***Learning Statistics w/ R - University of Adelaide***

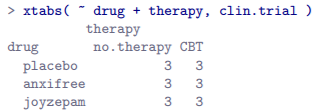
***Part V – Statistical Tools***

**16. Factorial ANOVA**

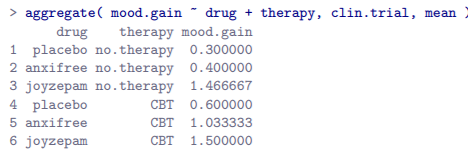
* Tools to compare 2 groups = most notably the **t-test** (Ch. 13) + then **analysis of variance (ANOVA)** as a method for comparing more than 2 groups (Ch. 14).
* Regression (Ch. 15) = somewhat different topic, but introduced a powerful new idea: *building statistical models that have MULTIPLE predictor variables used to explain a SINGLE outcome variable*
* A regression model could be used to predict # of errors a student makes in a reading comprehension test based on: 1) # of hours studied for the test + 2) their score on a standardized IQ test.
* Want to import this idea into the ANOVA framework
* Suppose we’re interested in using a reading comprehension test to measure student achievements in 3 different schools, + we also suspect girls + boys are developing at different rates (i.e. expected to have different performance on average).
* Each student is classified in 2 ways: gender + school 🡪 want to analyze scores in terms of both
* Tool for doing so = **factorial ANOVA** or **two-way ANOVA**, in contrast to the 1-way ANOVAs in Ch. 14

**16.1 Factorial ANOVA : Balanced designs, NO interactions**

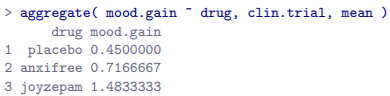
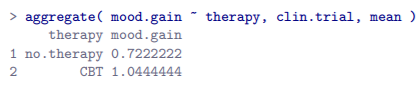
* W/ simple ANOVA, we assumed a fairly simple experimental design*: each person falls into 1 of several groups + we test if these groups have different means on some outcome variable*
* Broader class of experimental designs = **factorial design:** > 1 grouping variable.
* Ex: Effect of different drugs on mood experienced by each person
* When we did this, we found a significant effect of drug, but also ran an analysis to see if there was an effect of therapy + didn’t find one
* Something’s worrying about trying to run 2 separate analyses trying to predict the same outcome
* *Maybe there actually IS an effect of therapy on mood, but we couldn’t find it b/c it was being hidden by the effect of drug*
* Want to run a *single* analysis to include BOTH drug AND therapy as predictors.
* For this analysis, each person is **cross-classified** by drug given (factor w/ 3 levels) + therapy received (factor w/ 2 levels).
* Refer to this as a 3\*2 factorial design.
* Cross-tabulate drug by therapy using **xtabs()** + get the following table:



* Not only do we have participants corresponding to all possible combinations of the 2 factors, indicating our design is completely crossed, it turns out there are an equal # of people in each group = a **balanced design**.
* Balanced designs = simplest case.
* The story for *unbalanced* designs is quite tedious, so put it aside for the moment.
* Like one-way ANOVA, **factorial ANOVA** = a tool for testing certain types of hypotheses about *population means*.
* Sensible place to start = to be explicit about *what the hypotheses actually are.*
* However, before that point, it’s useful to have clean + simple notation to describe population means
* B/c observations are cross-classified in terms of 2 different factors, there are quite a lot of different means one might be interested in.
* Start by thinking about all different sample means we can calculate for this kind of design.
* Might be interested in this table of group means:



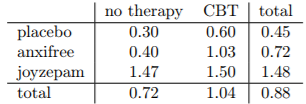
* See a cross-tabulation of group means for all possible combinations of the 2 factors (people w/ placebo + no therapy, people w/ placebo while getting CBT, etc.).
* However, can also construct tables that *ignore* 1 of the 2 factors.

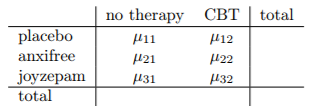
* If we can ignore 1 factor we can ignore both = calculating average mood gain across all 18 participants, regardless of drug or therapy



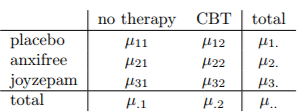
* At this point we have 12 different sample means to keep track of
* Organize all these numbers into a single table



* Each of these different means is a **sample statistic** = a quantity pertaining to specific observations made during a study.
* We *want* to make **inferences** about the corresponding **population parameters** = the TRUE means as they exist w/in some broader population.
* Those population means can also be organized into a similar table



* Each row = different level of Factor A (drug) + each column = level of Factor B (therapy)
* Let R = # of rows in the table + C = # of columns 🡺 refer to this as an **R\*C factorial ANOVA**.
* µ.rc = population mean associated w/ the r-th level of Factor A (row r) + the c-th level of Factor B (column c)
* Nice thing about subscript notation = it generalizes nicely
* If an experiment had involved a 3rd factor, we could just add a 3rd subscript.
* In principle, the notation extends to as many factors as you might care to include
* How to describe average mood gain across entire (hypothetical) population of people who might be given Joyzepam in an experiment like this, regardless of whether they were in CBT?
* Use **dot notation** to express this.
* In the case of Joyzepam: talking about a mean associated w/ the 3rd row in the table = averaging across 2 cell means (µ31 + µ32).
* Result = a **marginal mean** denoted µ3.
* Marginal mean for CBT = population mean associated w/ the 2nd column in the table, µ.2
* Grand mean = µ.. = mean obtained by averaging (**marginalizing**) over both.
* Technically, marginalizing != a regular mean, but = a **weighted average 🡪** *takes into account the frequency of the different events you’re averaging over.*
* However, w/ a balanced design, all cell frequencies are equal by definition, so the 2 are equivalent



* It is straightforward to formulate + express some hypotheses.
* Suppose goal = to find out 2 things
* 1) Does choice of drug have any effect on mood?
* 2) Does CBT have any effect on mood?
* These aren’t the only hypotheses we could formulate, but these are the 2 simplest to test
* 1st, if **drug** has no effect, we’d expect all row means, **µr.** to be identical = our null.
* On the other hand, if drug *does* matter, row means = different.
* Formally, we write down null + alternative in terms of the *equality of marginal means*:

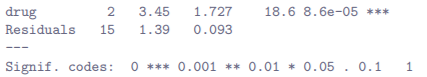


* These are the *exact same statistical hypotheses* formed w/ a one-way ANOVA but now w/ letters along w/ µ
* We’re talking about the same hypothesis, just the more-complicated ANOVA requires more careful notation due to the presence of multiple grouping variables
* Although hypothesis is identical, the *test* of that hypothesis is subtly different b/c we’re now acknowledging the existence of the SECOND grouping variable.
* 2nd hypothesis test is formulated the same way, but the null now corresponds to equality of *column* means:

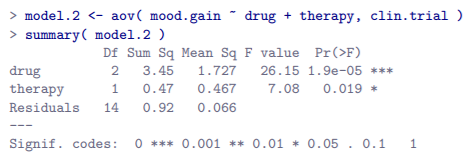


* These are basically the same as hypotheses testing in simpler one-way ANOVAs in Ch. 14.
* Expecting to see sums of squares (SS), mean squares (MS), degrees of freedom (df), + finally an F-statistic we can convert into a p-value
* If data you’re trying to analyze correspond to a balanced factorial design, running ANOVA is easy.
* Start by reproducing original analysis from Ch. 14 (single factor (drug) to predict outcome (mood)





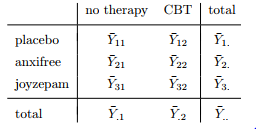
* Suppose we’re also curious to find out if **therapy** has a relationship to mood.
* All we have to do is extend the formula 🡺 specify mood.gain ~ drug + therapy



* Output = pretty simple to read 🡪 1st row = **between-group sum of squares (SS)** associated w/ **drug** along w/ a corresponding **between-group df**.
* It also calculates a **mean square value (MS),** **F-statistic**, a **p-value**.
* There is also a rows corresponding to **therapy** + to the **residuals** (**w/in groups variation**).
* Not only are all individual quantities familiar, the relationships between these different quantities has remained unchanged:
* Just like w/ one-way ANOVA, the **mean square** value = dividing SS by corresponding df regardless of talking about drug, therapy or residuals.
* Take it on faith that R calculates SS values correctly + try to verify all the rest of the #’s make sense.
* Note for drug, we divide 3.45 / 2, + end up w/ a **mean square** = 1.73 + for therapy, there’s only 1 dF, so 0.47 (SS) / 1 = 0.47 (MS).
* For F- statistics + p values, notice we have 2 of each: 1 corresponding to drug + 1 to therapy
* Regardless of factor, F-statistic = dividing MS value associated w/ the factor by the MS value associated w/ the residuals.
* A = 1st factor (drug) + R = residuals, then F-statistic associated w/ factor A = F.A is calculated as follows: **FA = MS.A / MS.R** + an equivalent formula exists for factor B (therapy).
* For **drugs**, take MS = 1.73 / residual MS 0.07 🡪 F-statistic = 26.15.
* Corresponding for therapy 🡪 divide 0.47 / 0.07 = 7.08 as the F-statistic.
* Last part of ANOVA table = calculation of p-values
* For each of our 2 factors, we’re trying to test the null that there is NO relationship between that factor + outcome
* To that end, we’ve (apparently) followed a similar strategy to one-way ANOVA + have calculated an F-statistic for each hypotheses.
* To convert these to p-values, must note the sampling distribution for the F-statistic under the null (*factor in question is irrelevant*) is an **F distribution** + that the 2 dF values are those corresponding to the factor + those corresponding to the residuals.
* For drug = F-distribution w/ 2 + 14 Df + for therapy, sampling distribution is F w/ 1 + 14 dF
* If we really wanted to, could calculate p-value using pf()



* This is indeed the p value reported in the ANOVA table above.
* ANOVA table for a more-complicated analysis should be read in much the same way as the ANOVA table for a simpler analysis
* In short, it’s telling us the factorial ANOVA for our 3\*2 design found a significant effect of drug (F(2,14) = 26.15, p < .001) as well as a significant effect of therapy (F(1,14) = 7.08, p < .02).
* More technically correct terminology 🡪 “*There are 2* ***main effects*** *of drug + therapy*”
* Probably seems redundant to refer to these as **main effects**, but it makes sense.
* Later on, we talk about the possibility of interactions between the 2 factors, + so we make a distinction between **main effects** + **interaction effects**.
* It’s genuinely true that factorial ANOVA = built in more or less the same way as a simpler one-way ANOVA model, but this feeling of familiarity starts to evaporate in the details.
* Remember, the hypothesis tests for the **main effects** (drug + therapy) are F-tests
* Assume for now we have only 2 predictor variables, Factor A + Factor B.
* If Y = outcome, use Y(r,c,i) = outcome associated w/ the i-th member of group (r,c) (level/row r for Factor A + level/column c for Factor B).
* Thus, if Y¯ = a sample mean, can use the same notation as before to refer to group means, marginal means + grand means:
* Y¯rc = sample mean associated w/ the r-th level of Factor A + the c-th level of Factor B,
* Y¯r. = marginal mean for the r-th level of Factor A
* Y¯.c = marginal mean for the c-th level of Factor B
* Y¯ .. = grand mean.
* In other words, our sample means can be organized into the same table as the population means.



* Drug has 3 levels + therapy has 2 levels, + so we’re trying to run a 3\*2 factorial ANOVA.
* Factor A (row factor) has R levels + Factor B (column factor) has C levels = R\*C factorial ANOVA.
* Now compute the sum of squares values for each of the 2 factors in a relatively familiar way.
* Factor A 🡪 between-group sum of squares = calculated by assessing the extent to which the (row) marginal means Y¯ 1. , Y¯ 2. etc., are different from the grand mean Y¯.. 🡺 calculate the sum of squared difference between Y¯i. values + Y¯.. values.
* Specifically, if there are N people in each group, calculate:



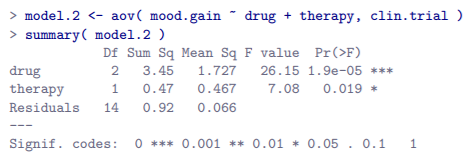
* As w/ one-way ANOVA, the most interesting part of this formula is



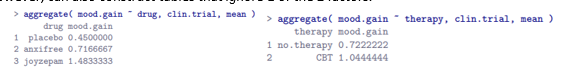
* This corresponds to the squared deviation associated w/ level r.
* Calculates this squared deviation for all R levels of the factor, adds them up, + then multiplies the result by N \* C.
* This last part = there are multiple cells in our design that have level r on Factor A + in fact, there are C of them, 1 corresponding to each possible level of Factor B
* There are 2 different cells in the design corresponding to anxifree: for people w/ no.therapy, + for CBT group.
* Not only that, w/in each of these cells there are N observations.
* So, to convert our SS value into a quantity that calculates the between-groups sum of squares on a *per observation* basis, we have to multiply by N \* C.
* The formula for factor B is of course the same thing, just w/ some subscripts shuffled around:



* Now that we have these formulas, we can check them against the R output from earlier



* We calculated all marginal means (row marginal means Y¯r. + column marginal means Y¯.c)



* And also calculated the grand mean



* Repeat those calculations,



* Calculate the sum of squares associated w/ the main effect of drug for the total of N = 3 people in each group, + C = 2 different types of therapy.
* Or, to put it another way, there are 3\*2 = 6 people who received any particular drug.



* Repeat the same kind of calculation for the main effect of therapy, again w/ N = 3 people in each group, but w/ R = 3 different drugs 🡺 3\*3 = 9 people who received CBT + 9 who received a placebo



* These SS values are analogous to the between-group sum of squares values from one-way ANOVA
* However, it’s *not a good idea to think of them as between-groups SS values anymore b/c we have 2 different grouping variables + it’s easy to get confused.*
* For an F test 🡪 also need to calculate **w/in-groups sum of squares**.
* Can refer to **w/in-groups SS** as the **residual sum of squares, SSres**.
* Easiest way to think about the residual SS values in *this* context = leftover variation in the outcome variable after taking into account the differences in the marginal means (after removing SS.A + SS.B)
* Start by calculating **total sum of squares** in the same way as for one-way ANOVA = take the difference between each observation Y(r, c, i) + the grand mean Y¯.., square the differences, + add them all up



* Triple summation = looks more complicated than it is.
* In the 1st 2 = summing across all levels of Factor A/possible rows, across all levels of Factor B/all possible columns c)
* Each r, c combination corresponds to a *single group*, + each group contains N people, so we have to sum across all those people (all i values) too.
* In other words, we’re summing across all observations in the data set/possible r, c, I combos
* At this point, we know the total variability of the outcome variable, SS.T, + know how much of that variability can be attributed to Factor A (SS.A) + to Factor B (SS.B)
* The **residual sum of squares** is thus defined to be the variability in Y that CAN’T be attributed to either of our 2 factors.



* Whole point of calling it a residual = it’s the *leftover* variation
* Common to refer to **SS.A + SS.B** as the variance attributable to the ANOVA model**, SS.M**,
* Often say **total sum of squares = the model sum of squares + the residual sum of squares**
* Not a surface similarity: ANOVA + regression are actually the same thing under the hood
* To get SSR, 1st calculate total sum of squares:



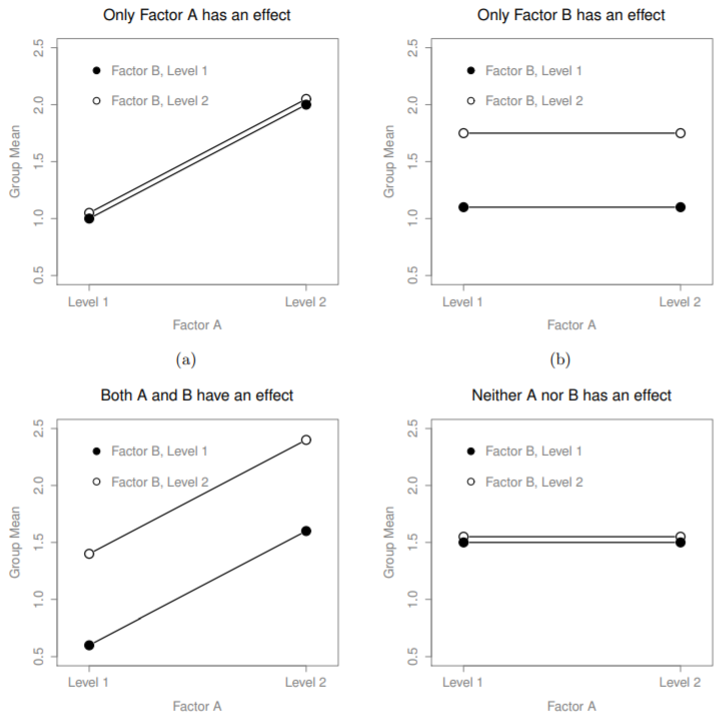
* Then we use it to calculate residual sum of squares (see its same as in ANOVA table)

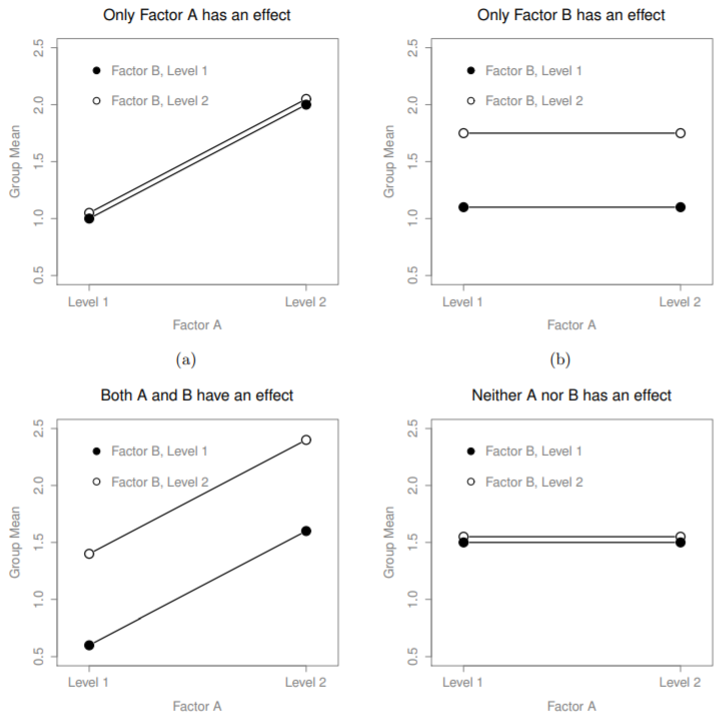


* The dF are calculated in much the same way as for one-way ANOVA.
* For any given factor, dF = # of levels minus 1 (R - 1 for row variable/Factor A, C - 1 for column variable/Factor B).
* So, for drug, df = 2, + for therapy df = 1.
* For drug, we observe 3 separate group means, which are constrained by 1 grand mean so therefore the dF = 2.
* For the residuals, the logic is similar, but not quite the same 🡪 total # of observations in our experiment = 18 + the constraints correspond to the 1 grand mean, the 2 additional group means drug introduces, + the 1 additional group mean therapy introduces, + so dF = 14.
* As a formula, this is  = 
* One-way ANOVA looked to see if there are any differences between drugs + a 2nd one-way ANOVA checked if there were any differences between therapies.
* Null + alternatives tested by the 1-way ANOVAs = identical to those tested by the factorial ANOVA
* Looking even more carefully at the ANOVA tables, the sum of squares associated w/ the factors are identical in the 2 different analyses (drug = 3.45 + therapy = 0.92), as are dF (drug = 2, therapy = 1)
* *But they don’t give the same answers*
* Most notably, the one-way ANOVA for therapy showed no significant effect (p-value = 0.21).
* However, main effect of therapy in the two-way ANOVA DID get a significant effect (p = .019)
* The 2 analyses are clearly not the same b/c of how the residuals are calculated.
* The whole idea behind an F-test = to compare the variability that can be attributed to a particular factor w/ the variability that CANNOT be accounted for (the residuals).
* A one-way ANOVA for therapy ignores the effect of drug, so **ANOVA dumps all the drug-induced variability into the residuals**
* This makes the data look more noisy than they really are, + the effect of therapy *which was correctly found to be significant in the two-way ANOVA* now becomes non-significant.
* If we ignore something that *actually matters* (drug) when trying to assess the contribution of something else (therapy), our analysis will be distorted.
* Perfectly okay to ignore variables that’re genuinely irrelevant to a phenomenon of interest
* Ex: color of the walls turned out to be non-significant in a three-way ANOVA (mood.gain ~ drug + therapy + wall.colour) = perfectly okay to disregard it
* **What you shouldn’t do is drop variables that actually make a difference**

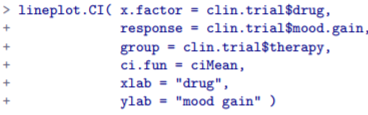
**16.2 Factorial ANOVA 2: balanced designs, interactions allowed**

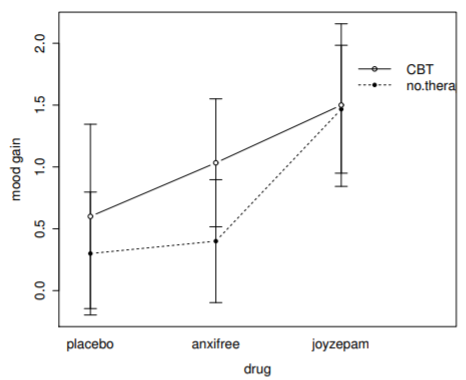
* The ANOVA model we’ve been talking about so far covers a range of different patterns we might observe in data.
* In a two-way ANOVA design, there are 4 possibilities:
* (a) only Factor A matters
* (b) only Factor B matters
* (c) both A + B matter
* (d) neither A nor B matters.



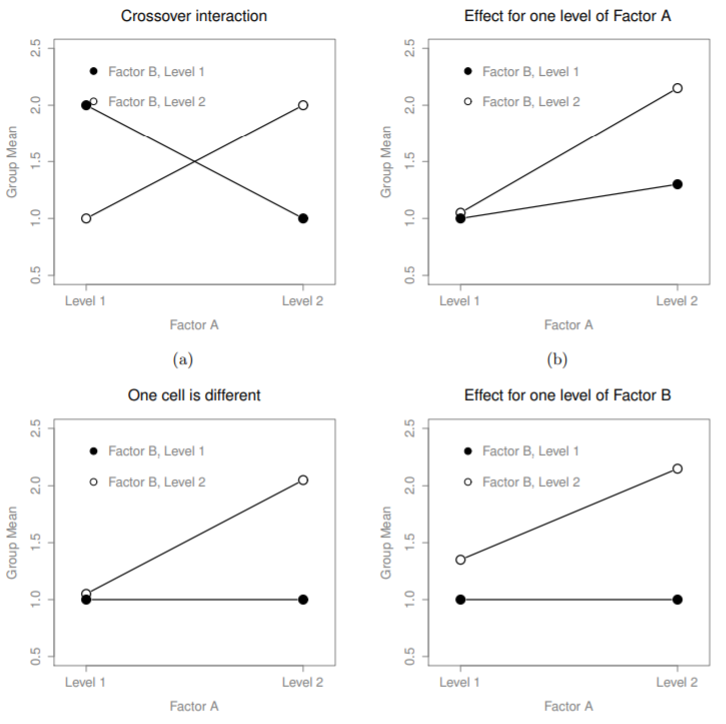


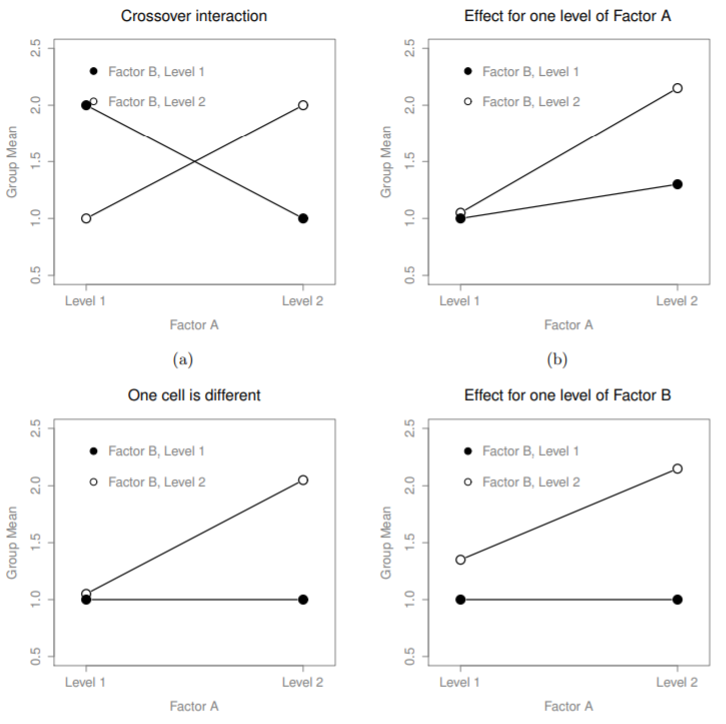
* These 4 patterns of data are all quite realistic: there are many data sets that produce exactly those patterns.
* However, they are not the whole story, + the ANOVA model we’ve been talking about so far is not sufficient to fully account for a table of group means.
* So far, we’ve talked about the idea that both drugs + therapy can influence mood, but no way of talking about the possibility of an *interaction* between the 2.
* An **interaction between A + B** is said to occur *whenever the effect of Factor A is different, depending on the level of Factor B*.
* Suppose operation of Anxifree + Joyzepam is governed quite different physiological mechanisms, + 1 consequence of this = Joyzepam has more or less the same effect on mood regardless of whether one is in therapy, Anxifree = much more effective when administered in conjunction w/ CBT
* *The ANOVA developed prior does not capture this idea*.
* To get some idea of whether an interaction is actually happening here, it helps to plot the various group means, say w/ **interaction.plot**(), which won’t draw error bars for you.
* To include error bars, use **lineplot.CI**() in the **sciplots** package
* don’t forget **ciMean**() is in **lsr** package, so you need to have lsr loaded





* Our main concern = the fact that the 2 lines aren’t parallel
* Effect of CBT (difference between solid + dotted lines) when drug = Joyzepam appears to be ~0, even smaller than the effect of CBT when a placebo is used
* However, w/ Anxifree, the effect of CBT is larger than w/ placebo
* Is this effect *real*, or is this just random variation due to chance?
* Our original ANOVA cannot answer this question, b/c we make no allowances for the idea interactions even exist
* Key idea = an **interaction effect**
* **mood.gain ~ drug + therapy + drug:therapy** = there’re only 2 factors involved in a model (drug + therapy), there are 3 distinct terms (drug, therapy + drug:therapy).
* In addition to the main effects of drug + therapy, we have a new component to the model, the **interaction term** drug:therapy
* Idea behind an interaction effect = *effect of Factor A is different depending on the level of Factor B*
* But what does that mean in terms of data?

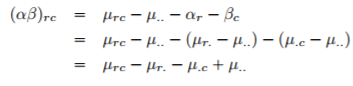




* Here are several patterns that, although different, all count as an interaction effect.
* It’s not entirely straightforward to translate this qualitative idea into something mathematical a statistician can work w/.
* As a consequence, the idea of an interaction effect being formalized in terms of null + alternative hypotheses is slightly difficult
* Basic idea = Be a little more explicit about our main effects.
* Consider main effect of Factor A (drug), originally formulated in terms of the null that the 2 marginal means **µr.** are all equal to each other.
* Obviously, if all these are equal to each other, they must also be equal to the grand mean µ.. as well,
* So we define the effect of Factor A at level r = the difference between the marginal mean µr. + the grand mean µ... 🡪 **αr** = µr. - µ..
* By definition, all αr values must sum to 0, for the same reason the average of the marginal means µr. must be = grand mean µ...
* Can similarly define main effect of Factor B at level i to be the difference between the column marginal mean **µ.c** + the grand mean **µ..** 🡪 βc = µ.c - µ..
* Once again, βc values must sum to 0.
* αr + βc values allows one to be precise about what it means to say that there is no interaction effect.
* If there is NO interaction at all, αr + βc will perfectly describe the group means µ(r, c).
* Specifically, it means µ(r, c) = µ.. + αr + βc
* That is, *there’s nothing special about the group means that you couldn’t predict perfectly by knowing all the marginal means* 🡺 that’s our null
* The alternative is that µ(r, c) != µ.. + αr + βc for *at* *least 1 group* (r, c)
* Statisticians usually define the specific interaction associated w/ group (r, c) to be some number, αβ(r, c), so the alternative is that **µ(r, c) = µ.. + αr + βc + αβ(r, c)** where *αβ(r, c) is non-zero* for at least 1 group.
* To calculate the **sum of squares for the interaction terms**, **SS.A:B**, 1st notice how the interaction effect is defined in terms of *the extent to which the actual group means differ from what you’d expect by just looking at the marginal means.*
* The previous refer to population parameters rather than sample statistics = don’t actually know what they are 🡪 can estimate them using sample means in place of population means.
* For Factor A, a good way to estimate the main effect at level r = difference between the sample marginal mean Y¯(r, c) + the sample grand mean Y¯..
* We’d use this as our estimate of the effect 🡪 
* Similarly, estimate of the main effect of Factor B at level c can be defined as follows: 
* Going back to the formulas used to describe the SS values for the 2 main effects, notice these effect terms are exactly the quantities we were squaring + summing

* What’s the analog of this for interaction terms? 🡪 1st rearrange the formula for the group means µ(r, c) under the alternative:



* Then substitute sample statistics in place of population means to get an estimate of the interaction effect for group (r, c),



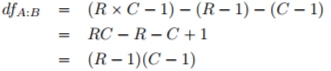
* Now all we have to do is sum all these estimates across all R levels of Factor A + all C levels of Factor B to get the formula for the **sum of squares associated w/ the interaction** as a whole:



* We multiply by N b/c there are N observations in each group, + we want SS values to reflect the variation *among observations* accounted for by the interaction, NOT the variation among *groups*.
* Now, w/ a formula for calculating SS.A:B, recognize that the interaction term is part of the model, so the total sum of squares associated w/ the model, SS.M = the sum of the 3 relevant SS values,
* The residual sum of squares SS.R is still “the leftover variation”, namely SST - SSM, but w/ the interaction term this becomes:



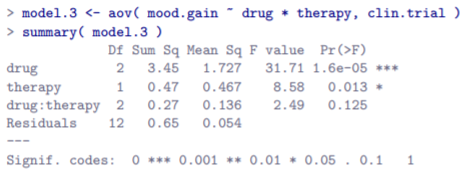
* As a consequence, SS.R will be smaller than in our original ANOVA that didn’t include interactions.
* Calculating interaction dF is, again, slightly trickier than corresponding calculation for main effects.
* To start, think about the ANOVA model as a whole.
* Once we include interaction effects in the model, we’re allowing every single group a unique mean, µ(r, c)
* For an R\*C factorial ANOVA, this means there are R\*C quantities of interest in the model, + only the 1 constraint: *all group means need to average out to the grand mean.*
* i.e. the model as a whole needs to have (R\*C) - 1 dF.
* But main effect of Factor A has R - 1 dF + main effect of Factor B has C - 1 dF, which means the dF associated w/ the interaction is



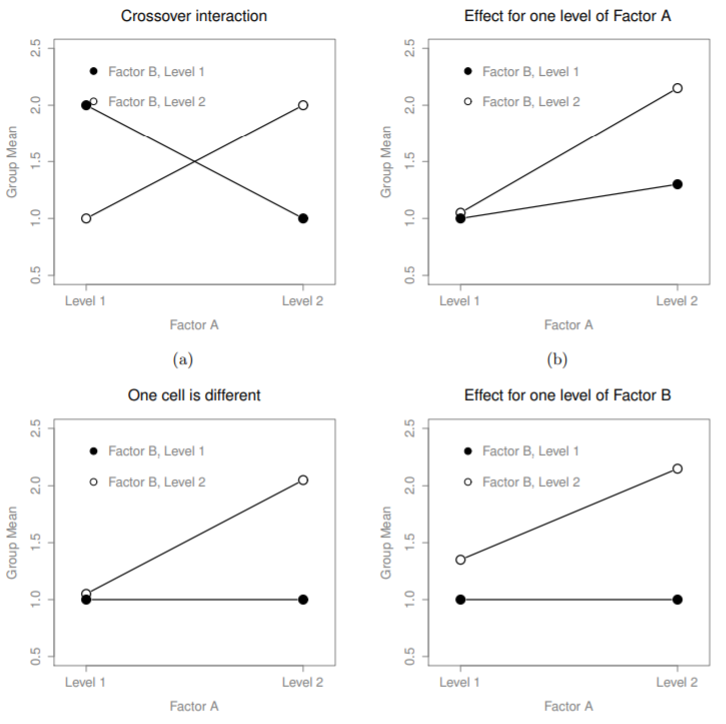
* This is the product of the dF associated w/ the row factor + the dF associated w/ the column factor
* B/c we’ve added interaction terms, which absorb some dF, there are fewer residual dF left over.
* Specifically, note if the model w/ interaction has a total of (R\*C) – 1 dF, + there are N observations in the data set constrained to satisfy 1 grand mean, then the residual dF now become N - (R\*C) – 1 + 1 = **N - (R\*C)**
* Adding interaction terms to the ANOVA model in R is straightforward 🡪 include the interaction term drug:therapy.



OR



* While we DO have a significant main effect of both drug (F(2,12) = 31.7, p < .001) + therapy (F(1,12) = 8.6, p = .013), *there is no significant interaction between the 2* (F(2,12) = 2.5, p = 0.125).
* There’s a couple of important things to consider when interpreting the results of factorial ANOVA
* 1st, the same issue w/ one-way ANOVA 🡪 if you obtain a significant main effect of (say) drug, it *doesn’t tell you anything about which drugs are different to one another*.
* To find that out, you need to run additional analyses (Sections 16.7 + 16.8)
* The same is true for interaction effects: knowing there’s a significant interaction doesn’t tell you anything about what kind of interaction exists.
* Again, you’ll need to run additional analyses.
* Secondly, there’s a very peculiar interpretation issue that arises when you *obtain a significant interaction effect but no corresponding main effect.*
* This happens sometimes 🡪 like in the crossover interaction



* In this case, neither main effect would be significant, but the *interaction effect* would be.
* This is difficult to interpret
* General advice = don’t pay much attention to the main effects when an interaction is present
* Although the tests of the main effects are perfectly valid from a mathematical point of view, when there is a significant interaction effect, the main effects rarely test interesting hypotheses.
* Recall that the null for a main effect = the marginal means are equal to each other, + recall that a marginal mean is formed by averaging across several different groups.
* But w/ a significant interaction effect, you know the groups that comprise the marginal mean aren’t homogeneous, so it’s not obvious why you’d even care about those marginal means.
* Suppose we had a 2\*2 design comparing 2 different treatments for phobias (systematic desensitization + flooding), + 2 different anxiety-reducing drugs (Anxifree + Joyzepam).
* Suppose we found Anxifree had no effect w/ desensitization, + Joyzepam had no effect w/ flooding, but both were effective for the other treatment.
* This is a classic crossover interaction + the ANOVA would show no main effect of drug, but a significant interaction.
* To say there’s “no main effect” means if we average over the 2 different treatments, the average effect of Anxifree + Joyzepam is the same.
* This is important b/c when treating someone for phobias, it is never the case a person can be treated using an *average* of flooding + desensitization 🡪 doesn’t make sense.
* For 1 treatment, 1 drug is effective; + for the other treatment, the other drug is effective = The **interaction** is the important thing + the main effect is irrelevant.
* This sort of thing happens a lot: the **main effect tests** = tests of marginal means, + when an interaction is present we often find ourselves not being interested in marginal means b/c they imply averaging over things that the interaction tells us shouldn’t be averaged
* **It’s not always the case that a main effect is meaningless when an interaction is present**.
* Often you can get a big main effect + a very small interaction, in which case you can still say things like “drug A is generally more effective than drug B”, b/c there was a big effect of drug A, but you’d need to modify it by adding that the difference in effectiveness was different for different psychological treatments.
* In any case, the main point = whenever you get a significant interaction, stop + think about what the main effect actually means in this context.
* Don’t automatically assume that the main effect is interesting.

**16.3 Effect size, estimated means, + confidence intervals**

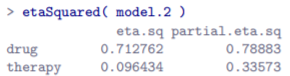
* The main thing you will probably want to calculate for a factorial ANOVA = **effect size** for each term in a model
* May also want to R to give you some estimates for the group means + associated CI’s.
* **Effect size** calculations for a factorial ANOVA = similar to those used in one-way ANOVA (Ch. 14.4)
* Specifically, we can use **η2 (eta-squared)** as simple way to measure how big the overall effect is for any particular term.
* η2 = dividing the sum of squares associated w/ a term by the total sum of squares.
* To determine size of main effect of Factor A:



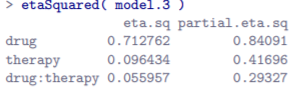
* This can be interpreted in much the same way as R2 in regression 🡪 tells you the proportion of variance in the outcome variable that can be accounted for by the main effect of Factor A.
* It therefore ranges from 0 (no effect at all) to 1 (accounts for all variability in the outcome)
* Moreover, the sum of all η2 values, taken across all terms in a model, will sum to the total R2 for the ANOVA model.
* If the ANOVA model fits *perfectly* (no w/in-groups variability at all), the η2 values sum to 1.
* Rarely, if ever, happens in real life
* A 2nd measure of effect = **partial** **η2** = **p.η2** 🡪 when measuring effect size for a particular term, you want to *deliberately ignore the other factor effects in the model*
* i.e. pretend the effect of all these other terms is zero
* Then we calculate what the η2 value *would have been*.
* To do so, remove the sum of squares associated w/ the other terms from the denominator.
* Ex: p.η2 for main effect of Factor A 🡪 denominator = sum of SS values for Factor A + the residuals



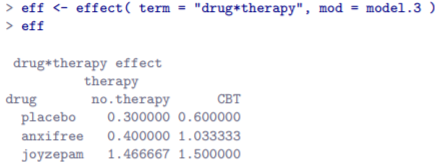
* This always gives a larger number than η2
* Once again, you get a number between 0 + 1, where 0 = no effect.
* However, it’s slightly trickier to interpret what a large partial η2 value means.
* Can’t actually compare the partial η2 values across terms
* Suppose there is no w/in-groups variability at all 🡪 If so, SSR = 0.
* This means every term has a partial η2 = 1
* **But that doesn’t mean all terms in the model are equally important, or that they are equally large**.
* All it means = all terms in the model have effect sizes large *relative to the residual variation*.
* It is NOT comparable across terms.
* Ex: 1st look at the effect sizes for the original ANOVA *w/out the interaction term*



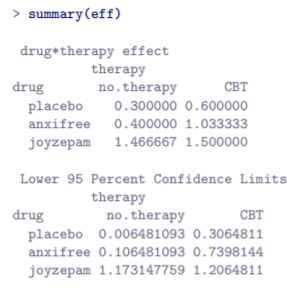
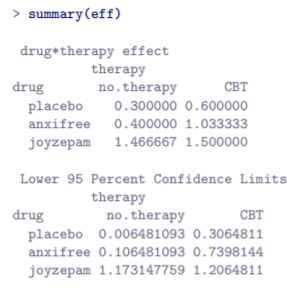
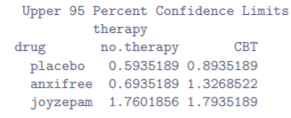
* Looking at η2 first 🡪 see that drug accounts for 71.3% of the variance in mood.gain, whereas therapy only accounts for 9.6%, leaving a total of 19.1% of the variation unaccounted for (residuals)
* Overall, this implies we have a very large effect of drug (too large for real-life, actually) + a modest effect of therapy.
* Looking at partial η2, b/c the effect of therapy isn’t all that large, controlling for it doesn’t make much of a difference = partial η2 for drug doesn’t increase very much
* In contrast, b/c the effect of drug was very large, controlling for it DOES make a big difference, + so when we calculate partial η2 for therapy, it rises up a good amount
* But what does these partial η2 values *actually mean*?
* General interpretation of partial η2 for the main effect of Factor A = as a statement about a hypothetical experiment in which *only* Factor A was being varied.
* partial η2 tells you how much *variance in the outcome variable you’d expect to see accounted for in that experiment w/ only Factor A being varied.*
* However, this interpretation (like many things associated w/ main effects) doesn’t make sense w/ a large + significant interaction effect.
* Speaking of interaction effects, calculate effect sizes for a model that includes the interaction term.



* η2 values for the main effects don’t change, but the *partial* η2 values do
* In many situations you will find yourself wanting to report **estimates of all group means** based on the results of an ANOVA, as well as CI’s associated w/ them.
* You can use **effect**() in the **effects** package to do this
* If the ANOVA you have is a **saturated model** (contains all possible main effects + all possible interaction effects), the estimates of the group means are identical to the sample means, though the CI’s will use a **pooled estimate of the standard errors**, rather than a separate one for each group

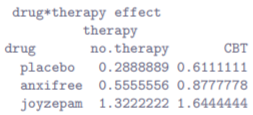


* Notice these are the same numbers we got when computing the sample means earlier (group.means computed using aggregate())
* 1 useful thing we can do using the effect variable **eff** is extract the CI’s using summary()

* The estimated mean mood gain for the placebo group w/ no therapy = 0.300, w/ a 95% CI from 0.006 to 0.594.
* \*\*\*Note: These are NOT the same CIs you’d get if you calculated them separately for each group, b/c of the fact that the ANOVA model assumes homoscedasticity + therefore uses a **pooled estimate of the standard deviation**.
* *When the model doesn’t contain the interaction term, the estimated group means will be different from the sample means.*
* Instead of reporting the sample mean, effect() will calculate the group means that would be expected on the basis of the marginal means (i.e., assuming no interaction).
* The estimate reported for the mean for level r on the (row) Factor A + level c on the (column) Factor B, **µ(r, c),** would be = **µ.. + αr + βc**
* If there are genuinely no interactions between the 2 factors, this is actually a better estimate of the population mean than the raw sample mean would be.
* Command to obtain these estimates = identical to the last, except w/ model.2 (no interaction term)





* \*\*\*R gives an error message letting you know the resulting estimates are based on the assumption that no interactions exist.
* Makes sense it would do this 🡪 W/ drug\*therapy as our input, we’re telling R we want it to output the **estimated group means** (rather than marginal means), but the actual input drug\*therapy might mean you want interactions included, or you might not.
* No *actual* ambiguity here, b/c the model itself either does or doesn’t have interactions, a warning just makes sure you’ve specified the actual model you care about.
* Can use **Effect**() to ignore this error message
* As before, we can obtain CIs w/ **summary( eff )** but the output looks the same as last time

**16.4 Assumption checking**

* As w/ one-way ANOVA, the key assumptions of factorial ANOVA are **homoscedasticity** (all groups have the same SD), **normality of the residuals**, + **independence of the observations**.
* The 1st 2 can be tested for, the 3rd is something you need to assess by asking if there are any special relationships between different observations.
* Additionally, if NOT using a **saturated model** (omitted the interaction terms), you’re also assuming omitted terms aren’t important.
* Can check this new 4th one by running an ANOVA w/ omitted terms included + see if significant
* Homoscedasticity + normality of the residuals checks are no to one-way ANOVA checks
* **Levene test for homoscedasticity** (14.7) 🡪 leveneTest() assumes a saturated model (included all of relevant terms), b/c the test is *primarily concerned w/ the w/in-group variance*
* Doesn’t make a lot of sense to calculate this any way other than w/ respect to the *full* model.
* 

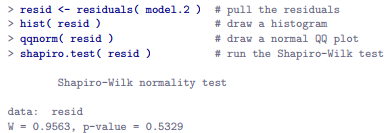


* Run Levene test w/ a saturated model





* The Levene test is non-significant = can safely assume homoscedasticity is not violated.
* Due to the way leveneTest() is implemented, however, if you use a formula like **mood.gain ~ drug + therapy + drug:therapy**, or input an ANOVA object based on a formula like this, you actually get the error message.
* Shouldn’t happen b/c this is a fully crossed model, however, there’s a quirky shortcut in the way leveneTest() checks whether a model is fully crossed that means it doesn’t recognize this as a fully crossed model.
* Essentially its checking that you used **\*** (ensures a model is fully crossed) + not “+” or “:” in the model formula.
* So if you’ve manually typed out all relevant terms for a fully crossed model, leveneTest() doesn’t detect it.
* As w/ one-way ANOVA, test for normality of residuals in a straightforward fashion (14.9)
* 1st use **residuals**() to extract the residuals from a model + examine them in a few different ways
* Generally a good idea to examine them *graphically* w/ + QQ plots (**qqnorm**())
* Formal test for normality of the residuals= **Shapiro-Wilk test** (**shapiro.test**())



* Non-significance of Shapiro-Wilk test = normality isn’t violated

**16.5 The F test as a model comparison**

* In the context of ANOVA, we’ve been referring to the **F-test** as a way of testing whether a particular term in the model (e.g., main effect of Factor A) is significant.
* This interpretation is perfectly valid, but not necessarily the most useful way to think about the test.
* In fact, it’s actually fairly limiting about what the F-test does.
* Suppose I want to see if there are any effects of *any kind* that involve **therapy** (don’t care if it’s a main or an interaction effect)
* 1 thing I could do is look at the output for **model.3** (DID see a main effect of therapy (p = .013), but did NOT see an interaction effect (p = .125))
* That’s *kind of* telling us what we want to know, but it’s not quite the same thing.
* What we *really* want = a single test that jointly checks main effect of therapy + the interaction effect
* Given the way I’ve been describing the ANOVA F-test up to this point, you’d be tempted to think that this isn’t possible.
* On the other hand, recall in regression (15.10), we were able to use F-tests to make comparisons between a wide variety of regression models.
* Something of that sort is possible w/ ANOVA
* \*\*\*Understand that the **F-test**, as used in both ANOVA + regression, is really a comparison of 2 statistical models.
* 1 of these models = the full model (alternative hypothesis), + the other model is a simpler model missing 1+ terms that the full model includes (null hypothesis).
* *The null model cannot contain any terms that are NOT in the full model*.
* Full model = model.3 🡪 contains main effect for therapy, main effect for drug, + the drug by therapy interaction term, drug:therapy
* The null model = model.1 🡪 contains only the main effect of drug.
* Say our full model can be written as a formula w/ several different terms, say **Y ~ A + B + C + D**.
* Our null model only contains *some subset* of these terms, say **Y ~ A + B**.
* Some of these terms might be main effect terms, others = interaction terms 🡪 really doesn’t matter
* Only thing that matters = want to treat some of these terms as the **starting point** (i.e. terms in the null model, A + B) + see if including the *other* terms (i.e., C + D) leads to significant improvement in model performance, over + above what could be achieved by a model that includes *only* A + B.
* In essence, we have null + alternatives that look like this:



* F-test is constructed from 2 kinds of quantity: **sums of squares (SS) + dF (df**), whichdefine a **mean square value (MS = SS/df),** + we obtain our F statistic by contrasting the MS associated w/ the thing we’re interested in (the model) w/ the MS value associated w/ everything else (the residuals).
* Want to figure out how to talk about the SS value associated w/ the difference between 2 models.
* Start w/ the fundamental rule used throughout regression 🡺 **SS.T = SS.M + SS.R** 🡺 total sums of squares (overall variability of the outcome variable) can be decomposed into 2 parts: variability associated w/ the model SS.M + the residual/leftover variability, SS.R.
* Useful to rearrange this equation 🡪say the SS value associated w/ a model is **SSM = SST - SSR**
* In our scenario, we have 2 models: the null (M0) + the full (M1):



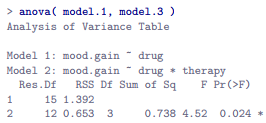
* Think about what we actually care about 🡪 *difference between the full model + the null model*
* So, to preserve the idea that we’re doing an analysis of the variance (ANOVA) in the outcome variable, define the SS associated w/ the difference to be equal to the difference in the SS values



* Now that we have our dF, we can calculate mean squares + F values in the usual way.
* Specifically 🡪 interested in the mean square *for the difference between models*, + the mean square *for the residuals associated w/ the full model* (M1), which are given by

* Finally, taking the ratio of these two gives us our F statistic: **F = MS∆ / MSR1**
* Null model here = model.1 🡪 stipulates there is a main effect of drug, but no other effects exist, expressed via the formula **mood.gain ~ drug**.
* The alternative model = model.3 🡪 stipulates there is a main effect of drug, a main effect of therapy, + an interaction 🡪 **mood.gain ~ drug + therapy + drug:therapy**
* Key = if we compare model.1 to model.3, we’re lumping the main effect of therapy + the interaction term together
* Input both models to anova() + it will run the exact F-test outlined above



* 1st , go back + look at the ANOVA tables for model.1 + model.3 to reassure that the RSS values printed in this table really do correspond to the residual sum of squares associated w/ these 2 models.



* Now, following the procedure above, the between model sum of squares = the difference between these 2 residual sum of squares values.



* Next, convert these SS values into MS values by dividing by the dF
* dF associated w/ the full-model residuals hasn’t changed from our original ANOVA for model.3 🡪 total sample size N, minus the total # of groups G relevant to the model.
* 18 people in the trial + 6 possible groups (2 therapies ˆ 3 drugs) 🡪 dF.full = 12.
* dF for the null model are calculated similarly + the only difference = there are only 3 relevant groups (3 drugs), so dF.null = 15
* B/c the dF associated w/ the difference = the difference in the 2 dF, we have 15 - 12 = 3 dF.



* W/ our 2 MS values, we can divide one by the other + obtain an F-statistic



* Just as we had hoped, this turns = identical to the F-statistic anova() produced earlier

**16.6 ANOVA as a linear model**

* One of the most important things to understand about ANOVA + 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 w/ testing for group differences, + regression is primarily concerned w/ understanding the correlations between variables. + as far as it goes, that’s perfectly true. But when you look under the hood, so to speak, the underlying mechanics of ANOVA + regression are awfully similar. In fact, if you think about it, you’ve already seen evidence of this. ANOVA + regression both rely heavily on sums of squares (SS), both make use of F tests, + so on. Looking back, it’s hard to escape the feeling that Ch.s 14 + 15 were a bit repetitive. The reason for this is that ANOVA + 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 + outcomes is the equation for a straight line, so it’s quite obviously a linear model. + 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 w/ is the somewhat uglier linear model: Yp b1X1p ` b2X2p ` b0 ` p where Yp is the outcome value for the p-th observation (e.g., p-th person), X1p is the value of the first predictor for the p-th observation, X2p is the value of the second predictor for the p-th observation, the b1, b2 + b0 terms are our regression coefficients, + p is the p-th residual. If we ignore the residuals p + just focus on the regression line itself, we get the following formula: Yˆ p b1X1p ` b2X2p ` b0 where Yˆ p is the value of Y that the regression line predicts for person p, as opposed to the actuallyobserved value Yp. 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 w/ a really simple example: rewriting a 2 ˆ 2 factorial ANOVA as a linear model. - 521 - 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), + whether or not the student actually read the textbook (the reading variable). We’ll say that attend = 1 if the student attended class, + attend = 0 if they did not. Similarly, we’ll say that reading = 1 if the student read the textbook, + 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 Yp denote the grade of the p-th student in the class. This is not quite the same notation that we used earlier in this Ch.: previously, we’ve used the notation Y(r, c)i to refer to the i-th person in the r-th group for predictor 1 (the row factor) + 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(r, c)i, but it has the shortcoming that it doesn’t actually keep track of the group memberships! That is, if I told you that Y0,0,3 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) + didn’t read the textbook (i.e. reading = 0), + who ended up failing the class (grade = 35). But if I tell you that Yp 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 X1p + X2p that keep track of this information. In the case of our hypothetical student, we know that X1p 0 (i.e., attend = 0) + X2p 0 (i.e., reading = 0). So the data might look like this: person, p grade, Yp attendance, X1p reading, X2p 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 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: > 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 + 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, + so you might instead be expecting to see data like this: - 522 - > 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 + 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, + 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 + 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 + attending the class both matter a lot. 16.6.2 ANOVA w/ 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 , + the two binary variables X1 + X2. What I want to you to recognise is that our 2ˆ2 factorial ANOVA is exactly equivalent to the regression model Yp b1X1p ` b2X2p ` b0 ` 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 X1 + X2 are now binary variables (i.e., values can only be 0 or 1), whereas in a regression analysis we expect that X1 + X2 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 exe(r, c)ise, proving that the two are identical. However, I’m going to go out on a limb + 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, + then rely on R to show that that ANOVA analyses + regression analyses aren’t just similar, they’re identical - 523 - for all intents + purposes.9 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 + reading as factors, + 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 So, by reading the key numbers off the ANOVA table + the table of means that we presented earlier, we can see that the students obtained a higher grade if they attended class pF1,5 26.1, p .0056q + if they read the textbook pF1,5 52.3, p .0008q. Let’s make a note of those p-values + 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 + 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, b/c the predictor only takes on two possible values, but it doesn’t violate any of the assumptions of linear regression. + 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 + is comfortable w/ 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 + 7th students in our data set (i.e. p 6 + 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 X2,6 0 + X2,7 0. However, student number 7 did turn up to lectures (i.e., attend = 1, X1,7 1) whereas student number 6 did not (i.e., attend = 0, X1,6 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 Yˆ 6 b1X1,6 ` b2X2,6 ` b0 pb1 ˆ 0q ` pb2 ˆ 0q ` b0 b0 So we’re expecting that this student will obtain a grade corresponding to the value of the inte(r, c)ept term b0. What about student 7? This time, when we insert the numbers into the formula for the regression 9This is cheating in some respects: b/c ANOVA + regression are provably the same thing, R is lazy: if you read the help documentation closely, you’ll notice that the aov() function is actually just the lm() function in disguise! But we shan’t let such things get in the way of our story, shall we? - 524 - line, we obtain the following: Yˆ 7 b1X1,7 ` b2X2,7 ` b0 pb1 ˆ 1q ` pb2 ˆ 0q ` b0 b1 ` b0 B/c this student attended class, the predicted grade is equal to the inte(r, c)ept term b0 plus the coefficient associated w/ the attend variable, b1. So, if b1 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 (X1,1 1) + read the textbook (X2,1 1)? If we plug these numbers into the regression, we get Yˆ 1 b1X1,1 ` b2X2,1 ` b0 pb1 ˆ 1q ` pb2 ˆ 1q ` b0 b1 ` b2 ` b0 So if we assume that attending class helps you get a good grade (i.e., b1 ą 0) + if we assume that reading the textbook also helps you get a good grade (i.e., b2 ą 0), then our expectation is that student 1 will get a grade that that is higher than student 6 + student 7. + 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 b2 ` b0. I won’t bore you w/ yet another regression formula. Instead, what I’ll do is show you the following table of expected grades: read textbook? no yes attended? no b0 b0 ` b2 yes b0 ` b1 b0 ` b1 ` b2 As you can see, the inte(r, c)ept term b0 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, b1 represents the boost that you’re expected to get if you come to class, + b2 represents the boost that comes from reading the textbook. In fact, if this were an ANOVA you might very well want to characterise b1 as the main effect of attendance, + b2 as the main effect of reading! In fact, for a simple 2 ˆ 2 ANOVA that’s exactly how it plays out. Okay, now that we’re really starting to see why ANOVA + regression are basically the same thing, let’s actually run our regression using the rtfm.1 data + the lm() function to convince ourselves that this is really true. Running the regression in the usual way gives us the following output:10 > regression.model <- lm( grade ~ attend + reading, data = rtfm.1 ) > summary( regression.model ) BLAH BLAH BLAH Coefficients: Estimate Std. Error t value Pr(>|t|) (Inte(r, c)ept) 43.500 3.354 12.969 4.86e-05 \*\*\* attend 18.000 3.873 4.648 0.00559 \*\* 10In the example given above, I’ve typed summary( regression.model ) to get the hypothesis tests. However, the summary() function does produce a lot of output, which is why I’ve used the BLAH BLAH BLAH text to hide the unnecessary parts of the output. But in fact, you can use the coef() function to do the same job. If you the command coef( summary( regression.model )) you’ll get exactly the same output that I’ve shown above (minus the BLAH BLAH BLAH). Compare + contrast this to the output of coef( regression.model ). - 525 - reading 28.000 3.873 7.230 0.00079 \*\*\* BLAH BLAH BLAH There’s a few interesting things to note here. Firstly, notice that the inte(r, c)ept 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 b1 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 b0 ` b1, which is equal to 43.5 ` 18.0 61.5. Again, this is similar to the observed group mean of 62.5, + 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 + our regression. Look at the p-values associated w/ the attend variable + 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 + the ANOVA calculates an F-statistic. However, if you can remember all the way back to Ch. 9, I mentioned that there’s a relationship between the t-distribution + the F-distribution: if you have some quantity that is distributed according to a t-distribution w/ k dF + you square it, then this new squared quantity follows an F-distribution whose dF are 1 + k. We can check this w/ 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 w/ 21.604, which is identical to the corresponding F statistic in our ANOVA. Finally, one last thing you should know. B/c R understands the fact that ANOVA + 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 + 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 + 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 ) BLAH BLAH BLAH Coefficients: - 526 - Estimate Std. Error t value Pr(>|t|) (Inte(r, c)ept) 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 \*\*\* BLAH BLAH BLAH The only thing that is different is that R labels the two variables differently: the output now refers to attendyes + 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 + no = 0. This is important. Suppose we wanted to say that yes = 0 + no = 1. We could still run this as a regression model, but now all of our coefficients will go in the opposite direction, b/c 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$reading [1] yes yes yes no no no no yes Levels: no yes Notice that order in which R prints out the levels is no + then yes. Now let’s apply the relevel() function: > relevel( x = rtfm.2$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 w/ 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 factor > 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 ) BLAH BLAH BLAH Coefficients: Estimate Std. Error t value Pr(>|t|) (Inte(r, c)ept) 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 \*\*\* BLAH BLAH BLAH As you can see, there are now a few changes. Most obviously, the attendno + 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 inte(r, c)ept has changed too. In our original regression, the baseline corresponded to a student who didn’t attend class + didn’t read the textbook, so we got a - 527 - 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 + did attend class, + for that student we would expect a grade of 89.5. 16.6.4 How to encode non binary factors as contrasts At this point, I’ve shown you how we can view a 2 ˆ 2 ANOVA into a linear model. + it’s pretty easy to see how this generalises to a 2 ˆ 2 ˆ 2 ANOVA or a 2 ˆ 2 ˆ 2 ˆ 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 ˆ 2 ANOVA that we ran earlier in this Ch. 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 + 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 + the placebo. To do this, we can create a second binary contrast, called drugjoyzepam, which equals 1 if the drug is joyzepam, + 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: 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, + drugjoyzepam will be 0. The reverse is true for Joyzepam: drugjoyzepam is 1, + 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, + a value of FALSE if the drug is Joyzepam or the placebo. The as.numeric() function just converts TRUE to 1 + 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 + 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.11 We can use it to create a new data frame, clin.trial.2 11Advanced users may want to look into the model.matrix() function, which produces similar output. Alternatively, you can use a command like contr.treatment(3)[clin.trial$drug,]. I’ll talk about the contr.treatment() function later. - 528 - that contains the same data as clin.trial, but w/ the two factors represented in terms of the contrast variables: > clin.trial.2 <- expandFactors( clin.trial ) > clin.trial.2 druganxifree drugjoyzepam therapyCBT mood.gain 1 0 0 0 0.5 2 0 0 0 0.3 3 0 0 0 0.1 4 1 0 0 0.6 5 1 0 0 0.4 6 1 0 0 0.2 7 0 1 0 1.4 8 0 1 0 1.7 BLAH BLAH BLAH 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, + we’ve already seen that ANOVA + 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 The equivalence between ANOVA + 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, + 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 ˆ 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 3.45 1.727 26.15 1.9e-05 \*\*\* therapy 1 0.47 0.467 7.08 0.019 \* Residuals 14 0.92 0.066 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.12 Next, let’s run a regression, using druganxifree, drugjoyzepam + therapyCBT as the predictors. Here’s what we get: > drug.regression <- lm( mood.gain ~ druganxifree + drugjoyzepam + therapyCBT, clin.trial.2 ) > summary( drug.regression ) BLAH BLAH BLAH Coefficients: Estimate Std. Error t value Pr(>|t|) (Inte(r, c)ept) 0.289 0.121 2.38 0.032 \* 12Future versions of this book will try to be a bit more consistent w/ the naming scheme for variables. One of the many problems w/ having to write a lengthy text very quickly to meet a teaching deadline is that you lose some internal consistency. - 529 - druganxifree 0.267 0.148 1.80 0.094 . drugjoyzepam 1.033 0.148 6.97 6.6e-06 \*\*\* therapyCBT 0.322 0.121 2.66 0.019 \* BLAH BLAH BLAH 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 + the drugjoyzepam contrast separately, as if they were two completely unrelated variables. It’s not surprising of course, b/c the poor lm() function has no way of knowing that drugjoyzepam + druganxifree are actually the two different contrasts that we used to encode our three-level drug factor. As far as it knows, drugjoyzepam + druganxifree are no more related to one another than drugjoyzepam + therapyCBT. However, you + 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, + 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, + omit both of the drug-related variables, + 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 4.38 2 14 0.92 2 3.45 26.1 1.9e-05 \*\*\* Ah, that’s better. Our F-statistic is 26.1, the dF are 2 + 14, + 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 + regression are essentially the same: they are both linear models, + 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 Ch. we’re going to rely heavily on this idea. 16.6.6 DF as parameter counting! At long last, I can finally give a definition of dF that I am happy w/. DF 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 inte(r, c)ept. 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, + a second one for the inte(r, c)ept. The alternative model (mood.gain ~ druganxifree + drugjoyzepam + therapyCBT) has four parameters: one regression coefficient for each of the three contrasts, + one more for the inte(r, c)ept. So the dF associated w/ the difference between these two models is df1 4 ´ 2 2. - 530 - 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 inte(r, c)ept, which is written as outcome ~ 1 in R, + the alternative model is the full regression model. The null model in this case contains 1 regression coefficient, for the inte(r, c)ept term. The alternative model contains K ` 1 regression coefficients, one for each of the K predictor variables + one more for the inte(r, c)ept. So the df value that you see in this F test is equal to df1 K ` 1 ´ 1 K. What about the second df value that appears in the F-test? This always refers to the dF associated w/ 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 + an inte(r, c)ept, then you’ve specified K ` 1 numbers. So, w/out 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 pK ` 1q ` pN ´ K ´ 1q N, + 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, + 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) + the number of parameters used by the simpler regression model that you’re actually interested in (i.e., K ` 1), + so the second dF 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, + K ` 1 4 regression coefficients associated w/ the ANOVA model, so the dF for the residuals is df2 18 ´ 4 14. 16.6.7 A 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, + will perform the appropriate analysis. After all, as I’ve been saying throughout this section, ANOVA + regression are both linear models, + 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 + 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 w/ this sort of thing. Okay, warnings aside, it’s actually kind of neat to run your ANOVA using the lm() function in the - 531 - way I did above. B/c 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 ) However, b/c the drug + 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 ) 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 + drugjoyzepam as if they were two distinct binary factors. This is b/c 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. 16.7 Different ways to specify contrasts In the previous section, I showed you a method for converting a factor into a collection of contrasts. In the method I showed you, we specify a set of binary variables, in which define a table like this one: drug druganxifree drugjoyzepam placebo 0 0 anxifree 1 0 joyzepam 0 1 Each row in the table corresponds to one of the factor levels, + each column corresponds to one of the contrasts. This table, which always has one more row than columns, has a special name: it is called a contrast matrix. However, there are lots of different ways to specify a contrast matrix. In this section I discuss a few of the standard contrast matrices that statisticians use, + how you can use them in R. If you’re planning to read the section on unbalanced ANOVA later on (Section 16.10) it’s worth reading this section carefully. If not, you can get away w/ skimming it, b/c the choice of contrasts doesn’t matter much for balanced designs. 16.7.1 Treatment contrasts In the particular kind of contrasts that I’ve described above, one level of the factor is special, + acts as a kind of baseline category (i.e., placebo in our example), against which the other two are defined. The name for these kinds of contrast is treatment contrasts. The name reflects the fact that these contrasts are quite natural + sensible when one of the categories in your factor really is special b/c it actually does represent a baseline. That makes sense in our clinical trial example: the placebo condition corresponds to the situation where you don’t give people any real drugs, + so it’s special. The other two conditions are defined in relation to the placebo: in one case you replace the placebo w/ Anxifree, + in the other case your replace it w/ Joyzepam. - 532 - R comes w/ a variety of functions that can generate different kinds of contrast matrices. For example, the table shown above is a matrix of treatment contrasts for a factor that has 3 levels. But suppose I want a matrix of treatment contrasts for a factor w/ 5 levels? The contr.treatment() function will do this: > contr.treatment( n=5 ) 2 3 4 5 1 0 0 0 0 2 1 0 0 0 3 0 1 0 0 4 0 0 1 0 5 0 0 0 1 Notice that, by default, the first level of the factor is always treated as the baseline category (i.e., it’s the one that has all zeros, + doesn’t have an explicit contrast associated w/ it). In Section 16.6.3 I mentioned that you can use the relevel() function to change which category is the first level of the factor.13 There’s also a special function in R called contr.SAS() that generates a treatment contrast matrix in which the last category is treated as the baseline: > contr.SAS( n=5 ) 1 2 3 4 1 1 0 0 0 2 0 1 0 0 3 0 0 1 0 4 0 0 0 1 5 0 0 0 0 However, you can actually select any category you like as the baseline w/in the contr.treatment() function, by specifying the base argument in that function. See the help documentation for more details. 16.7.2 Helmert contrasts Treatment contrasts are useful for a lot of situations, + they’re the default in R. However, they make most sense in the situation when there really is a baseline category, + you want to assess all the other groups in relation to that one. In other situations, however, no such baseline category exists, + it may make more sense to compare each group to the mean of the other groups. This is where Helmert contrasts, generated by the contr.helmert() function, can be useful. The idea behind Helmert contrasts is to compare each group to the mean of the previous ones. That is, the first contrast represents the difference between group 2 + group 1, the second contrast represents the difference between group 3 + the mean of groups 1 + 2, + so on. This translates to a contrast matrix that looks like this: > contr.helmert( n=5 ) [,1] [,2] [,3] [,4] 1 -1 -1 -1 -1 2 1 -1 -1 -1 3 0 2 -1 -1 4 0 0 3 -1 5 0 0 0 4 One useful thing about Helmert contrasts is that every contrast sums to zero (i.e., all the columns sum to zero). This has the consequence that, when we interpret the ANOVA as a regression, the inte(r, c)ept term corresponds to the grand mean µ..q if we are using Helmert contrasts. Compare this to treatment 13The lsr package contains a more general function called permuteLevels() that can shuffle them in any way you like. - 533 - contrasts, in which the inte(r, c)ept term corresponds to the group mean for the baseline category. This property can be very useful in some situations. It doesn’t matter very much if you have a balanced design, which we’ve been assuming so far, but it will turn out to be important later when we consider unbalanced designs in Section 16.10. In fact, the main reason why I’ve even bothered to include this section on specifying is that contrasts become important if you want to understand unbalanced ANOVA. 16.7.3 Sum to zero contrasts The third option that I should briefly mention are sum to zero contrasts, which are used to construct pairwise comparisons between groups. Specifically, each contrast encodes the difference between one of the groups + a baseline category, which in this case corresponds to the last group: > contr.sum( n=5 ) [,1] [,2] [,3] [,4] 1 1 0 0 0 2 0 1 0 0 3 0 0 1 0 4 0 0 0 1 5 -1 -1 -1 -1 Much like Helmert contrasts, we see that each column sums to zero, which means that the inte(r, c)ept term corresponds to the grand mean when ANOVA is treated as a regression model. When interpreting these contrasts, the thing to recognise is that each of these contrasts is a pairwise comparison between group 5 + one of the other four groups. Specifically, contrast 1 corresponds to a group 1 minus group 5 comparison, contrast 2 corresponds to a group 2 minus group 5 comparison, + so on. 16.7.4 Viewing + setting the default contrasts in R Every factor variable in R is associated w/ a contrast matrix. It has to be, otherwise R wouldn’t be able to run ANOVAs properly! If you don’t specify one explictly, or R will implicitly specify one for you. Here’s what I mean. When I created the clin.trial data, I didn’t specify any contrast matrix for either of the factors. You can see this by using the attr() function to print out the contrasts attribute of the factors. For example: > attr( clin.trial$drug, contrasts ) NULL The NULL output here means that R is telling you that the drug factor doesn’t have any attribute called contrasts for which it has any data. There is no contrast matrix stored anywhere explicitly for this factor. However, if we now ask R to tell us what contrasts are set up for this factor, it give us this: > contrasts( clin.trial$drug ) anxifree joyzepam placebo 0 0 anxifree 1 0 joyzepam 0 1 These are the same treatment contrast that we set up manually in Section 16.6. How did R know to set up treatment contrasts, even though I never actually told it anything about what contrasts I wanted? The answer is that R has a hidden list of default options that it looks up to resolve situations like this. You can print out all of the options by typing options() at the command prompt, but it’s not a very - 534 - enlightening read. There are a lot of options, + we’re only interested in contrasts right now. Instead of printing out all of the options, we can ask for just one, like this: > options( contrasts ) $contrasts unordered ordered contr.treatment contr.poly What this is telling us is that the default contrasts for unordered factors (i.e., nominal scale variables) are treatment contrasts, + the default for ordered factors (i.e., interval scale variables) are polynomial contrasts. I don’t discuss ordered factors much in this book, + so I won’t go into what polynomial contrasts are all about. The key thing is that the options() function also allows you to reset these defaults (though only for the current session: they’ll revert to the original settings once you close R). Here’s the command: > options(contrasts = c(contr.helmert, contr.poly)) Once we’ve done this, we can inspect the contrast settings again: > options(contrasts) $contrasts [1] contr.helmert contr.poly Now we see that the default contrasts for unordered factors have changed. So if I now ask R to tell me what contrasts are associated w/ the drug factor, it gives a different answer b/c I changed the default: > contrasts( clin.trial$drug ) [,1] [,2] placebo -1 -1 anxifree 1 -1 joyzepam 0 2 Those are Helmert contrasts. In general, if you’re changing the default settings for something in R, it’s a good idea to reset them to their original values once you’re done. So let’s do that: > options(contrasts = c(contr.treatment, contr.poly)) 16.7.5 Setting the contrasts for a single factor In the previous section, I showed you how to alter the default contrasts. However, suppose that all you really want to do is change the contrasts associated w/ a single factor, + leave the defaults as they are. To do this, what you need to do is specifically assign the contrast matrix as an attribute’ of the factor. This is easy to do via the contrasts() function. For instance, suppose I wanted to use sum to zero contrasts for the drug factor, but keep the default treatment contrasts for everything else. I could do that like so: > contrasts( clin.trial$drug ) <- contr.sum(3) + if I now inspect the contrasts, I get the following - 535 - > contrasts( clin.trial$drug) [,1] [,2] placebo 1 0 anxifree 0 1 joyzepam -1 -1 However, the contrasts for everything else will still be the defaults. You can check that we have actually made a specific change to the factor itself by checking to see if it now has an attribute, using the command attr( clin.trial$drug, contrasts ). This will print out the same output shown above, b/c the contrast has in fact been attached to the drug factor, + does not rely on the defaults. If you want to wipe the attribute + revert the defaults, use a command like this: > contrasts( clin.trial$drug ) <- NULL 16.7.6 Setting the contrasts for a single analysis One last way of changing contrasts. You might find yourself wanting to change the contrasts only for one specific analysis. That’s allowed too, b/c the aov() + lm() functions have a contrasts argument that you can use. To change contrasts for one specific analysis, we first set up a list variable that names14 the contrast types that you want to use for each of the factors: > my.contrasts <- list( drug = contr.helmert, therapy = contr.helmert ) Next, fit the ANOVA model in the usual way, but this time we’ll specify the contrasts argument: > mod <- aov( mood.gain ~ drug\*therapy, clin.trial, contrasts = my.contrasts ) If you try a command like summary(aov) you won’t see any difference in the output b/c the choice of contrasts does not affect the outcome when you have a balanced design (this won’t always be true later on). However, if you want to check that it has actually worked, you can inspect the value of mod$contrasts: > mod$contrasts $drug [,1] [,2] placebo -1 -1 anxifree 1 -1 joyzepam 0 2 $therapy [,1] no.therapy -1 CBT 1 As you can see, for the purposes of this one particular ANOVA, R has used Helmert contrasts for both variables. If I had omitted the part of the command that specified the contrasts argument, you’d be looking at treatment contrasts here b/c it would have reverted to whatever values the contrasts() function prints out for each of the factors. 14Technically, this list actually stores the functions themselves. R allows lists to contain functions, which is really neat for advanced purposes, but not something that matters for this book. - 536 - 16.8 Post hoc tests Time to switch to a different topic. Let’s suppose you’ve done your ANOVA, + it turns out that you obtained some significant effects. B/c of the fact that the F-tests are omnibus tests that only really test the null hypothesis that there are no differences among groups, obtaining a significant effect doesn’t tell you which groups are different to which other ones. We discussed this issue back in Ch. 14, + in that Ch. our solution was to run t-tests for all possible pairs of groups, making corrections for multiple comparisons (e.g., Bonferroni, Holm) to control the Type I error rate across all comparisons. The methods that we used back in Ch. 14 have the advantage of being relatively simple, + being the kind of tools that you can use in a lot of different situations where you’re testing multiple hypotheses, but they’re not necessarily the best choices if you’re interested in doing efficient post hoc testing in an ANOVA context. There are actually quite a lot of different methods for performing multiple comparisons in the statistics literature (Hsu, 1996), + it would be beyond the scope of an introductory text like this one to discuss all of them in any detail. That being said, there’s one tool that I do want to draw your attention to, namely Tukey’s Honestly Significant Difference, or Tukey’s HSD for short. For once, I’ll spare you the formulas, + just stick to the qualitative ideas. The basic idea in Tukey’s HSD is to examine all relevant pairwise comparisons between groups, + it’s only really appropriate to use Tukey’s HSD if it is pairwise differences that you’re interested in.15 For instance, in model.2, where we specified a main effect for drug + a main effect of therapy, we would be interested in the following four comparisons: • The difference in mood gain for people given Anxifree versus people given the placebo. • The difference in mood gain for people given Joyzepam versus people given the placebo. • The difference in mood gain for people given Anxifree versus people given Joyzepam. • The difference in mood gain for people treated w/ CBT + people given no therapy. For any one of these comparisons, we’re interested in the true difference between (population) group means. Tukey’s HSD constructs simultaneous CIs for all four of these comparisons. What we mean by 95% simultaneous CI is that there is a 95% probability that all of these CIs contain the relevant true value. Moreover, we can use these CIs to calculate an adjusted p value for any specific comparison. The TukeyHSD() function in R is pretty easy to use: you simply input the model that you want to run the post hoc tests for. For example, if we were looking to run post hoc tests for model.2, here’s the command we would use: > TukeyHSD( model.2 ) Tukey multiple comparisons of means 95% family-wise confidence level Fit: aov(formula = mood.gain ~ drug + therapy, data = clin.trial) $drug 15If, for instance, you actually would find yourself interested to know if Group A is significantly different from the mean of Group B + Group C, then you need to use a different tool (e.g., Scheffe’s method, which is more conservative, + beyond the scope this book). However, in most cases you probably are interested in pairwise group differences, so Tukey’s HSD is a pretty useful thing to know about. - 537 - diff lwr upr p adj anxifree-placebo 0.2666667 -0.1216321 0.6549655 0.2062942 joyzepam-placebo 1.0333333 0.6450345 1.4216321 0.0000186 joyzepam-anxifree 0.7666667 0.3783679 1.1549655 0.0003934 $therapy diff lwr upr p adj CBT-no.therapy 0.3222222 0.0624132 0.5820312 0.0186602 The output here is (I hope) pretty straightforward. The first comparison, for example, is the Anxifree versus placebo difference, + the first part of the output indicates that the observed difference in group means is .27. The next two numbers indicate that the 95% (simultaneous) CI for this comparison runs from ´.12 to .65. B/c the CI for the difference includes 0, we cannot reject the null hypothesis that the two group means are identical, + so we’re not all that surprised to see that the adjusted p-value is .21. In contrast, if you look at the next line, we see that the observed difference between Joyzepam + the placebo is 1.03, + the 95% CI runs from .64 to 1.42. B/c the interval excludes 0, we see that the result is significant pp ă .001q. So far, so good. What about the situation where your model includes interaction terms? For instance, in model.3 we allowed for the possibility that there is an interaction between drug + therapy. If that’s the case, the number of pairwise comparisons that we need to consider starts to increase. As before, we need to consider the three comparisons that are relevant to the main effect of drug + the one comparison that is relevant to the main effect of therapy. But, if we want to consider the possibility of a significant interaction (+ try to find the group differences that underpin that significant interaction), we need to include comparisons such as the following: • The difference in mood gain for people given Anxifree + treated w/ CBT, versus people given the placebo + treated w/ CBT • The difference in mood gain for people given Anxifree + given no therapy, versus people given the placebo + given no therapy. • etc There are quite a lot of these comparisons that you need to consider. So, when we run the TukeyHSD() command for model.3 we see that it has made a lot of pairwise comparisons (19 in total). Here’s the output: > TukeyHSD( model.3 ) Tukey multiple comparisons of means 95% family-wise confidence level Fit: aov(formula = mood.gain ~ drug \* therapy, data = clin.trial) $drug diff lwr upr p adj anxifree-placebo 0.2666667 -0.09273475 0.6260681 0.1597148 joyzepam-placebo 1.0333333 0.67393191 1.3927348 0.0000160 joyzepam-anxifree 0.7666667 0.40726525 1.1260681 0.0002740 $therapy diff lwr upr p adj CBT-no.therapy 0.3222222 0.08256504 0.5618794 0.012617 $‘drug:therapy‘ - 538 - diff lwr upr p adj anxifree:no.therapy-placebo:no.therapy 0.10000000 -0.539927728 0.7399277 0.9940083 joyzepam:no.therapy-placebo:no.therapy 1.16666667 0.526738939 1.8065944 0.0005667 placebo:CBT-placebo:no.therapy 0.30000000 -0.339927728 0.9399277 0.6280049 anxifree:CBT-placebo:no.therapy 0.73333333 0.093405606 1.3732611 0.0218746 joyzepam:CBT-placebo:no.therapy 1.20000000 0.560072272 1.8399277 0.0004380 joyzepam:no.therapy-anxifree:no.therapy 1.06666667 0.426738939 1.7065944 0.0012553 placebo:CBT-anxifree:no.therapy 0.20000000 -0.439927728 0.8399277 0.8917157 anxifree:CBT-anxifree:no.therapy 0.63333333 -0.006594394 1.2732611 0.0529812 joyzepam:CBT-anxifree:no.therapy 1.10000000 0.460072272 1.7399277 0.0009595 placebo:CBT-joyzepam:no.therapy -0.86666667 -1.506594394 -0.2267389 0.0067639 anxifree:CBT-joyzepam:no.therapy -0.43333333 -1.073261061 0.2065944 0.2750590 joyzepam:CBT-joyzepam:no.therapy 0.03333333 -0.606594394 0.6732611 0.9999703 anxifree:CBT-placebo:CBT 0.43333333 -0.206594394 1.0732611 0.2750590 joyzepam:CBT-placebo:CBT 0.90000000 0.260072272 1.5399277 0.0050693 joyzepam:CBT-anxifree:CBT 0.46666667 -0.173261061 1.1065944 0.2139229 It looks pretty similar to before, but w/ a lot more comparisons made. 16.9 The method of planned comparisons Okay, I have a confession to make. I haven’t had time to write this section, but I think the method of planned comparisons is important enough to deserve a quick discussion. In our discussions of multiple comparisons, in the previous section + back in Ch. 14, I’ve been assuming that the tests you want to run are genuinely post hoc. For instance, in our drugs example above, maybe you thought that the drugs would all have different effects on mood (i.e., you hypothesised a main effect of drug), but you didn’t have any specific hypothesis about how they would be different, nor did you have any real idea about which pairwise comparisons would be worth looking at. If that is the case, then you really have to resort to something like Tukey’s HSD to do your pairwise comparisons. The situation is rather different, however, if you genuinely did have real, specific hypotheses about which comparisons are of interest, + you never ever have any intention to look at any other comparisons besides the ones that you specified ahead of time. When this is true, + if you honestly + rigourously stick to your noble intentions to not run any other comparisons (even when the data look like they’re showing you deliciously significant effects for stuff you didn’t have a hypothesis test for), then it doesn’t really make a lot of sense to run something like Tukey’s HSD, b/c it makes corrections for a whole bunch of comparisons that you never cared about + never had any intention of looking at. Under those ci(r, c)umstances, you can safely run a (limited) number of hypothesis tests w/out making an adjustment for multiple testing. This situation is known as the method of planned comparisons, + it is sometimes used in clinical trials. In a later version of this book, I would like to talk a lot more about planned comparisons. 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, + you’ll find that you can’t read more than two or three - 539 - papers in psychology w/out 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 + 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 w/ 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 w/ a lot more care than balanced designs, + 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 resea(r, c)h 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 resea(r, c)hers in the field don’t actually know that there’s several different types of unbalanced ANOVAs, + 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 The coffee data As usual, it will help us to work w/ some data. The coffee.Rdata file contains a hypothetical data set (the coffee data frame) that produces an unbalanced 3 ˆ 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 + sugar that people add to the coffee. Suppose we took 18 people, + gave them some coffee to drink. The amount of coffee / caffeine was held constant, + we varied whether or not milk was added: so milk is a binary factor w/ two levels, yes + 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: > some( coffee ) milk sugar babble 4 yes real 5.6 5 yes real 4.9 7 yes none 3.8 8 yes none 3.7 12 yes fake 5.6 14 no real 6.1 15 no real 6.3 16 no none 5.5 17 no none 5.0 18 yes fake 5.0 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 - 540 - 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.16 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 ways17 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, w/ the sums of squares, F-values etc all conforming to the formulas that I gave at the start of the Ch.. 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 + 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, + 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 dF 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 16This discrepancy in standard deviations might (+ should) make you wonder if we have a violation of the homoscedasticity assumption. I’ll leave it as an exe(r, c)ise for the reader to check this using the leveneTest() function. 17Actually, this is a bit of a lie. ANOVAs can vary in other ways besides the ones I’ve discussed in this book. For instance, I’ve completely ignored the difference between fixed-effect models, in which the levels of a factor are fixed by the experimenter or the world, + random-effect models, in which the levels are random samples from a larger population of possible levels (this book only covers fixed-effect models). Don’t make the mistake of thinking that this book – or any other one – will tell you everything you need to know about statistics, any more than a single book could possibly tell you everything you need to know about psychology, physics or philosophy. Life is too complicated for that to ever be true. This isn’t a cause for despair, though. Most resea(r, c)hers get by w/ a basic working knowledge of ANOVA that doesn’t go any further than this book does. I just want you to keep in mind that this book is only the beginning of a very long story, not the whole story. - 541 - the strategy that is the important thing here, not the SS values themselves. Recall from Section 16.5 + 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 + 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, b/c 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 ˆ 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 + 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 inte(r, c)ept, + 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, b/c 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, + pick sugar. So the second model in our sequence of models is babble ~ sugar, + it forms the alternative hypothesis for our first test. We now have our first hypothesis test: 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 exe(r, c)ise 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: 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, b/c 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, + the corresponding hypothesis test is 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 + 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 ) - 542 - 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 + .0018 respectively. Next, let’s see if we can replicate the analysis using tools that we’re a little more familiar w/. 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) + anova(mod.3, mod.4) + to run the second + 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.18 The big problem w/ 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 resea(r, c)her has no reason to prefer one ordering over another. This is presumably the case for our milk + 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 w/ 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: 18The one thing that might seem a little opaque to some people is why the residual dF in this output look different from one another (i.e., ranging from 12 to 17) whereas in the original one the residual dF is fixed at 12. It’s actually the case that R uses a residual df of 12 in all cases (that’s why the p values are the same in the two outputs, + it’s enough to verify that pf(6.7495, 2,12, lower.tail=FALSE)) gives the correct answer of p .010863, for instance, whereas pf(6.7495, 2,15, lower.tail=FALSE)) would have given a p-value of about .00812. It’s the residual dF in the full model (i.e., the last one) that matters here. - 543 - > 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, + 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 + 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, + so we should move on to discuss Type II tests + 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, w/ 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 + B:C), then second order interactions (e.g., A:B:C) + so on. W/in 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 ) + 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 + then C, but then the interactions would be entered in the order B:C first, then A:B + then finally A:C. Reordering the terms w/in 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 + then using a command like anova(mod.1, mod.2, mod.3, mod.4) to fo(r, c)e 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 - 544 - 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, + 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: 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: 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: 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 w/ three factors, A, B + C, + we wanted to consider all possible main effects + 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: 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) + one second order interaction (i.e., A:B:C). The null model always contains 6 of thes 7 terms: + 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. + b/c 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 + sugar, but no effect of milk? That seems crazy. The right answer simply must be that it’s meaningless19 to talk about the main effect if the interaction is significant. In general, this seems to be what most statisticians advise us to do, + I think that’s the right advice. But if it really 19Or, at the very least, rarely of interest. - 545 - 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 + alternative hypotheses for each test, + 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 experience20 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) (Inte(r, c)ept) 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 + 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, + we can see that our Type III tests are significant for all terms, even the inte(r, c)ept. 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; + 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) (Inte(r, c)ept) 434.29 1 1647.8882 3.231e-14 \*\*\* 20Yes, I’m actually a big enough nerd that I’ve written my own functions implementing Type II tests + Type III tests. I only did it to convince myself that I knew how the different Types of test worked, but it did turn out to be a handy exe(r, c)ise: the etaSquared() function in the lsr package relies on it. There’s actually even an argument in the etaSquared() function called anova. By default, anova=FALSE + the function just prints out the effect sizes. However, if you set anova=TRUE it will spit out the full ANOVA table as well. This works for Types I, II + III. Just set the types argument to select which type of test you want. - 546 - 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, + 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 + not to be trusted? To some extent that’s true, + 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] ) # contrast 1 sums to 0 > random.contrasts[, 2] <- random.contrasts[, 2] - mean( random.contrasts[, 2] ) # contrast 2 sums to 0 > random.contrasts # print it to check that we really have an arbitrary contrast matrix... [,1] [,2] [1,] 0.38898807 -0.78454935 [2,] -0.04337123 0.70004953 [3,] -0.34561683 0.08449982 > contrasts( coffee$sugar ) <- random.contrasts # random contrasts for sugar > contrasts( coffee$milk ) <- contr.Helmert(2) # Helmert contrasts for the milk factor > mod.R <- lm( babble ~ sugar \* milk, coffee ) # R will use the contrasts that we assigned > Anova( mod.R, type = 3 ) Anova Table (Type III tests) Response: babble Sum Sq Df F value Pr(>F) (Inte(r, c)ept) 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 + III tests now, + 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 + looking to see what happens when you remove each term. However, both have some - 547 - serious flaws: Type I tests are dependent on the order in which you enter the terms, + Type III tests are dependent on how you code up your contrasts. B/c 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, + as a result they are a little easier to interpret. Type II tests are broadly similar to Type III tests: start w/ a full model, + 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 + 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 + C as well as the simpler interactions A:B, A:C + 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 + alternative models are: 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 + 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, + it yields the following as our Type II test of the main effect of A. 21 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: 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 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 21Note, of course, that this does depend on the model that the user specified. If original ANOVA model doesn’t contain an interaction term for B:C, then obviously it won’t appear in either the null or the alternative. But that’s true for Types I, II + III. They never include any terms that you didn’t include, but they make different choices about how to construct tests for the ones that you did include. - 548 - even simpler. The main effect of sugar corresponds to an F-test comparing these two models: Null model: babble ~ milk Alternative model: babble ~ sugar + milk The test for the main effect of milk is Null model: babble ~ sugar Alternative model: babble ~ sugar + milk Finally, the test for the interaction sugar:milk is: 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 + Type III tests. They don’t depend on the order in which you specify factors (unlike Type I), + they don’t depend on the contrasts that you use to specify your factors (unlike Type III). + although opinions may differ on this last point, + it will definitely depend on what you’re trying to do w/ 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 resea(r, c)h 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.22 16.10.6 Effect sizes (+ non-additive sums of squares) The etaSquared() function in the lsr package computes η 2 + partial η 2 values for unbalanced designs + for different Types of tests. It’s pretty straightforward. All you have to do is indicate which type of tests you’re doing, 22I find it amusing to note that the default in R is Type I + the default in SPSS is Type III (w/ Helmert contrasts). Neither of these appeals to me all that much. Relatedly, I find it depressing that almost nobody in the psychological literature ever bothers to report which Type of tests they ran, much less the order of variables (for Type I) or the contrasts used (for Type III). Often they don’t report what software they used either. The only way I can ever make any sense of what people typically report is to try to guess from auxiliary cues which software they were using, + to assume that they never changed the default settings. Please don’t do this... now that you know about these issues, make sure you indicate what software you used, + if you’re reporting ANOVA results for unbalanced data, then specify what Type of tests you ran, specify order information if you’ve done Type I tests + specify contrasts if you’ve done Type III tests. Or, even better, do hypotheses tests that correspond to things you really care about, + then report those! - 549 - > 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 + out pops the η 2 + 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 p sugar 0.22537682 0.4925493 3.0696323 2 1.5348161 5.823808 0.017075099 milk 0.07019886 0.2321436 0.9561085 1 0.9561085 3.627921 0.081060698 sugar:milk 0.43640732 0.6527155 5.9438677 2 2.9719339 11.276903 0.001754333 Residuals 0.23219530 NA 3.1625000 12 0.2635417 NA 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 w/ the effects in the model, + add that to the residual SS, they’re supposed to add up to the total sum of squares. +, on top of that, the whole idea behind η 2 is that – b/c 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. B/c I’ve included the η 2 value associated w/ 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. B/c, the whole idea here was that the variance in the outcome variable can be divided up into the variability attributable to the model, + the variability in the residuals. Right? Right? + 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 + 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, + 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 + III, the individual SS values are not obliged to the total sum of squares, + will only do so if you have balanced data. I’ll explain why this happens + 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, + 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 - 550 - 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 w/ one another, + it becomes difficult to tell the difference between the effect of Factor A + the effect of Factor B. In the extreme case, suppose that we’d run a 2 ˆ 2 design in which the number of participants in each group had been as follows: sugar no sugar milk 100 0 no milk 0 100 Here we have a spectacularly unbalanced design: 100 people have milk + sugar, 100 people have no milk + no sugar, + that’s all. There are 0 people w/ milk + no sugar, + 0 people w/ sugar but no milk. Now suppose that, when we collected the data, it turned out there is a large (+ statistically significant) difference between the milk + sugar group + the no-milk + no-sugar group. Is this a main effect of sugar? A main effect of milk? Or an interaction? It’s impossible to tell, b/c the presence of sugar has a perfect association w/ the presence of milk. Now suppose the design had been a little more balanced: sugar no sugar milk 100 5 no milk 5 100 This time around, it’s technically possible to distinguish between the effect of milk + the effect of sugar, b/c we have a few people that have one but not the other. However, it will still be pretty difficult to do so, b/c the association between sugar + milk is still extremely strong, + 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 + 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 + Type III tests are more conservative. Variance that cannot be clearly attributed to a specific effect doesn’t get attributed to any of them, + it goes missing