14.4: 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…

14.4.1 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."[205] 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…

14.4.2 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…

```
names( my.anova )
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "contrasts"     "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  3.453333  1.391667
## Deg. of Freedom        2        15
##
## Residual standard error: 0.3045944
## 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 SS_b=3.45, the within groups sums of squares is SS_w=1.39, and the degrees of freedom are 2 and 15 repectively.

14.4.3 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.[206] When we do that…

```
summary( my.anova )
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## drug         2  3.453  1.7267   18.61 8.65e-05 ***
## Residuals   15  1.392  0.0928
## ---
## 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.

This page titled 14.4: Running an ANOVA in R is shared under a CC BY-SA 4.0 license and was authored, remixed, and/or curated by Danielle Navarro via source content that was edited to the style and standards of the LibreTexts platform.