Pairwise Post Hoc Analysis for One-way ANOVA

Neil J. Hatfield

Last Updated: March 19, 2023

In this tutorial, we are going to explore doing Pairwise Post Hoc analysis in the context of One-way ANOVA models. We will do this in the context of parametric shortcuts.

1 Setting up `R` and Loading Data

As is always the case, we must do a bit of housekeeping to ensure that `R` will work for us nicely.

1.1 Loading Packages, Setting Options, and Additional Tools

For this tutorial/guide, we will make use of the following packages: `tidyverse`, `knitr`, `kableExtra`, `parameters`, `emmeans`, `DescTools`, `dunn.test`, and `multcompView`.

We will also need to set our two options: telling `R` that table cells with missing values should visually look empty and that we require treatment/group effects for each factor to sum to zero.

Finally, we will want to load the set of tools I've created as there are two functions that will directly help us with Post Hoc analysis.

Here is a reminder for how to write this code in `R`:

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

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

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

1.2 Loading Data

For this guide/tutorial, we are going to make use of the **Free Amino Acids in Cheese** (or the Cheese study). For information on the Cheese study, look at Table 5.2 and Example 5.5 from page 91 of Oehlert. You can also make use of the Spring 2023 Song Knowledge Study for the Your Turn examples.

Take a moment to think about how you would read in the data then compare your code with the following:

```r
# Demo code for loading data ----
## Song Data
songData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/songKnowledge_Sp23.csv",
  header = TRUE,
  sep = ","
)
### Set year to an ordered factor
songData$Year <- factor(
  x = songData$Year,
  levels = c("Junior", "Senior", "Other")
)

## Cheese study
### Manual Entry
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
  )
)

# Notice that I used as.factor in the data.frame call;
## this is an approach you can use when manually creating a data frame for ANOVA
```

2 Key Decisions for Post Hoc Analysis

When we think about Post Hoc Analysis (either the pairwise explorations of this guide/tutorial or contrasts/linear combinations of a separate guide/tutorial), there are some steps that we need to take first.

2.1 Study Design Decisions

Our first steps for Post Hoc Analysis actually relate to our study design. Post Hoc analyses are different from our initial approach in that we engage in these analyses ***after*** we've seen the data. In fact, we only engage in Post Hoc analyses when we reject the null hypothesis for the (One-way) ANOVA model. To help guard ourselves against any claims that we might be up to problematic behaviors, we often add a couple of sentences to our study design such as the following:

> After getting and cleaning the data, we'll use the parametric ANOVA F test to answer the research question at a 5% level of significance. For Post Hoc analysis, we'll use an overall Type I Error Rate of 5% using Tukey's HSD.

Some of the above statement should feel familiar: 1) choice of the parametric shortcut, 2) statement of our Unusualness Threshold (level of significance), and 3) a statement of our overall Type I Risk. However, this third element has some slightly different language–"Error Rate" vs. "Risk". There is an additional element of method ("Tukey's HSD").

There are several different ways in which we can operationalize or quantify Type I Risk. We refer to these ways as "Type I Error Rates" and there are five common types. We have to pick one of these types of error rates to then select a method.

2.1.1 Testing Families

To help us pick a method, we need to conceptualize our post hoc "testing family"; that is, if we reject the null hypothesis from the "omnibus" ANOVA test (via shortcut, permutation, bootstrapping, etc.), what combinations of groups/treatments do we want to test? Here are few examples of testing families:

- Pairwise Combinations
  - All possible pairs
    - Ex: Seniors vs. Juniors, Seniors vs. Others, and Juniors vs. Others
  - Special Pairs

        - ∗ Compare to Null Treatment
            - · Ex: Strain A vs. No starting strain, Strain B vs. No starting Strain, Straings A & B vs. No starting strain
        - ∗ Comparing to Standard Care
            - · Ex: New Drug 1 vs. Current, New Drug 2 vs. Current, New Drug 3 vs. Current
        - ∗ Comparing to "Best" Treatment
            - · Ex: Replace "Current" with what we've previously identified as "best"
- (Linear) Combinations ("Contrasts")
    - Ex: Graduating vs. Not Yet Graduating
    - Ex: Oral Medication (aspirin, acetaminophen, ibuprofen) vs. Topical Medications (lidocaine, capsaicin, benzocaine) for pain relief

In this tutorial/guide, we'll focus on pairwise families. (We'll have a separate tutorial/guide for the combinations.) One of the key aspects of conceptualizing your post hoc testing family is to count the number of comparisons, m, that make up the family.

2.1.2 Choose Your Type I Error Rate and Method

Once you conceptualize your testing family, you can yourself "Which Type I Error Rate do I want to control?" The following table shows the five different Type I Error Rates we can choose from. I've ordered the table from the most conservative to the least conservative; top to bottom. This order also reflects moving from the strongest guards against Type I Errors (top) to the weakest guards against Type I Errors (bottom).

Selected Type I Error Rate	Possible Adjustment 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.

If you aren't sure what method to pick, ask yourself the following questions:

- Am I looking at all possible pairwise comparison?
    - **Yes**: Use Tukey (-Kramer) HSD; **No**: Use Bonferroni
- Did someone data snoop?
    - **Yes**: Must use Scheffé
- Want to use middle ground?
    - **Yes**: Use Benjamini-Hochberg, provided you have independent tests
- Have no idea?
    - **Yes**: Tukey (-Kramer) makes a good default

2.2 Checking Appropriateness and Assumptions

Post Hoc analyses involve hypothesis tests. And just like any other hypothesis test, these all have their own set of conditions for whether they are appropriate and for whether we've met their assumptions (note: all methods we're looking at in this course are shortcuts). However, we have a useful advantage.

Since we're doing these tests in a planned, Post Hoc fashion, ***after*** we check the appropriateness of (One-way) ANOVA methods and assess the assumptions of the shortcut, our lives become easy. In essence, if we have satisfied the assumptions for the omnibus shortcuts, then we have satisfied the assumptions for the Post Hoc tests. Similarly, if ANOVA is appropriate, then we know that our Post Hoc methods are appropriate. Thus, taking the time to carefully check appropriateness AND assumptions for the omnibus test pays dividends for us.

2.3 Consistency of Approach

The last decision we have to make is really a consistency check. We typically want to continue to use the same approach as we did for the omnibus test. That is to say, if we used a simulation method (permutation, bootstrapping) or a shortcut to get the sampling distribution for F in our omnibus test, then we should continue using that method for Post Hoc analysis.

Typically, we want to use parametric shortcuts as they are more powerful (better Type II error control) *provided* their assumptions are satisfied. If you start with a parametric shortcut (i.e., the ANOVA F test), then you need to use a parametric shortcut for the Post Hoc analysis.

3 Post Hoc for the Parametric Shortcut

For Post Hoc analysis, we need to have fitted our ANOVA model. Here is the code and modern ANOVA table for the Cheese Study; I'll leave you to do the same with the Song Knowledge study.

```r
# Demo Code for Fitting a One-way ANOVA Model ----
cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheeseData,
  na.action = "na.omit"
)

# Make a modern ANOVA table
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")
  )
```

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					

We will say for the Cheese Study that we'll reject the null hypothesis and decide to act as if the strain of starter bacteria does impact the level of free amino acids. I will use the Cheese study as my coding example. You should mimic my code not only with the Cheese study (i.e., reproducing my results) but also apply the code to the Song Knowledge study.

I'm going to demonstrate multiple approaches using a variety of packages.

3.1 The emmeans Approach

The `emmeans` package deals with estimating the *marginal means* given a statistical model fitted to data ("emmeans" stands for estimated marginal means). When we do Post Hoc analysis, we want to ensure that we are working from the model we just tested. Further, the `emmeans` approach will serve as a basis for future analyses that we'll cover (e.g., contrasts, post hoc for advanced models).

```r
# Demo Code for emmeans Post Hoc ----
cheesePHPairs <- emmeans::emmeans(
  object = cheeseModel, # Your aov/lm object
  specs = pairwise ~ strain, # Creates all pairs of the levels of the factor listed
  adjust = "tukey", # How you want to control the error rate
  level = 0.9 # 1 - Your overall Type I Error Rate
)

## Make a professional looking table
knitr::kable(
  x = cheesePHPairs$contrasts, # Grab the appropriate sub-object
  digits = 3,
  caption = "Pairwise Post Hoc Comparison via Tukey HSD",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value"),
  align = "lccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 3: Pairwise Post Hoc Comparison via Tukey HSD

Pair	Difference	SE	DF	t	p-value
A - AB	-1.892	0.397	4	-4.770	0.030
A - B	-0.875	0.397	4	-2.207	0.264
A - None	0.245	0.397	4	0.618	0.921
AB - B	1.017	0.397	4	2.564	0.187
AB - None	2.137	0.397	4	5.388	0.019
B - None	1.120	0.397	4	2.825	0.146

You can directly compare the *p*-values shown in Table 3 to your chosen Unusualness Threshold to decide whether to reject the null hypothesis that there is no statistically significant difference between the pair of levels.

The `emmeans` approach is a nice one to take and has two distinct advantages:

1) This gives you a single set of code which you can use with ten different adjust methods (see below).
2) The `specs` argument gives us some great flexibility (we'll see that flexibility in Unit 4).

The downside is that you do need to install and load the `emmeans` package in order to use this method. Additionally, this method does not produce confidence intervals for the differences.

Available adjustment methods for control our Type I error rate include `"bonferroni"`, `"tukey"`, `"sidak"`, `"scheffe"`, `"holm"`, `"hochberg"`, `"hommel"`, `"BH"` or `"fdr"`, `"BY"`, or `"none"` (for no Type I Error Rate adjustment).

3.1.1 Quick Note on Pair Order

The `emmeans` package (and all others) has its own internal pair construction method and not every approach will make the pairwise comparisons in the same order. For example, in Table 3, we have the pair of A and None. We can express this pair in two ways: [A – None] and [None – A]. While these will result in different differences, there is a relationship between them. They are opposites of one another: $(A - B) = -1 * (B - A)$. If you need to express the pairwise comparison in the other order, all you need to do is multiple the difference (or the limits of a confidence interval) by -1.

3.2 Tukey's [& Kramer's] Honest Significant Difference

Using Tukey's or Tukey-Kramer's HSD is a decent default method to use to control the SCI Type I Error Rate. The distinction between Tukey's HSD and Tukey-Kramer's HSD is that the former is for exactly balanced designs; the later is for imbalanced designs. Both are accessed through the same function, `TukeyHSD` and R will automatically use the correct one. (Note: other functions also automatically adjust Tukey to Tukey-Kramer.)

```r
# Demo Code for Using Tukey HSD Post Hoc ----
hsdPH <- TukeyHSD(
  x = cheeseModel, # Your aov/lm object
  conf.level = 0.95 # 1 -- Your overall Type I Error level
)

## Kable Code for Tukey HSD
knitr::kable(
  x = hsdPH$strain, # Notice the factor's name
  digits = 3,
  caption = "Post Hoc Tukey HSD Comparisons",
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 4: Post Hoc Tukey HSD Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
AB-A	1.892	0.277	3.506	0.030
B-A	0.875	-0.739	2.489	0.264
None-A	-0.245	-1.859	1.369	0.921
B-AB	-1.017	-2.631	0.598	0.187
None-AB	-2.137	-3.751	-0.522	0.019
None-B	-1.120	-2.734	0.494	0.146

You have confidence intervals (Lower Bound, Upper Bound) as well as p-values that are adjusted for the HSD. You can directly compare these to your Unusualness Threshold, $UT \leq \mathcal{E}_I$.

The `TukeyHSD` is part of the base build of R and does not require any special packages.

3.2.1 Quick Note on Pair Order

Notice that `tukeyHSD` (Table 4) used a different comparison ordering than `emmeans` (Table 3). However, if you multiple the differences by -1, you get the same result. Further notice that the p-values are unaffected by this difference.

3.3 `DescTools` Method

The `DescTools` package also provides a function that will allow you to pairwise comparisons for Post Hoc analysis; `PostHocTest`.

```
# Demo Code for DescTool Pairwise Post Hoc ----
dtPH <- DescTools::PostHocTest(
  x = cheeseModel, # Your aov/lm object
  method = "newmankeuls", # Your chosen method
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
)

## Kable Code for DescTools
knitr::kable(
  x = dtPH$strain, # Notice the use of the factor name
  digits = 3,
  caption = paste( # Creates a nice title; copy at will
    "Post Hoc",
    attr(dtPH, "method"),
    "Comparisons"
  ),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 5: Post Hoc Newman-Keuls Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
AB-A	1.892	0.777	3.006	0.019
B-A	0.875	0.030	1.720	0.092
None-A	-0.245	-1.090	0.600	0.570
B-AB	-1.017	-1.862	-0.171	0.062
None-AB	-2.137	-3.422	-0.851	0.019
None-B	-1.120	-2.235	-0.005	0.099

This table is much like the one for Tukey's HSD.

You can change which method gets used by changing the value of the `method` argument:

Chosen Method	Set `method =`
Tukey HSD	`"hsd"`
Dunn's (Bonferroni)	`"bonf"`
Fisher's Least Significant Difference	`"lsd"`
Scheffé	`"scheffe"`
Newman-Keuls	`"newmankeuls"`

If I'm after using the Least Significant Difference, then I use the `DescTools` method.

3.4 Special Comparisons (Dunnett's)

In certain studies, you might want to do all possible pairwise comparisons. You might want to compare treatments to a null treatment, a standard care treatment, or the current "gold standard". To make such comparisons, we will need to use the `DescTools` package and Dunnett's Test (`DunnettTest`).

```
# Demo Code for Special Pairwise Comparisons ----
## Dunnett's Test via DescTools
dunnett <- DescTools::DunnettTest(
  formula = acids ~ strain,
  data = cheeseData,
  control = "None", # Enter the level that you want to compare to
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
)

## Kable Code for Dunnett's Test
knitr::kable(
  x = dunnett$`None`, # Note the use of special treatment level
  digits = 3,
  caption = paste("Post Hoc Comparisons--Dunnett's Test"),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 7: Post Hoc Comparisons–Dunnett's Test

	Difference	Lower Bound	Upper Bound	Adj. p-Value
A-None	0.245	-0.899	1.389	0.869
AB-None	2.136	0.993	3.280	0.013
B-None	1.120	-0.024	2.264	0.104

4 Post Hoc Effect Sizes

When you have used the parametric shortcuts, you'll want to use the `anova.PostHoc` function from my ANOVATools.R helper file. You'll pass your model object (the result of calling `aov` or `lm`) as the input to this function.

- Note 1: this will generate ALL pairwise comparisons. If you only did a subset, you'll want to remove the extra rows. (The `anova.PostHoc` function returns a data table.)
- Note 2: some special characters (e.g., "&") will get get replaced with periods. I am working on fixing this.

```
# Demo Code for Post Hoc Effect Sizes ----
anova.PostHoc(cheeseModel) %>%
  knitr::kable(
    digits = 3,
    caption = "Post Hoc Comparison Effect Sizes",
    col.names = c("Pairwise Comparison","Cohen's d", "Hedge's g",
                  "Prob. Superiority"),
    align = 'lccc',
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 8: Post Hoc Comparison Effect Sizes

Pairwise Comparison	Cohen's d	Hedge's g	Prob. Superiority
A vs. AB	-5.443	-3.110	0.000
A vs. B	-1.634	-0.934	0.124
A vs. None	0.803	0.459	0.715
AB vs. B	2.161	1.235	0.937
AB vs. None	12.809	7.319	1.000
B vs. None	2.545	1.454	0.964

Cohen's d and Hedge's g are both measures of the distance between the *Sample Arithmetic Mean* values for the two groups scaled by pooled standard deviations (Hedge's g uses a weighted pooling). Thus, you can think of these as saying that there are 5.443 standard deviations between the performance for using both A & B strains and just the A strain in terms of free amino acid levels. Cohen's d and Hedge's g are also sensitive to the order of comparison. Just like the pairwise differences, you can multiply these values by -1 to get reverse the order of subtraction.

The Probability of Superiority measure the percent of the time a randomly selected member of the first group will have the higher numeric value of the response than a randomly selected member of the second group. Thus, we can say that only 12% of the time we randomly select a cheese given just the A strain will that cheese have more free amino acids than a randomly selected cheese given just the B strain.

Just like the pairwise differences and the other effects sizes, Probability of Superiority is sensitive to the order of comparison (subtraction). However, **unlike** the pairwise differences and the other effect sizes, Probabilities of Superiority *can not* be multiplied by -1 to change the order of comparison. Further, you also *can not* just take the complement. If we wanted the Probability that B was greater than A, we would need to re-calculate the effect sizes. We are given $P[A > B]$ whose complement is $1 - P[A > B] = P[B \geq A] = P[B \geq A] + P[B = A]$. The `probSup` function in my toolkit will return the appropriate probability of superiority given a value of Cohen's d.

5 Visualizing Pairwise Comparisons

There are a number of ways that you can visualize your pairwise comparisons. A useful package in this area is `multcompView` ("Visualizations of Paired Comparisons) package.

5.1 Connecting Letter Report

The Connecting Letters Report is similar to the Oehlert's Underline diagrams (p. 88). Rather than using lines, we use lowercase letters. If different treatments share the same letter, they are statistically indistinguishable from one other. If they have different letters, then there is a statistical difference between those treatments.

```
# Demo Code for Connecting Letters Report ----
## Note: You can not use the emmeans object here
multcompView::multcompLetters4(
  object = cheeseModel, # Your aov/lm object
  comp = hsdPH, # Your stored comparisons
  threshold = 0.1 # Your Overall Type I Error Rate
)
```

```
## $strain
##   AB    B    A None
##  "a" "ab"  "b"  "b"
```

As you can see from the above raw output, the Connecting Letters report in raw does not look very nice. However, we can improve upon this by joining the Connecting Letters Report with box plots.

5.2 Box Plot + Connecting Letters

```r
# Demo Code for a Box Plot with Connecting Letters ----
## Note 1: you can not easily set your threshold; uses 5%
## Note 2: you can not use anything other than Tukey
multcompView::multcompBoxplot(
  formula = acids ~ strain,
  data = cheeseData,
  compFn = "TukeyHSD",
  plotList = list(
    boxplot = list(fig = c(0, 0.85, 0, 1)),
    multcompLetters = list(
      fig = c(0.8, 0.9, 0.1, 0.9),
      fontsize = 12,
      fontface = "bold"
    )
  )
)
```

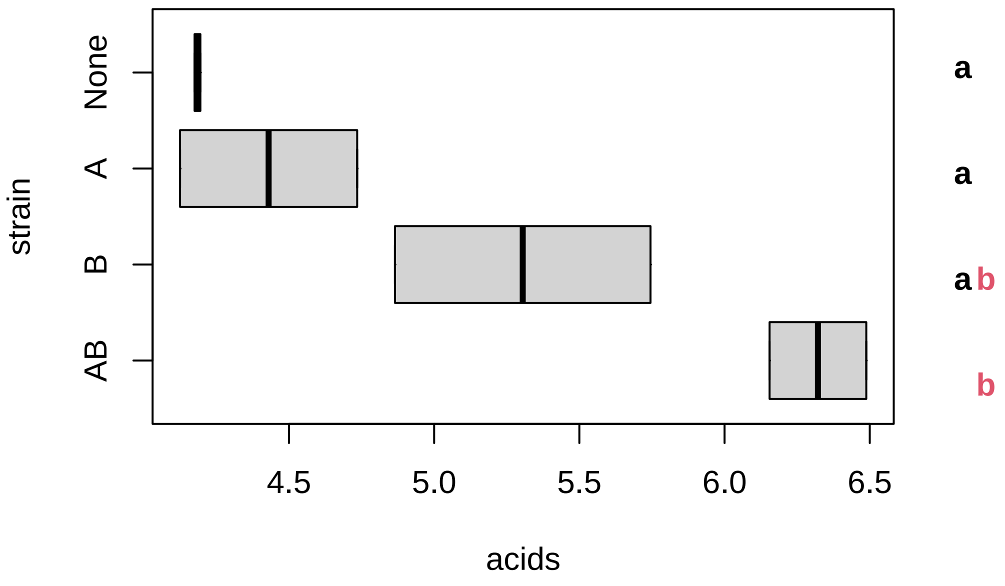

Figure 1: Box Plot and Connecting Letters Report for Free Amino Acids in Cheese Study

Play with the values of `fig = c(x1, x2, y1, y2)` to get the best arrangement for your case.

- `x1` is the start proportion of the horizontal (left edge is 0)
- `x2` is then end proportion of the horizontal (right edge is 1)
- `y1` is the start proportion of the vertical (bottom edge is 0)
- `y2` is the end proportion of the vertical (top edge is 1)

6 Putting Things Together–Your Turn

Use the above examples to explore the Song Knowledge study in terms of Post Hoc Analysis. Generate not only which years in school are significantly different from each other, but what are effect sizes and see if you can re-create the following plot.

7 Code Appendix

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

# Load packages ----
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "emmeans", "DescTools",
              "dunn.test", "multcompView")
lapply(packages, library, character.only = TRUE)

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

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

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

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

# Demo code for loading data ----
## Song Data
songData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/songKnowledge_Sp23.csv",
  header = TRUE,
  sep = ","
)
### Set year to an ordered factor
songData$Year <- factor(
  x = songData$Year,
  levels = c("Junior", "Senior", "Other")
)

## Cheese study
### Manual Entry
cheeseData <- data.frame(
  strain = as.factor(
    c("None", "None", "A", "A",
      "B", "B", "AB", "AB")
  ),
  acids = c(
```

```r
    4.195, 4.175, 4.125, 4.735,
    4.865, 5.745, 6.155, 6.488
  )
)

# Notice that I used as.factor in the data.frame call;
## this is an approach you can use when manually creating a data frame for ANOVA

# Demo Code for Fitting a One-way ANOVA Model ----
cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheeseData,
  na.action = "na.omit"
)

# Make a modern ANOVA table
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 for emmeans Post Hoc ----
cheesePHPairs <- emmeans::emmeans(
  object = cheeseModel, # Your aov/lm object
  specs = pairwise ~ strain, # Creates all pairs of the levels of the factor listed
  adjust = "tukey", # How you want to control the error rate
  level = 0.9 # 1 - Your overall Type I Error Rate
)

## Make a professional looking table
knitr::kable(
  x = cheesePHPairs$contrasts, # Grab the appropriate sub-object
  digits = 3,
  caption = "Pairwise Post Hoc Comparison via Tukey HSD",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value"),
  align = "lccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

```
# Demo Code for Using Tukey HSD Post Hoc ----
hsdPH <- TukeyHSD(
  x = cheeseModel, # Your aov/lm object
  conf.level = 0.95 # 1 -- Your overall Type I Error level
)

## Kable Code for Tukey HSD
knitr::kable(
  x = hsdPH$strain, # Notice the factor's name
  digits = 3,
  caption = "Post Hoc Tukey HSD Comparisons",
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for DescTool Pairwise Post Hoc ----
dtPH <- DescTools::PostHocTest(
  x = cheeseModel, # Your aov/lm object
  method = "newmankeuls", # Your chosen method
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
)

## Kable Code for DescTools
knitr::kable(
  x = dtPH$strain, # Notice the use of the factor name
  digits = 3,
  caption = paste( # Creates a nice title; copy at will
    "Post Hoc",
    attr(dtPH, "method"),
    "Comparisons"
  ),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for Special Pairwise Comparisons ----
## Dunnett's Test via DescTools
dunnett <- DescTools::DunnettTest(
  formula = acids ~ strain,
  data = cheeseData,
  control = "None", # Enter the level that you want to compare to
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
```

```
)

## Kable Code for Dunnett's Test
knitr::kable(
  x = dunnett$`None`, # Note the use of special treatment level
  digits = 3,
  caption = paste("Post Hoc Comparisons--Dunnett's Test"),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for Post Hoc Effect Sizes ----
anova.PostHoc(cheeseModel) %>%
  knitr::kable(
    digits = 3,
    caption = "Post Hoc Comparison Effect Sizes",
    col.names = c("Pairwise Comparison","Cohen's d", "Hedge's g",
                  "Prob. Superiority"),
    align = 'lccc',
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for Connecting Letters Report ----
## Note: You can not use the emmeans object here
multcompView::multcompLetters4(
  object = cheeseModel, # Your aov/lm object
  comp = hsdPH, # Your stored comparisons
  threshold = 0.1 # Your Overall Type I Error Rate
)

# Demo Code for a Box Plot with Connecting Letters ----
## Note 1: you can not easily set your threshold; uses 5%
## Note 2: you can not use anything other than Tukey
multcompView::multcompBoxplot(
  formula = acids ~ strain,
  data = cheeseData,
  compFn = "TukeyHSD",
  plotList = list(
    boxplot = list(fig = c(0, 0.85, 0, 1)),
    multcompLetters = list(
      fig = c(0.8, 0.9, 0.1, 0.9),
      fontsize = 12,
      fontface = "bold"
    )
  )
)

# Your Turn Code------------------------------------
multcompView::multcompBoxplot(
  formula = score ~ year,
  data = songData,
  compFn = "TukeyHSD",
  plotList = list(
    boxplot = list(fig = c(0, 0.85, 0, 1)),
    multcompLetters = list(
      fig = c(0.8, 0.9, 0.1, 0.9),
      fontsize = 12,
      fontface = "bold"
    )
  )
)
```