Post Hoc Analysis for ANOVA

Neil J. Hatfield

3/13/2021

In this tutorial, we are going to explore doing Post Hoc analysis in the context of ANOVA models. So that we have some data to work with, we will make use of Table 5.2/Example 5.5 from page 91 of Oehlert, **Free Amino Acids in Cheese**. Take a moment to create this data frame for yourself. (Check your code against what appears in the Code Appendix.)

I will leave both the checking of base requirements as well as assumptions for both the parametric and nonparametric shortcuts to you. However, Table 1 provides the modern ANOVA table for our cheese data. (Reminder: don't forget to tell R to use the Sum to Zero constraint: `options(contrasts = c("contr.sum", "contr.poly"))`.)

Table 1: ANOVA Table for Free Amino Acids in Cheese Study

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

Choose Your Type I Error Rate

Conceptualize your testing family and then ask yourself "Which Type I Error Rate do I want to control?"

Selected Type I Error Rate	Possible Methods
Simultaneous Confidence Intervals	Bonferroni, Tukey HSD, Tukey-Kramer HSD, Scheffé
Strong Familywise/Maximum Experimentwise	Hochberg, Holm, REGWR, Šidák, 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

You have have chosen which Type I Error Rate you're going to control, pick your over all level $\mathcal{E}_I \in (0, 0.15]$ (or use the level assigned by the client). Then you may apply the chosen method. For these examples, I'll use $\mathcal{E}_I = 0.1$; this will corresponded to a confidence level of 0.9.

I've organized the rest of this tutorial into two major sections: Parametric Shortcuts and Nonparametric Shortcuts. You will want to be consistent in your work. That is to say, if you've used the parametric shortcut for the omnibus ANOVA test, then you should stick with parametric shortcuts for the Post Hoc. I then discuss how you can get Effect Sizes for both sets of cases using some helper functions I've created for us to use. I'll end the tutorial with a few ways that you can visualize the pairwise differences.

Parametric Shortcuts

In many cases, you will be able to use essentially the same code and apply multiple methods; typically, you just have to change one argument to switch which method you want to use. Thus, I've divide this Parametric Shortcuts section up by the main function call you'll be making.

Tukey's [& Kramer's] HSD (Recommended Default)

When you just don't know what to do, controlling the SCI/MEER and using Tukey's or Tukey-Karmer's HSD is not a bad default position. Keep in mind that if you aren't building confidence intervals, then you are controlling the MEER. The distinction between Tukey's HSD and Tukey-Karmer's HSD is that the former is for exact balanced designs; the later is for imbalanced designs. Both are accessed through the same function, `TukeyHSD` and R will automatically use the correct one.

```
hsdPH <- TukeyHSD(
  x = cheeseModel, # Your aov/lm object
  conf.level = 0.9 # 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 3: Post Hoc Tukey HSD Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
A & B-A	1.892	0.606	3.177	0.030
B-A	0.875	-0.411	2.161	0.264
None-A	-0.245	-1.531	1.041	0.921
B-A & B	-1.017	-2.302	0.269	0.187
None-A & B	-2.137	-3.422	-0.851	0.019
None-B	-1.120	-2.406	0.166	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.

Pairwise Method

A second option that is built into the base build of R is that of `pairwise.t.test`. This is an extension of the `t.test` function that you would use for a standard two-sample test for testing location parameters. However, `pairwise.t.test` is built for handling the Multiple Comparison/Simultaneous Inference Problem.

```
pairwisePH <- pairwise.t.test(
  x = cheese$acids, # call the response vector
  g = cheese$strain, # call your treatment vector
  p.adjust.method = "bonferroni" # Which method you want
)

## Kable Code for Pairwise.t.Test
knitr::kable(
  x = pairwisePH$p.value,
  digits = 3,
  caption = paste( # This creates a nice looking table caption
    "Post Hoc", # feel free to copy
    gsub(
      pattern = "(^|[[:space:]])([[:alpha:]])",
      replacement = "\\1\\U\\2",
      x = pairwisePH$p.adjust.method,
      perl = TRUE
    ),
    "Comparisons"
  ),
  align = rep('c', nrow(pairwisePH$p.value))
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  ) %>%
  kableExtra::footnote(
    general = "Rows and Columns are Treatment Levels.",
    footnote_as_chunk = TRUE
  )
```

Table 4: Post Hoc Bonferroni Comparisons

	A	A & B	B
A & B	0.053		
B	0.552	0.374	
None	1.000	0.034	0.286

Note: Rows and Columns are Treatment Levels.

The reported p-values are adjusted so that you may compare them against your Usualness Threshold. You can change which method gets used by changing the value of the `p.adjust.method` argument:

Chosen Method	Set `p.adjust.method =`
Bonferroni	`"bonferroni"`
Holm	`"holm"`
Hochberg	`"hochberg"`

Chosen Method	Set `p.adjust.method =`
Benjamini & Hochberg	`"BH"` OR `"fdr"`

`DescTools` Method

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

```
## DescTools Pairwise Method
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 6: Post Hoc Newman-Keuls Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
A & B-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-A & B	-1.017	-1.862	-0.171	0.062
None-A & B	-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"`

I often prefer using `DescTools` over `pairwise.t.test` as you do get a format for Post Hoc results that is often easier for people to look at and understand.

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`).

```
# Special Comparisons-Dunnett's Test
dunnett <- DescTools::DunnettTest(
  formula = acids ~ strain,
  data = cheese,
  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 8: Post Hoc Comparisons–Dunnett's Test

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

Nonparametric Shortcuts

While there are several routes you can take for Nonparametric Post Hoc analysis, there are not as many as for parametric shortcuts. Further, I'm only going to detail two approaches: Dunn's Test and the DSCF Test.

Dunn's Test

Dunn's Test (not to be confused with Dunnett's Test) comes from the `dunn.test` package and provides multiple nonparametric versions of the methods you've already seen.

```
# dunn.test is a bit "noisy" in that it will print
# extraneous output to your console and to your report.
# Use quietly from purrr (part of tidyverse) to stop this

dunn <- purrr::quietly(dunn.test::dunn.test)(
  x = cheese$acids, # response vector
  g = cheese$strain, # factor vector
  method = "bonferroni", # Your chosen method
  alpha = 0.1, # Your Overall Type I Error Rate
  kw = FALSE, # Turns Off Kruskal Wallis Output
  table = FALSE, # Turns off a default output table
  list = FALSE # Used with step up/down methods
)$result # Don't forget this call to get the result of dunn.test

## Kable Code for Dunn's Test
knitr::kable(
  x = data.frame(
    comparison = dunn$comparisons,
    pvalues = dunn$P.adjusted
  ),
  digits = 4,
  caption = "Post Hoc Dunn's Test--[insert method here] Adjustment", # Fill this in
  col.names = c("Comparison", "Adj. p-Value"),
  align = 'lc'
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 9: Post Hoc Dunn's Test–[insert method here] Adjustment

Comparison	Adj. p-Value
A - A & B	0.1237
A - B	0.6620
A & B - B	1.0000
A - None	1.0000
A & B - None	0.1237
B - None	0.6620

You can then compare these p-values to your Unusalness Threshold.

You can change which method you use by changing the value of the `method` argument:

Chosen Method	Set `method =`
Bonferroni	`"bonferroni"`
Šidák	`"sidak"`

Chosen Method	Set `method =`
Holm	`"holm"`
Holm-Šidák	`"hs"`
Hochberg	`"hochberg"`
Benjamini-Hochberg	`"bh"`

For the last four, set `list = TRUE` to have the results be put into the proper ordering and marked for rejection of the null hypothesis.

DSCF Test

The nonparametric counterpart of Tukey's/Tukey-Kramer's HSD is the Dwass-Steel-Critchlow-Fligner All Pairs Test; DSCF Test for short. You will need to use the `PMCMRplus` package to access the `dscfAllPairsTest` function

```
# Post Hoc Method Two--DSCF Test
dscf <- PMCMRplus::dscfAllPairsTest(
  formula = acids ~ strain,
  data = cheese,
  na.action = "na.omit"
)

# Kable Code for DSCF
knitr::kable(
  x = dscf$p.value,
  digits = 3,
  caption = paste("Post Hoc-Dwass-Steel-Critchlow-Fligner Tests"),
  align = rep('c', nrow(dscf$p.value))
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 11: Post Hoc-Dwass-Steel-Critchlow-Fligner Tests

	A	A & B	B
A & B	0.408		
B	0.408	0.408	
None	1.000	0.408	0.408

As you can tell, the output of this method is formatted like that of the `pairwise.t.test`.

Effect Sizes

For the Omnibus tests we made use of effect sizes to provide a sense of how much of the variation in our response the alternative model explained/accounted for. We did this through the use of ω^2, η^2, and ϵ^2. When doing Post Hoc analysis, we will also want to get measures of practical significance.

For post hoc tests, particularly pairwise comparisons, we will use two sets of effect sizes:

- Standardized Distances
  - A measure of the standardized distance between location parameters
  - Statistics: Cohen's d, Hedge's g, and Hodges-Lehmann $\widehat{\Delta}$
- Common Language Effect Size
  - An intutitive comparison based on using lotteries to compare two randomly selected individuals
  - Statistic: the Probability of Superiority

I've created some helper functions that will calculate the appropriate effect sizes and append them to a table for display in a report.

To access these functions, you'll need to include the following code in your R file:

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

My recommendation is to stick this right next to where you set the sum to zero constraint. Both of them should be towards the top of your document, near where you have placed all of your library calls.

Parametric Example

When you have used the Parametric Shortcuts, you'll want to use the `anova.PostHoc` function and pass your model object (the result of calling `aov` or `lm`) as the input.

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

```
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 12: Post Hoc Comparison Effect Sizes

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

Nonparametric Example

When you have used Nonparametric Shortcuts (either through Dunn's Test or the DSCF Test), you'll want to use the `kw.PostHoc` function. You'll need to provide two inputs: the response vector and the treatment vector.

```
kw.PostHoc(
    response = cheese$acids,
    treatments = cheese$strain
  ) %>%
knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparison Effect Sizes",
  col.names = c("Pairwise Comparison","Hodges Lehmann Estimate",
                "Prob. Superiority"),
  align = 'lcc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 13: Post Hoc Comparison Effect Sizes

Pairwise Comparison	Hodges Lehmann Estimate	Prob. Superiority
A - A & B	-1.892	0.000
A - B	-0.875	0.137
A & B - B	1.017	0.736
A - None	0.245	0.500
A & B - None	2.137	1.000
B - None	1.120	0.863

Visualizing Pairwise Comparisons

When using the parametric shortcuts, you can visualize your pairwise comparisons in a couple of different ways by using the `multcompView` package.

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.

```
# Connecting Letters Report
multcompView::multcompLetters4(
  object = cheeseModel, # Your aov/lm object
  comp = hsdPH, # Your stored comparisons
  threshold = 0.1 # Your Overall Type I Error Rate
)
```

```
## $strain
```

```
## A & B     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.

Box Plot + Connecting Letters

```
# Box plot with connecting letters
## NOTE: Does not allow you to set a threshold
multcompView::multcompBoxplot(
  formula = acids ~ strain,
  data = cheese,
  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"
    )
  )
)
```

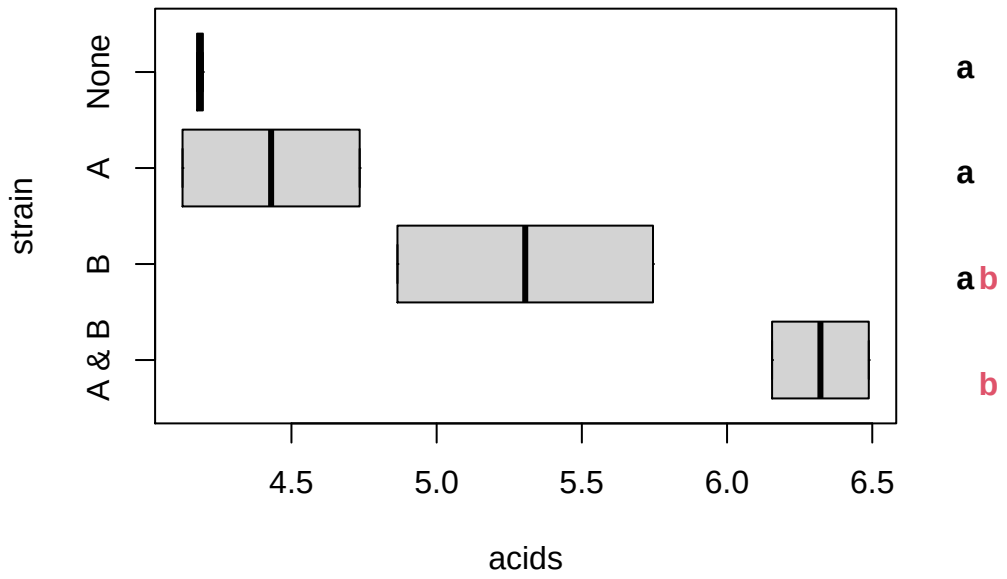

Figure 1: Box Plot and Connecting Letters Report for Free Amino Acids in Cheese Study

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)

Code Appendix

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

packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "DescTools", "dunn.test",
              "PMCMRplus", "multcompView")
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
cheese <- data.frame(
  strain = as.factor(
    c("None", "None", "A", "A",
      "B", "B", "A & B", "A & B")
  ),
  acids = c(
    4.195, 4.175, 4.125, 4.735,
    4.865, 5.745, 6.155, 6.488
  )
)

cheeseModel <- aov(
  formula = acids ~ strain,
  data = cheese,
  na.action = "na.omit"
)

parameters::model_parameters(
  model = cheeseModel,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Omega Sq.", "Eta Sq.", "Epslion 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("scale_down", "HOLD_position")
  )

hsdPH <- TukeyHSD(
  x = cheeseModel, # Your aov/lm object
  conf.level = 0.9 # 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"
  )

pairwisePH <- pairwise.t.test(
  x = cheese$acids, # call the response vector
  g = cheese$strain, # call your treatment vector
  p.adjust.method = "bonferroni" # Which method you want
)

## Kable Code for Pairwise.t.Test
knitr::kable(
  x = pairwisePH$p.value,
  digits = 3,
  caption = paste( # This creates a nice looking table caption
    "Post Hoc", # feel free to copy
    gsub(
      pattern = "(^|[[:space:]])([[:alpha:]])",
      replacement = "\\1\\U\\2",
      x = pairwisePH$p.adjust.method,
      perl = TRUE
    ),
    "Comparisons"
  ),
  align = rep('c', nrow(pairwisePH$p.value))
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  ) %>%
  kableExtra::footnote(
    general = "Rows and Columns are Treatment Levels.",
```

```
    footnote_as_chunk = TRUE
  )

## DescTools Pairwise Method
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"
  )

# Special Comparisons-Dunnett's Test
dunnett <- DescTools::DunnettTest(
  formula = acids ~ strain,
  data = cheese,
  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"
  )
```

```r
# dunn.test is a bit "noisy" in that it will print
# extraneous output to your console and to your report.
# Use quietly from purrr (part of tidyverse) to stop this

dunn <- purrr::quietly(dunn.test::dunn.test)(
  x = cheese$acids, # response vector
  g = cheese$strain, # factor vector
  method = "bonferroni", # Your chosen method
  alpha = 0.1, # Your Overall Type I Error Rate
  kw = FALSE, # Turns Off Kruskal Wallis Output
  table = FALSE, # Turns off a default output table
  list = FALSE # Used with step up/down methods
)$result # Don't forget this call to get the result of dunn.test

## Kable Code for Dunn's Test
knitr::kable(
  x = data.frame(
    comparison = dunn$comparisons,
    pvalues = dunn$P.adjusted
  ),
  digits = 4,
  caption = "Post Hoc Dunn's Test--[insert method here] Adjustment", # Fill this in
  col.names = c("Comparison", "Adj. p-Value"),
  align = 'lc'
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
# Post Hoc Method Two--DSCF Test
dscf <- PMCMRplus::dscfAllPairsTest(
  formula = acids ~ strain,
  data = cheese,
  na.action = "na.omit"
)

# Kable Code for DSCF
knitr::kable(
  x = dscf$p.value,
  digits = 3,
  caption = paste("Post Hoc-Dwass-Steel-Critchlow-Fligner Tests"),
  align = rep('c', nrow(dscf$p.value))
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

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

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"
  )

kw.PostHoc(
    response = cheese$acids,
    treatments = cheese$strain
  ) %>%
knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparison Effect Sizes",
  col.names = c("Pairwise Comparison","Hodges Lehmann Estimate",
                "Prob. Superiority"),
  align = 'lcc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Connecting Letters Report
multcompView::multcompLetters4(
  object = cheeseModel, # Your aov/lm object
  comp = hsdPH, # Your stored comparisons
  threshold = 0.1 # Your Overall Type I Error Rate
)

# Box plot with connecting letters
## NOTE: Does not allow you to set a threshold
multcompView::multcompBoxplot(
  formula = acids ~ strain,
  data = cheese,
  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"
    )
  )
```

```
)
```