Nonparametric Shortcuts in ANOVA

Neil J. Hatfield

Last Updated: April 24, 2023

In this tutorial we will take a look at nonparametric shortcuts for ANOVA models, specifically the Kruskal-Wallis H Test and the Friedman Twoway ANOVA Test.

1 Important Notes

There are several important notes to keep in mind when you consider a nonparametric shortcut.

1) **Nonparametric doesn't mean assumption-free.** These shortcuts still require assumptions but these assumptions tend to be less demanding than those of parametric shortcuts. For example, nonparametric shortcuts almost never assume a *named* distribution such as the Gaussian, opting for a more general class of distributions such as "continuous".
2) If the assumptions of a parametric shortcut are satisfied, you should use that method. Parametric shortcuts are more powerful (i.e., better ability to detect departures from the null model) than nonparametric when their assumptions are met.
3) Be consistent with your usage: if you use a nonparametric shortcut for your omnibus tests, then use nonparametric shortcuts for your post hoc analysis. Don't switch back-and-forth between approaches.
4) The types of ANOVA models that you'll be able to explore using these shortcuts are limited to
    - One-way layouts (single factor)
    - Some Two-way layouts
        - One-way + Block
        - Within Subjects Repeated Measures One-way ANOVA (i.e., One-way + Block)
        - Two factors with NO interaction

1.1 Data Ranks

Many nonparametric shortcuts are *rank-based* methods and the Kruskal-Wallis H test and Friedman Two-way ANOVA test are no different. The underlying approach here is to use a case's (measurement unit's) *rank* instead of their *magnitude*. As a quick example, suppose that we have three people's whose heights are 71", 62", 84". If we used their magnitudes, then we would directly use the 71, 62, and 84, respectively. If we used their ranks instead, we'd use 2, 1, 3, respectively. This shift to ranks also comes with a slight modification to the underlying ANOVA model: instead of focusing on the performance metrics (i.e., the *Arithmetic Means*), we'll look at measures of middle (i.e., *Medians*).

2 Setting up `R`

As usual, there are a few housekeeping items to do to ensure that our current `R` session is ready for our work. The following code is how we can ensure that we have `R` ready for our use.

```
# Demo code to set up R ----
## Load packages
packages <- c("tidyverse", "hasseDiagram", "knitr", "kableExtra",
              "car", "psych", "coin", "rcompanion", "dunn.test")
lapply(packages, library, character.only = TRUE, quietly = TRUE)

## Set options
```

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

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

3 Data Contexts

3.1 Honey Study

Our first context will be the **Honey Study**. We want to see how three different varietals (types of plant) impact the amount of honey bees produce (in pounds) when they draw from a single varietal. In our study, we have nine colonies that we've randomly selected from a large population of honey bee (*Apis mellifera*) colonies via lottery. We then used a separate lottery to randomly assign three colonies to each of three varietals: clover, orange blossom, and alfalfa.

```r
# Demo code for loading Honey data ----
honeyData <- data.frame(
  Amount = c(150, 50, 100, 85, 90, 95, 130, 50, 80),
  Varietal = rep(c("Clover", "Orange Blossom", "Alfalfa"), each = 3)
)
## Set Varietal to factor
honeyData$Varietal <- as.factor(honeyData$Varietal)
```

The order in which the data appear in the code is measurement order.

3.2 Alfalfa Pest Study

Our second context will be an exploration on different methods to control the alfalfa weevil (*Hypera postica Gyllenhal*) for different varieties of alfalfa. In particular, we have five different types of alfalfa covering experimental (i.e., new), recently introduced, and established species. For the weevil management methods, we have four methods.

- Conventional–using petro-chemicals as prescribed for agriculture usages.
- Integrated Pest Management (IPM)–using a variety of methods with minimal usage of petro-chemicals
- Organic–practices avoiding any use of petro-chemicals.
- None–alfalfa is only cut at regular intervals

The response in this study will be the number of whole alfalfa weevil larvae found in the plot during the 2-hour collection window. To learn more about this study, check out the chapter by MacFarland & Yates.

```r
# Demo Code for loading Alfalfa study data ----

## Be careful when breaking URLs into chunks; break on forward slashes
rootPath <- "https://static-content.springer.com/esm/chp%3A10.1007%2F978-3-319-30634-6_7/"
filePath <- "MediaObjects/385146_1_En_7_MOESM1_ESM.csv"

alfalfaData <- read.table(
  file = paste0(rootPath, filePath),
  header = TRUE,
  sep = ","
)

alfalfaData$Treatment <- as.factor(alfalfaData$Treatment)
alfalfaData$Variety <- as.factor(alfalfaData$Variety)
```

4 Explore Your Data

Just as with the Parametric Shortcut, you should always begin by exploring your data. Check out the other guides/tutorials I've posted on data visualizations and descriptive statistics. You should create a data narrative that weaves both of these types of elements together and helps your readers build their understanding of your data.

5 Checking Appropriateness

One important think to keep in mind is that checking whether ANOVA methods are appropriate is different from assessing the assumptions of a particular test. We must meet these conditions regardless of whether we're using a parameteric or nonparametric shortcut. Thus, we still need to ensure that

1) we are working with a qualitative/categorical factor,
2) we are working with a quantitative response,
3) we are working with an additive model,
4) we have estimable effects, and
5) we have estimable errors/residuals.

We can check these in the same manner as we have all semester.

6 Kruskal-Wallis *H* Test (One-way ANOVA)

The Kruskal-Wallis H Test is a nonparametric shortcut for dealing with One-way ANOVA contexts. You can arrive at a "one-way layout" in two ways: either you've designed a study to only have one factor OR you've collapsed separate factors into a single "factor". (You should try to limit using the second approach to those situations where you have no interest in the main effects.)

The underlying model for the Kruskal-Wallis test is

$$Y_{ij} = \theta_{\bullet\bullet} + \tau_i + \epsilon_{ij}$$

where $\theta_{\bullet\bullet}$ represents the ***Grand Median***, τ_i represents the effect of factor level i relative to the *Median*, and ϵ_{ij} represents the residuals for this model.

6.0.1 Example-Honey Study

In investigating the effect of the type of varietal (species of flower) has on the production of excess honey, we constructed the Hasse diagram in Figure 1. With our nine hives of the same species of bee, we can see that we have sufficient degrees of freedom to estimate the effects for our three levels of varietal and have degrees of freedom for our error term. Given that we're measuring our response (excess honey) in pounds, along with the additive model shown in Figure 1, a one-way ANOVA model is a valid approach.

```
# Demo code for Hasse Diagram for the Honey Study ----
modelLabels <- c("1 Make Honey 1", "3 Varietal 2", "9 (Hives) 6")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE),
  nrow = 3,
  ncol = 3,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)
```

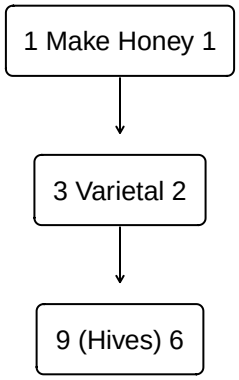


Figure 1: Hasse Diagram for Honey Study

6.1 Assessing Assumptions

The Kruskal-Wallis H test makes two assumptions:

1) Independence of Observations
2) The response follows some *continuous* distribution that differs between groups by location (*Medians*) at most.

If these assumptions are satisfied, the our test statistic, H, follows a χ^2 with $k-1$ *degrees of freedom* (where we have k levels to our factor).

6.1.1 Independence of Observations

We can assess the Independence of Observations by looking at the residuals in an index plot (as we did for the parametric shortcut). However, for One-way ANOVA types of problems, we can also look at index plots using the response values instead of the residuals. (Note: this is only true for when you have one factor and nothing else in the design.)

```
# Demo code of an index plot using the response for Honey Study ----
ggplot(
  data = honeyData,
  mapping = aes(
    x = 1:nrow(honeyData),
    y = Amount
  )
) +
  geom_point() +
  geom_line() +
  geom_hline(
    yintercept = mean(honeyData$Amount, na.rm = TRUE),
    color = "red",
    linetype = "dashed"
  ) +
  theme_bw() +
  xlab("Measurement Order") +
  ylab("Amount of Honey (lbs)") +
  scale_x_continuous(breaks = 1:9)
```

6.1.1.1 Honey Study Example

Figure 2 shows the index plot using the response of the Honey study (i.e., amount of honey produced in pounds). Just as before, we are still looking to see if there are any patterns to this plot. The presence of patterns indicates a threat to the assumption of Independent Observations. Even though there are only nine measurement units in the Honey study, I don't see any patterns in Figure 2. Additionally, I can't think of any threats to independence from the study design (each hive was placed sufficiently far apart to minimize competition and cross-contamination).

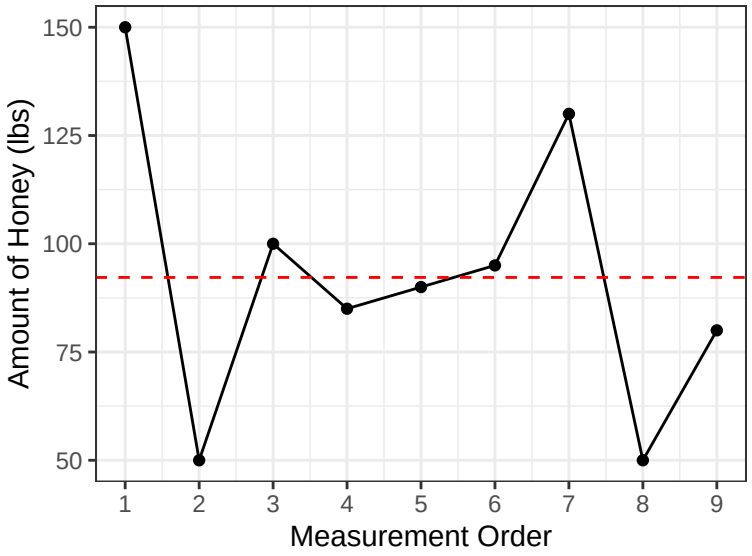


Figure 2: Index Plot for Honey Study

6.1.2 Continuous Distribution

The second assumption (response follows some continuous distribution) is a bit complicated. What we mean here is that up to differences in the location parameter, the response follows the same kind of continuous (or continuous adjacent) distribution family. For example, the responses in each group all come from log-normal distributions that have different values for μ (the location parameter, not the Expected Value). Or they all come from χ^2 distributions except each group has a different value for ν (degrees of freedom). Whatever named distribution best describes our data can only have differences in the value of the location parameter; all scale parameters should be the same.

We are bit flexible with this assumption, as we will allow for "continuous adjacent" data. Likert scale data (e.g., 1-strongly disagree to 5-strongly agree; the Hedonic tasting scale) aren't actually continuous but are ordinal, which allows for us to meaningfully convert these scale scores into ranks.

While we could use QQ Plots with a bunch of different distributions (any pre-programmed distribution in `R` may be used instead of `"norm"` for the `distribution` argument), we will instead just ask ourselves "Is the response continuous or at least ordinal?" and "Do I have reason to suspect that one of the groups/treatments creates extremely different behavior in the response attribute?" These two questions will guide us in assessing whether or not this assumption is satisfied.

6.1.2.1 Honey Study Example Our response in the Honey Study is the excess honey (lbs) that each hive produces during the same time span. Weight is a continuous attribute. Further, we have no reason to believe that the type of Varietal (kind of flower) will alter the process which underpins honey production beyond the total amount of honey produced. Thus, we will act as though the continuous assumption is satisfied.

6.1.3 What About Homoscedasticity?

If you attempt to line up the assumptions for the parametric shortcut with those of the Kruskal-Wallis, you might find yourself asking about homoscedasticity. Strictly speaking, we still have this assumption in that the only difference in the continuous distribution of our response is location parameter–that is, they all use the same scale. If we also want to free the scale, we have what is referred to as the *k-Sample Behrens-Fisher Problem*; Rust and Fligner proposed a modification to Kruskal-Wallis to account for this (beyond this guide).

6.2 Fitting the Model

In this guide, we'll only look at using base `R` for the Kruskal-Wallis test. There are other packages that will give you additional options, but the base `R` approach works well.

```r
# Demo code for running the Kruskal-Wallis test in base R ----
honeyOmni <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honeyData,
  na.action = "na.omit"
)
```

Notice that all we've done is swap `aov` for `kruskal.test`. We are still using a `formula` argument in the format `response ~ factor`, the `data` argument, and the optional `na.action` safety argument.

6.3 Results

While fitting the model in `R` was a quick function name change, reporting the results is different. This is due to the fact that there is ANOVA table for the Kruskal-Wallis setting. Instead, we often list out the values in our narrative paragraphs. Just like the parametric shortcut, you must still have set up your decision rule, especially your Type I Risk, $\mathcal{E}_I$, and your Unusualness Threshold (UT). For the Honey Study, I'm going to use $\mathcal{E}_I = 0.05$ and $UT = 0.03$.

If you want to create a table to display your results, you can; just don't call the table an "ANOVA Table".

6.3.1 Omnibus

To quickly view our results from the Kruskal-Wallis test, we just need to call the result object.

```r
## Demo Code for showing Kruskal-Wallis results ----
honeyOmni
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Amount by Varietal
## Kruskal-Wallis chi-squared = 0.56022, df = 2, p-value = 0.7557
```

Keep in mind that we should not be displaying raw output in our reports. Since we've stored these results, we can weave them into our narrative text. We just need to know how to call the right elements.

Table 1: Commands to Extract KW Results

Value	Object Name	Code Example	Final Result
H	`honeyOmni$statistic`	`round(honeyOmni$statistic, digits = 2)`	0.56
DF	`honeyOmni$parameter`	`honeyOmni$parameter`	2
p-value	`honeyOmni$p.value`	`round(honeyOmni$p.value, digits = 4)`	0.7557

To get a measure of effect size, we will use *Epsilon-Squared*. However, we must use the function from the `rcompanion` package to account for using the Kruskal-Wallis test appropriately.

```r
# Demo Code for measuring practical significance for Kruskal-Wallis ----
## Honey Study
honeyEffectSize <- rcompanion::epsilonSquared(
  x = honeyData$Amount,
  g = honeyData$Varietal,
  digits = 4
)

## Display results
honeyEffectSize
```

```
## epsilon.squared
##         0.07003
```

Notice that instead of using a formula, we use `x` to denote the response and `g` to pass along the grouping (factor) information. Additionally, you'll notice that even though `digits = 4`, there are five numbers after the decimal point. Here the `digits` argument refers to the number of *significant* digits to display.

We still interpret ϵ^2 as the proportion of variation explained by our model, just like we interpret the effect sizes in the parametric shortcut.

6.3.1.1 Honey Study Example

> Given that a one-way ANOVA model is appropriate to investigate whether the varietal impacts the amount of excess honey produced and we decided that we did not meet the assumptions for the parametric ANOVA F test, we turned towards the nonparametric Kruskal-Wallis H test. After checking that the data satisfy the assumptions, we found that $H = 0.56$ with 2 degrees of freedom. This results in a p-value of 0.7557. Since this is larger than our stated Unusualness Threshold ($UT = 0.03$), we will fail to reject the null and decide to act as if varietal does not impact the amount of excess honey the bees produced.

6.3.1.2 Generating Code for write up

> Given that a one-way ANOVA model is appropriate to investigate whether the varietal impacts the amount of excess honey produced and we decided that we did not meet the assumptions for the parametric ANOVA F test, we turned towards the nonparametric Kruskal-Wallis H test. After checking that the data satisfy the assumptions, we found that \(H=‘ r round(honeyOmni$statistic, digits = 2)‘\) with ‘ r honeyOmni$parameter‘ degrees of freedom. This results in a p-value of ‘ r round(honeyOmni$p.value, digits = 4) ‘. Since this is larger than our stated Unusualness Threshold (\(UT = 0.03\)), we will fail to reject the null and decide to act as if varietal does not impact the amount of excess honey the bees produced.

Note 1: I didn't make use of the effect size since I'm failing to reject the null hypothesis. Generally, we only include effect sizes when we have a statistical discovery (i.e., rejecting the null).

Note 2: the ‘ is the "back tick" from the key just to the left of the 1-key on a standard (American) keyboard. This symbol immediately followed by `r` will start an in-line code chunk.

6.3.2 Post Hoc Analysis

There are two sets approaches that you can use for pairwise comparisons in the nonparametric setting. (We won't worry about contrasts in this guide.) For both of the approaches, I'm going to use the **Alfalfa Pest Study** data, ignoring the block. We will aim to control our SCI at 0.1 and use a $UT = 0.1$.

```
# Demo Code for fitting Oneway Alfalfa Study ----
## KW Results
alfalfaModel0 <- kruskal.test(
  formula = Larvae ~ Treatment,
  data = alfalfaData,
  na.action = "na.omit"
)

alfalfaModel0
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  Larvae by Treatment
## Kruskal-Wallis chi-squared = 9.8731, df = 3, p-value = 0.01968
```

6.3.2.1 Dunn's Test

Dunn's Test (which is different from *Dunnet's Test*) is part of the `dunn.test` package and will give you flexibility for using a number of different methods to control our Type I Error Rate.

```
# Demo Code for using Dunn's Test ----
## 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 = alfalfaData$Larvae, # response vector
  g = alfalfaData$Treatment, # 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',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 2: Post Hoc Dunn's Test–[insert method here] Adjustment

Comparison	Adj. p-Value
Conventional - IPM	0.6267
Conventional - None	0.0069
IPM - None	0.2197
Conventional - Organic	0.1348
IPM - Organic	1.0000
None - Organic	0.8912

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"`
Holm	`"holm"`
Holm-Šidák	`"hs"`
Hochberg	`"hochberg"`

Chosen Method	Set `method =`
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.

6.3.2.2 DSCF Test The Dwass-Steel-Critchlow-Fligner (DSCF) Test is the nonparametric equivalent of the Tukey/Tukey-Kramer HSD test. To use this approach, you'll need to have loaded my ANOVATools.

```
# Demo Code for the DSCF Test ----
dscf <- dscfTest(
  response = alfalfaData$Larvae,
  factor = alfalfaData$Treatment
)
# Kable Code for DSCF
knitr::kable(
  x = dscf,
  digits = 3,
  col.names = c("Comparison", "Observed W", "Adj. p-value"),
  caption = paste("Post Hoc-Dwass-Steel-Critchlow-Fligner Tests"),
  align = 'lcc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 4: Post Hoc-Dwass-Steel-Critchlow-Fligner Tests

Comparison	Observed W	Adj. p-value
Conventional vs. IPM	2.511	0.285
Conventional vs. None	3.397	0.077
Conventional vs. Organic	2.806	0.194
IPM vs. None	2.806	0.194
IPM vs. Organic	1.482	0.721
None vs. Organic	-1.920	0.526

You would then compare these adjusted p-values to your Unusualness Threshold.

6.3.3 Post Hoc Effect Sizes

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

```
kw.PostHoc(
    response = alfalfaData$Larvae,
    treatments = alfalfaData$Treatment
  ) %>%
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 5: Post Hoc Comparison Effect Sizes

Pairwise Comparison	Hodges Lehmann Estimate	Prob. Superiority
Conventional - IPM	-29	0.270
Conventional - None	-54	0.000
IPM - None	-31	0.165
Conventional - Organic	-37	0.123
IPM - Organic	-10	0.365
None - Organic	19	0.689

The Probability of Superiority has the same kind of interpretation as in the parametric shortcut: approximately 27% of the time we randomly select a plot getting conventional weevil treatment and one getting the IPM treatment, the conventional plot will have the higher number of whole larvae.

The Hodges-Lehmann Estimate, $\widehat{\Delta}$, measures the middle difference of all possible pairings between two groups. If we imagine taking all of the plots which got either conventional pest control or the IPM treatment, we can imagine all of the pairings of plot from each group (i.e., the 25 pairings). For each pairing, we can find the difference in number of whole weevil larvae found. For these two groups/treatments, half of all the pairings show that the conventional group has at least 29 fewer whole larvae than the IPM group.

7 Friedman Two-way ANOVA Test

The Friedman Two-way ANOVA Test is a nonparametric shortcut for dealing with a subset of two-way ANOVA designs. Specifically, designs which we can map to a Randomized Complete Block Design (One-way + Block). This includes the Within Subjects Repeated Measures design.

The underlying model for the Friedman test is

$$Y_{ij} = \theta_{\bullet\bullet\bullet} + \beta_i + \tau_j + \epsilon_{ijk}$$

where $\theta_{\bullet\bullet\bullet}$ represents the ***Grand Median***, β_i represents the effect due to block i, τ_j represents the effect of factor level j relative to the *Median*, and ϵ_{ijk} represents the residuals for this model.

7.1 Assessing Assumptions

The Friedman test makes three assumptions:

1) Independence of Observations
2) The response follows some *continuous* distribution that differs between groups by location (*Medians*) at most.
3) We do not have any significant (and meaningful) interaction between our block and our factor.

If these assumptions are satisfied, the our test statistic, S, follows a χ^2 with $k-1$ *degrees of freedom* (where we have k levels to our factor).

7.1.1 Independence of Observations

Use the approaches we've made use of all semester.

7.1.1.0.1 Your Turn

How would you assess the Independence of Observations assumption for the Alfalfa Pest Study?

7.1.2 Continuous Distribution

Use the same approach as for the Kruskal-Wallis: Is the response continuous or at least ordinal? Do I have reason to suspect that one of the groups/treatments creates extremely different behavior in the response attribute?

7.1.2.1 Your Turn

How would you assess the Continuous Distribution assumption for the Alfalfa Pest Study?

7.1.3 No Interaction Between Block and Factor

Just as we've done in the past, we will want to look at an interaction plot. A major difference here is to not use the *SAM* but the *Sample Median*.

```
# Demo Code for Interaction Plot in Alfalfa Pest Study ----
ggplot2::ggplot(
  data = alfalfaData,
  mapping = aes(
    x = Variety,
    y = Larvae,
    color = Treatment,
    shape = Treatment,
    linetype = Treatment,
    group = Treatment
  )
) +
  stat_summary(fun = "median", geom = "point") + # Notice the change in function
  stat_summary(fun = "median", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Variety") +
  ylab("Number of Whole Larvae Found") +
  labs(color = "Treatment") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)
```

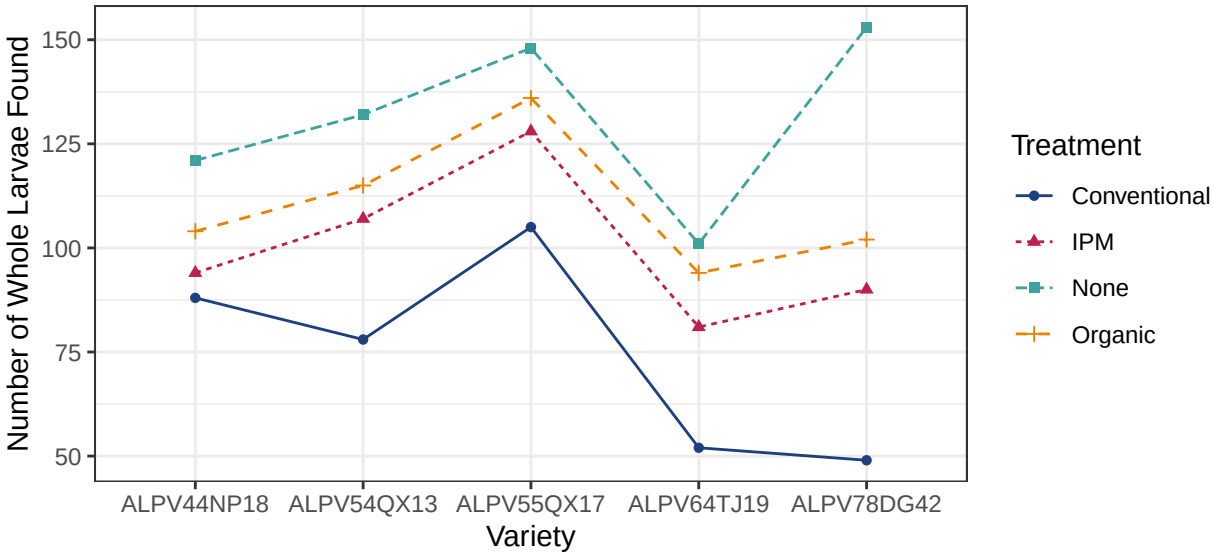


Figure 3: Interation Plot for Alfalfa Pest Study

In looking at Figure 3 we are still looking for parallelism between corresponding line segments. Keep in mind that we do not need perfection. The only somewhat worrying segment is for the Conventional treatment between the ALPV44NP18 and ALPV54QX13 varieties.

7.2 Fitting the Model

7.3 Results

7.3.1 Omnibus

7.3.2 Post Hoc Analysis

8 OLD

8.1 Results

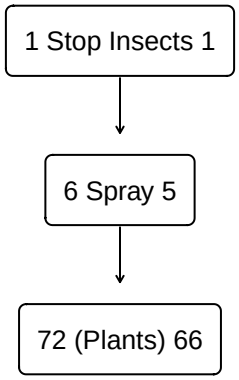
8.2 Generating Code

9 Putting Everything Together–Your Turn

Now is an opportunity for you to get some practice. I'm going to use the data frame `InsectSprays`, which is built into R. You may load this data into your session with the command `data("InsectSprays")`. (This is same data set as in the Parametric Shortcut guide/tutorial.)

Reminder of the study context: the original researchers were exploring the effectiveness of various sprays on reducing the number of instances of particular type of insect for a crop. Each observation is randomly sampled plant from a field. NOTE: we do not know the measurement order.

Explore how you would use the Kruskal-Wallis H test with the Insect Sprays data. If you get stuck, check out the code appendix to see the code that I used.




```
##
##  Asymptotic Kruskal-Wallis Test
##
## data:  count by spray (A, B, C, D, E, F)
## chi-squared = 54.691, df = 5, p-value = 1.511e-10

## epsilon.squared
##          0.7703
```

10 Code Appendix

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

packages <- c("tidyverse", "hasseDiagram", "knitr", "kableExtra",
              "car", "psych", "rcompanion", "dunn.test")
lapply(packages, library, character.only = TRUE)

# Tell Knitr to use empty space instead of NA in printed tables
options(knitr.kable.NA = "")
# Set constraint
options(contrasts = c("contr.sum", "contr.poly"))

# Load extra tools
source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
# Demo code to set up R ----
## Load packages
packages <- c("tidyverse", "hasseDiagram", "knitr", "kableExtra",
              "car", "psych", "coin", "rcompanion", "dunn.test")
lapply(packages, library, character.only = TRUE, quietly = TRUE)

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

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

# Demo code for loading Honey data ----
honeyData <- data.frame(
  Amount = c(150, 50, 100, 85, 90, 95, 130, 50, 80),
  Varietal = rep(c("Clover", "Orange Blossom", "Alfalfa"), each = 3)
)
## Set Varietal to factor
honeyData$Varietal <- as.factor(honeyData$Varietal)

# Demo Code for loading Alfalfa study data ----

## Be careful when breaking URLs into chunks; break on forward slashes
rootPath <- "https://static-content.springer.com/esm/chp%3A10.1007%2F978-3-319-30634-6_7/"
filePath <- "MediaObjects/385146_1_En_7_MOESM1_ESM.csv"

alfalfaData <- read.table(
  file = paste0(rootPath, filePath),
  header = TRUE,
  sep = ","
)

alfalfaData$Treatment <- as.factor(alfalfaData$Treatment)
alfalfaData$Variety <- as.factor(alfalfaData$Variety)

# Demo code for Hasse Diagram for the Honey Study ----
modelLabels <- c("1 Make Honey 1", "3 Varietal 2", "9 (Hives) 6")
```

```r
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE),
  nrow = 3,
  ncol = 3,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Demo code of an index plot using the response for Honey Study ----
ggplot(
  data = honeyData,
  mapping = aes(
    x = 1:nrow(honeyData),
    y = Amount
  )
) +
  geom_point() +
  geom_line() +
  geom_hline(
    yintercept = mean(honeyData$Amount, na.rm = TRUE),
    color = "red",
    linetype = "dashed"
  ) +
  theme_bw() +
  xlab("Measurement Order") +
  ylab("Amount of Honey (lbs)") +
  scale_x_continuous(breaks = 1:9)

# Demo code for running the Kruskal-Wallis test in base R ----
honeyOmni <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honeyData,
  na.action = "na.omit"
)

## Demo Code for showing Kruskal-Wallis results ----
honeyOmni

# Demo Code for measuring practical significance for Kruskal-Wallis ----
## Honey Study
honeyEffectSize <- rcompanion::epsilonSquared(
  x = honeyData$Amount,
  g = honeyData$Varietal,
  digits = 4
)

## Display results
honeyEffectSize

# Demo Code for fitting Oneway Alfalfa Study ----
## KW Results
alfalfaModel0 <- kruskal.test(
  formula = Larvae ~ Treatment,
  data = alfalfaData,
  na.action = "na.omit"
)
```

```
alfalfaModel0
# Demo Code for using Dunn's Test ----
## 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 = alfalfaData$Larvae, # response vector
  g = alfalfaData$Treatment, # 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',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for the DSCF Test ----
dscf <- dscfTest(
  response = alfalfaData$Larvae,
  factor = alfalfaData$Treatment
)
# Kable Code for DSCF
knitr::kable(
  x = dscf,
  digits = 3,
  col.names = c("Comparison", "Observed W", "Adj. p-value"),
  caption = paste("Post Hoc-Dwass-Steel-Critchlow-Fligner Tests"),
  align = 'lcc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
kw.PostHoc(
    response = alfalfaData$Larvae,
    treatments = alfalfaData$Treatment
  ) %>%
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"
  )
# Demo Code for Interaction Plot in Alfalfa Pest Study ----
ggplot2::ggplot(
  data = alfalfaData,
  mapping = aes(
    x = Variety,
    y = Larvae,
    color = Treatment,
    shape = Treatment,
    linetype = Treatment,
    group = Treatment
  )
) +
  stat_summary(fun = "median", geom = "point") + # Notice the change in function
  stat_summary(fun = "median", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Variety") +
  ylab("Number of Whole Larvae Found") +
  labs(color = "Treatment") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

# Your Turn Code ---------------------------------------------------------------

# Hasse Diagram for Insect Spray Study
modelLabels <- c("1 Stop Insects 1", "6 Spray 5", "72 (Plants) 66")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE),
  nrow = 3,
  ncol = 3,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Load Data
data("InsectSprays")

# Run Kruskal Wallis Test
coin::kruskal_test(
  formula = count ~ spray,
  data = InsectSprays,
  ties.method = "mid-ranks"
)

# Effect size for KW on Insect Spray
```

```
rcompanion::epsilonSquared(
  x = InsectSprays$count,
  g = InsectSprays$spray,
  digits = 4
)
```