Contrasts and ANOVA

Neil J. Hatfield

Last Updated: 2022-11-02

In this tutorial, we are going to explore setting up and testing Contrasts in the context of ANOVA models **using parametric shortcuts *only***. The structure for this guide/tutorial will be:

- Setting Up `R` and Loading Data
- Key Decisions for Post Hoc Analysis
  - Study Design Decisions
  - Checking Appropriateness and Assumptions
- Setting Up Your Contrasts
- Checking Your Contrasts
- Testing Your Contrasts in `R`
  - Using Base Packages
  - Using the `DescTools` Package
- Reporting and Interpreting Results

Keep in mind that contrasts generally fit within the space of Post Hoc analysis after an initial omnibus test. We are going to restrict ourselves to just parametric shortcuts.

Setting Up `R` and Loading Data

For this particular guide/tutorial, we will load (most) of our usual packages: `tidyverse`, `knitr`, and `kableExtra`. We will also load `parameters` and `DescTools`.

```r
# Demo code for setting up R for contrasts
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "DescTools")
lapply(packages, library, character.only = TRUE)

options(knitr.kable.NA = "")
options(contrasts = c("contr.sum", "contr.poly"))

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
```

Don't forget that we also need to set options including the all important constraint that our factor effects must add to zero (i.e. `c("contr.sum", "contr.poly")`).

Loading Data

We will make use of the **Free Amino Acids in Cheese** (i.e., the Cheese Study; pg. 91 of Oehlert) for the examples in this guide. You will also want to load the Song Knowledge study data for the final example. For this guide, I'm not going to show the loading code here. However, if you get stuck in writing the code, check out the Code Appendix at the end of the guide/tutorial.

Key Decisions for Post Hoc Analysis

Just as with pairwise approaches to Post Hoc analysis, we have the same sets of steps to take for contrasts.

Study Design Decisions

Our first steps for contrasts is to make sure that contrasts are actually warranted by our research questions. Notice that I wrote the plural form of question. In the pairwise case of Post Hoc analysis, the pairwise SRQs are implied by our main SRQ. The same is not true for contrasts: we actually need to articulate what contrasts we're interested in investigating.

> In addition to investigating whether the strain of starting bacteria has an impact on the amount of free amino acids in the cheese, we have two additional research questions. First, we want to know whether having any of the two starter strains leads to a difference in free amino acids as compared to not having either starter strains. Second, we want to also investigate if getting both strains is statistically different from getting one of the two strains.

The above paragraph lays out the statistical research "questions" (in statement form) for two contrasts. The first contrast proposes combining three of the levels of our factor (i.e., A, B, and both–AB) together and comparing them to the fourth level (i.e., none). The second question has us compare the level of both strains (i.e., AB) against the combination of individual strains (i.e., A and B). (Yes, the wording for this second question is rather close to what we might anticipate for pairwise comparisons.)

As an example for an algebraic form of the hypotheses for contrasts, consider the following for the first Cheese study contrasts:

$$\begin{aligned} H_0: &\quad \frac{\mu_A + \mu_B + \mu_{AB}}{3} = \mu_N \\ H_1: &\quad \frac{\mu_A + \mu_B + \mu_{AB}}{3} \neq \mu_N \end{aligned}$$

Testing Families

Just as with pairwise comparison based Post Hoc analysis, we still need to conceptualize a testing family for contrasts. One approach you can take is to simply imagine all of your contrasts belonging to one testing family. This can give you two testing families for Post Hoc analysis: one for (all) the pairwise comparisons and one for contrasts. Once you conceptualize your testing family of contrasts, be sure you record the number of comparisons, m. For the above example, $m = 2$.

Choose Your Type I Error Rate and Method

Just as with pairwise Post Hoc analysis, we still have to select a Type I Error Rate and method. The table from the Post Hoc guide/tutorial appears below. I've placed the Scheffé method in bold face as this method is built especially for contrasts and guards against any data snooping that might have occurred.

Selected Type I Error Rate	Possible Parametric Methods
Simultaneous Confidence Intervals	Bonferroni, Šidák, Tukey HSD, Tukey-Kramer HSD, **Scheffé**
Strong Familywise/Maximum Experimentwise	Hochberg, Holm, REGWR, Gabriel, Dunnett, DSCF
False Discovery Rate	Benjamini-Hochberg, Student-Newman-Keuls
Experimentwise Error Rate	[ANOVA Omnibus Tests,] Protected LSD
Comparisonwise Error Rate	Most Two Sample Tests, Unprotected LSD

After picking which Error Rate you want to control, you choose an appropriate method. There are some pieces of guidance you can follow for choosing the particular method. However, a lot comes down to personal preference.

Checking Appropriateness and Assumptions

Again, we'll bank upon the appropriateness checks and your assessment of assumptions for the omnibus test (i.e., the ANOVA F test) for our contrasts.

As a refresher for the Cheese study, here is a reproduction of the modern ANOVA table for this study:

Table 2: ANOVA Table for Free Amino Acids in Cheese Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
strain	5.6279	3	1.8760	11.9318	0.0183	0.8995	0.8039	0.8241
Residuals	0.6289	4	0.1572					

Setting Up Your Contrasts

There are two parts to setting up your contrast in R: making sure you know the correct order of your factor levels and then saving the weights.

Check the Order

Checking the order of your factor levels is a quick application of the `levels` function:

```r
# Demo code to check the order of factor levels
levels(cheeseData$strain)
```

```
## [1] "A"    "AB"   "B"    "None"
```

The output tells us that our contrasts will need to be in the order (A, AB, B, None).

You must always check the ordering that the software is using. Suppose that we used "(A & B)" instead of "AB" for both strains together. The ordering would then be ((A & B), A, B, None). Any changes you make to the names of the levels will impact the ordering.

Save the Weights

Now that we know the order of the factor levels, we can create contrasts by saving the weights for each level as a vector. For example,

```r
# Demo code to save the contrasts weights as vectors
c1 <- c(1/3, 1/3, 1/3, -1)
c2 <- c(-1/2, 1, -1/2, 0)
```

The first contrast, `c1`, pools the cheeses that received either Strain A, Strain B, or Both together and compares them against those which received the base set of cultures ("None"). The second contrast, `c2`, compares the pooling of Strain A and Strain B against the combination of both strains. By saving the contrasts, they are now available for our use.

Checking Your Contrasts

There are a couple of conditions that we need to check with our contrasts before we actually test them.

Weights Sum to Zero

The first check we need to do is make sure that contrasts weights sum to zero. You can do this visually/mentally, or you can ask `R` to do so. One argument to have `R` do this check is to make sure you didn't make a typo when programming the contrasts. You can perform this with the `sum` function:

```
# Demo code to check that the weights add to zero
sum(c1)
```

```
## [1] -5.551115e-17
```

```
sum(c2)
```

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

Notice that `sum(c1)` is a non-zero value. This is a byproduct of computer arithmetic. Given that this value is -5.55×10^{-17}, we'll go ahead and say this is zero.

On the other hand, if you were to see a sum such as 0.072, that would be sign that something is wrong with our weights.

Checking Orthogonality

When we have multiple contrasts, we often like to have "orthogonal contrasts". We like orthogonal contrasts for two reasons. First, this means that they are independent from each other. Second, orthogonal contrasts will perfectly partition the sums of squares for the factor they are applied to. This fact is especially useful in situations where we want to compare a set of new treatments to a standard care or null treatment.

Two contrasts, w_i and $w_i^\star$, are said to be orthogonal if the weighted sum of their i-th components add to zero. That is,

$$\sum_{i=1}^{g} \frac{w_i \cdot w_i^\star}{n_i}$$

In this formula, the weighting of the weights comes from the sample size for each component, n_i. For our two Cheese study contrasts, we would have the following:

$$\begin{aligned}\sum_{i=1}^{4} \frac{w_i \cdot w_i^\star}{n_i} &\Rightarrow \frac{1/3 * 1}{2} + \frac{1/3 * -1/2}{2} + \frac{1/3 * -1/2}{2} + \frac{-1 * 0}{2} \\ &= \frac{1}{6} + \frac{-1}{12} + \frac{-1}{12} + 0 \\ &= 0\end{aligned}$$

Thus our two contrasts, `c1` and `c2`, are orthogonal to each other.

We can use `R` to help us check orthogonality of our contrasts:

```
# Demo code for checking orthogonality of two contrasts
# Create a vector of sample sizes in the same order as the contrast vectors
n <- c(2,2,2,2)

sum((c1*c2)/n)
```

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

Testing Your Contrasts in `R`

Once you have set up your contrasts, you have two routes to take for testing them: base packages or using the `DescTools` package.

Base Package Approach

The base package approach allows you to explore a variety of contrasts on multiple different factors. However, analyzing contrasts takes a series of steps.

The Steps of Doing Contrasts in Base `R`

1) Construct the contrast weight vectors
2) Connect the contrast weight vectors to each factor
3) (Re-) Run the ANOVA F Test
4) Store the Correct ANOVA Output
5) Adjust the p-values
6) Make a Professional Looking Table

Exploring the Code

We've already discussed how we can construct, store, and check our contrast weight vectors. We will use both `c1` and c2‘ in our work.

The next step is to connect these contrasts to our factor. There's nothing inherent in how we constructed the vectors that says "Hey, these are for the strain of bacteria!". We have to tell `R` that this is what we want. To do this, we'll used the `contrasts` function:

```r
# Demo code for setting contrasts in the Cheese study
contrasts(cheeseData$strain) <- cbind(c1, c2)

# Generic example
# contrasts(dataFrame$factor) <- weightMatrix
```

Now we need to (re-)run the ANOVA model. If you ran the ANOVA model to get residuals but hadn't yet added on the contrasts, then you need to re-run everything. This will ensure that the contrasts get tested with the parametric shortcut.

```r
# Demo code for testing contrasts in the Cheese study
cheeseContrasts <- aov(
  formula = acids ~ strain,
  data = cheeseData
)
```

Now we get to a bit of a tricky point: we need to isolate the appropriate portion of the model results. For quick examinations we would use `summary` to do this. However, neither of these methods will display the tests for the contrasts. Thus, we need to make use of a new argument: `split`

```r
contrastOut <- summary(
  object = cheeseContrasts,
  split = list( # Apply meaningful labels to your contrasts here
    strain = list(
      "AB, A, and B vs. None" = 1,
      "AB vs. A or B" = 2
    )
  )
)
```

The `split` argument allows us to call the contrasts we want to explore. The structure of the `split` argument is a list of lists. The outer list has elements which start with the name of the factor; `stain = list(`. The inner lists begin with the meaningful labels (e.g., `"(A&B), A, and B vs. None"`) and then a number which corresponds to order in which you listed the weight vectors in the `cbind` call.

Now that we have stored the output that we need, `contrastOut`, we can impose our adjustments for the multiple comparison problem and create a professional looking table.

```r
# Demo code for adjusting p-values and making a professional table
adjustPValues(
  contrastObject = contrastOut,
  method = "fdr"
) %>%
  knitr::kable(
    digits = 4,
    col.names = c(
      "DF", "SS", "MS", "F", "Raw p-value", "Adj. p-value"),
    caption = "ANOVA Table for Free Amino Acids in Cheese Study",
    booktabs = TRUE,
    align = rep("c", 5)
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 3: ANOVA Table for Free Amino Acids in Cheese Study

	DF	SS	MS	F	Raw p-value	Adj. p-value
strain	3	5.6279	1.8760	11.9318	0.0183	
strain: AB, A, and B vs. None	1	2.0434	2.0434	12.9969	0.0227	0.0453
strain: AB vs. A or B	1	2.8188	2.8188	17.9287	0.0133	0.0266
Residuals	4	0.6289	0.1572			

The `adjustPValues` function comes from my set of helper tools (`ANOVATools`). You need to pass this function the output of the `summary` call with the `split` argument and you need to state which method you want to use. The `method` argument supports all of the following methods:

Chosen Method	Set `method =`
Bonferroni	`"bonferroni"`
Holm	`"holm"`
Hochberg	`"hochberg"`
Benjamini & Hochberg	`"BH"` OR `"fdr"`

We can pipe the output of `adjustPValues` into a `kable` call to make our professional table. We will not be using the `parameters` package with contrasts to make modern ANOVA tables. (Still working on some bugs.) In Table 3, the two contrasts appear as the second and third rows of our table.

`DescTools` Package

The `DescTools` package includes the `ScheffeTest` function that will allow you test your contrasts with the Scheffé method.

Unlike the base package approach, you will not get an ANOVA table, but rather, you'll get a table like what you would see when doing pairwise post hoc analysis. The process here has three steps:

1) Fit your ANOVA model just as you typically would.
    - You do not need to bind your contrasts to your factor beforehand.
2) Save the output of the `DescTools::ScheffeTest` to an object.
3) Make a professional looking table.

Examine the following example code:

```r
# Demo code for testing contrasts
# DescTools Scheffe Test Approach
## Step 1 -- Fit the ANOVA model as usual
cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheeseData,
  na.action = "na.omit"
)

## Step 2--Save output of Scheffe Test
scheffeCheese <- DescTools::ScheffeTest(
  x = cheeseModel,
  contrasts = cbind(c1, c2),
  conf.level = 0.9, # 1 -- Your Overall Type I Error Rate
)

## Step 3--Make a Professional looking table
knitr::kable(
  x = scheffeCheese[[1]], # Grab the output
  digits = 4,
  col.names = c(
    "Difference", "Lower Bound", "Upper Bound", "p-value"),
  caption = "Scheffe Test Results",
  booktabs = TRUE,
  align = rep("c", 4)
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 5: Scheffe Test Results

	Difference	Lower Bound	Upper Bound	p-value
A,AB,B-None	1.1672	0.0192	2.3151	0.0953
AB-A,B	1.4540	0.2364	2.6716	0.0585

We will not worry about effect sizes with contrasts (shocking, I know).

Reporting and Interpreting Results

For this example, I'm going to suppose that we choose to use the Scheffé method to control our Type I Error rate at no more than 10%.

Given the results in Table 5, we can see that for our first contrast, adding either Strain A, Strain B, or both strains of starting bacteria, appear to lead to higher levels of free amino acids in the cheese than the naturally occurring cultures (i.e., "none" for additional strains). We would anticipate getting a difference at least as large (magnitude) around 9.5% of the time when there is no difference between the three-way combination and no additional bacteria.

For our second question, we see a slight more unusual result. Comparing the joint treatment of both strains with the combination of the individual inoculations, we anticipate observing a difference at least as large (in magnitude) as 1.45 units/cheese around 6% of the time if there actually was no difference. This suggests that if we want more umami-rich and firmer cheese (the result of higher free amino acids), then we should use both Strain and Strain B in the cheese making process rather than just one or the other.

Putting Things Together–Your Turn

Use what you've seen here in the context of the Song Knowledge study to test the contrast of graduating students vs. non-graduating students.

Note: since there is just one contrast in this family, you won't have to worry about adjustments to p*-values.*

Code Appendix

```r
# Setting Document Options
knitr::opts_chunk$set(
  echo = FALSE,
  warning = FALSE,
  message = FALSE,
  fig.align = "center"
)

packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "DescTools")
lapply(packages, library, character.only = TRUE)

options(knitr.kable.NA = "")
options(contrasts = c("contr.sum", "contr.poly"))

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
# Demo code for setting up R for contrasts
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "DescTools")
lapply(packages, library, character.only = TRUE)

options(knitr.kable.NA = "")
options(contrasts = c("contr.sum", "contr.poly"))

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")

# Create the Cheese data frame
cheeseData <- data.frame(
  strain = as.factor(
    c("None", "None", "A", "A",
      "B", "B", "AB", "AB")
  ),
  acids = c(
    4.195, 4.175, 4.125, 4.735,
    4.865, 5.745, 6.155, 6.488
  )
)

# Song Knowledge study
songData <- read.csv(
file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/songKnowledge_Spring2022.csv",
header = TRUE,
sep = ","
)
# Set year to an ordered factor
songData$year <- factor(
x = songData$year,
levels = c("sophomore", "junior", "senior")
)

# Create a modern ANOVA table to get started.
cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheeseData,
  na.action = "na.omit"
)
```

```r
parameters::model_parameters(
  model = cheeseModel,
  effectsize_type = c("eta", "omega", "epsilon")
) %>%
  knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "ANOVA Table for Free Amino Acids in Cheese Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("HOLD_position")
  )

# Demo code to check the order of factor levels
levels(cheeseData$strain)

# Demo code to save the contrasts weights as vectors
c1 <- c(1/3, 1/3, 1/3, -1)
c2 <- c(-1/2, 1, -1/2, 0)

# Demo code to check that the weights add to zero
sum(c1)
sum(c2)

# Demo code for checking orthogonality of two contrasts
# Create a vector of sample sizes in the same order as the contrast vectors
n <- c(2,2,2,2)

sum((c1*c2)/n)

# Demo code for setting contrasts in the Cheese study
contrasts(cheeseData$strain) <- cbind(c1, c2)

# Generic example
# contrasts(dataFrame$factor) <- weightMatrix

# Demo code for testing contrasts in the Cheese study
cheeseContrasts <- aov(
  formula = acids ~ strain,
  data = cheeseData
)

contrastOut <- summary(
  object = cheeseContrasts,
  split = list( # Apply meaningful labels to your contrasts here
    strain = list(
      "AB, A, and B vs. None" = 1,
      "AB vs. A or B" = 2
    )
  )
)

# Demo code for adjusting p-values and making a professional table
adjustPValues(
  contrastObject = contrastOut,
  method = "fdr"
) %>%
  knitr::kable(
    digits = 4,
    col.names = c(
      "DF", "SS", "MS", "F", "Raw p-value", "Adj. p-value"),
    caption = "ANOVA Table for Free Amino Acids in Cheese Study",
    booktabs = TRUE,
    align = rep("c", 5)
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo code for testing contrasts
# DescTools Scheffe Test Approach
## Step 1 -- Fit the ANOVA model as usual
cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheeseData,
  na.action = "na.omit"
)

## Step 2--Save output of Scheffe Test
scheffeCheese <- DescTools::ScheffeTest(
  x = cheeseModel,
  contrasts = cbind(c1, c2),
  conf.level = 0.9, # 1 -- Your Overall Type I Error Rate
)

## Step 3--Make a Professional looking table
knitr::kable(
  x = scheffeCheese[[1]], # Grab the output
  digits = 4,
  col.names = c(
    "Difference", "Lower Bound", "Upper Bound", "p-value"),
  caption = "Scheffe Test Results",
  booktabs = TRUE,
  align = rep("c", 4)
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```