Nonparametric Shortcut: One-way ANOVA

Neil J. Hatfield

3/1/2021

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.

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 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

Our first step will be to load data into our R session. Recall that after checking, we decided that the Honey data did not satisfy the assumptions for the parametric shortcut. Thus, we will make use of the Honey data here.

```
## 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)
```

Is ANOVA Even Appropriate?

The following is taken from the guide on the parametric shortcut. These are still applicable to the nonparametric shortcut.

Recall the base requirements for One-way ANOVA are:

- you are working with a categorized factor,
- you are working with an additive model,
- you have estimable effects, and
- you have estimable errors/residuals.

You can check these four requirements rather quickly. First, use the `str` function to ensure that R is thinking about the `Varietal` (or `temp`) as a factor. Second, examine your Hasse diagram and algebraic model–did you build upon the additivity of terms? (If you've followed the methods that we've been using, then yes.) Third, did you tell R to set the constraint? Look at the term nodes in the Hasse diagram. Do each of the

terms you want to test have non-zero (and non-negative) *Degrees of Freedom*? (Yes, you're good; no, there are problems.) Fourth, look at the error term in your Hasse diagram. Do you have a positive (i.e., non-zero, non-negative) number of *Degrees of Freedom*? (Yes, you're good; no, there are problems.)

Notice the work that the Hasse diagram is doing for us. They not only help us visualize our model but they also set up our screens, and allow for us to check whether we meet the basic requirements for doing ANVOA. The Hasse Diagram App will through an error if you run into problems with your *Degrees of Freedom*.

Example-Honey Data

We will begin the Honey example from class. Before we look at any p-values and make any decisions, we need to ensure that the prerequisites and then assumptions are met.

Figure 1 shows the Hasse diagram for our Honey Study. We're interested in testing the type of varietal. Both that node and our error node have positive values for *Degrees of Freedom*. This taken with our study design, support our using One-way ANOVA methods.

```
# DEMO CODE

# Hasse Diagram for the Honey Study
modelLabels <- c("1 Make Honey 1", "3 Type 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
)
```

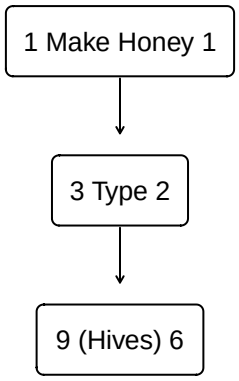


Figure 1: Hasse Diagram for Honey Study

Assessing Assumptions

For the Kruskal-Wallis H test, we will only make two assumptions:

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

Rather than focusing on residuals, we would want to use the observed values of the response.

```
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)")
```

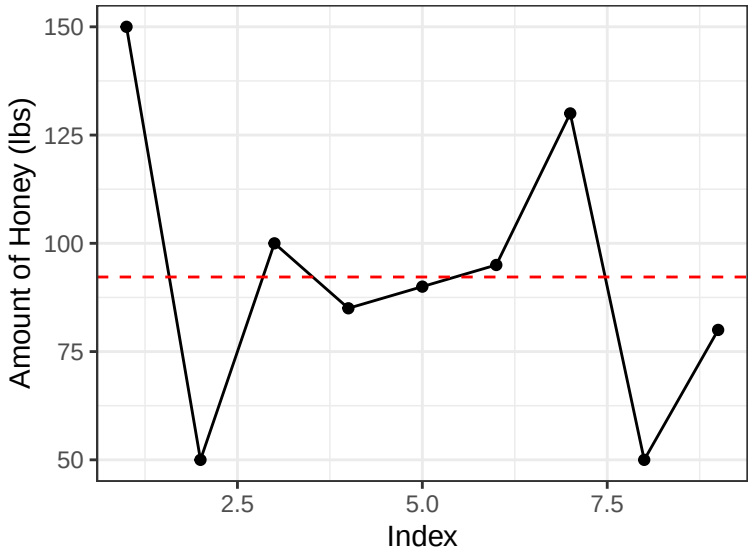


Figure 2: Index Plot for Honey Study

For assessing the assumption about a continuous distribution, we will not make use of *any* data visualizations or tests. Rather, we will think critically about the response and reason whether or not the response follows some continuous distribution.

The reason that we won't look at a QQ plot is that there are just too many continuous distributions for us to check.

Our response is the "amount of honey produced (lbs)". If we think about this quantity, we quickly realize that this is a continuous quantity. Thus, if we think about the long-run behavior of the underlying process behind the quantity, we will have to have a continuous distribution.

Base Packages

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

```r
# DEMO CODE

# Kruskal-Wallis Test-Base R
results1 <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)
```

Quick Look at Results

To quickly look at the results of the Kruskal-Wallis test, you would just need to call the `results1` object (or run the test without the `results1 <-` portion of code).

```r
#DEMO CODE
results1
```

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

Calling Values for Reports

Remember, this quick look method is fine if you are just looking at the results for yourself. However, if you are going to be reporting the values, you'll want to do so more professionally. We do not make an ANOVA table for the Kruskal-Wallis H test, thus, these values will need to be reported in the narrative.

Value	Code Name	Code Example	Final Result
H	`results1$statistic`	`round(results1$statistic, digits = 2)`	$H = 0.56$
DF	`results1$parameter`	`results1$parameter`	2
p-value	`results1$p.value`	`round(results1$p.value, digits = 4)`	0.7557

Using `coin`

The `coin` package has the `kruskal_test` which provides an alternative approach to doing the Kruskal-Wallis H test. One benefit to using this approach is that it is applicable with One-way ANOVA situations as well as One-way ANOVA + Block situations. Thus, this method covers more situations.

```r
# DEMO CODE

# Kruskal-Wallis Test-coin package
results2 <- coin::kruskal_test(
  formula = Amount ~ Varietal,
  data = honey,
  ties.method = "mid-ranks"
)
```

Quick Look

```r
# DEMO CODE
results2
```

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

Just as before, the quick look lets you see the results of the Kruskal-Wallis test.

Calling Values

Calling the values when you use the `coin` package is a bit different from the base packages, since the output is a `S4` object. (Don't worry about what that means.)

Value	Code Name	Code Example	Final Result
H	`coin::statistic(results2)`	`round(coin::statistic(results2), digits = 2)`	$H = 0.56$
DF	`results2@statistic@df`	`results2@statistic@df`	2
p-value	`coin::pvalue(results2)`	`round(coin::pvalue(results2), digits = 4)`	0.7557

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.

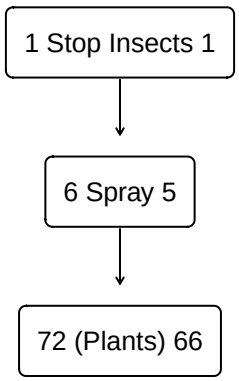
```
# Demo Code

# Practical Significance
rcompanion::epsilonSquared(
  x = honey$Amount,
  g = honey$Varietal,
  digits = 4
)
```

```
## 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.

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")`.




```
##
##  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", "coin", "rcompanion")
lapply(packages, library, character.only = TRUE)

# Tell Knitr to use empty space instead of NA in printed tables
options(knitr.kable.NA = "")

## 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)

# DEMO CODE

# Hasse Diagram for the Honey Study
modelLabels <- c("1 Make Honey 1", "3 Type 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
)

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

# Kruskal-Wallis Test-Base R
results1 <- kruskal.test(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)

#DEMO CODE
results1

# DEMO CODE

# Kruskal-Wallis Test-coin package
results2 <- coin::kruskal_test(
  formula = Amount ~ Varietal,
  data = honey,
  ties.method = "mid-ranks"
)

# DEMO CODE
results2

# Demo Code

# Practical Significance
rcompanion::epsilonSquared(
  x = honey$Amount,
  g = honey$Varietal,
  digits = 4
)

# 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
)
```

```r
# 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
)
```