Nonparametric Shortcut: One-way ANOVA

Neil J. Hatfield

3/17/2022

In this tutorial, we are going to explore using `R` to fit a One-way ANOVA model to two data sets using the nonparametric shortcut known as the Kruskal-Wallis H Test.

If you have not yet gone through the Oneway Parametric Shortcut tutorial, I recommend that you go look at that guide first.

Setting up `R` and Reading in Data

As usual, there are a few housekeeping items to do to ensure that our current `R` session is ready for our work.

Loading Packages and Setting Options

The first thing that you'll want to do is load what really becomes our core set of packages: `tidyverse`, `hasseDiagram`, `knitr`, `kableExtra`, `car`, and `psych`. I'll omit the code example for loading packages here. (Check out the code appendix for this tutorial.)

Additionally, we will want to set the same options as we did for the Parametric Shortcut:

```r
#Demo code for setting options
# 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"))
```

You will also want to load the helper functions (as a good habit to get into).

```r
# Demo code for loading Neil's helper functions
source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
```

Additional Packages

To do the Kruskal-Wallis H test does not require you to use any special packages, unless you want to. I will show you two ways to conduct the Kruskal-Wallis test: one using base `R` packages and one using the `coin` package. To get a measure of practical significance, you'll want to use the `rcompanion` package and the `epsilonSquared` function.

Load Data

For this exploration we're going to use two data sets: our Honey Study and the Song Knowledge Study. The Honey Study highlights the instance where we opted to NOT use the parametric shortcut. We're looking at the Song Knowledge study so that we have an example where we've used both the Parametric and Nonparametric shortcuts. This will allow us to make comparisons.

```r
# Demo code

# Honey Data
honey <- 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
honey$Varietal <- as.factor(honey$Varietal)

# Song Data
songData <- read.csv(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/songKnowledge2022.csv",
  header = TRUE,
  sep = ","
)
# Set year to an ordered factor
songData$year <- factor(
  x = songData$year,
  levels = c("sophomore", "junior", "senior")
)
```

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.

Is ANOVA Even Appropriate?

Whether we're aiming to us the Parametric Shortcut or the Nonparametric Shortcut, we must still check to see whether ANOVA methods are even appropriate for our SRQ.

As mentioned in the Parametric Shortcut tutorial, the base requirements for One-way ANOVA are:

- From Unit 2
  - you are working with a qualitative/categorical factor,
  - you are working with a quantitative response,

- From Unit 3
  - you are working with an additive model,
  - you have estimable effects, and
  - you have estimable errors/residuals.

We can check out these requirements using the `str` function for the first two and our Hasse diagram for the last three. Check out the Parametric Shortcut guide/tutorial for more information.

Hasse Diagram 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

# 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
)
```

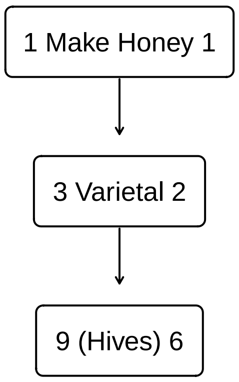


Figure 1: Hasse Diagram for Honey Study

Your Turn

Try creating the code (either on your own or via the app) for the Hasse diagram for the Song Knowledge and writing up whether or not ANOVA methods are appropriate for investigating the statistical research question.

Assessing Assumptions

The Kruskal-Wallis H test is the nonparametric shortcut for One-way ANOVA situations. For this test we only have two assumptions:

1) The response follows some *continuous* distribution.
2) Independence of Observations

Continuous Distribution Assumption

For the first assumption (response follows some continuous distribution) has a bit more nuance than first glance. What we really 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).

The "continuous adjacent" highlights that we can also allow for responses that are ordinal in nature (e.g., Likert scales) but aren't truly continuous.

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.

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.

Indepdence 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.)

```r
# Demo code of an index plot using the response
# Recall the honey data are in measurement order
ggplot(
  data = honey,
  mapping = aes(
    x = 1:nrow(honey),
    y = Amount
  )
) +
  geom_point() +
  geom_line() +
  geom_hline(
    yintercept = mean(honey$Amount),
    color = "red",
    linetype = "dashed"
  ) +
  theme_bw() +
  xlab("Index") +
  ylab("Amount of Honey (lbs)")
```

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).

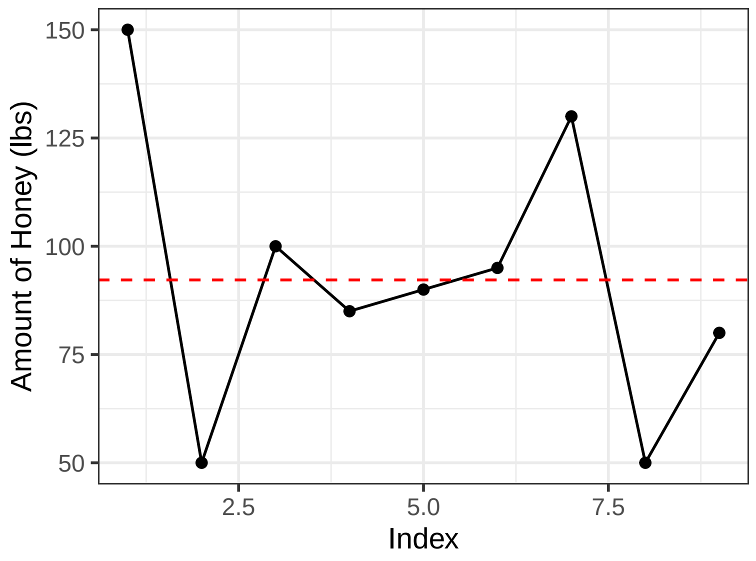


Figure 2: Index Plot for Honey Study

Your Turn

For the Song Knowledge study, decide whether or not the two assumptions for the Kruskal-Wallis H test are satisfied. Why or why not?

Fitting the Model

Before we look at using `R` to fit the model, I want to take a moment to highlight that when we make use of the Kruskal-Wallis H test as our non-parametric shortcut, we make a couple of important swaps to the our model.

- We will swap the *GSAM*, $\mu_{\bullet\bullet}$, for the *Grand Median*, $\theta_{\bullet\bullet}$
- We will swap out the factor/treatment effects **relative to the *GSAM***, α_i, for the effects **relative to the *Grand Median***, τ_i
- We will still have a residual/error term (e_{ij})

This means that our algebraic model for the alternative now looks like $H_A : Y_{ij} = \theta_{\bullet\bullet} + \tau_i + e_{ij}$ for some $\tau_i \neq 0$.

As I mentioned at the start of this guide/tutorial, there are (at least) two ways to do the Kruskal-Wallis H test. I'll show you two methods.

Base Packages

The first method for doing the Kruskal-Wallis H test is from the base packages of `R` (specifically, the `stats` package which you automatically have when you install `R`). To run the test, you use the `kruskal.test` function.

```r
# Demo code for running the Kruskal-Wallis test in base R

# Honey Study
honeyResultsBase <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)
```

Notice how similar the structure of the call is to the `aov` call for the parametric shortcut. That is, both use the `formula`, `data`, and `na.action` arguments. These arguments have the same meaning and structure as in the parametric case. You can even drop the `na.action` argument if you wish.

Notice that we saved the output of `kruskal.test` into an object called `honeyResultsBase`. This will let us call values from the output easily later on.

`coin` Package

A second way to fit the model for the Kruskal-Wallis H test is by using the `coin` package's `kruskal_test` function:

```r
# Demo code for running the Kruskal-Wallis test from the coin package

# Honey study
honeyResultsCoin <- coin::kruskal_test(
  formula = Amount ~ Varietal,
  data = honey,
  ties.method = "mid-ranks"
)
```

The first two arguments to `kruskal_test` are exactly the same as in `kruskal.test`. The third argument, `ties.method`, is something new. While we won't go into what this option means or what other choices you can make here, you can use `ties.method = "mid-ranks"` for our class.

Looking at Results

Another way that the nonparametric shortcut (Kruskal-Wallis H test) is different from the parametric shortcut (ANOVA F test) is that there is no ANOVA table. Instead, we often list out the values directly into our narrative.

Decision Rule

However, 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 this tutorial, I'm going to use $\mathcal{E}_I = 0.05$ and $UT = 0.03$.

Look at Results

Unlike the `aov` output, we do not need to use either `summary` or `anova` to display the results of the test for us. We can either run the test without saving the results (not shown) or call the object we saved the results into. This is true whether you used base `R` or the `coin` package.

```r
# Demo code for quickly looking at the results of the Kruskal-Wallis test

# Base packages
honeyResultsBase
```

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

```r
# Coin package
honeyResultsCoin
```

```
##
##  Asymptotic Kruskal-Wallis Test
##
```

```
## data:  Amount by
##   Varietal (Alfalfa, Clover, Orange Blossom)
## chi-squared = 0.56022, df = 2, p-value = 0.7557
```

You'll notice that the raw outputs are similar. The `coin` package kindly lists the levels of our factor, but that's about the extent of meaningful differences between the two options.

What is true in both cases is that the raw output should NOT appear in the body of your report. The report should look professional.

Calling Values for Reports

To make make our reports look professional, we weave the results of the Kruskal-Wallis H test into our narratives. There are two ways that you can do this:

1) You can run the output into your console and then manually type of the values into your narrative, OR
2) You can use some in-line code in your R Markdown file to write the values for you.

A perk of the first option, is that you can see the value and can immediately interpret that value. However, a draw back is that if you discover a data entry error, you have to manually update all of the values yourself.

For the second option, we have the reverse: the values automatically update themselves, but we don't as readily see the values when we're writing.

I try to split the difference. I'll run the output so I can see the numbers, and I'll use in-line code to write the values into my narrative: best of both worlds.

The key to this method is knowing how to extract the values from your output objects, `honeyResultsBase` and `honeyResultsCoin`. Luckily, I have tables to help you with this. There are three values we work into our narratives: H, the degrees of freedom (DF), and the p-value.

Call Values from Base `R`

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

Call Values from `coin`

Value	Object Name	Code Example	Final Result
H	`coin::statistic(honeyResultsCoin)`	`round(coin::statistic(honeyResultsCoin), digits = 2)`	-0.56
DF	`honeyResultsCoin@statistic@df`	`honeyResultsCoin@statistic@df`	2
p-value	`coin::pvalue(honeyResultsCoin)`	`round(coin::pvalue(honeyResultsCoin), digits = 4)`	0.7557

Notice that the calls from the `coin` package do not use `$` like base `R`; rather, they use `@` or special functions `statistic` and `pvalue`.

After we talk about getting a measure of practical significance, I'll give a demonstration for writing up the results.

Practical Significance

To get a measure of effect size, we will use Epsilon Squared. However, we must us 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 = honey$Amount,
  g = honey$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 treatment 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.

Writing Up Results

Since there is no ANOVA table, we simply incorporate the results of the Kruskal-Wallis H test into our narrative. Here is an example for the Honey study.

Results

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.

[NOTE: 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).]

Generating Code

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(honeyResultsBase$statistic, digits = 2)‘\\) with ‘ r honeyResultsBase$parameter‘ degrees of freedom. This results in a p-value of ‘ r round(honeyResultsBase$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: 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.

Your Turn

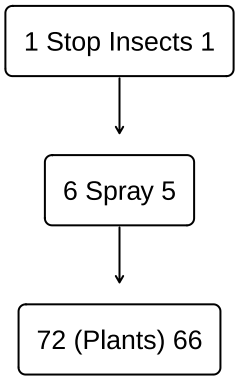
Using the Honey study as an example, build your own paragraph for the Song Knowledge study using the Kruskal-Wallis *H* test.

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
```

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", "coin", "rcompanion")
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 for setting options
# 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"))

# Demo code for loading Neil's helper functions
source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")

# Demo code

# Honey Data
honey <- 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
honey$Varietal <- as.factor(honey$Varietal)

# Song Data
songData <- read.csv(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/songKnowledge2022.csv",
  header = TRUE,
  sep = ","
)
# Set year to an ordered factor
songData$year <- factor(
  x = songData$year,
  levels = c("sophomore", "junior", "senior")
)

# DEMO CODE

# 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,
```

```r
  ncol = 3,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Demo code of an index plot using the response
# Recall the honey data are in measurement order
ggplot(
  data = honey,
  mapping = aes(
    x = 1:nrow(honey),
    y = Amount
  )
) +
  geom_point() +
  geom_line() +
  geom_hline(
    yintercept = mean(honey$Amount),
    color = "red",
    linetype = "dashed"
  ) +
  theme_bw() +
  xlab("Index") +
  ylab("Amount of Honey (lbs)")

# Demo code for running the Kruskal-Wallis test in base R

# Honey Study
honeyResultsBase <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)

# Demo code for running the Kruskal-Wallis test from the coin package

# Honey study
honeyResultsCoin <- coin::kruskal_test(
  formula = Amount ~ Varietal,
  data = honey,
  ties.method = "mid-ranks"
)

# Demo code for quickly looking at the results of the Kruskal-Wallis test

# Base packages
honeyResultsBase

# Coin package
honeyResultsCoin

# Demo Code for measuring practical significance for Kruskal-Wallis

# Honey Study
honeyEffectSize <- rcompanion::epsilonSquared(
  x = honey$Amount,
  g = honey$Varietal,
  digits = 4
)

# Display results
honeyEffectSize

# 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
)
```