

Parametric Shortcut: One-way ANOVA

Neil J. Hatfield

Last Updated: October 09, 2023

In this tutorial, we are going to explore using R to fit a One-way ANOVA model using the parametric shortcut known as the One-way ANOVA F Test. I'll interweaving a couple of different examples throughout the text.

1 Setting Up R and Reading in Data

There are two parts to this portion of the tutorial: getting R in order and reading in the data. If you are already familiar with these activities, you can skip ahead. HOWEVER, make sure you check out the code for setting up our constraint.

1.1 Getting R Ready

The first thing that we need to do is ensure that R is set up so that we can have success. For this particular tutorial, we will be using the following packages: `tidyverse`, `hasseDiagram`, `knitr`, `kableExtra`, `car`, `psych`, and `parameters`. You can load these packages into your session by running the following code in the Console or by placing this code into a code chunk in your R Markdown file:

```
# Demo code for loading packages ----
packages <- c("tidyverse", "hasseDiagram", "knitr",
             "kableExtra", "car", "psych", "parameters")
lapply(packages, library, character.only = TRUE)
```

1.1.1 Specifying R Options

While there are many options we can set, two options are particularly useful to set at the start: table formatting and on model constraint.

1.1.2 Table Formatting

To make professional looking tables in a R Markdown file, I recommend using the `knitr` and `kableExtra` packages. One thing to keep in mind is that by default, R does not like truly empty table cells (typically printing "NA"). Thus, to keep our tables as uncluttered as possible, we need to instruct R to leave empty table cells visually empty. We can do this with the following code:

```
# Demo code for controlling table options ----
# Tell Knitr to use empty space instead of NA in printed tables
options(knitr.kable.NA = "")
```

I typically place this code right after I load my packages.

1.1.3 Set Our Constraint

Recall that in order to ensure that we have estimable functions, we must set a side condition or constraint on our treatments:

$$\sum_i^k \alpha_i = 0$$

This is not what R does by default, but we can tell R to adopt this constraint with the following code:

```
## Demo code for setting the constraint ----
options(contrasts = c("contr.sum", "contr.poly"))
```

Again, I place this just after I set my table options. NOTE: you have to specify this option BEFORE you build any models in R. (If you forget, you can run the command, but you should then re-fit your models and re-ask R to provide you with your estimates.)

1.1.4 Some Additonal Handy Tools

Over the years I have created some additional tools which can be useful as you perform ANOVA analyses. To access them, you must first load them into your R session/R Markdown document:

```
# Demo code for loading Neil's extra tools ----
source("https://raw.githubusercontent.com/neilhatfield/STAT461/main/rScripts/ANOVATools.R")
```

This is typically the last line of code in my first code chunk of any R Markdown file for ANOVA.

1.2 Load Data

Once you have set the R options, now comes reading in data. For this tutorial, we're going to work with three different data sets: our Song Knowledge data, data from a honey study, and data from Example 3.2 Resin Lifetimes from the Oehlert textbook.

The following code demonstrates how we can read in the three data sets:

```
# Demo Code for Loading Data ----
## Song Data
songData <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/main/dataFiles/songKnowledge_Fa23.csv",
  header = TRUE,
  sep = ",",
)
### Set year to an ordered factor
songData$Year <- factor(
  x = songData$Year,
  levels = c("Junior", "Senior", "Other")
)

## Honey Data--Manual Entry
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 (no particular order)
honey$Varietal <- as.factor(honey$Varietal)

## Resin Lifetimes Data
resin <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/main/dataFiles/resinLifetimes.dat",
  header = TRUE,
  sep = "" # Notice the change in separator
)
#### Set temp to factor
resin$temp <- as.factor(resin$temp)

### Change the name of the y column to something more meaningful
names(resin)[which(names(resin) == "y")] <- "log10Lifetime"
```

2 Explore Your Data

At this point in time, you should engage in exploratory data analysis including creating professional looking data visualizations (I recommend using the `ggplot2` package) as well as looking at descriptive statistics by groups (I recommend using `psych` or `dplyr` packages). For more details on both of these topics, see the starting guides for Data Visualizations and Descriptive Statistics I've posted.

2.1 Your Turn

Explore the data from any of the three studies. What do you find? What do you learn?

3 Is ANOVA Even Appropriate?

Recall 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.

The first two requirements (from Unit 2) you can check quickly in R by using the `str` (“structure”) function or by clicking on the blue circle with a white triangle to the left of each data frame’s name in the Environment tab of R Studio. This is *not* something that you put into any reports. Rather, this is something you should check for yourself. In essence, this is to make sure R is thinking about your data in the same way that you are.

```
# Checking the first two base requirements
```

```
str(songData)
```

```
## 'data.frame': 12 obs. of 2 variables:
## $ Year : Factor w/ 3 levels "Junior","Senior",...: 1 1 1 1 3 3 3 3 2 2 ...
## $ Score: int 7 12 3 6 1 2 6 9 0 3 ...
```

```
str(honey)
```

```
## 'data.frame': 9 obs. of 2 variables:
## $ Amount : num 150 50 100 85 90 95 130 50 80
## $ Varietal: Factor w/ 3 levels "Alfalfa","Clover",...: 2 2 2 3 3 3 1 1 1
```

```
str(resin)
```

```
## 'data.frame': 37 obs. of 2 variables:
## $ temp : Factor w/ 5 levels "175","194","213",...: 1 1 1 1 1 1 1 1 2 2 ...
## $ log10Lifetime: num 2.04 1.91 2 1.92 1.85 1.96 1.88 1.9 1.66 1.71 ...
```

What we want to see is that for our factor(s), R has the word “Factor” immediately after their (column) name. For our response, we want to see either “num” or “int” after their (column) name. If we see these AND they match our expectations of the data, then we can say that we’ve met these two requirements.

The last three base requirements (from Unit 3) stem from the Hasse diagram. Essentially, if you can build the Hasse diagram and have positive (i.e., non-zero and non-negative) degrees of freedom everywhere, then all three of these are satisfied.

Hasse diagrams can be included in your reports. The following is an example for how we might do so with the Honey study. For putting Hasse diagrams into your R Markdown files, I recommend using the [Hasse Diagram App](#) and copying the R code generated there.

3.1 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 for a Hasse Diagram ----
```

```
## 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
)

```

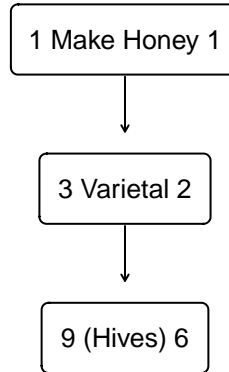


Figure 1: Hasse Diagram for Honey Study

3.1.1 Your Turn

Try creating the code (either on your own or via the app) for the Hasse diagrams for the Song Knowledge and the Resin Lifetime studies (see p. 32 of Oehlert).

4 Fit the ANOVA Model

In order to check out the assumptions of the Parametric Shortcut (a.k.a. “the ANOVA F Test”), we first need to fit the ANOVA model in R. This will enable us to access the residuals in an easy way. I must give a word of caution here: don’t look at the results of the model just yet. You must first assess all of the assumptions so that you can build your trust in the results.

To fit our ANOVA model, we will primarily use the `aov` function that is part of base R. (You can also use the `lm` function with the same arguments.) We will want to save our model to a named object so that we can call them at later times. Here is how we would build/fit the model:

```

# Demo code for building ANOVA models ----
## Song Knowledge study
songModel <- aov(
  formula = Score ~ Year,
  data = songData,
  na.action = "na.omit"
)

## Honey study
honeyModel <- aov(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)

## Resin Lifetime study
resinModel <- aov(
  formula = log10Lifetime ~ temp,
  data = resin,
  na.action = "na.omit"
)

```

Notice that in all three examples, we used the same three arguments to `aov`: `formula`, `data`, and `na.action`.

- The `formula` argument is how we express our model. In essence, you put in all of our factors/terms. R automatically knows to account for the first and last nodes of our Hasse diagrams (the action screen and measurement units).
 - The structure of the formulas are `responseName ~ factorName` for one-way ANOVA.
- The `data` argument is where you tell `aov` the name of the data frame you want to use.
- The `na.action` argument is a safety precaution. You are instructing R that if there is an observation with missing values, then R is to omit that observation from the model. (You may leave this argument off if you desire and are willing to run the risk of problems.)

5 Assessing Assumptions

Before we look at the results of fitting the ANOVA model, we must first assess the assumptions. R automatically does the parametric shortcut test when we call the results of fitting the model—regardless of whether the parametric shortcut is actually valid. Thus, we need to be convinced that we’ve met the assumptions of the test well enough to trust the results.

For the parametric shortcut (a.k.a. “the ANOVA F test”), there are three assumptions:

- 1) Our residuals need to follow a Gaussian distribution,
- 2) We have homoscedasticity (around the model) among the residuals, and
- 3) We have Independent Observations

For the first two assumptions, we will need to access the residuals from our model. We can do this in two different ways: `songModel$residuals` or `residuals(honeyModel)`. Either method will give the same set of things.

Remember, we want to *assess*, *not test* our assumptions. Thus, we will be relying on specific data visualizations (and some descriptive statistics) to help us. All of which can go into your reports.

5.1 Assessing the Gaussian Assumption

The first assumption is that our residuals follow a Gaussian (“normal”) distribution, A QQ plot is the tool of choice; one of the better versions of this plot comes from the `car` package. The following code demonstrates creating QQ Plots for the Song Knowledge and Honey studies:

```
# Demo code for making QQ plots ----
## Chunk options for side-by-side plots
### fig.show="hold", out.width="100%", fig.height=3.5
## R option for side-by-side plots
par(mfrow = c(1,2), mar = c(4, 4, 0.1, 0.1))

## Song Knowledge study
car::qqPlot(
  x = songModel$residuals,
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (points)"
)

# Honey study
car::qqPlot(
  x = residuals(honeyModel),
  distribution = "norm",
  envelope = FALSE,
  id = TRUE,
  pch = 1,
  ylab = "Residuals (lbs)"
)
```

```
## [1] 2 1
```

You’ll notice that in the `qqPlot` function there are several arguments that allow you to customize the plots. The two required arguments are `x` (where you’ll call the residuals) and `distribution` (where you will use “norm”).

The `id`, `pch`, and `ylab` arguments are meant to improve the visual display of the graph.

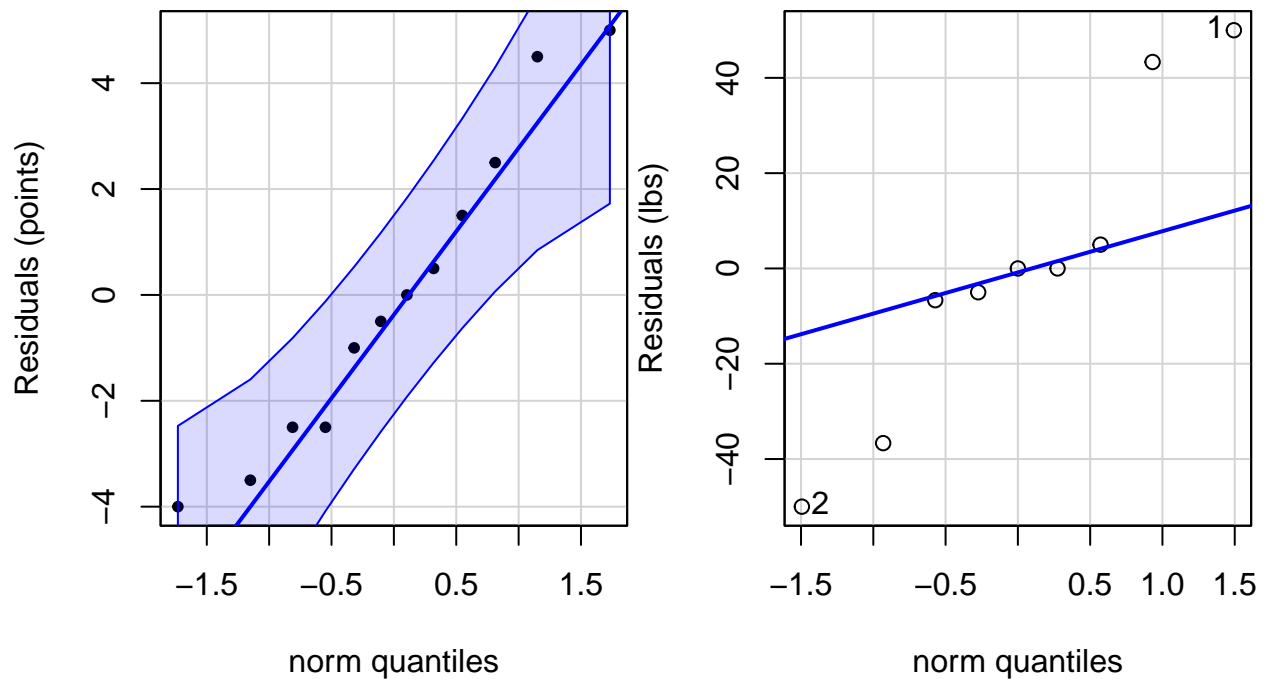


Figure 2: QQ Plot Residuals for Song Knowledge (left) and Honey (right) Studies

- The `id` argument will label the two points with the most extreme vertical distances from the solid blue reference line. These labels can be distracting, thus I recommend using `id = FALSE`.
 - You'll also notice that there is an errant `## [1] 2 1` in this document, just above the two QQ plots. This is consequence of setting `id = TRUE` and will make your reports look *less* professional.
- The `pch` (plotting character) controls the type of dot used. The default value is 1 for an open circle; solid dots are typically better. Thus, you'll want to use `pch = 19` or `pch = 20` for solid dots; the 19 version is slightly larger in size than 20.
- The `ylab` is a way to make the label for the vertical axis a bit nicer. You can use `ylab = "Residuals"` for your plots. (Where possible, include the units of measure.)

The `envelope` argument is particularly useful when assessing the Gaussian assumption for ANOVA. To turn off the envelope you use `envelope = FALSE`. To add the envelope, you used `envelope = num` where `num` is your chosen confidence level such as 0.90 for the Song Knowledge context. This helps in our assessment as any points that are away from the line but yet still inside the envelope aren't of much concern. This lets us focus on residuals which are truly far from the blue reference line. The level of confidence we use has a direct impact on how concerned we should be for violations of the Gaussian assumption. For 90%, then we are okay with at most ~10% of the observed residuals being outside the envelope. This decision banks upon us capitalizing that balanced designs have moderate robustness to violating the Gaussian assumption.

For the Song Knowledge QQ Plot (left of Figure 2), we can see that no residuals are outside the envelope. Since none of the residuals are beyond the 90% envelope, we can say that the Gaussian assumption is satisfied.

Take a look that the QQ Plot for the Honey Study (right of Figure 2). How would you judge the Gaussian assumption? Notice that the lack of envelope makes the judgement a bit harder. However, we can note that there are four observations (out of 9) whose residuals are quite far from the blue reference line (look at the scale of the vertical axis). This suggests that our residuals do not follow a Gaussian distribution.

5.1.1 Your Turn

Try replicating the QQ plot code for the Resin Lifetime study. How would you assess the Gaussian assumption?

5.1.2 Secondary Tool

Another tool that you can use for assessing the Gaussian assumption are two descriptive statistics: *Sample Skewness* and *Sample Excess Kurtosis*. These are generated by the `psych` package's `describe` function (i.e., `psych::describe(songModel$residuals)`). You can also call them directly with `psych::skew(songModel$residuals)` and `psych::kurtosi(songModel$residuals)`. Ideally, we want the value of *Sample Skewness* and *Sample Excess Kurtosis* to be as close to zero as possible.

5.1.2.1 Important Note Something to keep in mind about the values of *Sample Skewness* and *Sample Excess Kurtosis* is that their magnitudes are challenging to interpret. The context of the study matters. Thus, there is **NO** magic cutoff/rule of thumb for deciding how large (in magnitude) is *too* large for either of these statistics.

Some magnitudes are easier to make a decision than others. For example, consider the following values of *Sample Excess Kurtosis*: 1.2×10^{-1} , -2.1×10^0 , and 2.1×10^2 . Statisticians tend to be in general agreement that the first and third are easy to classify (not that far from 0 and way off from 0, respectively). However, the -2.1 is harder to decide. Thus, these statistics are almost never used alone but in conjunction with a QQ plot or some other data visualization (e.g., a worm plot or de-trended QQ plot).

For the Song Knowledge context, the value of *Sample Skewness* is 0.31 and the value of *Sample Excess Kurtosis* is -1.29. When we partner these values with the QQ plot, we can say that we're satisfied with the Gaussian assumption being met.

For the Honey study, the value of *Sample Skewness* is 0.11 and the value *Sample Excess Kurtosis* is -1.17. Using these values in conjunction with the QQ plot re-affirms that we should question whether we've satisfied the Gaussian assumption with the honey data.

Notice that the two values of *Sample Excess Kurtosis*, -1.29 and -1.17, are not that far from one another. This highlights the above point about not using these statistics in isolation of other tools.

5.2 Assessing Homoscedasticity

The second assumption for the parametric shortcut is that of homoscedasticity. This assumption deals with there being the same or similar amounts of variation within each group (around our model). For this assumption, you can make use of the values of the *Sample Arithmetic Standard Deviation* or *Sample Arithmetic Variance* IF you are looking by groups. The `describeBy` function from the `psych` package is your friend here. However, there is also visual method which you can use: the strip chart.

A strip chart is a variation on a scatter plot. If you have done regression, you might have looked at a scatter plot of residuals by fitted values. A strip chart is the same idea. However, rather than having many different fitted values, we'll only have the same number as groups. This is what creates the strips. For this visualization, we will use the `ggplot2` package rather than the output of the `plot` function. (We always want to aim for professional looking plots.)

```
# Demo code for making strip charts ----
# Chunk options for side-by-side
### fig.subcap=c("Song Knowledge Study", "Honey Study"), fig.ncol=2,
### out.width="50%"
## There are no special par calls needed with this approach (only good for PDF outputs)

## Song Knowledge study
ggplot(
  data = data.frame(
    residuals = songModel$residuals,
    fitted = songModel$fitted.values
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  theme_bw() +
  xlab("Fitted values (points)") +
  ylab("Residuals (points)")

## Honey study
ggplot(
  data = data.frame(
    residuals = residuals(honeyModel),
    fitted = fitted(honeyModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  theme_bw() +
  xlab("Fitted values (lbs)") +
```

```
ylab("Residuals (lbs)")
```

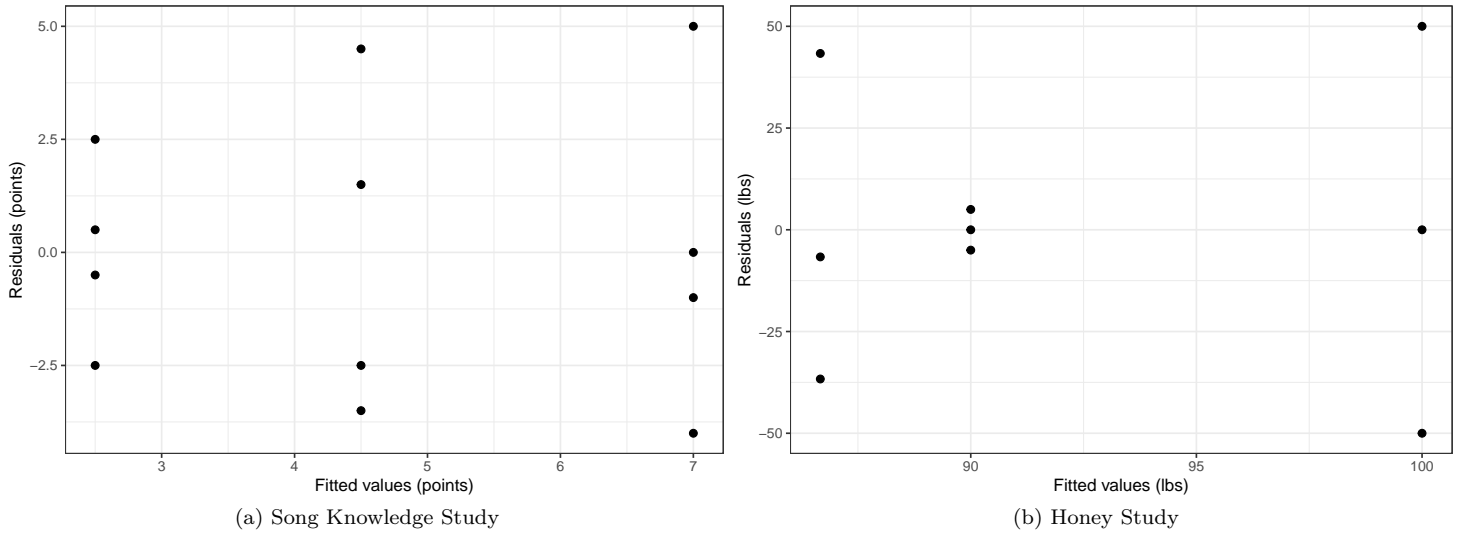


Figure 3: Strip Chart Examples

When we look at strip charts for assessing homoscedasticity, we're primarily focused two aspects: comparing the lengths of the strips and if there are any patterns.

Let's first focus on the Song Knowledge portion of Figure 3. We have three strips of 4 points (note: two points for the lowest fitted value are on top of each other). The lengths of these strips are such that the first strip is just about half that of the third strip. This is a rule of thumb that we want to look out for: if any group has more than twice the length of another group, we should start worrying about homoscedasticity. If we turn our attention to the Honey portion of Figure 3, we can see the same kind of worry. Having a single group with different amount of variation is not a terminal violation, especially if we have a balanced design. However, we will want to proceed with caution.

A bigger issue for homoscedasticity is that of patterns in a strip chart. The most common pattern to look out for is a megaphone or funnel shape. We have a (right-opening) megaphone if the length of the strips get larger as you move from the left to the right along the horizontal axis. We have a funnel (left-opening megaphone) if the length of the strips gets smaller as you move from left to right. Keep an eye out for other patterns or shapes. These indicate deeper issues that result in violating the homoscedasticity assumption.

Turning back to Figure 3, we need to keep in mind that having only three strips makes judging patterns challenging. When I look at the Song Knowledge strip chart, I see a funnel shape. When I look at the Honey strip chart, I see a bow tie (wide on the outsides, narrow in the middle).

Taking the two pieces together (vertical spacing and patterns), I would say that for the Song Knowledge data, we do not satisfy homoscedasticity assumption. For the Honey study, we've violated the homoscedasticity assumption. In both cases, I would recommend either applying some type of variance-stabilizing transformation (see Table 6.3 in the Oehlert text) or using the nonparametric shortcut known as Kruskal-Wallis.

5.2.1 Your Turn

Try creating the strip chart for the Resin Lifetime study. How would you assess the Homoscedasticity assumption?

5.3 Assessing Independence of Observations

The last assumption is not only the most important but the hardest assumption to check. I want to stress that this assumption is about the **Independence of Observations**. There are many kinds of independence in Statistics (e.g., independence of attributes), thus you need to clearly articulate which kind of independence you're talking about.

5.3.1 Knowledge of Study Design and Sample

The first method we have available to us is not graphical or statistical in nature. Rather, we focus on what we know about the study design, how the data were collected, and the make up of the sample (i.e., the measurement units). For example, did we take precautions to draw a random sample for our measurement units? Did we take a convenience sample? Did we end up

getting a chunk of closely related family members? Think through all possible ways that we could end up with measurement units directly impacting each other and then check the study to see if any of those methods could have/did slip through our guards.

For the Song Knowledge study, while all students who were present that day took part, not everyone agreed to participate for our data. Further, we took a random sample stratified by year in school resulting in the 12 students in our sample. Given how we designed our study, we might cautiously argue that we have satisfied the assumption of Independent Observations.

5.3.2 Graphical and Statistical Methods

In order to use graphical and/or statistical methods, we must know the order in which measurements were taken or how they were arranged spatially. In the case of the Song Knowledge study, we do not have this information, thus we **can not** use these methods.

5.3.2.1 Index Plots For the Honey data, we know that the order of the values reflects the measurement order (provided you entered the data in the same way as I did). Since we know ordering, we can make use of a visualization known as an index plot (Figure 4).

```
# Demo code for index plots ----

## Honey study
ggplot(
  data = data.frame(
    residuals = honeyModel$residuals,
    index = 1:length(honeyModel$residuals)
  ),
  mapping = aes(x = index, y = residuals)
) +
  geom_point(size = 1.5) +
  geom_line() +
  theme_bw() +
  geom_hline(
    yintercept = 0,
    linetype = "dashed",
    color = "red"
  ) +
  xlab("Measurement order") +
  ylab("Residuals")
```

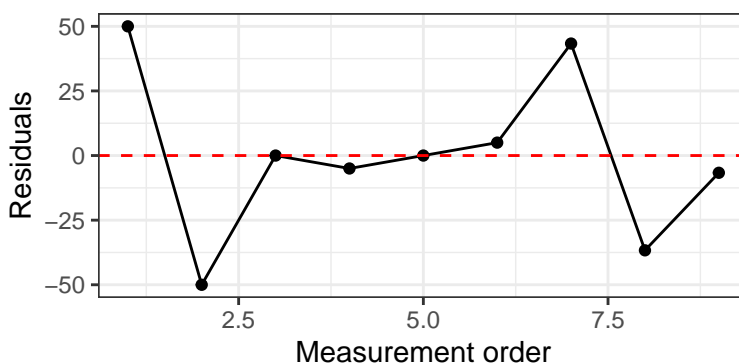


Figure 4: Index Plot for Honey Residuals

What we’re looking for is a pattern. A “cloud” of points or the lack of any pattern is what we’re hoping for. If instead we see something that reminds us of things we saw in college algebra, pre-calculus, trig, calculus, or another math classes, then we have a problem. For the Honey data, I do not see any indication of a pattern.

5.3.2.2 Durbin-Watson In addition to the index plots, we can also use the Durbin-Watson statistic to measure the level of autocorrelation (or serial correlation) in our data. Again, we must know the measurement order of our data.

If there is no autocorrelation (i.e., independent observations), then the DW statistic should be around 2. A **rough** rule of thumb is that DW values that are less than 1.5 or greater than 2.5 are causes for concern. However, the DW statistic is

sensitive to things such as long runs of the same treatment/group in the data. This means that there is a lot of wiggle room in this rule of thumb. Only use the DW statistic in conjunction with the index plot AND your knowledge of the study. Use `car::durbinWatsonTest(honeyModel)$dw` to get the value of the Durbin-Watson statistic.

For the Honey data, the value of the Durbin-Watson statistic is 2.57, which is fairly close to the 2.5 cutoff, so I'm not concerned about violations of the Independence of Observations assumptions.

5.3.3 Your Turn

Use the information in Oehlert about the Resin Lifetime example as well as fact that the data appear in measurement order in the DAT file to assess the assumption of Independent Observations. What do you come up with?

6 Conducting the Parametric Shortcut (ANOVA F Test)

There are a couple of things that we will want to make sure that we have before we actually look at the results of the ANOVA F test. First, we need to make a set of choices related to our decision rule. Second, we need to assess the assumptions of the shortcut.

6.1 Decision Rule

For this shortcut we will make use of a p -value decision rule. This is standard practice for ANOVA problems. However, we need to be sure that we've made two choices before we look at the results.

First, we need to be sure that we've set our chosen which Type I error rate we're going to use (more on this later) and set the Type I risk level, \mathcal{E}_I .

Second, we need to decide what our *Unusualness Threshold*, UT , is going to serve as the way to decide whether we reject or fail to reject the null hypothesis. The Unusualness Threshold (also called "Level of Significance") is the maximum percentage of the time we anticipate seeing "unusual events" given the null hypothesis. This is a probability value and we're free to choose the value provided that $UT \leq \mathcal{E}_I$. (There are some subtle distinctions between the ideas of Unusualness Threshold/Level of Significance and Type I risk that we won't get into.)

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

6.2 Assessment of Assumptions

When you go to make your final determination about whether the data meet the assumptions for the parametric shortcut, keep in mind whether we have a balanced design. Balanced designs give us a bit more flexibility to accommodate minor to moderate violations to the Gaussian and Homoscedasticity. While both the Song Knowledge study and the Honey study are balanced designs, I'm going to propose two different decisions. I'm going to state that the violations are too numerous for the Honey study to proceed. However, I'll cautiously proceed with the Song Knowledge study.

If you have been following along with the Resin data, I feel that we can proceed (cautiously) with the Resin Lifetimes data.

6.3 Quick Look Method

Technically, we've already done the parametric shortcut (ANOVA F test). When we used the `aov` command, R took the shortcut automatically. Thus, all we need to do is look at the results. If we are just looking for ourselves, the "quick look" method is great. However, this method is absolutely **TERRIBLE** for reports. To quickly look at the results you use either the `summary` or `anova` functions on your model objects (the outputs of the `aov` call):

```
# Demo code for quickly looking at results ----
## Summary example with Song Knowledge
summary(songModel)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Year          2  40.67   20.33   1.906  0.204
## Residuals     9  96.00   10.67
```

```
## Anova example with Resin Lifetime
anova(resinModel)
```

```
## Analysis of Variance Table
##
```

```
## Response: log10Lifetime
##           Df Sum Sq Mean Sq F value    Pr(>F)
## temp         4 3.5376 0.88441   96.363 < 2.2e-16 ***
## Residuals    32 0.2937 0.00918
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `anova` function is special wrapper of the `summary` function that adds just a bit more information about the design. However, we can do MUCH better.

6.4 Professional ANOVA Tables

There are two kinds of professional looking ANOVA tables. The first is a classical table that is essentially the output of the `summary/anova` call (sans the significance codes) in a table format. However, these are *lacking*. The modern ANOVA table incorporates effect sizes (practical significance) into the table along with statistical significance.

To make these modern tables, we will need the `parameters` package along with `knitr` and `kableExtra` packages.

6.4.1 Song Knowledge Study Example

```
# Demo Code for Professional, Modern ANOVA Tables ----
## Song Knowledge study
parameters::model_parameters(
  model = songModel,
  effectsize_type = c("eta", "omega", "epsilon") # Effect sizes
) %>%
  knitr::kable(
    digits = 4,
    col.names = c(
      "Source", "SS", "df", "MS", "F", "p-value",
      "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
    caption = "Modern ANOVA Table for Spring '23 Song Knowledge Study",
    booktabs = TRUE,
    align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("HOLD_position")
  )
```

Table 1: Modern ANOVA Table for Spring '23 Song Knowledge Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
Year	40.6667	2	20.3333	1.9063	0.2041	0.2976	0.1312	0.1415
Residuals	96.0000	9	10.6667					

6.4.1.1 Interpretation of Table 1 We can see that a STAT461 undergraduate's year in school accounts for about 1.1 times as much variation as the residuals in Table 1. Drawing upon the balanced design to grant some robustness to the heteroscedasticity, we would anticipate seeing an F ratio at least as large as this ~37% of the time we would repeat the experiment if there truly was no impact of year in school on quiz score. Since our p -value is greater than our Unusualness Threshold (UT), we have an event that is "usual" (or typical) under the null hypothesis model and we would fail to reject the null hypothesis. We will decide to act as if a student's year in school does not impact how well they do on a song knowledge quiz.

6.4.2 Resin Study Example

```
# Demo Code for Professional, Modern ANOVA Tables ----
## Resin Lifetime study
parameters::model_parameters(
```

```

model = resinModel,
effectsize_type = c("eta", "omega", "epsilon")
) %>%
knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "Modern ANOVA Table for Resin Lifetimes Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
) %>%
kableExtra::kable_styling(
  font_size = 10,
  latex_options = c("HOLD_position")
) %>%
kableExtra::footnote(
  general = "Computer rounding has made the p-value look like zero.",
  general_title = "Note. ",
  footnote_as_chunk = TRUE
)

```

Table 2: Modern ANOVA Table for Resin Lifetimes Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
temp	3.5376	4	0.8844	96.363	0	0.9233	0.9116	0.9138
Residuals	0.2937	32	0.0092					

Note. Computer rounding has made the p-value look like zero.

6.4.2.1 Interpretation of Table 2 The operating temperature accounts for ~ 96 times as much variation in the \log_{10} lifetimes for the resin as the residuals (see Table 2). If the null hypotheses were true, then observing such a large F ratio (or a value even larger) would be exceedingly rare. Since this percentage (i.e., the p -value) is less than our chosen Unusualness Threshold of 3%, we will declare that we have an unusual event given the null model. We take this as evidence against the null hypothesis and reject the null hypothesis that the temperature does not impact lifetime of the resin. From a practical standpoint, the operating temperature accounts for essentially 91% of all variation in lifetimes.

6.4.3 Rounding p -values

Notice that both Table 1 and Table 2 look much more professional than the raw output we got from using `summary` and `anova`. Further, by using this approach with the `model_parameters` function, we've gotten our three estimates of effect size for the model (i.e., practical significance). There is a downside in this approach: the p -value in Table 2 has been made to look equal to zero. We know that while p -values may be essentially equal to zero, they are never actually zero. We added a footnote to Table 2.

Alternatively, we could adopt the convention of reporting that the p -value is less than a given threshold. Part of my suite of ANOVA tools is the `pvalRound` function. This function takes an input p and the argument `digits` to determine whether the input is less than $10^{-\text{digits}}$. If so, the function returns a character string such as " < 0.0001 " for `digits = 4` (the default). If the input p is larger than the $10^{-\text{digits}}$, the function returns the input unchanged.

```

# Demo Code to fix p-value rounding issues ----
## Using pvalRound with Resin Lifetimes study
parameters::model_parameters(
  model = resinModel,
  effectsize_type = c("eta", "omega", "epsilon")
) %>%
dplyr::mutate(
  p = ifelse(
    test = is.na(p),
    yes = NA,

```

```

    no = pvalRound(p, digits = 4)
  )
) %>%
knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "ANOVA Table for Resin Lifetimes Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
) %>%
kableExtra::kable_styling(
  font_size = 10,
  latex_options = c("HOLD_position")
)

```

Table 3: ANOVA Table for Resin Lifetimes Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
temp	3.5376	4	0.8844	96.363	< 0.0001	0.9233	0.9116	0.9138
Residuals	0.2937	32	0.0092					

Either table (Table 2 or Table 3) is sufficient. You only need to worry about fixing a p -value looking like 0 IF you have a p -value that is sufficiently close to zero (i.e., less than 0.0001). While 0.0001 is a fairly common threshold, you could update the `pvalRounding` function to use another threshold such as 0.001. I do not recommend going smaller than 0.0001 (`digits = 4`).

7 Reporting Estimates

One the last things we can do related to the ANOVA omnibus analysis is to estimate the various parameters in our models. We can do this in two way: point and interval. Thinking back to HW #1.2, both of these approaches have their strengths and their weaknesses.

7.1 Point Estimation

Here's how we can get the point estimates for our Grand Mean and factor effects.

Make sure that you've told R to use the Sum to Zero constraint before proceeding.

To get these point estimates, we will use the `dummy.coef` function. To make the results look professional, we'll again make use the `knitr` and `kableExtra` packages.

```

# Demo making a profession table of coefficients/point estimates ----
## If you want to quickly look, just enter dummy.coef(songModel) into the console

## Song Knowledge study
pointEst <- dummy.coef(songModel)
# pointEst # Look at the output of pointEst so you know in what order the estimates appear
pointEst <- unlist(pointEst)
names(pointEst) <- c("Grand Mean", "Junior", "Senior",
  "Other")

data.frame("Estimate" = pointEst) %>%
  knitr::kable(
    digits = 2,
    caption = "Point Estimates from the Song Knowledge Study",
    booktabs = TRUE,
    align = "c"
  )

```

```

) %>%
kableExtra::kable_styling(
  font_size = 12,
  latex_options = c("HOLD_position")
)

```

Table 4: Point Estimates from the Song Knowledge Study

	Estimate
Grand Mean	4.67
Junior	2.33
Senior	-2.17
Other	-0.17

If you just call `dummy.coef(songModel)` in the console, you'll see a value labelled (**Intercept**); this is the value of the *Grand Sample Arithmetic Mean*. The other values should be labelled by factor name and level name. These point estimates are what will fill our various screens (plus the residuals).

Keep in mind that the estimates in Table 4 are *rates*. Thus, for the Song Knowledge study, we would interpret this value as 4.67 points per student; our entire sample accumulated 4.67 times as many points as sampled students.

We can also see the factor level (treatment) effects ($\hat{\alpha}_i$) estimates. For Seniors, they accumulated an additional -2.17 points per student where as the Juniors accumulated 2.33 points per student and the Others accumulated -0.17 points per student. This suggests that Juniors and Seniors perform better than baseline (*GSAM*).

7.1.1 Your Turn

Attempt to come up with the code that makes the following table for the Resin Lifetime study. Additionally, practice interpreting these point estimates.

Table 5: Point Estimates from the Resin Lifetime Study

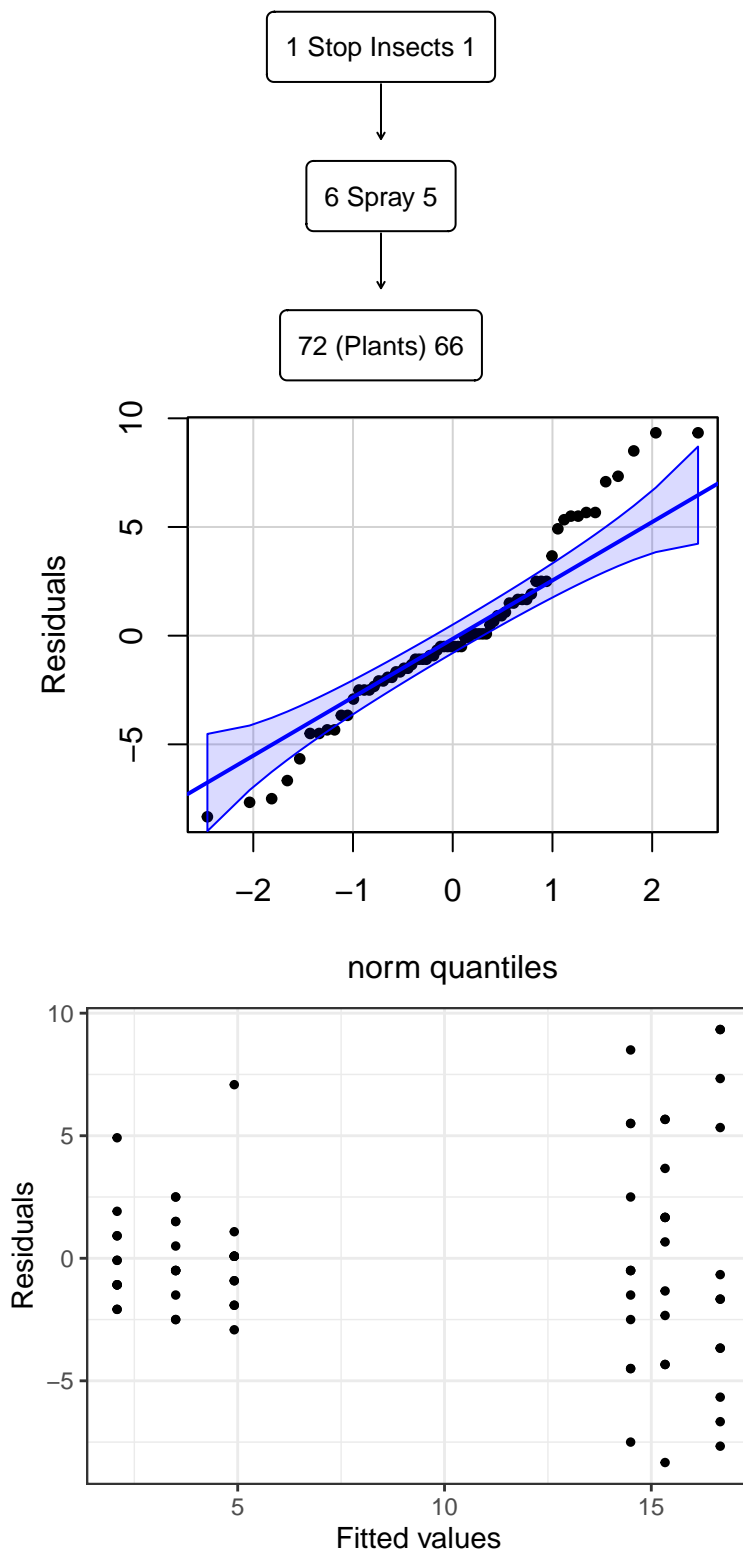
	Estimate
Grand Mean	1.44
175°C	0.49
194°C	0.19
213°F	-0.06
231°F	-0.24
250°F	-0.38

8 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")`.

A bit of background: 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 these data and build the elements that you would include a report as shown above. I'm going to provide several outputs, but no narrative. Use these as reminders for what to do. If you get stuck, check out the code appendix to see the code that I used.



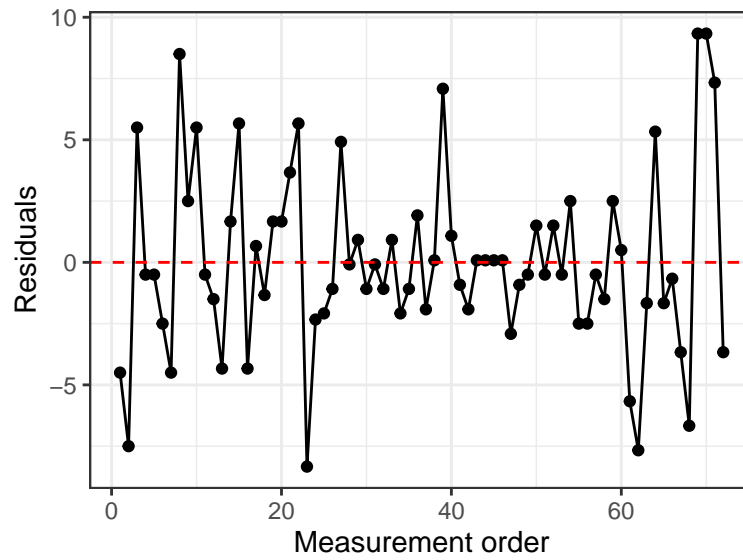


Table 6: ANOVA Table for Insect Spray Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
spray	2668.833	5	533.7667	34.7023	< 0.0001	0.7244	0.7006	0.7036
Residuals	1015.167	66	15.3813					

Table 7: Point Estimates from the Insect Spray Study

	Estimate
Grand Mean	9.50
Spray A	5.00
Spray B	5.83
Spray C	-7.42
Spray D	-4.58
Spray E	-6.00
Spray F	7.17

9 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", "parameters")
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.githubusercontent.com/neilhatfield/STAT461/main/rScripts/ANOVATools.R")

# Demo code for loading packages ----
packages <- c("tidyverse", "hasseDiagram", "knitr",
             "kableExtra", "car", "psych", "parameters")
lapply(packages, library, character.only = TRUE)

# Demo code for controlling table options ----
# Tell Knitr to use empty space instead of NA in printed tables
options(knitr.kable.NA = "")

## Demo code for setting the constraint ----
options(contrasts = c("contr.sum", "contr.poly"))

# Demo code for loading Neil's extra tools ----
source("https://raw.githubusercontent.com/neilhatfield/STAT461/main/rScripts/ANOVATools.R")

# Demo Code for Loading Data ----
## Song Data
songData <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/main/dataFiles/songKnowledge_Fa23.csv",
  header = TRUE,
  sep = ",",
)
#### Set year to an ordered factor
songData$Year <- factor(
  x = songData$Year,
  levels = c("Junior", "Senior", "Other")
)

## Honey Data--Manual Entry
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 (no particular order)
honey$Varietal <- as.factor(honey$Varietal)

## Resin Lifetimes Data
```

```

resin <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/main/dataFiles/resinLifetimes.dat",
  header = TRUE,
  sep = "" # Notice the change in separator
)
#### Set temp to factor
resin$temp <- as.factor(resin$temp)

### Change the name of the y column to something more meaningful
names(resin)[which(names(resin) == "y")] <- "log10Lifetime"

# Checking the first two base requirements
str(songData)
str(honey)
str(resin)

# Demo Code for a Hasse Diagram ----
## 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
)

# Demo code for building ANOVA models ----
## Song Knowledge study
songModel <- aov(
  formula = Score ~ Year,
  data = songData,
  na.action = "na.omit"
)

## Honey study
honeyModel <- aov(
  formula = Amount ~ Varietal,
  data = honey,
  na.action = "na.omit"
)

## Resin Lifetime study
resinModel <- aov(
  formula = log10Lifetime ~ temp,
  data = resin,
  na.action = "na.omit"
)

# Demo code for making QQ plots ----
## Chunk options for side-by-side plots
### fig.show="hold", out.width="100%", fig.height=3.5
## R option for side-by-side plots
par(mfrow = c(1,2), mar = c(4, 4, 0.1, 0.1))

## Song Knowledge study

```

```

car::qqPlot(
  x = songModel$residuals,
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (points)"
)

# Honey study
car::qqPlot(
  x = residuals(honeyModel),
  distribution = "norm",
  envelope = FALSE,
  id = TRUE,
  pch = 1,
  ylab = "Residuals (lbs)"
)

# Demo code for making strip charts ----
# Chunk options for side-by-side
### fig.subcap=c("Song Knowledge Study", "Honey Study"), fig.ncol=2,
### out.width="50%"
## There are no special par calls needed with this approach (only good for PDF outputs)

## Song Knowledge study
ggplot(
  data = data.frame(
    residuals = songModel$residuals,
    fitted = songModel$fitted.values
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  theme_bw() +
  xlab("Fitted values (points)") +
  ylab("Residuals (points)")

## Honey study
ggplot(
  data = data.frame(
    residuals = residuals(honeyModel),
    fitted = fitted(honeyModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  theme_bw() +
  xlab("Fitted values (lbs)") +
  ylab("Residuals (lbs)")

# Demo code for index plots ----

## Honey study
ggplot(
  data = data.frame(
    residuals = honeyModel$residuals,
    index = 1:length(honeyModel$residuals)
  ),

```

```

mapping = aes(x = index, y = residuals)
) +
geom_point(size = 1.5) +
geom_line() +
theme_bw() +
geom_hline(
  yintercept = 0,
  linetype = "dashed",
  color = "red"
) +
xlab("Measurement order") +
ylab("Residuals")
# Demo code for quickly looking at results ----
## Summary example with Song Knowledge
summary(songModel)

## Anova example with Resin Lifetime
anova(resinModel)

# Demo Code for Professional, Modern ANOVA Tables ----
## Song Knowledge study
parameters::model_parameters(
  model = songModel,
  effectsize_type = c("eta", "omega", "epsilon") # Effect sizes
) %>%
knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "Modern ANOVA Table for Spring '23 Song Knowledge Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
) %>%
kableExtra::kable_styling(
  font_size = 10,
  latex_options = c("HOLD_position")
)

# Demo Code for Professional, Modern ANOVA Tables ----
## Resin Lifetime study
parameters::model_parameters(
  model = resinModel,
  effectsize_type = c("eta", "omega", "epsilon")
) %>%
knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "Modern ANOVA Table for Resin Lifetimes Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
) %>%
kableExtra::kable_styling(
  font_size = 10,
  latex_options = c("HOLD_position")
) %>%
kableExtra::footnote(

```

```

    general = "Computer rounding has made the p-value look like zero.",
    general_title = "Note. ",
    footnote_as_chunk = TRUE
  )

# Demo Code to fix p-value rounding issues ----
## Using pvalRound with Resin Lifetimes study
parameters::model_parameters(
  model = resinModel,
  effectsize_type = c("eta", "omega", "epsilon")
) %>%
  dplyr::mutate(
    p = ifelse(
      test = is.na(p),
      yes = NA,
      no = pvalRound(p, digits = 4)
    )
  ) %>%
  knitr::kable(
    digits = 4,
    col.names = c(
      "Source", "SS", "df", "MS", "F", "p-value",
      "Eta Sq.", "Omega Sq.", "Epsilon Sq."
    ),
    caption = "ANOVA Table for Resin Lifetimes Study",
    booktabs = TRUE,
    align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("HOLD_position")
  )

# Demo making a profession table of coefficients/point estimates ----
## If you want to quickly look, just enter dummy.coef(songModel) into the console

## Song Knowledge study
pointEst <- dummy.coef(songModel)
# pointEst # Look at the output of pointEst so you know in what order the estimates appear
pointEst <- unlist(pointEst)
names(pointEst) <- c("Grand Mean", "Junior", "Senior",
  "Other")

data.frame("Estimate" = pointEst) %>%
  knitr::kable(
    digits = 2,
    caption = "Point Estimates from the Song Knowledge Study",
    booktabs = TRUE,
    align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo making a profession table of coefficients/point estimates

# Resin Lifetime Data
pointEst <- dummy.coef(resinModel)
pointEst <- unlist(pointEst)

```

```

names(pointEst) <- c("Grand Mean", "175°C", "194°C",
                    "213°F", "231°F", "250°F")

data.frame("Estimate" = pointEst) %>%
  knitr::kable(
    digits = 2,
    caption = "Point Estimates from the Resin Lifetime Study",
    booktabs = TRUE,
    align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Putting Everything Together--Your Turn Code-----

# Hasse Diagram for Insect Spray Study
modellLabels <- 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 = modellLabels
)

# Load Data
data("InsectSprays")

# Fit Model
isModel <- aov(
  formula = count ~ spray,
  data = InsectSprays
)

# Assess Gaussian Assumption with a QQ Plot
car::qqPlot(
  x = isModel$residuals,
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals"
)

# Supplement with Skewness and Kurtosis
isRSkew <- psych::skew(isModel$residuals)
isRKurt <- psych::kurtosi(isModel$residuals)

# Use a strip plot to assess homoscedasticity
ggplot(
  data = data.frame(
    residuals = isModel$residuals,
    fitted = isModel$fitted.values
  ),

```

```

mapping = aes(x = fitted, y = residuals)
) +
geom_point(size = 1) +
theme_bw() +
xlab("Fitted values") +
ylab("Residuals")

# RED HERRING
# We don't know measurement order so this plot does not actually
# provide any assistance to us.

# Index Plot for Resin Residuals
ggplot(
  data = data.frame(
    residuals = isModel$residuals,
    index = 1:length(isModel$residuals)
  ),
  mapping = aes(x = index, y = residuals)
) +
geom_point(size = 1.5) +
geom_line() +
theme_bw() +
geom_hline(
  yintercept = 0,
  linetype = "dashed",
  color = "red"
) +
xlab("Measurement order") +
ylab("Residuals")

dw <- car::durbinWatsonTest(resinModel)$dw

# Insect Sprays ANOVA Table
parameters::model_parameters(
  model = isModel,
  effectsize_type = c("eta", "omega", "epsilon")
) %>%
dplyr::mutate(
  p = ifelse(
    test = is.na(p),
    yes = NA,
    no = pvalRound(p)
  )
) %>%
knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "ANOVA Table for Insect Spray Study",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
) %>%
kableExtra::kable_styling(
  font_size = 12,
  latex_options = c("HOLD_position")
)

# Point Estimates for Insect Sprays

```

```

pointEst <- dummy.coef(isModel)
pointEst <- unlist(pointEst)
names(pointEst) <- c("Grand Mean", "Spray A", "Spray B",
                    "Spray C", "Spray D", "Spray E", "Spray F")

data.frame("Estimate" = pointEst) %>%
  knitr::kable(
    digits = 2,
    caption = "Point Estimates from the Insect Spray Study",
    booktabs = TRUE,
    align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```