "Null model is `outcome ~ ...`",  "Alte
```

Term being tested is	Null model is `outcome ~ ...`	Alternative model is `outcome ~ ...`
A	B + C + B:C	A + B + C + B:C
B	A + C + A:C	A + B + C + A:C
C	A + B + A:B	A + B + C + A:B

Term being tested is	Null model is `outcome ~ ...`	Alternative model is `outcome ~ ...`
`A:B`	`A + B + C + A:C + B:C`	`A + B + C + A:B + A:C + B:C`
`A:C`	`A + B + C + A:B + B:C`	`A + B + C + A:B + A:C + B:C`
`B:C`	`A + B + C + A:B + A:C`	`A + B + C + A:B + A:C + B:C`
`A:B:C`	`A + B + C + A:B + A:C + B:C`	`A + B + C + A:B + A:C + B:C + A:B:C`

In the context of the two way ANOVA that we've been using in the coffee data, the hypothesis tests are even simpler. The main effect of `sugar` corresponds to an F-test comparing these two models:

```
knitr::kable(tibble::tribble(
                  ~V1,                                          ~V2,
         "Null model:",     "`babble ~ milk`",
  "Alternative model:", "`babble ~ sugar + milk`"
  ), col.names = c("", ""))
```

Null model:	`babble ~ milk`
Alternative model:	`babble ~ sugar + milk`

The test for the main effect of `milk` is

```
knitr::kable(tibble::tribble(
                  ~V1,                                          ~V2,
         "Null model:",     "`babble ~ sugar`",
  "Alternative model:", "`babble ~ sugar + milk`"
  ), col.names = c("", ""))
```

Null model:	`babble ~ sugar`
Alternative model:	`babble ~ sugar + milk`

Finally, the test for the interaction `sugar:milk` is:

```
knitr::kable(tibble::tribble(
                  ~V1,                                          ~V2,
         "Null model:",     "`babble ~ sugar + milk`",
  "Alternative model:", "`babble ~ sugar + milk + sugar:milk`"
  ), col.names = c("", ""))
```

Null model:	`babble ~ sugar + milk`
Alternative model:	`babble ~ sugar + milk + sugar:milk`

Running the tests are again straightforward. We use the `Anova()` function, specifying `type=2` :

```
mod <- lm( babble ~  sugar*milk, coffee )
 Anova( mod, type = 2 )
```

```
## Anova Table (Type II tests)
##
## Response: babble
##             Sum Sq Df F value   Pr(>F)
## sugar       3.0696  2  5.8238 0.017075 *
## milk        0.9561  1  3.6279 0.081061 .
## sugar:milk 5.9439  2 11.2769 0.001754 **
## Residuals  3.1625 12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Type II tests have some clear advantages over Type I and Type III tests. They don't depend on the order in which you specify factors (unlike Type I), and they don't depend on the contrasts that you use to specify your factors (unlike Type III). And although opinions may differ on this last point, and it will definitely depend on what you're trying to do with your data, I do think that the hypothesis tests that they specify are more likely to correspond to something that you actually care about. As a consequence, I find that it's usually easier to interpret the results of a Type II test than the results of a Type I or Type III test. For this reason, my tentative advice is that, if you can't think of any obvious model comparisons that directly map onto your research questions but you still want to run an ANOVA in an unbalanced design, Type II tests are probably a better choice than Type I or Type III.[252]

16.10.6 Effect sizes (and non-additive sums of squares)

The `etaSquared()` function in the `lsr` package computes η2 and partial η2 values for unbalanced designs and for different Types of tests. It's pretty straightforward. All you have to do is indicate which `type` of tests you're doing,

```
etaSquared( mod, type=2 )
```

```
##                eta.sq eta.sq.part
## sugar      0.22537682   0.4925493
## milk       0.07019886   0.2321436
## sugar:milk 0.43640732   0.6527155
```

and out pops the η^2 and partial η^2 values, as requested. However, when you've got an unbalanced design, there's a bit of extra complexity involved. To see why, let's expand the output from the `etaSquared()` function so that it displays the full ANOVA table:

```
es <- etaSquared( mod, type=2, anova=TRUE )
es
```

```
##                eta.sq eta.sq.part        SS df        MS         F
## sugar      0.22537682   0.4925493 3.0696323  2 1.5348161  5.823808
## milk       0.07019886   0.2321436 0.9561085  1 0.9561085  3.627921
## sugar:milk 0.43640732   0.6527155 5.9438677  2 2.9719339 11.276903
## Residuals  0.23219530          NA 3.1625000 12 0.2635417        NA
##                       p
## sugar      0.017075099
## milk       0.081060698
## sugar:milk 0.001754333
## Residuals           NA
```

Okay, if you remember back to our very early discussions of ANOVA, one of the key ideas behind the sums of squares calculations is that if we add up all the SS terms associated with the effects in the model, and add that to the residual SS, they're supposed to

add up to the total sum of squares. And, on top of that, the whole idea behind η^2 is that – because you're dividing one of the SS terms by the total SS value – is that an η^2 value can be interpreted as the proportion of variance accounted for by a particular term.

Now take a look at the output above. Because I've included the η^2 value associated with the residuals (i.e., proportion of variance in the outcome attributed to the residuals, rather than to one of the effects), you'd expect all the η^2 values to sum to 1. Because, the whole idea here was that the variance in the outcome variable can be divided up into the variability attributable to the model, and the variability in the residuals. Right? Right? And yet when we add up the η^2 values for our model…

```
sum( es[,"eta.sq"] )
```

```
## [1] 0.9641783
```

… we discover that for Type II and Type III tests they generally don't sum to 1. Some of the variability has gone "missing". It's not being attributed to the model, and it's not being attributed to the residuals either. What's going on here?

Before giving you the answer, I want to push this idea a little further. From a mathematical perspective, it's easy enough to see that the missing variance is a consequence of the fact that in Types II and III, the individual SS values are not obliged to the total sum of squares, and will only do so if you have balanced data. I'll explain why this happens and what it means in a second, but first let's verify that this is the case using the ANOVA table. First, we can calculate the total sum of squares directly from the raw data:

```
ss.tot <- sum( (coffee$babble - mean(coffee$babble))^2 )
 ss.tot
```

```
## [1] 13.62
```

Next, we can read off all the SS values from one of our Type I ANOVA tables, and add them up. As you can see, this gives us the same answer, just like it's supposed to:

```
 type.I.sum <- 3.5575 + 0.9561 + 5.9439 + 3.1625
 type.I.sum
```

```
## [1] 13.62
```

However, when we do the same thing for the Type II ANOVA table, it turns out that the SS values in the table add up to slightly less than the total SS value:

```
type.II.sum <- 0.9561 + 3.0696 + 5.9439 + 3.1625
type.II.sum
```

```
## [1] 13.1321
```

So, once again, we can see that there's a little bit of variance that has "disappeared" somewhere.

Okay, time to explain what's happened. The reason why this happens is that, when you have unbalanced designs, your factors become correlated with one another, and it becomes difficult to tell the difference between the effect of Factor A and the effect of Factor B. In the extreme case, suppose that we'd run a 2×2 design in which the number of participants in each group had been as follows:

```
knitr::kable(tibble::tribble(
        ~V1,     ~V2,        ~V3,
     "milk",   "100",        "0",
  "no milk",     "0",      "100"
  ), col.names = c(         "", "sugar", "no sugar"))
```

	sugar	no sugar
milk	100	0
no milk	0	100

Here we have a spectacularly unbalanced design: 100 people have milk and sugar, 100 people have no milk and no sugar, and that's all. There are 0 people with milk and no sugar, and 0 people with sugar but no milk. Now suppose that, when we collected the data, it turned out there is a large (and statistically significant) difference between the "milk and sugar" group and the "no-milk and no-sugar" group. Is this a main effect of sugar? A main effect of milk? Or an interaction? It's impossible to tell, because the presence of sugar has a perfect association with the presence of milk. Now suppose the design had been a little more balanced:

```
knitr::kable(tibble::tribble(
        ~V1,     ~V2,        ~V3,
     "milk",   "100",        "5",
  "no milk",     "5",      "100"
  ), col.names = c(         "", "sugar", "no sugar"))
```

	sugar	no sugar
milk	100	5
no milk	5	100

This time around, it's technically possible to distinguish between the effect of milk and the effect of sugar, because we have a few people that have one but not the other. However, it will still be pretty difficult to do so, because the association between sugar and milk is still extremely strong, and there are so few observations in two of the groups. Again, we're very likely to be in the situation where we *know* that the predictor variables (milk and sugar) are related to the outcome (babbling), but we don't know if the *nature* of that relationship is a main effect of one predictor, or the other predictor or the interaction.

This uncertainty is the reason for the missing variance. The "missing" variance corresponds to variation in the outcome variable that is clearly attributable to the predictors, but we don't know which of the effects in the model is responsible. When you calculate Type I sum of squares, no variance ever goes missing: the sequentiual nature of Type I sum of squares means that the ANOVA automatically attributes this variance to whichever effects are entered first. However, the Type II and Type III tests are more conservative. Variance that cannot be clearly attributed to a specific effect doesn't get attributed to any of them, and it goes missing.