Parametric Shortcut: One-way ANOVA

Neil J. Hatfield

3/15/2021

In this tutorial, we are going to explore using R to fit a One-way ANOVA model to two data sets using the parametric shortcut known as the One-way ANOVA F Test.

I'll be interweaving a couple of different examples throughout the text. The general structure to this tutorial is:

- Setting up `R` and Reading in Data
- Explore Your Data
- Discussion on whether ANOVA methods are appropriate
- Fitting the ANOVA Model
- Assessing the Assumptions of the Parametric Shortcut for One-way ANOVA
- Conducting the Omnibus Test
  - Quick Look at Results
  - Making Professional Looking Tables
  - Interpreting Results
- Reporting Point Estimates
  - Formatting
  - Interpretations
- Post Hoc Analysis
  - Choice of Type I Error Rate
  - Recommended Default (Tukey-Kramer HSD)
  - Other Approaches
  - Special Comparisons
  - Effect Sizes

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.

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`, `parameters`, `DescTools`, and `multcompView`. 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:

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

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.

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:

```r
# 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.

Set Our Constraints

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:

```r
## 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.

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:

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

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

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:

```r
# 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")
)

# 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.github.com/neilhatfield/STAT461/master/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"
```

Explore Your Data

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

Is ANOVA Even Appropriate?

Recall the base requirements for One-way ANOVA are:

- you are working with a qualitative/categorical factor,
- you are working with a quantitative response,
- you are working with an additive model,
- you have estimable effects, and
- you have estimable errors/residuals.

The first two requirements you can check quickly in `R` by using the `str` 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 but is just for you. In essence, this check to make sure `R` is thinking about your data correctly.

```
# Checking the first two base requirements
str(songData)
```

```
## 'data.frame':    15 obs. of  2 variables:
##  $ year : Factor w/ 3 levels "sophomore","junior",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ score: int  2 9 8 5 10 10 12 9 5 9 ...
```

```
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 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.

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.

```r
# 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
)
```

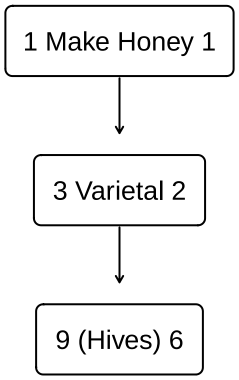


Figure 1: Hasse Diagram for Honey Study

Your Turn

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

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 trust in what the model results are.

To fit our ANOVA model, we will primarily use the `aov` function that is part of base R. 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:

```r
# 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: `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.)

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. 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 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 to help us. (All of which can go into your reports.)

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:

```r
# Demo code for making QQ plots
## Configure 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"
)

# QQ plot for Honey Residuals
car::qqPlot(
  x = residuals(honeyModel),
  distribution = "norm",
  envelope = FALSE,
  id = TRUE,
  pch = 19,
  ylab = "Residuals"
)
```

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

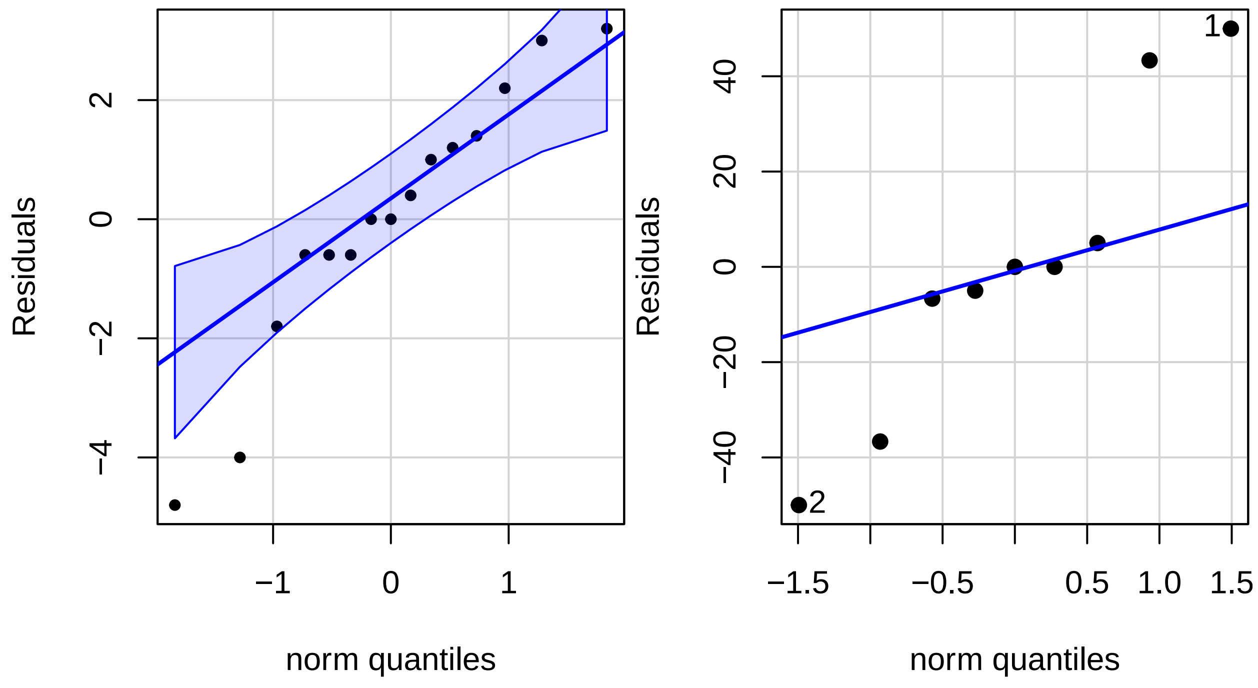


Figure 2: QQ Plot Residuals for Song Knowledge (left) and Honey (right) Studies

You'll notice that in the `qqPlot` function there are several arguments that allow you to customize the plots. The two arguments that are required 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. 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` that appears just above the plots. This is consequence of setting `id = TRUE`. 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. Finally, the `ylab` is a way to make the label for the vertical axis a bit nicer. You can use `ylab = "Residuals"` for your plots.

The `envelope` argument is particuarly useful when assessing the Gaussian assumption. To turn off the envelope you use `envelope = FALSE`. To add the envelope you used `envelope = num` where num is your 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 conern. 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 Guassian assumption.

For the Song Knowledge QQ Plot (left of Figure 2), we can see that 2 residuals are outside the envelope; 2 is close enough to 10% of 15 (i.e., 1.5) for us to proceed (perhaps cautiously) with the Gaussian assumption being 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. This suggests that our residuals do not follow a Gaussian distribution.

Your Turn

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

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.

For the Song Knowledge context, the value of *Sample Skewness* is -0.6 and the value of *Sample Excess Kurtosis* is -1.17. These values are questionable but when partnered with the QQ plot and our knowledge of the study design, they are cautiously acceptable.

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 satisified the Gaussian assumption with the honey data.

There is no hard and fast rule for assessing the Gaussian assumption.

Assessing Homoscedastcitiy

```
# DEMO CODE

# Strip chart for Honey Residuals
ggplot(
  data = data.frame(
    residuals = honeyModel$residuals,
    fitted = honeyModel$fitted.values
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 1) +
  theme_bw() +
  xlab("Fitted values (lbs)") +
  ylab("Residuals (lbs)")
```

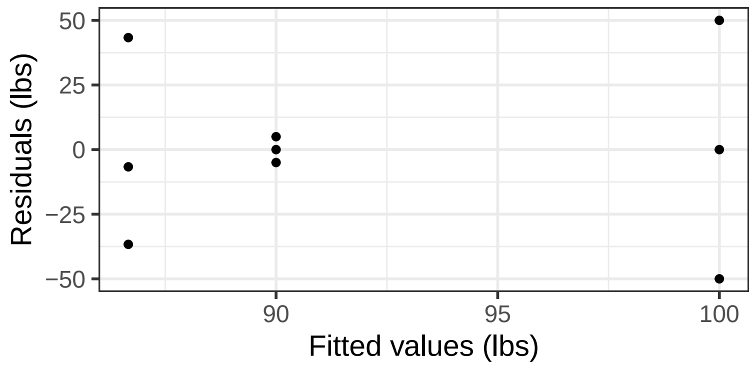


Figure 3: Stripchart for Honey Residuals

In Figure 3, we can see that one strip (located at a fitted value of 90) is rather close together. The vertical space that group uses is well under half the vertical space used by any other strip. This along with the essential bow-tie shape to the plot suggest that we do not have the same level of variation within each level of treatment.

Assessing Independence of Observations

In order to truly assess whether our observations are truly independent, we must know something about the order in which measurements were taken (or how they were arranged spatially).

For the honey data, we know that the order of the values reflects the measurement order. Thus, we can use an index plot.

```
# DEMO CODE

# Index Plot for Honey Residuals
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")
```

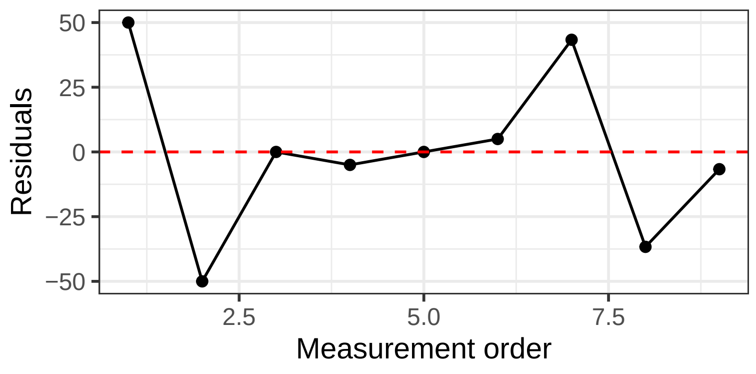


Figure 4: Index Plot for Honey Residuals

Given the order that measurements were made, there does not appear to be any strong indications of autocorrelation in Figure 4. If there was an issue, we would expect to see some patterning and/or clusters of values. Further, the Durbin-Watson statistic has a value of 2.57 which is fairly close to the 2.5 Rule of Thumb cutoff.

Use `car::durbinWatsonTest(honeyModel)$dw` to get the value of the Durbin-Watson statistic.

Conduct the Test

Since two of the assumptions are violated (and fairly badly), conducting the significance test is not recommended without first taking corrective actions. Thus, we will **NOT** use the parametric shortcut with the Honey Study data.

Example 2-Resin Lifetimes

For background on this example, please see Example 3.2 (page 32) in the Oehlert text.

The One-way ANOVA designs end up working well in this situation.

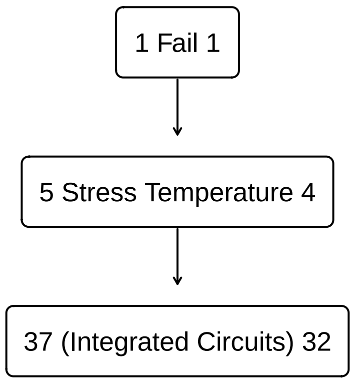


Figure 5: Hasse Diagram for Resin Study

Figure 5 shows that our factor has 5 (categorized) levels, an additive model, and that we have sufficient *Degrees of Freedom* to estimate all of our terms. Thus a One-way method is appropriate.

Assessing Assumptions

Again, we will need to first fit the One-way model to our data in order to assess the assumptions of the parametric shortcut.

We will now use `resinModel$residuals` to access the residuals from the model.

Assessing the Gaussian Assumption

We will turn to the QQ plot of the residuals to assess this assumption.

We have several residuals falling outside of the 90% confidence envelope (see Figure 6); approximately 35.1% of the residuals are outside. This seems high. The values of the *Sample Skewness* (-0.15) and *Sample Excess Kurtosis* (0.45) are not too far off what we would want to see (i.e., 0). We might say that the Gaussian assumption is "Questionable" and see what the other assumptions show us.

Assessing Homoscedastcitiy

When examining Figure 7, the second vertical strip from the left is the strip to compare with the others as this uses the least amount of vertical space. Comparing this strip to the first and fourth strips (from the left), revels that these groups use just about twice the vertical space as our reference. This puts us just on the cusp of violation of the Homoscedasticity Assumption.

Assessing Independence of Observations

Supposing that our resin data appear in measurement order, we can look at an index plot for checking independence of observations.

Figure 8 does not show any pattern that would suggest a violation of this assumption. Additionally, the Durbin-Watson statistic has a value of 2.37.

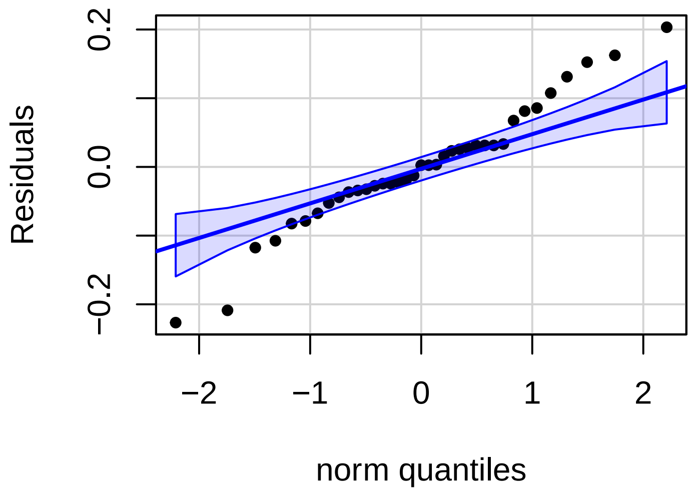


Figure 6: QQ Plot of Resin Lifetimes Residuals

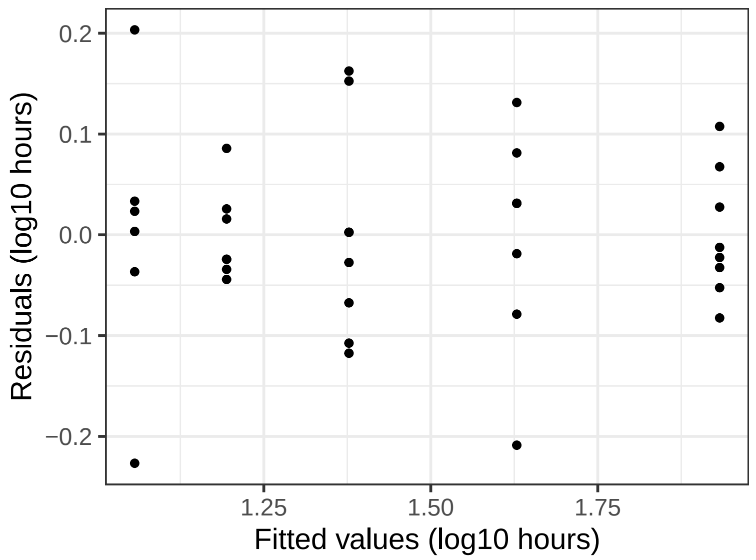


Figure 7: Stripchart for Resin Lifetimes Residuals

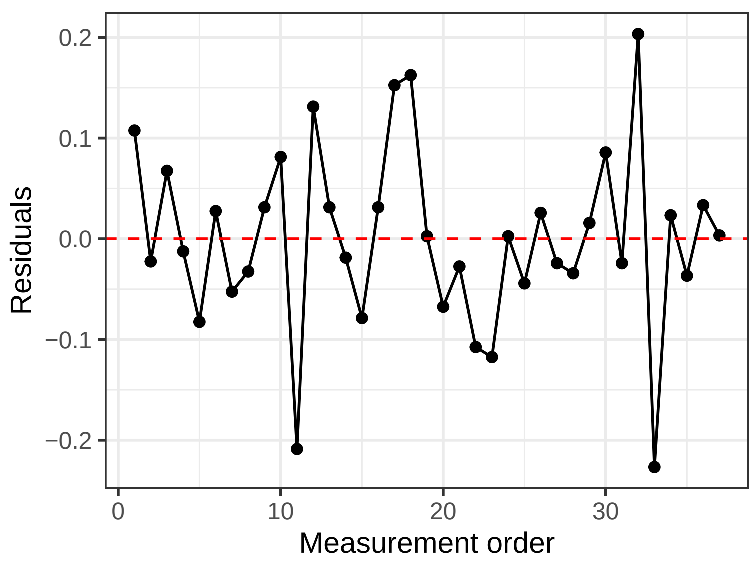


Figure 8: Index Plot for Resin Lifetimes Residuals

Conduct the Test

In this situation, we might decide to proceed with the parametric shortcut, albeit cautiously. Remember, the parametric shortcut is robust to moderate and minor violations of assumptions.

Before we go any further, we need to set our Unusualness Threshold (level of significance). Don't be boring and use $UT = 0.05$; think about how often something has to occur in order for you say "Hey, that's unusual." For me, the break between usual and unusual occurs around 3%; thus, I'm going to use $UT = 0.03$.

There are a couple of ways that you can see the results of the test. When we fit the model, we actually gave R everything necessary to do the test. We just now need to see the results.

Quick Look Method

The quick look method is great when you are just looking for yourself but TERRIBLE when you are going to write a report. You simply use the `summary` or `anova` functions to get the raw output as shown here:

```r
# DEMO CODE
# Displaying the results of the Parametric Shortcut via summary
summary(resinModel)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## temp         4  3.538  0.8844   96.36 <2e-16 ***
## Residuals   32  0.294  0.0092
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# DEMO CODE
# Displaying the results of the Parametric Shortcut via anova
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 formatting to the output. However, we can do MUCH better.

Professional Looking ANOVA Tables

We are going to combine professional looking tables from the Descriptive Statistics guide along with the notion of Modern ANOVA tables so that we get both statistical and practical significance information.

To get the effect sizes, we will make use of the `parameters` package's `model_parameters` function.

```r
# DEMO CODE

# Create professional looking modern ANOVA table
parameters::model_parameters(
  model = resinModel,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Omega Sq.", "Eta Sq.", "Epslion Sq."),
  caption = "ANOVA Table for Resin Lifetimes Study",
  format = "latex",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("scale_down", "HOLD_position")
  ) %>%
  kableExtra::footnote(
    general = "Computer rounding has made the p-value look like zero.",
    general_title = "Note. ",
    footnote_as_chunk = TRUE
  )
```

Table 1: ANOVA Table for Resin Lifetimes Study

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

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

Notice that Table 1 looks 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.

There is a downside in this approach: the p-value has been made to look equal to zero. We know that while p-values may be essentially equal to zero, they are not actually zero. We added a footnote to Table 1. However, we could attempt to fix this issue using the following function:

```r
# Set up the p-value rounding function.
pvalRound <- function(x){
  if (x < 0.0001) {
    return("< 0.0001")
  } else {
    return(x)
  }
}
```

```r
# DEMO CODE

# Professional looking modern ANOVA table with fixed p-value
parameters::model_parameters(
  model = resinModel,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  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",
    "Omega Sq.", "Eta Sq.", "Epslion Sq."),
  caption = "ANOVA Table for Resin Lifetimes Study",
  format = "latex",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("scale_down", "HOLD_position")
  )
```

Table 2: ANOVA Table for Resin Lifetimes Study

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

Either table is sufficient; you only need to worry about this IF you have a p-value that is sufficiently close to zero. You can then proceed with providing interpretations of F, the p-value, the effect sizes, and make a decision.

Reporting Point Estimates

One of the last things that we're going to do here is get the point estimates for our Grand Mean and treatment 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:

```
# DEMO CODE

# Getting coefficients/point estimates
dummy.coef(resinModel)
```

```
## Full coefficients are
##
## (Intercept):        1.43794
## temp:                   175         194         213         231         250
##                  0.49455952  0.19080952 -0.06044048 -0.24365476 -0.38127381
```

A couple of things to note: these values are slightly different from what Oehlert tabulated in Example 3.5 (p. 42). However, if you add the Grand Mean value (i.e., "(Intercept)") and the estimate for each treatment effect, you will get to the values of $\widehat{\mu_i}$ in Example 3.5 as well as the listed mean values in Listings 3.1 (p. 50) and 3.2 (p. 51). (I have programmed the homework answers based on R's evaluations.)

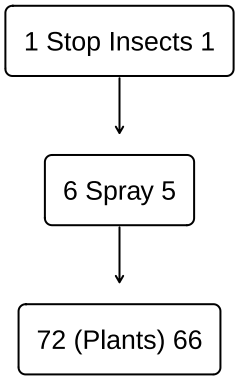
You should be prepared to interpret these estimates as well as format them in a professional layout. Note: `dummy.coef` will return a list of two elements: the grand mean and the treatment effects.

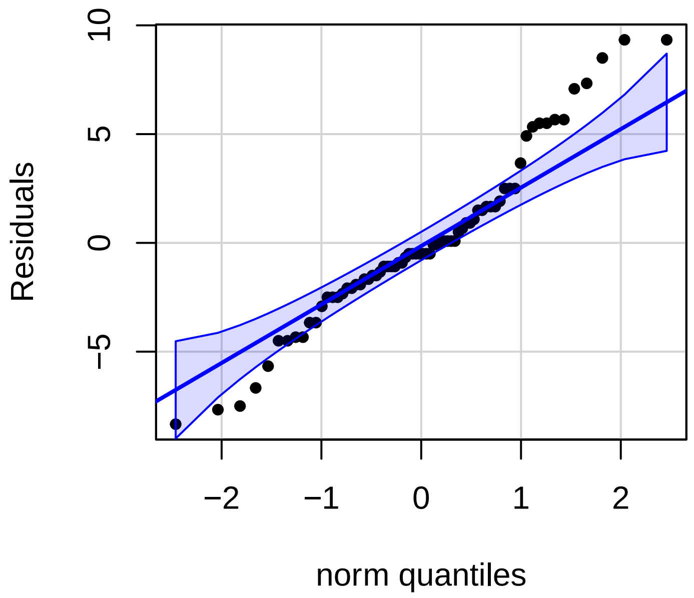
Your Turn

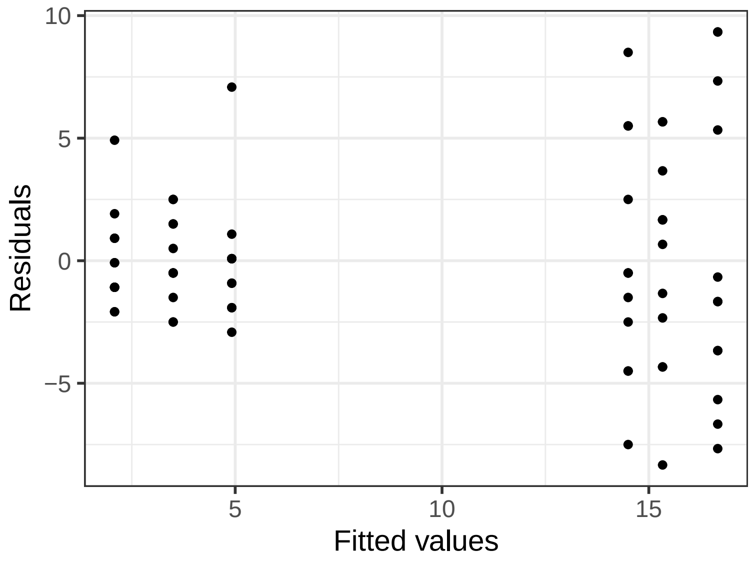
Now is an opportunity for you to get some practice. I'm going to use the data frame `InsectSprays`, which is built into R. You may load this data into your session with the command `data("InsectSprays")`.

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.




Residuals
10
5
0
−5
−2
−1
0
1
2
norm quantiles


Residuals
10
5
0
−5
5
10
15
Fitted values

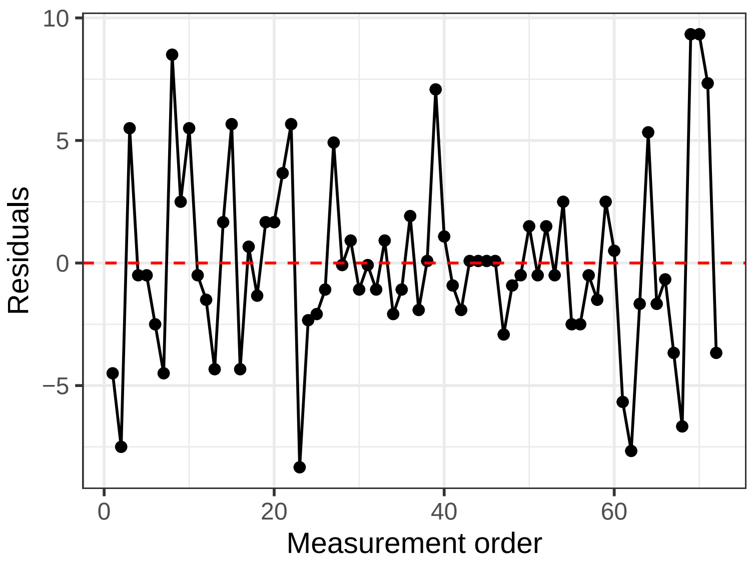


Table 3: ANOVA Table for Insect Spray Study

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

Table 4: 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

Code Appendix

```r
# 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", "DescTools", "multcompView")
lapply(packages, library, character.only = TRUE)

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

# Set constratint
options(contrasts = c("contr.sum", "contr.poly"))

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

# Demo code for loading packages for the tutorial
packages <- c("tidyverse", "hasseDiagram", "knitr",
              "kableExtra", "car", "psych",
              "parameters", "DescTools", "multcompView")
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.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")

# 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")
)

# Honey Data--Manual Entry
```

```r
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.github.com/neilhatfield/STAT461/master/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

# 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
)

# 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"
)
```

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

# Demo code for making QQ plots
## Configure 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"
)

# QQ plot for Honey Residuals
car::qqPlot(
  x = residuals(honeyModel),
  distribution = "norm",
  envelope = FALSE,
  id = TRUE,
  pch = 19,
  ylab = "Residuals"
)

# DEMO CODE

# Strip chart for Honey Residuals
ggplot(
  data = data.frame(
    residuals = honeyModel$residuals,
    fitted = honeyModel$fitted.values
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 1) +
  theme_bw() +
  xlab("Fitted values (lbs)") +
  ylab("Residuals (lbs)")

# DEMO CODE

# Index Plot for Honey Residuals
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")
# Hasse Diagram from Resin Study
modelLabels <- c("1 Fail 1", "5 Stress Temperature 4", "37 (Integrated Circuits) 32")
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
)

# Fit our One-way ANOVA model to the honey data
resinModel <- aov(
  formula = log10Lifetime ~ temp,
  data = resin,
  na.action = "na.omit"
)

# QQ plot for Resin Lifetime Residuals
car::qqPlot(
  x = resinModel$residuals,
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals"
)

# Strip chart for Resin Lifetime Residuals
ggplot(
  data = data.frame(
    residuals = resinModel$residuals,
    fitted = resinModel$fitted.values
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 1) +
  theme_bw() +
  xlab("Fitted values (log10 hours)") +
```

```r
  ylab("Residuals (log10 hours)")

# Index Plot for Resin Residuals
ggplot(
  data = data.frame(
    residuals = resinModel$residuals,
    index = 1:length(resinModel$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
# Displaying the results of the Parametric Shortcut via summary
summary(resinModel)

# DEMO CODE
# Displaying the results of the Parametric Shortcut via anova
anova(resinModel)

# DEMO CODE

# Create professional looking modern ANOVA table
parameters::model_parameters(
  model = resinModel,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  knitr::kable(
  digits = 4,
  col.names = c(
    "Source", "SS", "df", "MS", "F", "p-value",
    "Omega Sq.", "Eta Sq.", "Epslion Sq."),
  caption = "ANOVA Table for Resin Lifetimes Study",
  format = "latex",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("scale_down", "HOLD_position")
  ) %>%
  kableExtra::footnote(
    general = "Computer rounding has made the p-value look like zero.",
    general_title = "Note. ",
    footnote_as_chunk = TRUE
  )

# Set up the p-value rounding function.
pvalRound <- function(x){
  if (x < 0.0001) {
    return("< 0.0001")
  } else {
    return(x)
  }
}

# DEMO CODE

# Professional looking modern ANOVA table with fixed p-value
parameters::model_parameters(
  model = resinModel,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  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",
    "Omega Sq.", "Eta Sq.", "Epslion Sq."),
  caption = "ANOVA Table for Resin Lifetimes Study",
  format = "latex",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 10,
    latex_options = c("scale_down", "HOLD_position")
  )

# DEMO CODE

# Getting coefficients/point estimates
dummy.coef(resinModel)

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

# Hasse Diagram for Insect Spray Study
modelLabels <- c("1 Stop Insects 1", "6 Spray 5", "72 (Plants) 66")
```

```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
)

# 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,
  omega_squared = "raw",
  eta_squared = "raw",
  epsilon_squared = "raw"
) %>%
  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",
    "Omega Sq.", "Eta Sq.", "Epslion Sq."
    ),
  caption = "ANOVA Table for Insect Spray Study",
  format = "latex",
  booktabs = TRUE,
  align = c("l", rep("c", 8))
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )

# Point Estimates for Insect Sprays
pEst <- dummy.coef(isModel)
pEst <- unlist(pEst)
names(pEst) <- c("Grand Mean", "Spray A", "Spray B",
                 "Spray C", "Spray D", "Spray E", "Spray F")

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