Factorial Models

Neil J. Hatfield

4/15/2022

In this tutorial, we are going to explore Factorial Treatment Structures/Factorial Designs in R. For our purposes here, we will restrict our attention to [full] factorial models with all Fixed Factor Effects. (We will allow for our measurement units to be the only random effect term.) The general structure for this guide will be:

- Setting Up R
  - Loading Packages, Setting Options, and Additional Tools
- Data Contexts
  - Load! Aim! Ready! Release! (Gummy Bear Catapult)
  - Battery Manufacturing
- Exploring the Factorial Treatment Structure
- Fit the Models
  - Appropriateness of ANOVA
  - Check Interactions
  - Form the Model
- Assessing Assumptions
  - Gaussian Residuals
  - Homoscedasticity
  - Independence of Observations
- Results
  - Omnibus
  - Point Estimates
  - Post Hoc–Pairwise and Effect Sizes
  - Post Hoc–Contrasts
- Dealing with Imbalanced Designs

Setting Up R

Just as in the prior guides/tutorials, we have to first ensure that `R` is properly configured and prepared for our work. We will want to ensure that we load all of the appropriate packages, set our constraint, and load in any additional tools.

I've also added in the `psych` package to demonstrate an approach you can use for getting values of descriptive statistics in accordance with the factorial treatment structure. You'll also see the `openxlsx` package in the list; this happens to be my preferred way to read in XLSX files. Feel free to use your own method for reading in Excel files.

As a reminder, the following code does all of these things:

```
# Demo code to set up R
## Load packages
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "hasseDiagram", "car",
              "psych", "DescTools", "emmeans", "openxlsx")
lapply(packages, library, character.only = TRUE)
```

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

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

Data Contexts

For this guide/tutorial, we'll make use of two contexts: **Load! Aim! Ready! Release! (Gummy Bears)** and **Battery Manufacturing (Batteries)**.

Load! Aim! Ready! Release! (Gummy Bears)

This data comes from the design that we put together in class to explore the impacts of launch angle and launch position for our spoon-apults for launching gummy bears to see how far they will fly.

```r
# Demo code for loading and cleaning data
## Loading gummy bear data
catapultData <- openxlsx::readWorkbook(
  xlsxFile = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/gummyBears2022.xlsx",
  sheet = 1,
  colNames = TRUE,
  rowNames = FALSE
)

## Change variables
names(catapultData)[which(names(catapultData) == "Launch.Angle")] <- "Angle"
names(catapultData)[which(names(catapultData) == "Launch.Position")] <- "Position"
names(catapultData)[which(names(catapultData) == "Measurement.Order")] <- "Order"

## Fix the data
## R will treat high and High as TWO separate levels
## Same with low & Low, back & Back, and front & Front
catapultData <- catapultData %>%
  dplyr::mutate(
    Angle = dplyr::recode_factor(
      Angle,
      "high" = "High", # Old Value = New Value
      "High" = "High",
      "low" = "Low",
      "Low" = "Low"
    ),
    Position = dplyr::recode_factor(
      Position,
      "back" = "Back",
      "Back" = "Back",
      "front" = "Front",
      "Front" = "Front"
    ),
    Team = dplyr::case_when(
      Team == "Team 6" ~ Team,
      TRUE ~ paste("Team", Team)
    )
  )
```

Notice that I used a couple of functions from `dplyr` to help me clean and take control of the data. The `mutate` function allows you to change an entire column of a data frame in a consistent and relatively speedy way. The `recode_factor` function combines two things: alter values for consistency AND tell `R` to treat the attribute as a factor. Note: the order you list the new values will be the order `R` uses for the factor levels.

Omnibus SRQs and Hypotheses

Unlike in One-way ANOVA or RCBD with One Factor, we have several SRQs for factorial designs:

- Does the launch angle make a difference on how far a gummy bear flew?
  - $H_{1,0}$: There is no statistically significant impact on how far a gummy bear flies due to launch angle.
  - $H_{1,A}$: There is a statistically significant impact on how far a gummy bear flies due to launch angle.
- Does the launch position make a difference on how far a gummy bear flew?
  - $H_{2,0}$: There is no statistically significant impact on how far a gummy bear flies due to launch position.
  - $H_{2,A}$: There is a statistically significant impact on how far a gummy bear flies due to launch position.
- Does the interaction of launch angle and position make a difference on how far a gummy bear flew?
  - $H_{3,0}$: There is no statistically significant interaction effect on how far a gummy bear flies between launch angle and position.
  - $H_{3,A}$: There is a statistically significant interaction effect on how far a gummy bear flies between launch angle and position.

Battery Manufacturing

An engineer is designing a battery for use in a device that will people will use in some extreme temperatures. Unfortunately, the engineer may only alter one design parameter: the plate material for the battery of which he has three choices.

The device his batteries are for gets manufactured separately and is then shipped to the field, where the engineer has no control over the temperature the device will encounter. His experiences lead him to believe that environmental temperature will affect the battery life. He can control the temperature in the lab for product development testing.

He decides to test all three plate materials at three temperature levels—15ºF, 70ºF, and 125ºF—as these temperatures are consistent with reported end-use environments.

His questions:

1) What effects do material type and temperature have on life of battery?
2) Is there a choice of material that would give uniformly long life regardless of temperature?

```
## Demo code for loading and cleaning data
## Load battery data
batteryData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/batteryLife.dat",
  header = TRUE,
  sep = ","
)

## Clean data
batteryData$temperature <- dplyr::recode_factor(
  batteryData$temperature,
  `15` = "15ºF",
  `70` = "70ºF",
  `125` = "125ºF"
)
batteryData$material <- dplyr::recode_factor(
  batteryData$material,
  `1` = "Plate 1",
  `2` = "Plate 2",
  `3` = "Plate 3"
)
```

Your Turn

Write SRQs and hypotheses for the Battery Manufacturing study.

Exploring the Factorial Treatment Structure

Exploring the data in factorial settings becomes much more important as now you have many more ways to think about slicing up the data resulting in more ways to help people (and yourself) think about the data. Remember, data visualizations are some of your strongest and most helpful tools here.

Box Plots Revisited

You can use the multiple factors in a variety of ways in your data visualizations. For example, rather than looking at a side-by-side box plots along one factor, you could do a set for each factor or by the interaction. R's base `boxplot` function allows for you explore interactions by using the formula argument.

```
# Demo code for box plots using factorial treatment structure
## Battery Manufacturing
boxplot(
  formula = life ~ temperature:material,
  data = batteryData,
  ylab = "Life (hrs)",
  xlab = "Temp (ºF) x Material"
)
```

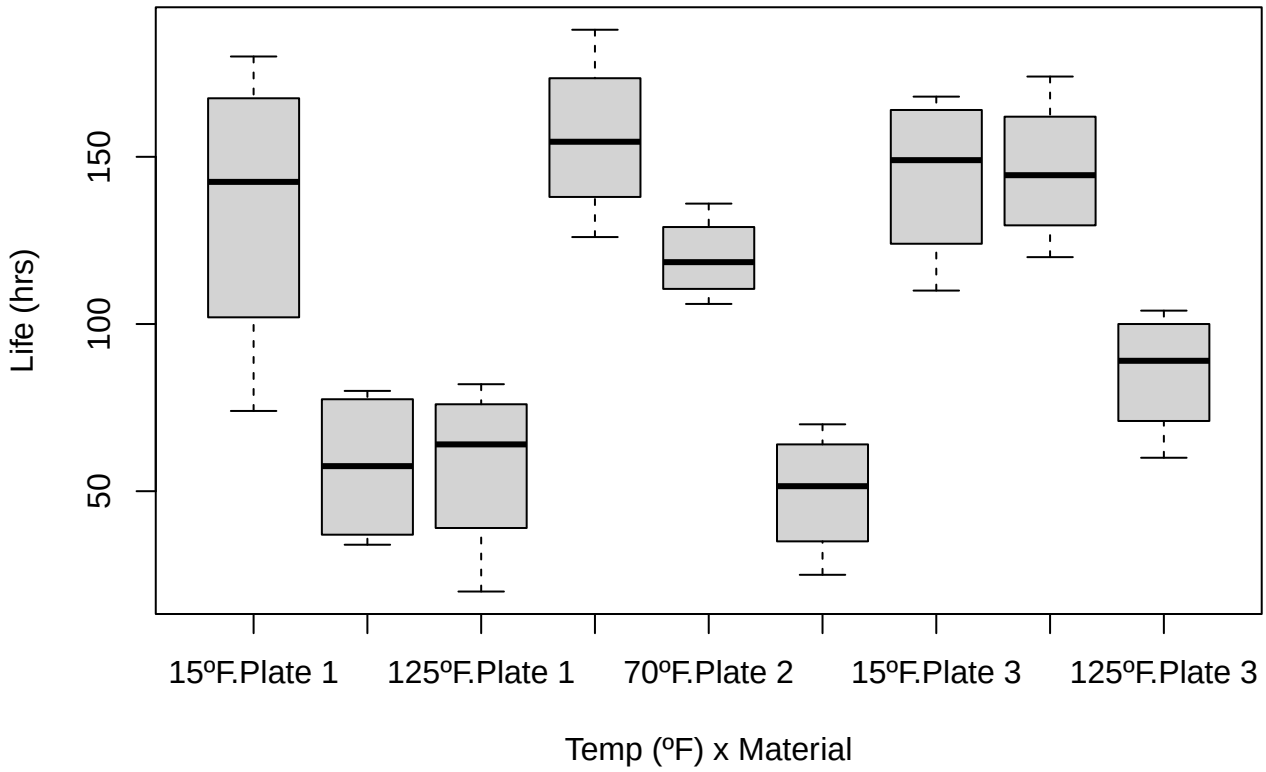

Figure 1: Box Plot of Battery Life Spans by Temperature and Plate Material

While this box plot is okay to look at, we could improve this plot greatly for professional work. The easiest method would be to use `ggplot2`.

```r
# Demo code of box plots incorporating factorial treatment structure
## ggplot2 and Gummy Bears Study
ggplot(
  data = catapultData,
  mapping = aes(
    x = Angle,
    y = Distance,
    fill = Position
  )
) +
  geom_boxplot() +
  theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (in)") +
  labs(
    fill = "Launch Position"
  ) +
  theme(
    legend.position = "right",
    text = element_text(size = 14)
  )
```

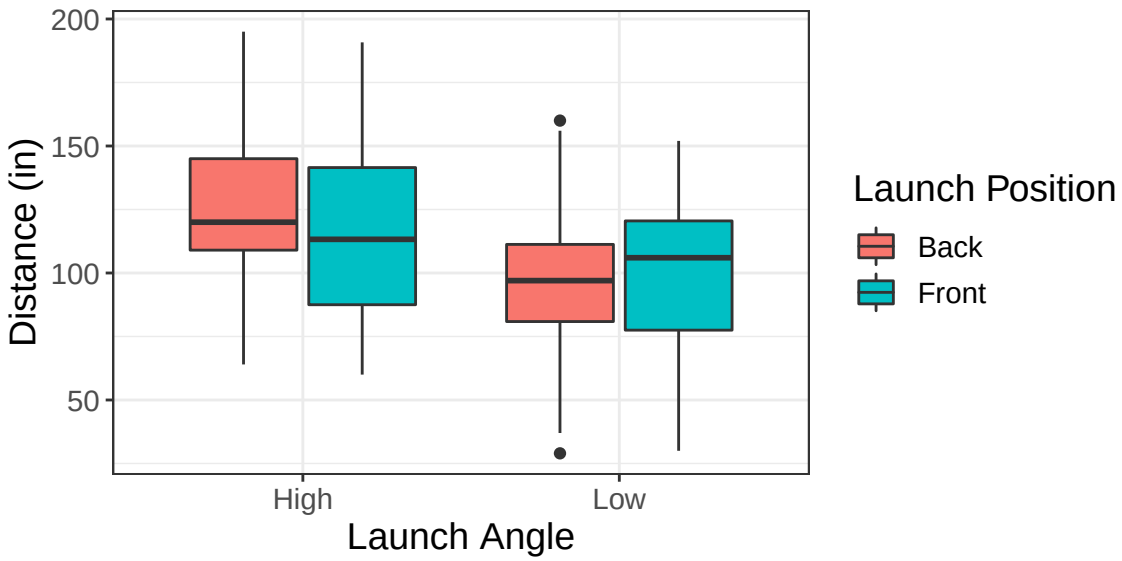

Figure 2: Box Plot With Multiple Factors–Gummy Bears Study

Descriptive Statistics

In addition to data visualizations, we also may make use of descriptive/incisive statistics. We've used the `describeBy` from the `psych` package in the past to break our response up into groups based upon our factor. We can do something similar in multi-factor situations as shown here:

```r
# Demo code for descriptive statistics for factorial treatment structure
## Battery Manufacturing
batteryStats <- psych::describeBy(
  x = batteryData$life,
  # Notice how we're getting the factorial treatments
  group = paste(batteryData$temperature, batteryData$material, sep = " x "),
  na.rm = TRUE,
  skew = TRUE,
  ranges = TRUE,
  quant = c(0.25, 0.75),
```

```
  IQR = TRUE,
  mat = TRUE,
  digits = 4
)

batteryStats %>%
  tibble::remove_rownames() %>%
  tibble::column_to_rownames(
    var = "group1"
  ) %>%
  dplyr::select(
    n, min, Q0.25, median, Q0.75, max, mad, mean, sd, skew, kurtosis
  ) %>%
  knitr::kable(
    caption = "Summary Statistics for Battery Life Spans",
    digits = 3,
    format.args = list(big.mark = ","),
    align = rep('c', 11),
    col.names = c("n", "Min", "Q1", "Median", "Q3", "Max", "MAD", "SAM", "SASD",
                  "Sample Skew","Sample Ex. Kurtosis"),
    booktabs = TRUE
  )  %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )
```

Table 1: Summary Statistics for Battery Life Spans

	n	Min	Q1	Median	Q3	Max	MAD	SAM	SASD	Sample Skew	Sample Ex. Kurtosis
125°F x Plate 1	4	20	48.50	64.0	73.00	82	17.791	57.50	26.851	-0.466	-1.864
125°F x Plate 2	4	25	40.00	51.5	61.00	70	18.532	49.50	19.261	-0.195	-2.015
125°F x Plate 3	4	60	76.50	89.0	98.00	104	16.309	85.50	19.279	-0.319	-2.001
15°F x Plate 1	4	74	116.00	142.5	161.25	180	37.065	134.75	45.353	-0.331	-1.938
15°F x Plate 2	4	126	144.00	154.5	166.25	188	24.463	155.75	25.617	0.105	-1.917
15°F x Plate 3	4	110	131.00	149.0	162.00	168	22.239	144.00	25.974	-0.308	-2.047
70°F x Plate 1	4	34	38.50	57.5	76.25	80	29.652	57.25	23.599	-0.006	-2.397
70°F x Plate 2	4	106	112.75	118.5	125.50	136	11.861	119.75	12.659	0.197	-1.968
70°F x Plate 3	4	120	134.25	144.5	156.00	174	22.239	145.75	22.544	0.114	-1.956

If you are using `dplyr`'s `summarize` function, you can achieve similar results by first calling `group_by` and then listing all of your factors. In this case we would want `dplyr::group_by(temperature, material)`.

```
# Demo code for descriptive statistics for factorial treatment structure
## Gummy Bears
catapultData %>%
  dplyr::group_by(Angle, Position) %>%
  summarize(
    n = n(),
    min = min(Distance),
    Q1 = quantile(Distance, probs = c(0.25)),
    med = median(Distance),
    Q3 = quantile(Distance, probs = c(0.75)),
    max = max(Distance),
    mad = mad(Distance),
    sam = mean(Distance),
    sd = sd(Distance),
    skew = psych::skew(Distance),
```

```
  kurtosis = psych::kurtosi(Distance)
) %>%
knitr::kable(
  caption = "Summary Statistics for Gummy Bear Study",
  digits = 3,
  format.args = list(big.mark = ","),
  align = rep('c', 13),
  col.names = c("Angle", "Position", "n", "Min", "Q1", "Median", "Q3", "Max", "MAD",
                "SAM", "SASD", "Sample Skew","Sample Ex. Kurtosis"),
  booktabs = TRUE
)  %>%
kableExtra::kable_styling(
  font_size = 12,
  latex_options = c("HOLD_position", "scale_down")
)
```

Table 2: Summary Statistics for Gummy Bear Study

Angle	Position	n	Min	Q1	Median	Q3	Max	MAD	SAM	SASD	Sample Skew	Sample Ex. Kurtosis
High	Back	75	64	109.000	120.00	145.00	195.0	26.687	125.190	28.451	0.265	-0.474
High	Front	75	60	87.500	113.25	141.50	190.8	38.918	114.935	31.050	0.284	-0.915
Low	Back	75	29	80.875	97.00	111.25	160.0	24.092	96.390	26.295	-0.114	0.098
Low	Front	75	30	77.500	106.00	120.50	152.0	25.204	98.473	27.877	-0.630	-0.479

Either approach (`psych::describeBy` or `dplyr::group_by` and `dplyr::summarize`) works for getting values of descriptive statistics in accordance to the factorial treatment structure.

Fit the Models

Before we write code to fit the factorial model in `R`, we should do two things: 1) double check that a factorial ANOVA approach is appropriate, and 2) check for interactions.

Appropriateness of ANOVA

As we have been doing since Unit 3, checking for whether ANOVA methods are appropriate for answering our SRQ comes down to the following:

- Do we have a quantitative response?
- Do we have two or more categorical/qualitative factors? (If only one, then we're not factorial ANOVA.)
- Do we have enough *Degrees of Freedom* to be able to estimate the Main Effects and Interactions?
- Do we have enough *Degrees of Freedom* for estimating residuals/errors?
- (For Main Effects Only Models: do we have an additive model?)

As before, our knowledge of the study design and the Hasse diagram can help us out.

Figure 3 shows the Hasse diagram for the Battery Manufacturing study. We know that the response is quantitative (number of hours of life). From the Hasse diagram, we have two factors (Plate Material and Temperature) which are both categorical. We are doing a full factorial structure (we have all possible interactions). Further, since we have positive *Degrees of Freedom* for each node in the Hasse diagram, we know that we should be able to estimate all main effects, interactions, and the residuals/errors.

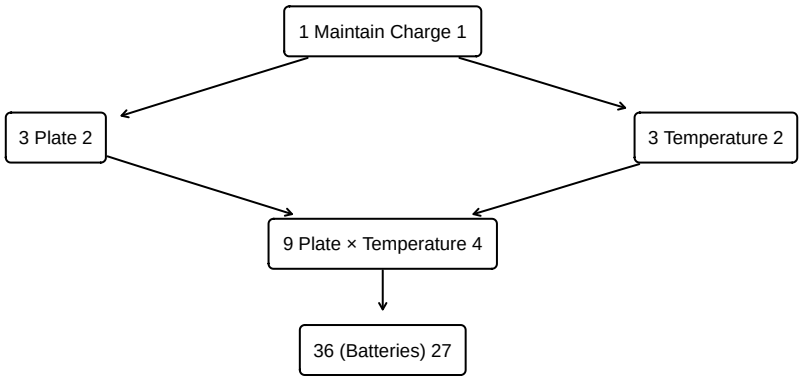

Figure 3: Hasse Diagram for Battery Manufacturing Study

Your Turn

Create a similar paragraph as I did but for the Gummy Bears situation.

Check Interactions

With a [Full] Factorial Design, we no longer have a truly additive model. The interaction term, in some ways, is a measure of how far our model departs from additivity. We want to see whether interactions are important or unimportant: data visualizations are our key to detect this. However, unlike with One-way ANOVA with a Block, we will not worry if we see interactions.

There are several ways that we can look at interaction plots. In the guide/tutorial on Block designs, we saw a way that we can use the `ggplot2` package to create an interaction plot. However, there is also a method that uses Base `R` (i.e., no extra packages).

Base `R` Interaction Plot

The function `interaction.plot` is part of the basic setup of `R` (included in the default `stats` package). This function allows us to explore the interaction between **TWO** factors at a time. This does mean that if you want to try to explore a three-way interaction, this method won't work.

```
# Demo code for using base R to create an interaction plot
## Battery Manufacturing Study
interaction.plot(
  x.factor = batteryData$temperature, # First Factor
  trace.factor = batteryData$material, # Second Factor
  response = batteryData$life, # Response
  fun = mean,
  type = "b", # Both points and lines
  col = c("black","red","blue"), # Set colors for trace
  pch = c(19, 17, 15),  # Set symbols for trace
  fixed = TRUE,
  legend = TRUE,
  xlab = "Operating Temperature",
  ylab = "Life Span (hours)",
  trace.label = "Plate Material")
```

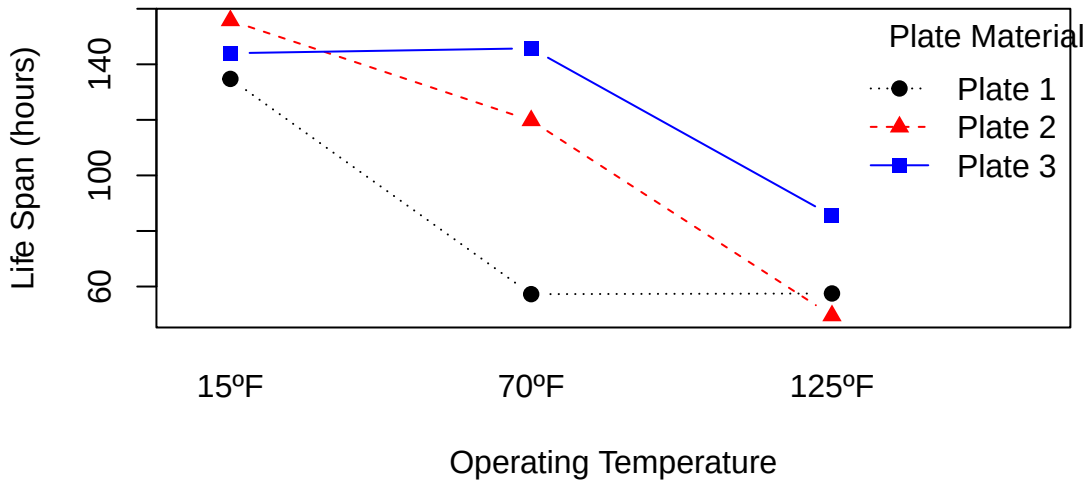

Figure 4: Interaction Plot using base R

For a plot from base R, this is actually a pretty decent. Remember, we're looking to see if as we move through the levels of one factor (operating temperature) and through the levels of the other factor (plate material), what appears to be happening to the response (life span). If there is no interaction, then we should see consistent behaviors throughout (perfect world of no interaction would have parallel line segments). The more inconsistent the behavior (for example, complete reversal of behavior, perpendicular line segments), the more impactful the interaction is between the two factors

Example Write up From Figure 4, we can see that there is some interaction between the operating temperature and plate material. For example, going from 15ºF to 70ºF, produced a drop in life span for materials one and two but for plate three there is a slight increase in life span. As we move from room temperature (70ºF) to the high of 125ºF, this time plate material one appears to hold steady in life span while the other two materials experience drops in life span.

Using `ggplot2` for Interaction Plots

As we saw in the Block guide/tutorial, we can use `ggplot2` to create an interaction plot. Again, you want to have some care here for how many factors you want to explore at any one time. With `ggplot2` you can move beyond two factors at a time, but be careful; you don't want to overwhelm your audience.

In the following example, I'm going to create an interaction plot for launch angle and launch position, but I'm also going to place the observations on the graph so we can also get a sense of variation that gets hidden by the trend lines.

```
# Demo code for using ggplot to make interaction plot w/observations showing
## Gummy Bears Study
ggplot(
  data = catapultData,
  mapping = aes(
    x = Angle,
    y = Distance,
    shape = Position,
    color = Position,
    linetype = Position,
    group = Position
  )
) +
  stat_summary(fun = "mean", geom = "point", size = 3) +
  stat_summary(fun = "mean", geom = "line", size = 1) +
  geom_jitter(width = 0.1, height = 0.1, alpha = 0.25, size = 1) +
  ggplot2::theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (in)") +
  labs(
    color = "Launch Position",
```

```
    shape = "Launch Position",
    linetype = "Launch Position"
  ) +
  scale_color_manual(values = c("red", "blue")) +
  theme(
    legend.position = "bottom",
    text = element_text(size = 12)
  )
```

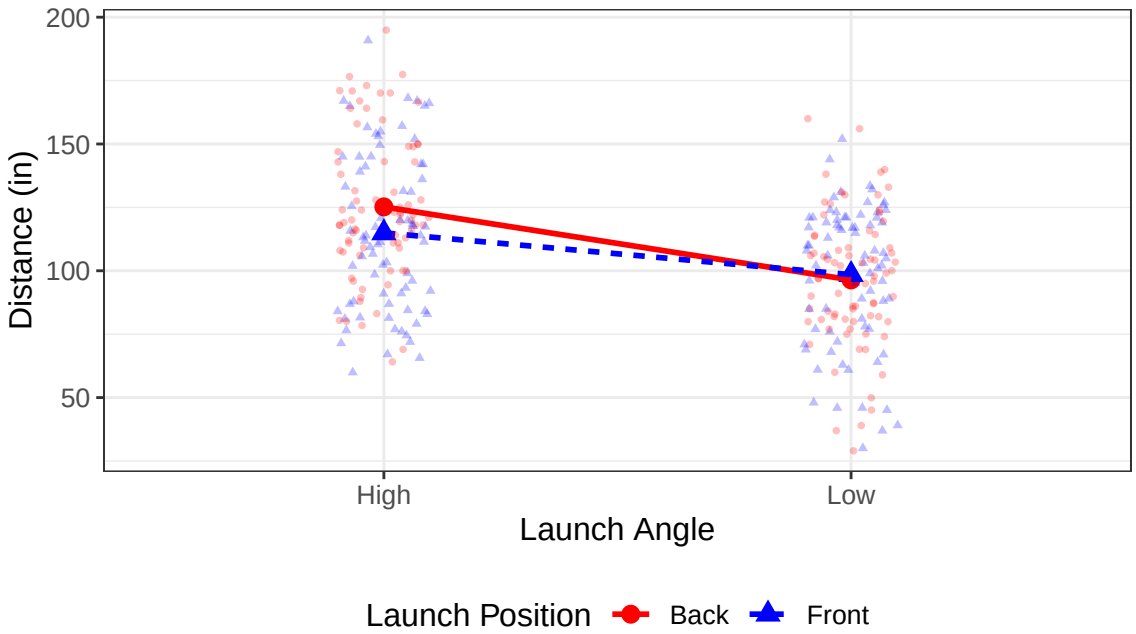

Figure 5: Interaction Plot for Gummy Bears Study

Whether you use the interaction plot code laid out in the Block guide/tutorial, the base `R` approach, or this most recent version is entirely up to you. What you want to be sure of is that the visualization helps you and your audience gain a deeper understanding of the data and context for the SRQ. You also want to make sure that the data visualization looks good.

Your Turn

Write up a paragraph that goes with Figure 5.

Form the Model

Once we've verified that a factorial ANOVA model is appropriate and made a decision about including interaction terms[1], we can turn our attention to forming the model in `R`.

There are a couple of different ways that you can specify factorial designs in `R`: you can manually type in the main the effects and interactions in the order you wish OR you can let `R` fill in all of the terms for you.

For `R`, to specify a main effect, you simply type the name of the factor in the formula just as we have been doing all semester.

For an interaction, you'll type the names of **all** main effects involved in the interaction, separating each name with a colon (:). For example, if we wanted the two-way interaction of A and B, we would type `A:B`; for a three-way interaction of A, B, and C, we would type `A:B:C`.

To have `R` automatically fill in all terms, you simply list each main effect and use `*` to separate terms. Thus, typing `y ~ A*B` is the same as `y ~ A + B + A:B`. Notice that in the `formula` argument the `*` symbol can take on multiple meanings: multiplication as in `2*A` and factorial expansion as in `A*B` going to `A + B + A:B`.

I'll show both approaches here:

[1] There is no harm in keeping interaction terms in the model, even if you don't think there is an interaction...provided you have sufficient *Degrees of Freedom*.

```r
# Demo code for forming the factorial models
## Fitting by hand--Battery Manufacturing Study
batteryModel <- aov(
  formula = life ~ temperature + material + temperature:material,
  data = batteryData
)

## Letting R handle the expansion--Gummy Bears Study
catapultModel <- aov(
  formula = Distance ~ Angle*Position,
  data = catapultData
)
```

Assessing Assumptions

For the parametric shortcut for factorial designs (the ANVOA F test), we have the same three assumptions as in the One-way case: Gaussian Residuals, Homoscedasticity, and Independence of Observations. We will assess them in the same ways as we have before.

Gaussian Residuals

Use a QQ plot like usual:

```r
# Demo code for QQ plot
# Battery Manufacturing
car::qqPlot(
  x = residuals(batteryModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (hours)"
)
```

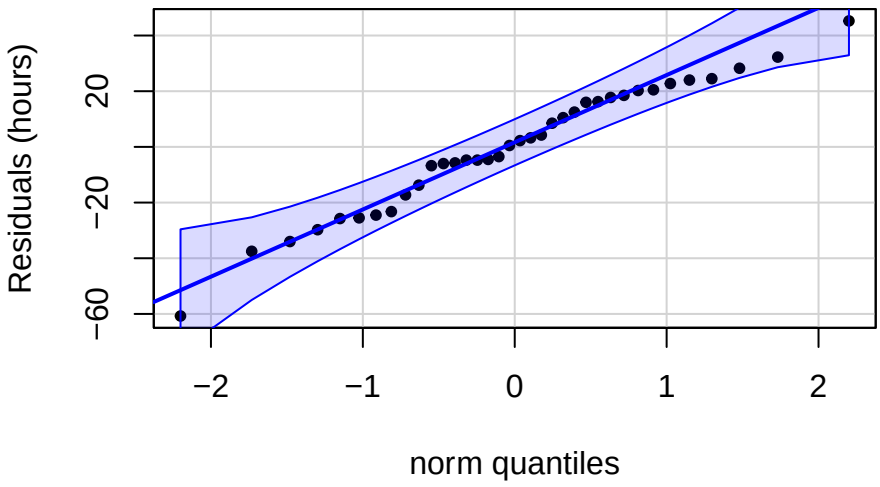

Figure 6: QQ Plot for Residuals in Battery Manufacturing Study

There is very little to be concerned about in our QQ plot; we will go ahead and proceed as if our residuals follow a Gaussian distribution.

Your Turn

Build a QQ plot and write accompanying text for assessing the Gaussian Residuals assumption for the Gummy Bears Study.

Homoscedasticity

Just as in the One-way ANOVA with a Block, we will want to look at a Tukey-Anscombe plot rather than a strip chart for our factorial designs.

```
ggplot(
  data = data.frame(
    residuals = residuals(batteryModel),
    fitted = fitted.values(batteryModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline(
    yintercept = 0,
    linetype = "dashed",
    color = "grey50"
  ) +
  geom_smooth(
    formula = y ~ x,
    method = stats::loess,
    method.args = list(degree = 1),
    se = FALSE,
    size = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (hours)") +
  ylab("Residuals (hours)")
```

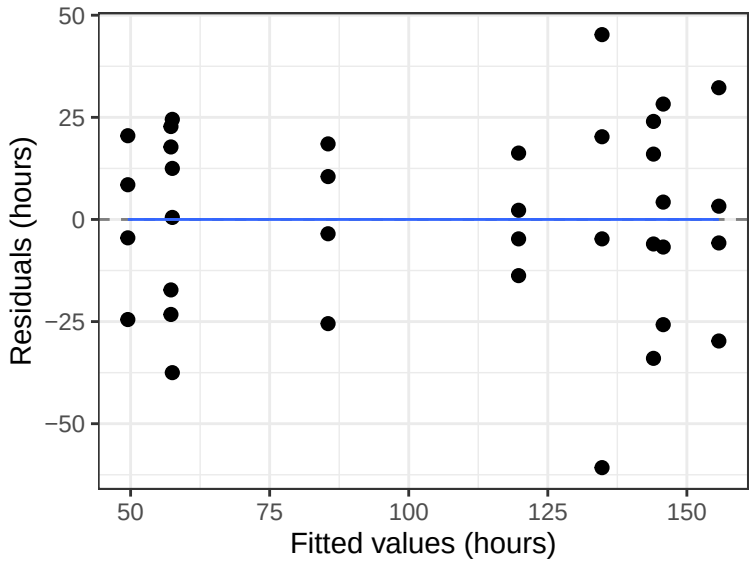

Figure 7: Tukey-Anscombe Plot for Battery Manufacturing Study

The first thing that I notice in the Tukey-Anscombe plot (Figure 7) is that the fourth strip from the left shows the least amount of variation while the fifth strip (from the left) shows the most. The fifth used more than twice the vertical space as the fourth, however, this is the only aspect that causes me a moment of hesitation. There are no discernible patterns to the plot and the blue reference line is perfectly horizontal indicating that we have [sufficient] homoscedasticity.

Your Turn

Create the appropriate plot and associated text for the Gummy Bear study.

Independence of Observations

For Independence of Observations, keep in mind that our Go To is to think about the study design. If we happen to know measurement order, then we can make use of an index plot and the DW statistic.

Battery Manufacturing Study

Unfortunately, we don't know measurement order in the Battery Manufacturing study, so index plots are not going to be useful here. However, we can think through the study design and reach a decision about independent observations.

In this particular case, we know that the the application of plate material was randomly assigned to instances of battery building process and that a set of batteries using each plate material were selected via a random process. Within each of those sets, the engineer assigned an operating temperature to each battery. Taken together, this information does not raise any flags that we have dependent observations based upon the design.

Gummy Bears Study

We do know measurement order for the Gummy Bears study... except that there is one slight wrinkle. We have 12 different "first" gummy bears launched and measured. To explore whether we have any violation of the Independence of Observations, we are going to need to think through the study design and make use of the measurement order.

What do we know about how the study was designed? Quite a bit, after all, we designed the study! We will take the 300 gummy bears as constituting a random sample from the population of gummy bears created by the manufacturer. While not a rigorous random process, Neil did draw individual gummy bears from a central stock pile to separate them into 12 groups of 25. We also know that we randomly assigned the treatments to catapult teams (our experimental units) and they arbitrarily took one of the bags of gummy bears. Taken together, we might take these facts as suggesting that we have independent observations ***up to the point the catapult teams took possession of the baggies of gummy bears***.

I happen to know that at least one group did not use all 25 gummy bears; rather they used a couple of gummy bears and repeated launched and measured them. This is a threat to Independence of Observations. To explore, we will look at index plots... broken out by each experimental unit (catapult teams).

```
ggplot(
  data = catapultData,
  mapping = aes(
    x = Order,
    y = Distance
  )
) +
  geom_point(size = 0.5) +
  geom_line() +
  theme_bw() +
  xlab("Measurement order") +
  ylab("Distance (in)") +
  facet_wrap(
    . ~ Team
  )
```

Figure 8 shows the index plots for the twelve catapult teams. I am going to leave to you the challenge of getting `R` to place the facets in the order Team 1, Team 2, Team 3, etc. instead of what currently appears.

When I look at Figure 8, I don't see twelve perfect carbon copies. This supports the idea that the twelve experimental units are independent. The two plots I'm most concerned about are Team 4, Team 8, and Team 9. In Team 4's plot, we have a run of increasing distances from launch 19 through launch 24. For Team 8, they reached a point where a general upward trend shows up starting launch 12; similarly for Team 9 around launch 11. I suspect that in these cases, the team got into a "groove" or rhythm. While in some circumstances this might be more problematic, given our context, we'll survive.

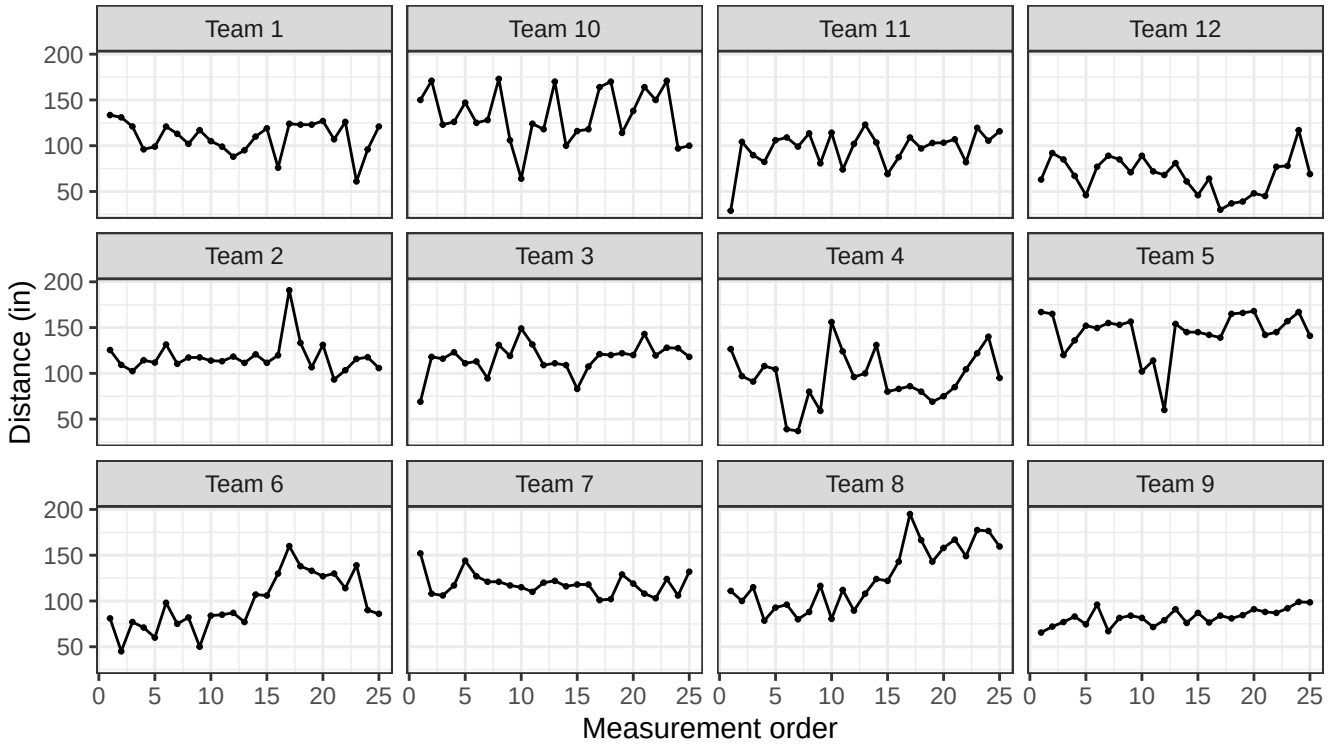

Figure 8: Index Plots for Gummy Bear Study

Recapping Assumptions

We will state that all three assumptions are satisfied for both the Battery Manufacturing and Gummy Bears studies.

Results

For results, we want to tackle both sets: Omnibus (F test, effect sizes, and point estimates) and Post Hoc (pairwise, effect sizes, and/or contrasts).

Omnibus

For the Battery Manufacturing and Gummy Bears studies, we have **balanced** designs. Thus, we do not need to worry about different types of *Sums of Squares*. However, I'm going to demonstrate how we can switch the types.

```
# Omnibus Test/Modern ANOVA Table
parameters::model_parameters(
    model = batteryModel,
    omega_squared = "partial", # Notice the use of partial
    eta_squared = "partial",
    epsilon_squared = "partial",
    type = 1, # Use 1, 2, or 3 for the Type of SSQs you want
    drop = "(Intercept)", # Drop an unneeded row for ANOVA
    verbose = FALSE # Makes the function "quiet"
) %>%
  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",
                "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
  caption = "ANOVA Table for Battery Manufacturing Study",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )
```

Table 3: ANOVA Table for Battery Manufacturing Study

Source	SS	df	MS	F	p-value	Partial Omega Sq.	Partial Eta Sq.	Partial Epsilon Sq.
temperature	39118.722	2	19559.361	28.9677	< 0.0001	0.6084	0.6821	0.6586
material	10683.722	2	5341.861	7.9114	0.002	0.2774	0.3695	0.3228
temperature:material	9613.778	4	2403.444	3.5595	0.0186	0.2214	0.3453	0.2483
Residuals	18230.750	27	675.213					

Table 3 gives us a modern ANOVA table for our full factorial in the Battery Manufacturing Study. Unlike for RCBD, we care about all of the rows this time. We interpret the F-ratio just as we have done previously. Same goes for the p-values. Supposing that we set $UT = 0.05$, we still use the p-values from the table in the regular way to make decisions between the hypotheses.

The effect sizes (partial omega/eta/epsilon-squared values) are still proportions of variation in the response explained and we can use the regular Rules of Thumb for qualitative sizing. The biggest catch is to recognize that these aren't *unique* explanations of variance.

In the Battery Manufacturing situation, we would decide to reject the null hypothesis for each of the main effects and for the interaction term. Of these, temperature appears to explain most of the variation in battery life.

Your Turn

Make the modern ANOVA table for the Gummy Bears situation. Write a paragraph to go with the table.

Point Estimates

You can get point estimates for your main effects and treatment effects using the `dummy.coef` function.

```
# Demo code for Point Estimates
## Battery Manufacturing Study
pointEst <- dummy.coef(batteryModel)
pointEst <- unlist(pointEst)
names(pointEst) <- c(
  "Grand Mean",
  levels(batteryData$temperature),
  levels(batteryData$material),
  outer(
    levels(batteryData$temperature),
    levels(batteryData$material),
    FUN = paste,
    sep = " x "
  )
)
```

```r
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	105.53
15°F	39.31
70°F	2.06
125°F	-41.36
Plate 1	-22.36
Plate 2	2.81
Plate 3	19.56
15°F x Plate 1	12.28
70°F x Plate 1	-27.97
125°F x Plate 1	15.69
15°F x Plate 2	8.11
70°F x Plate 2	9.36
125°F x Plate 2	-17.47
15°F x Plate 3	-20.39
70°F x Plate 3	18.61
125°F x Plate 3	1.78

I will make no claims that Table 4 is the most efficient way to present the point estimates. I'll leave how to better parse this table to you all.

What I will highlight is that the interpretations of the *GSAM* and the main effects remain consistent with what we've worked with previously. For example, ignoring all factors, our batteries lasted 105.53 times as long as the number of batteries we tested (i.e., the common baseline performance). At 70°F, the battery's performance was an additional 2.06 hours/battery greater than the common baseline. However, Plate 1 batteries had a reduction of 22.36 hours/battery from baseline.

The interpretation of interaction terms may be thought of as the "differences in differences" for performances or as how much we have to moderate the performance of a group from just the main effects? That is to say, how much does the battery performance change due to the interaction of the two factors. For example, let's consider Plate 1 and 15°F. The performance for the batteries in this group starts with baseline (105.53 hrs/battery) and gets a bonus for 15°F (+39.31 hrs/battery). However, this group also has a performance penalty for Plate 1 (-22.36 hrs/battery). However, the interaction of 15°F and Plate 1 moderates the performance penalty by giving back 12.28 hrs/battery. If we add these rates together, $105.53 + 39.31 - 22.36 + 12.28 \approx 134.76$, which is equal (up to rounding) to the value of the *SAM* we saw for this group in Table 1 or will see in Table 5.

Your Turn

Get the point estimates for the Gummy Bear Study. Give interpretations for the *GSAM*, one level of each main effect, and one interaction level.

Post Hoc–Pairwise Comparisons & Effect Sizes

While you *could* use the pairwise comparison functions we've previously used, a better approach is to embrace our Factorial Design and look at the *estimated marginal means*. These will hold certain factors constant and let others vary. To do this, we will need to use the **emmeans** package.

```
# Demo code for Post Hoc
## Marginal Means with Adjusted Confidence Intervals
## Battery Manufacturing Study
batteryPHMeans <- emmeans::emmeans(
  object = batteryModel,
  # The order of factors does not really matter for this
  specs = pairwise ~ temperature | material,
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)

as.data.frame(batteryPHMeans$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Temperature", "Plate Material", "Marginal Mean","SE", "DF",
                  "Lower Bound","Upper Bound"),
    caption = "Marginal Means-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 5: Marginal Means-Tukey 90% Adjustment

Temperature	Plate Material	Marginal Mean	SE	DF	Lower Bound	Upper Bound
15ºF	Plate 1	134.75	12.9924	27	112.6201	156.8799
70ºF	Plate 1	57.25	12.9924	27	35.1201	79.3799
125ºF	Plate 1	57.50	12.9924	27	35.3701	79.6299
15ºF	Plate 2	155.75	12.9924	27	133.6201	177.8799
70ºF	Plate 2	119.75	12.9924	27	97.6201	141.8799
125ºF	Plate 2	49.50	12.9924	27	27.3701	71.6299
15ºF	Plate 3	144.00	12.9924	27	121.8701	166.1299
70ºF	Plate 3	145.75	12.9924	27	123.6201	167.8799
125ºF	Plate 3	85.50	12.9924	27	63.3701	107.6299

```
# Demo code for Post Hoc
## Pairwise Comparisons
## Battery Manufacturing Study
batteryPHTemp <- emmeans::emmeans(
  object = batteryModel,
  # Order maters for this; comparisons | held constant
  specs = pairwise ~ temperature | material,
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)
```

```
as.data.frame(batteryPHTemp$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Plate Material", "Estimate","SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Pairwise Comparisons of Temperature-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 6: Pairwise Comparisons of Temperature-Tukey 90% Adjustment

Comparison	Plate Material	Estimate	SE	DF	t Statistic	p-value
15ºF - 70ºF	Plate 1	77.50	18.3741	27	4.2179	0.0007
15ºF - 125ºF	Plate 1	77.25	18.3741	27	4.2043	0.0007
70ºF - 125ºF	Plate 1	-0.25	18.3741	27	-0.0136	0.9999
15ºF - 70ºF	Plate 2	36.00	18.3741	27	1.9593	0.1419
15ºF - 125ºF	Plate 2	106.25	18.3741	27	5.7826	0.0000
70ºF - 125ºF	Plate 2	70.25	18.3741	27	3.8233	0.0020
15ºF - 70ºF	Plate 3	-1.75	18.3741	27	-0.0952	0.9950
15ºF - 125ºF	Plate 3	58.50	18.3741	27	3.1838	0.0099
70ºF - 125ºF	Plate 3	60.25	18.3741	27	3.2791	0.0078

```
# Demo code for Post Hoc
## Pairwise Comparisons
## Battery Manufacturing Study
batteryPHPlate <- emmeans::emmeans(
  object = batteryModel,
  # Order maters for this; comparisons | held constant
  specs = pairwise ~ material | temperature,
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)

as.data.frame(batteryPHPlate$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Temperature", "Estimate","SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Pairwise Comparisons of Plate Material-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 7: Pairwise Comparisons of Plate Material-Tukey 90% Adjustment

Comparison	Temperature	Estimate	SE	DF	t Statistic	p-value
Plate 1 - Plate 2	15°F	-21.00	18.3741	27	-1.1429	0.4967
Plate 1 - Plate 3	15°F	-9.25	18.3741	27	-0.5034	0.8703
Plate 2 - Plate 3	15°F	11.75	18.3741	27	0.6395	0.7998
Plate 1 - Plate 2	70°F	-62.50	18.3741	27	-3.4015	0.0058
Plate 1 - Plate 3	70°F	-88.50	18.3741	27	-4.8166	0.0001
Plate 2 - Plate 3	70°F	-26.00	18.3741	27	-1.4150	0.3475
Plate 1 - Plate 2	125°F	8.00	18.3741	27	0.4354	0.9012
Plate 1 - Plate 3	125°F	-28.00	18.3741	27	-1.5239	0.2959
Plate 2 - Plate 3	125°F	-36.00	18.3741	27	-1.9593	0.1419

The `adjust` argument of `emmeans` allows values as your method for controlling Type I Error rates: `"bonferroni"`, `"tukey"`, `"scheffe"`, `"sidak"`, `"holm"`, `"hochberg"`, `"hommel"`, `"BH"` (Benjamini and Hochberg), and `"fdr"`. You will get point estimates (marginal means) and their confidence intervals when you use the `postHocObject$emmeans` and you'll get pairwise comparisons with adjusted p-values when you use `postHocObject$contrasts`.

Effect Sizes

Unfortunately, my `anova.PostHoc` function does not currently work with with factorial models. However, the `emmeans` package provides us with a way to get Cohen's d, which then allows us to my `probSup` function to get the Probability of Superiority.

```
# Demo Code for Effect Sizes
## Battery Manufacturing Study
## Pairwise Temperature Comparisons
as.data.frame(
  eff_size(
    object = batteryPHTemp,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate(
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, material, effect.size, ps) %>%
  knitr::kable(
    digits = 3,
    col.names = c("Comparison", "Plate Material", "Cohen's d", "Probability of Superiority"),
    align = "lccc",
    caption = "Effect Sizes for Temperature",
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 8: Effect Sizes for Temperature

Comparison	Plate Material	Cohen's d	Probability of Superiority
(15°F - 70°F)	Plate 1	2.983	0.983
(15°F - 125°F)	Plate 1	2.973	0.982
(70°F - 125°F)	Plate 1	-0.010	0.497
(15°F - 70°F)	Plate 2	1.385	0.836
(15°F - 125°F)	Plate 2	4.089	0.998
(70°F - 125°F)	Plate 2	2.703	0.972
(15°F - 70°F)	Plate 3	-0.067	0.481
(15°F - 125°F)	Plate 3	2.251	0.944
(70°F - 125°F)	Plate 3	2.319	0.949

```
# Demo Code for Effect Sizes
## Battery Manufacturing Study
## Pairwise Plate Material Comparisons
as.data.frame(
  eff_size(
    object = batteryPHPlate,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate(
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, temperature, effect.size, ps) %>%
  knitr::kable(
    digits = 3,
    col.names = c("Comparison", "Temperature", "Cohen's d", "Probability of Superiority"),
    align = "lccc",
    caption = "Effect Sizes for Plate Material",
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 9: Effect Sizes for Plate Material

Comparison	Temperature	Cohen's d	Probability of Superiority
(Plate 1 - Plate 2)	15°F	-0.808	0.284
(Plate 1 - Plate 3)	15°F	-0.356	0.401
(Plate 2 - Plate 3)	15°F	0.452	0.625
(Plate 1 - Plate 2)	70°F	-2.405	0.044
(Plate 1 - Plate 3)	70°F	-3.406	0.008
(Plate 2 - Plate 3)	70°F	-1.001	0.240
(Plate 1 - Plate 2)	125°F	0.308	0.586
(Plate 1 - Plate 3)	125°F	-1.078	0.223
(Plate 2 - Plate 3)	125°F	-1.385	0.164

You would interpret these effects sizes exactly as you would in a One-way ANOVA context.

You can also see a bit of the influence of the interaction terms. Look at Table 9 and the Plate 1 - Plate 2 values across the three temperatures. Notice that the effect changes with the operating temperature.

Post Hoc–Contrasts

While not all that common, especially if you are using a model building perspective, you can still do contrasts with factorial designs.

For contrasts involving combinations of levels of a single factor, you would use the same approaches and methods as in the One-way ANOVA case.

Contrasts involving combinations of levels of interaction terms are not straightforward. My recommendation is that you think extremely carefully as to what the purpose of such a combination is and whether there are other ways to investigate.

Imbalanced Designs

As mentioned in class, when you have imbalanced designs for factorial models, we have to make a decision about which type of Sums of Squares we want to use.

Different Types of Sums of Squares

We have three types of Sums of Squares. Keep in mind that the default is Type I for the `aov` call. Most often in Factorial ANOVA settings, we are interested in Type II (model building–watch out of important interaction terms) and Type III (for testing differences amongst factor levels).

For all of these we can use the `type` argument of the `model_parameters` function that we use for our modern ANOVA tables.

Quick Example

For this example, I'm going to use a data set on different training methods' (fixed, 2 levels) and engery drink's (fixed, 2 levels) impacts on the time to run around a particular track.

Look at how the various values change across the tables.

```
# Demo Code of Imbalanced Designs
# Load Running Data
running <- read.table(
  file = "http://stat.ethz.ch/~meier/teaching/data/running.dat",
  header = TRUE
)

running$method <- as.factor(running$method)
running$drink <- as.factor(running$drink)

# Fit the anova model--same as usual
runningModel <- aov(
  formula = y ~ method*drink, # R interprets this as y ~ method + drink + method:drink
  data = running
)
```

```
# Demo Code
# Type I SSQs
parameters::model_parameters(
    model = runningModel,
    omega_squared = "partial",
    eta_squared = "partial",
```

```
    epsilon_squared = "partial",
    type = 1, # Type I SSQs
    drop = "(Intercept)",
    verbose = FALSE
) %>%
  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",
                "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
  caption = "ANOVA Table for Running-Type I SSQs",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )
```

Table 10: ANOVA Table for Running-Type I SSQs

Source	SS	df	MS	F	p-value	Partial Omega Sq.	Partial Eta Sq.	Partial Epsilon Sq.
method	2024.0201	1	2024.0201	263.7191	< 0.0001	0.7896	0.7998	0.7968
drink	455.2481	1	455.2481	59.3164	< 0.0001	0.4545	0.4733	0.4654
method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0383	0.0543	0.0400
Residuals	506.5440	66	7.6749					

```
# Demo Code
# Type II SSQs
parameters::model_parameters(
    model = runningModel,
    omega_squared = "partial",
    eta_squared = "partial",
    epsilon_squared = "partial",
    type = 2, # Type II SSQs
    drop = "(Intercept)",
    verbose = FALSE
) %>%
  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",
                  "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
    caption = "ANOVA Table for Running-Type II SSQs",
    align = c('l',rep('c',8)),
```

```
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )
```

Table 11: ANOVA Table for Running-Type II SSQs

Source	SS	df	MS	F	p-value	Partial Omega Sq.	Partial Eta Sq.	Partial Epsilon Sq.
method	1333.4120	1	1333.4120	173.7365	< 0.0001	0.7116	0.7247	0.7205
drink	455.2481	1	455.2481	59.3164	< 0.0001	0.4545	0.4733	0.4654
method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0383	0.0543	0.0400
Residuals	506.5440	66	7.6749					

```
# Demo Code
# Type III SSQs
parameters::model_parameters(
  model = runningModel,
  omega_squared = "partial",
  eta_squared = "partial",
  epsilon_squared = "partial",
  type = 3, # Type III SSQs
  drop = "(Intercept)",
  verbose = FALSE
) %>%
  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",
                  "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
    caption = "ANOVA Table for Running-Type III SSQs",
    align = c('l',rep('c',8)),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )
```

Table 12: ANOVA Table for Running-Type III SSQs

	Source	SS	df	MS	F	p-value	Partial Omega Sq.	Partial Eta Sq.	Partial Epsilon Sq.
1	method	1352.4038	1	1352.4038	176.2110	< 0.0001	0.7145	0.7275	0.7234
2	drink	484.2370	1	484.2370	63.0935	< 0.0001	0.4701	0.4887	0.4810
3	method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0383	0.0543	0.0400
5	Residuals	506.5440	66	7.6749					

Code Appendix

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

packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "hasseDiagram", "car",
              "psych", "DescTools", "emmeans", "openxlsx")
lapply(packages, library, character.only = TRUE)

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

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

# Demo code to set up R
## Load packages
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "hasseDiagram", "car",
              "psych", "DescTools", "emmeans", "openxlsx")
lapply(packages, library, character.only = TRUE)

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

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

# Demo code for loading and cleaning data
## Loading gummy bear data
catapultData <- openxlsx::readWorkbook(
  xlsxFile = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/gummyBears2022.xlsx",
  sheet = 1,
  colNames = TRUE,
  rowNames = FALSE
)

## Change variables
names(catapultData)[which(names(catapultData) == "Launch.Angle")] <- "Angle"
names(catapultData)[which(names(catapultData) == "Launch.Position")] <- "Position"
names(catapultData)[which(names(catapultData) == "Measurement.Order")] <- "Order"

## Fix the data
## R will treat high and High as TWO separate levels
## Same with low & Low, back & Back, and front & Front
catapultData <- catapultData %>%
  dplyr::mutate(
    Angle = dplyr::recode_factor(
      Angle,
      "high" = "High", # Old Value = New Value
      "High" = "High",
      "low" = "Low",
      "Low" = "Low"
```

```
    ),
    Position = dplyr::recode_factor(
      Position,
      "back" = "Back",
      "Back" = "Back",
      "front" = "Front",
      "Front" = "Front"
    ),
    Team = dplyr::case_when(
      Team == "Team 6" ~ Team,
      TRUE ~ paste("Team", Team)
    )
  )

## Demo code for loading and cleaning data
## Load battery data
batteryData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/batteryLife.dat",
  header = TRUE,
  sep = ","
)

## Clean data
batteryData$temperature <- dplyr::recode_factor(
  batteryData$temperature,
  `15` = "15ºF",
  `70` = "70ºF",
  `125` = "125ºF"
)
batteryData$material <- dplyr::recode_factor(
  batteryData$material,
  `1` = "Plate 1",
  `2` = "Plate 2",
  `3` = "Plate 3"
)

# Demo code for box plots using factorial treatment structure
## Battery Manufacturing
boxplot(
  formula = life ~ temperature:material,
  data = batteryData,
  ylab = "Life (hrs)",
  xlab = "Temp (ºF) x Material"
)

# Demo code of box plots incorporating factorial treatment structure
## ggplot2 and Gummy Bears Study
ggplot(
  data = catapultData,
  mapping = aes(
    x = Angle,
    y = Distance,
    fill = Position
  )
) +
  geom_boxplot() +
  theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (in)") +
```

```
  labs(
    fill = "Launch Position"
  ) +
  theme(
    legend.position = "right",
    text = element_text(size = 14)
  )

# Demo code for descriptive statistics for factorial treatment structure
## Battery Manufacturing
batteryStats <- psych::describeBy(
  x = batteryData$life,
  # Notice how we're getting the factorial treatments
  group = paste(batteryData$temperature, batteryData$material, sep = " x "),
  na.rm = TRUE,
  skew = TRUE,
  ranges = TRUE,
  quant = c(0.25, 0.75),
  IQR = TRUE,
  mat = TRUE,
  digits = 4
)

batteryStats %>%
  tibble::remove_rownames() %>%
  tibble::column_to_rownames(
    var = "group1"
  ) %>%
  dplyr::select(
    n, min, Q0.25, median, Q0.75, max, mad, mean, sd, skew, kurtosis
  ) %>%
  knitr::kable(
    caption = "Summary Statistics for Battery Life Spans",
    digits = 3,
    format.args = list(big.mark = ","),
    align = rep('c', 11),
    col.names = c("n", "Min", "Q1", "Median", "Q3", "Max", "MAD", "SAM", "SASD",
                  "Sample Skew","Sample Ex. Kurtosis"),
    booktabs = TRUE
  )  %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )

# Demo code for descriptive statistics for factorial treatment structure
## Gummy Bears
catapultData %>%
  dplyr::group_by(Angle, Position) %>%
  summarize(
    n = n(),
    min = min(Distance),
    Q1 = quantile(Distance, probs = c(0.25)),
    med = median(Distance),
    Q3 = quantile(Distance, probs = c(0.75)),
    max = max(Distance),
    mad = mad(Distance),
    sam = mean(Distance),
    sd = sd(Distance),
```

```
    skew = psych::skew(Distance),
    kurtosis = psych::kurtosi(Distance)
  ) %>%
  knitr::kable(
    caption = "Summary Statistics for Gummy Bear Study",
    digits = 3,
    format.args = list(big.mark = ","),
    align = rep('c', 13),
    col.names = c("Angle", "Position", "n", "Min", "Q1", "Median", "Q3", "Max", "MAD",
                  "SAM", "SASD", "Sample Skew","Sample Ex. Kurtosis"),
    booktabs = TRUE
  )  %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )

# Hasse Diagram
modelLabels <- c("1 Maintain Charge 1", "3 Plate 2", "3 Temperature 2",
                 "9 Plate × Temperature 4", "36 (Batteries) 27")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, TRUE,
           FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE,
           TRUE, TRUE, TRUE, FALSE),
  nrow = 5,
  ncol = 5,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Demo code for using base R to create an interaction plot
## Battery Manufacturing Study
interaction.plot(
  x.factor = batteryData$temperature, # First Factor
  trace.factor = batteryData$material, # Second Factor
  response = batteryData$life, # Response
  fun = mean,
  type = "b", # Both points and lines
  col = c("black","red","blue"), # Set colors for trace
  pch = c(19, 17, 15),  # Set symbols for trace
  fixed = TRUE,
  legend = TRUE,
  xlab = "Operating Temperature",
  ylab = "Life Span (hours)",
  trace.label = "Plate Material")

# Demo code for using ggplot to make interaction plot w/observations showing
## Gummy Bears Study
ggplot(
  data = catapultData,
  mapping = aes(
    x = Angle,
    y = Distance,
    shape = Position,
    color = Position,
    linetype = Position,
```

```
    group = Position
  )
) +
  stat_summary(fun = "mean", geom = "point", size = 3) +
  stat_summary(fun = "mean", geom = "line", size = 1) +
  geom_jitter(width = 0.1, height = 0.1, alpha = 0.25, size = 1) +
  ggplot2::theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (in)") +
  labs(
    color = "Launch Position",
    shape = "Launch Position",
    linetype = "Launch Position"
  ) +
  scale_color_manual(values = c("red", "blue")) +
  theme(
    legend.position = "bottom",
    text = element_text(size = 12)
  )

# Demo code for forming the factorial models
## Fitting by hand--Battery Manufacturing Study
batteryModel <- aov(
  formula = life ~ temperature + material + temperature:material,
  data = batteryData
)

## Letting R handle the expansion--Gummy Bears Study
catapultModel <- aov(
  formula = Distance ~ Angle*Position,
  data = catapultData
)

# Demo code for QQ plot
# Battery Manufacturing
car::qqPlot(
  x = residuals(batteryModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (hours)"
)


ggplot(
  data = data.frame(
    residuals = residuals(batteryModel),
    fitted = fitted.values(batteryModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline(
    yintercept = 0,
    linetype = "dashed",
    color = "grey50"
  ) +
  geom_smooth(
```

```
    formula = y ~ x,
    method = stats::loess,
    method.args = list(degree = 1),
    se = FALSE,
    size = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (hours)") +
  ylab("Residuals (hours)")

ggplot(
  data = catapultData,
  mapping = aes(
    x = Order,
    y = Distance
  )
) +
  geom_point(size = 0.5) +
  geom_line() +
  theme_bw() +
  xlab("Measurement order") +
  ylab("Distance (in)") +
  facet_wrap(
    . ~ Team
  )

# Omnibus Test/Modern ANOVA Table
parameters::model_parameters(
    model = batteryModel,
    omega_squared = "partial", # Notice the use of partial
    eta_squared = "partial",
    epsilon_squared = "partial",
    type = 1, # Use 1, 2, or 3 for the Type of SSQs you want
    drop = "(Intercept)", # Drop an unneeded row for ANOVA
    verbose = FALSE # Makes the function "quiet"
) %>%
  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",
                "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
  caption = "ANOVA Table for Battery Manufacturing Study",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )

# Demo code for Point Estimates
## Battery Manufacturing Study
```

```
pointEst <- dummy.coef(batteryModel)
pointEst <- unlist(pointEst)
names(pointEst) <- c(
  "Grand Mean",
  levels(batteryData$temperature),
  levels(batteryData$material),
  outer(
    levels(batteryData$temperature),
    levels(batteryData$material),
    FUN = paste,
    sep = " x "
  )
)

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 code for Post Hoc
## Marginal Means with Adjusted Confidence Intervals
## Battery Manufacturing Study
batteryPHMeans <- emmeans::emmeans(
  object = batteryModel,
  # The order of factors does not really matter for this
  specs = pairwise ~ temperature | material,
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)

as.data.frame(batteryPHMeans$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Temperature", "Plate Material", "Marginal Mean","SE", "DF",
                  "Lower Bound","Upper Bound"),
    caption = "Marginal Means-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo code for Post Hoc
## Pairwise Comparisons
## Battery Manufacturing Study
batteryPHTemp <- emmeans::emmeans(
  object = batteryModel,
  # Order maters for this; comparisons | held constant
  specs = pairwise ~ temperature | material,
```

```
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)

as.data.frame(batteryPHTemp$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Plate Material", "Estimate","SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Pairwise Comparisons of Temperature-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo code for Post Hoc
## Pairwise Comparisons
## Battery Manufacturing Study
batteryPHPlate <- emmeans::emmeans(
  object = batteryModel,
  # Order maters for this; comparisons | held constant
  specs = pairwise ~ material | temperature,
  adjust = "tukey", # Where you specify your chosen method
  level = 0.9 # 1--Type I Risk
)

as.data.frame(batteryPHPlate$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Temperature", "Estimate","SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Pairwise Comparisons of Plate Material-Tukey 90\\% Adjustment",
    align = rep("c", 7),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo Code for Effect Sizes
## Battery Manufacturing Study
## Pairwise Temperature Comparisons
as.data.frame(
  eff_size(
    object = batteryPHTemp,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate(
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
```

```
  dplyr::select(contrast, material, effect.size, ps) %>%
  knitr::kable(
    digits = 3,
    col.names = c("Comparison", "Plate Material", "Cohen's d", "Probability of Superiority"),
    align = "lccc",
    caption = "Effect Sizes for Temperature",
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code for Effect Sizes
## Battery Manufacturing Study
## Pairwise Plate Material Comparisons
as.data.frame(
  eff_size(
    object = batteryPHPlate,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate(
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, temperature, effect.size, ps) %>%
  knitr::kable(
    digits = 3,
    col.names = c("Comparison", "Temperature", "Cohen's d", "Probability of Superiority"),
    align = "lccc",
    caption = "Effect Sizes for Plate Material",
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Demo Code of Imbalanced Designs
# Load Running Data
running <- read.table(
  file = "http://stat.ethz.ch/~meier/teaching/data/running.dat",
  header = TRUE
)

running$method <- as.factor(running$method)
running$drink <- as.factor(running$drink)

# Fit the anova model--same as usual
runningModel <- aov(
  formula = y ~ method*drink, # R interprets this as y ~ method + drink + method:drink
  data = running
)

# Demo Code
```

```r
# Type I SSQs
parameters::model_parameters(
    model = runningModel,
    omega_squared = "partial",
    eta_squared = "partial",
    epsilon_squared = "partial",
    type = 1, # Type I SSQs
    drop = "(Intercept)",
    verbose = FALSE
) %>%
  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",
                "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
  caption = "ANOVA Table for Running-Type I SSQs",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )

# Demo Code
# Type II SSQs
parameters::model_parameters(
    model = runningModel,
    omega_squared = "partial",
    eta_squared = "partial",
    epsilon_squared = "partial",
    type = 2, # Type II SSQs
    drop = "(Intercept)",
    verbose = FALSE
) %>%
  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",
                  "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
    caption = "ANOVA Table for Running-Type II SSQs",
    align = c('l',rep('c',8)),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
```

```
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )

# Demo Code
# Type III SSQs
parameters::model_parameters(
  model = runningModel,
  omega_squared = "partial",
  eta_squared = "partial",
  epsilon_squared = "partial",
  type = 3, # Type III SSQs
  drop = "(Intercept)",
  verbose = FALSE
) %>%
  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",
                  "Partial Omega Sq.", "Partial Eta Sq.", "Partial Epsilon Sq."),
    caption = "ANOVA Table for Running-Type III SSQs",
    align = c('l',rep('c',8)),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("scale_down", "HOLD_position")
  )
```