Factorial Models

Neil J. Hatfield

Last Updated: November 13, 2023

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.) I will also cover Post Hoc analysis in the factorial setting. I'll end with a section highlighting the issues with imbalanced factorial designs.

1 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:

```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")
```

2 Data Contexts

For this guide/tutorial, we'll make use of two contexts: **Gummy Bears** and **Battery Manufacturing**.

2.1 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
gummyData <- openxlsx::readWorkbook(
  xlsxFile = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/gummyBears_Fall2023.xlsx",
  sheet = 1,
  colNames = TRUE,
  rowNames = FALSE
)

### Impose a logical ordering on the angle factor
gummyData$angle <- factor(
  x = gummyData$angle,
  levels = c("Flat", "Low", "High")
)

gummyData$position <- as.factor(gummyData$position)
gummyData$team <- as.factor(gummyData$team)
```

Notice that I used both `factor` and `as.factor`. The `as.factor` function makes use of the values that exist in the column and organizes the levels alphabetically. This works well for the launch position. For the launch angle factor, I wanted the levels to reflect the increasing magnitude of the angle, thus, I used the `factor` function's `levels` argument to specify the order. The risk here is if your entries to `levels` don't exactly match what is in your data, you'll have observations get dropped.

2.1.1 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.

2.2 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?

```r
## 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
### If you are using dplyr version 1.1.0+
batteryData$temperature <- dplyr::case_match(
  .x = batteryData$temperature,
  15 ~ "15ºF",
  70 ~ "70ºF",
  125 ~ "125ºF",
  .ptype = factor(levels = c("15ºF", "70ºF", "125ºF"))
)

batteryData$material <- dplyr::case_match(
  .x = batteryData$material,
  1 ~ "Plate 1",
  2 ~ "Plate 2",
  3 ~ "Plate 3",
  .ptype = factor(levels = c("Plate 1", "Plate 2", "Plate 3"))
)

### If you are using dplyr version < 1.1.0
# 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"
# )
```

For the Battery Manufacturing data I made use of the `case_match` function from the `dplyr` package (if using an older version of `dplyr`, use the `recode_factor` function). This allowed me not only transform/clean the reported values but also simultaneously tell R to treat the attributes as factors.

2.2.1 Your Turn

Write SRQs and hypotheses for the Battery Manufacturing study.

3 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.

3.1 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.

```r
# 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"
)
```

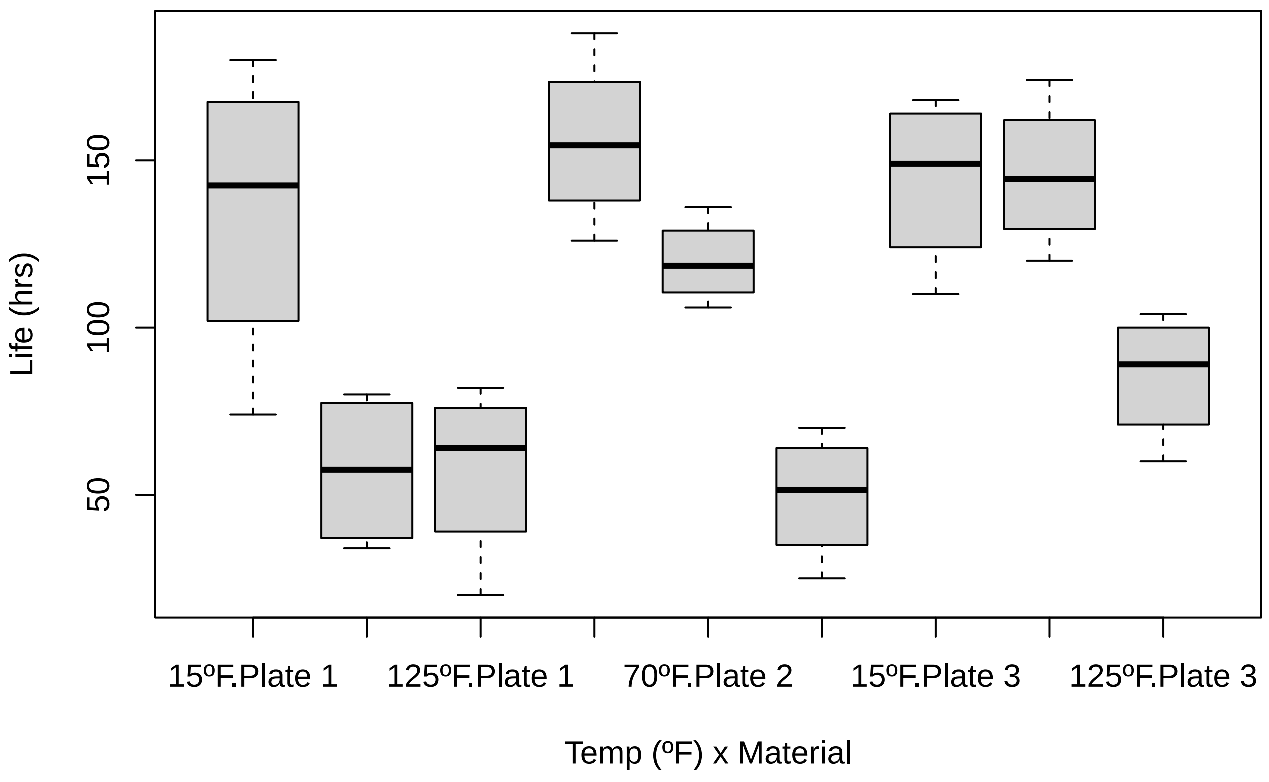

Figure 1: Box Plot of Battery Life Spans by Temperature and Plate Material

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 = gummyData,
  mapping = aes(
    x = angle,
    y = distance,
    fill = position
  )
) +
  geom_boxplot() +
  theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (cm)") +
  labs(
    fill = "Launch Position"
  ) +
  theme(
    legend.position = "bottom",
    text = element_text(size = 14)
  )
```

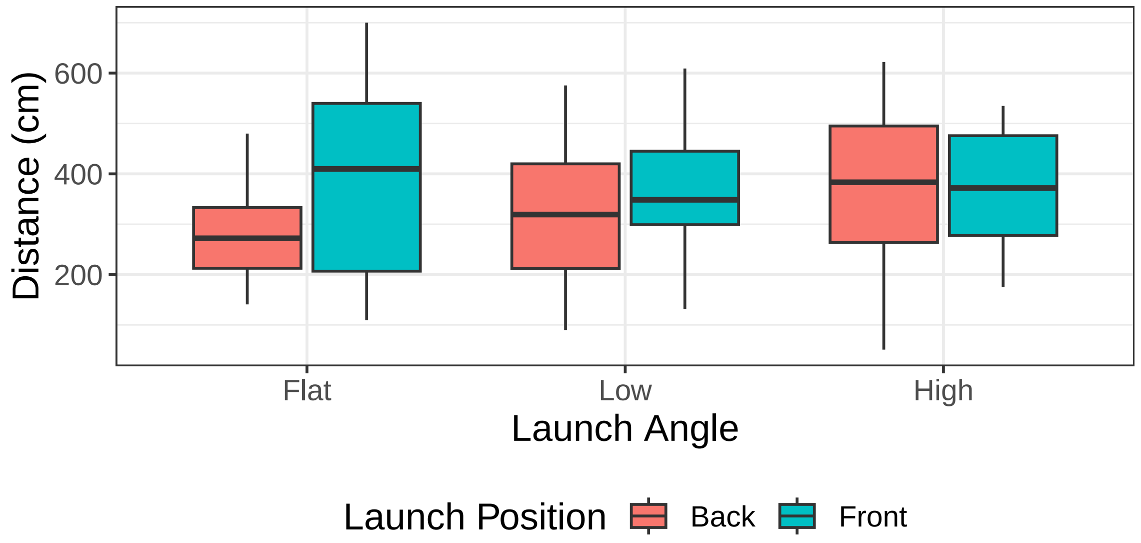

Figure 2: Box Plot With Multiple Factors–Gummy Bears Study

3.2 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:

```
# Demo code for descriptive statistics for factorial treatment structure ----
## Using the psych package's describeBy function
## Battery Manufacturing
batteryStats <- psych::describeBy(
  life ~ temperature + material, # Notice that we use the + as an "AND"
  data = batteryData,
  na.rm = TRUE,
  skew = TRUE,
  ranges = TRUE,
  quant = c(0.25, 0.75),
  IQR = TRUE,
  mat = TRUE,
  digits = 4
)

batteryStats %>%
  tibble::remove_rownames() %>%
  dplyr::select(
    group1, group2, 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("l", "c"), times = c(2, 11)),
    col.names = c("Temperature", "Material", "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

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

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)`. The following example is for the gummy bears data.

```
# Demo code for descriptive statistics for factorial treatment structure ----
## Using dplyr's summarize function with custom choices of statistics
## Gummy Bears
gummyData %>%
  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),
    .groups = "drop"
  ) %>%
  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
Flat	Back	22	140.8	212.55	272.00	332.875	479.9	92.440	279.991	89.860	0.510	-0.536
Flat	Front	22	109.3	206.80	409.70	539.750	700.0	272.205	386.009	192.774	0.011	-1.526
Low	Back	22	90.0	212.00	319.25	419.875	575.5	166.051	309.841	139.827	0.157	-1.195
Low	Front	22	131.6	299.00	348.50	444.950	609.1	88.956	369.336	114.578	0.227	-0.469
High	Back	22	51.0	263.75	383.25	495.000	622.0	175.317	375.486	155.700	-0.356	-1.009
High	Front	22	175.0	277.50	371.75	475.650	534.8	153.078	374.973	115.182	-0.127	-1.437

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.

4 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.

4.1 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.

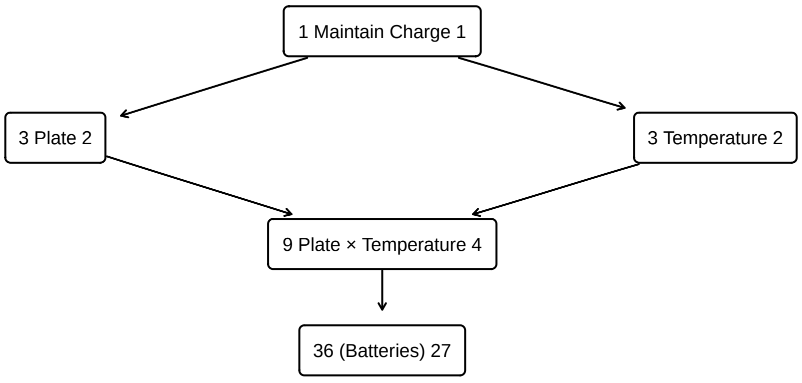

Figure 3: Hasse Diagram for Battery Manufacturing Study

4.1.1 Your Turn

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

4.2 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).

4.2.1 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.

```r
# 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")
```

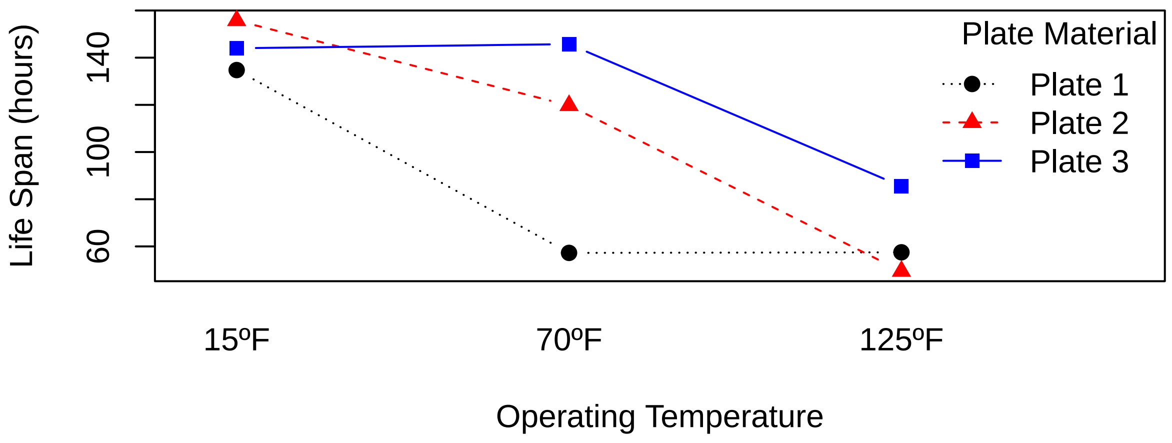

Figure 4: Interaction Plot using base R

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

4.2.1.1 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.

4.2.2 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.

```r
# Demo code for using ggplot to make interaction plot w/observations showing ----
## Gummy Bears Study
ggplot(
  data = gummyData,
  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", linewidth = 1) +
  geom_jitter(width = 0.1, height = 0.1, alpha = 0.25, size = 1) +
  ggplot2::theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (cm)") +
  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)
  )
```

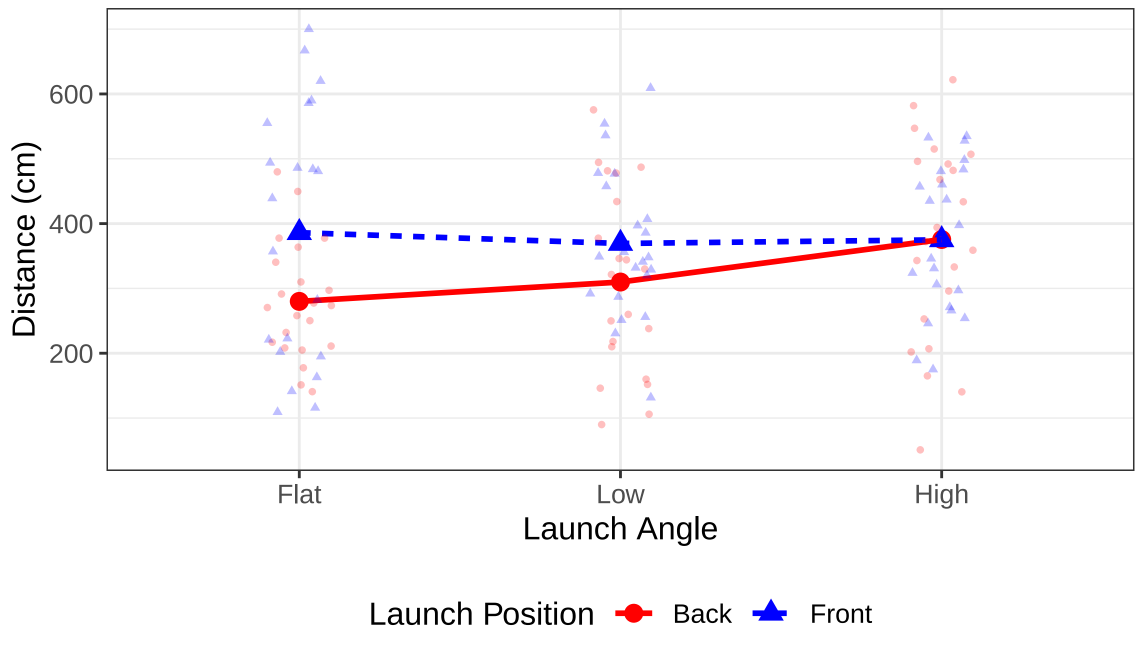

Figure 5: Interaction Plot for Gummy Bears Study

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.

4.2.3 Your Turn

Write up a paragraph that goes with Figure 5.

4.3 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:

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

[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
## Letting R handle the expansion--Gummy Bears Study
gummyModel <- aov(
  formula = distance ~ angle*position,
  data = gummyData
)
```

5 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.

5.1 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)"
)
```

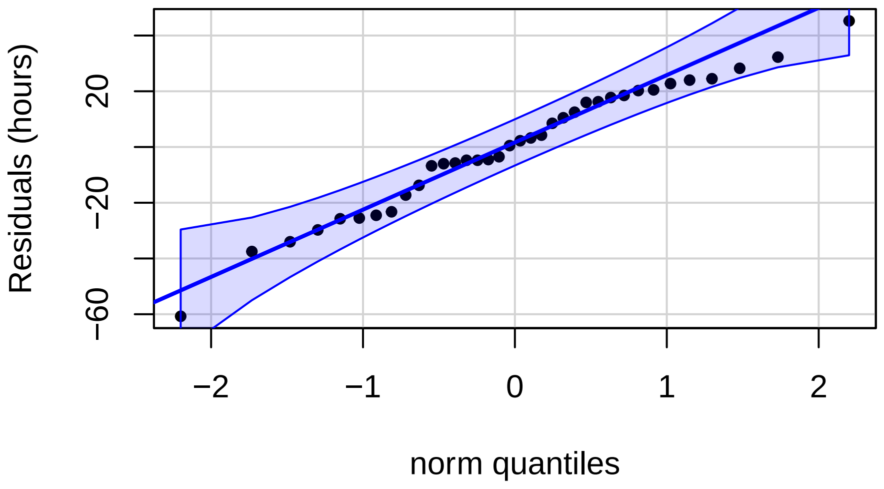

Figure 6: QQ Plot for Residuals in Battery Manufacturing Study

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.

5.1.1 Your Turn

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

5.2 Homoscedasticity

With Factorial Designs, we will want to look at a Tukey-Anscombe plot rather than a strip chart. This allows us to better incorporate our full model.

```r
## Tukey-Anscombe Plot for Battery Study ----
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,
    linewidth = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (hours)") +
  ylab("Residuals (hours)")
```

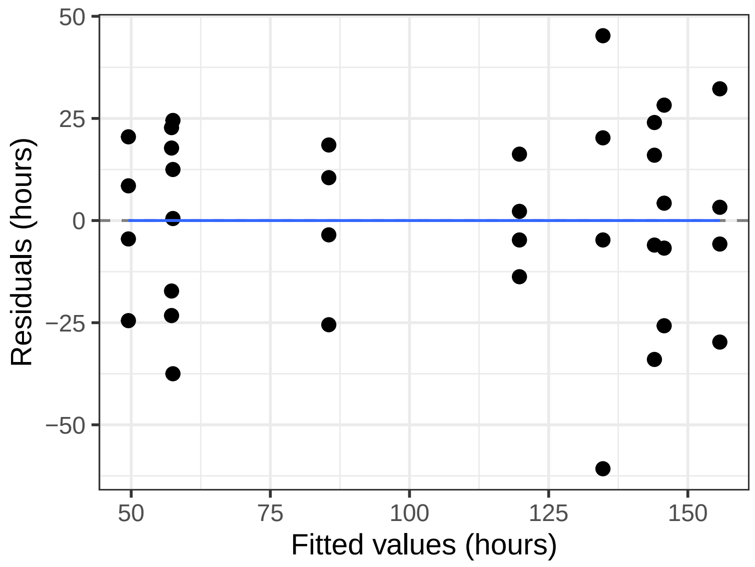

Figure 7: Tukey-Anscombe Plot for Battery Manufacturing Study

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.

5.2.1 Your Turn

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

5.3 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.

5.3.1 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.

5.3.2 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 240 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 20. 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***.

```r
# Index Plots for Gummy Bear Study ----
ggplot(
  data = gummyData,
  mapping = aes(
    x = order,
    y = distance
  )
) +
  geom_point(size = 0.5) +
  geom_line() +
  theme_bw() +
  xlab("Measurement order") +
  ylab("Distance (cm)") +
  facet_wrap(
    facets = vars(team),
    scales = "fixed"
  )
```

Figure 8 shows the index plots for the twelve catapult teams.

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 plot I'm most concerned about is from Group/Team 3D. We'll need to investigate further.

5.4 Recapping Assumptions

We will state that all three assumptions are satisfied in the Battery Manufacturing study. For the Gummy Bears study, we have issues we need to resolve first.

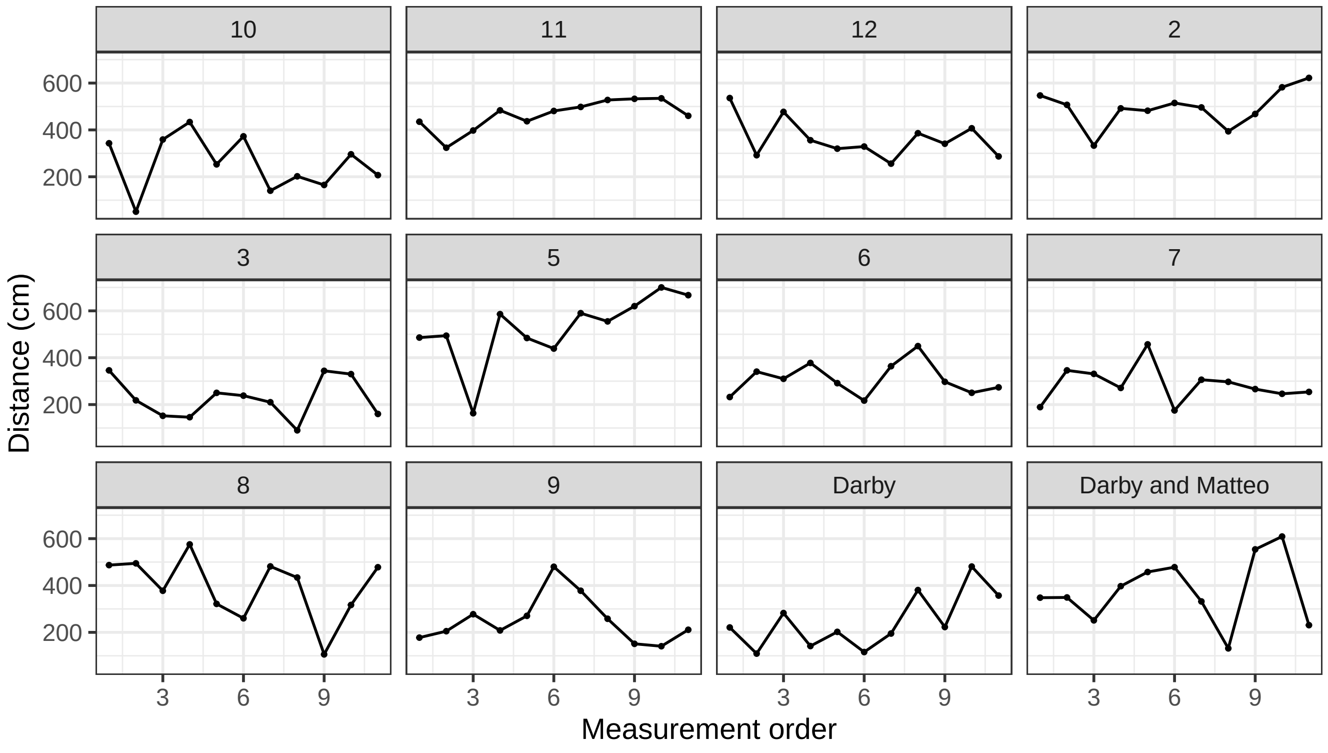

Figure 8: Index Plots for Gummy Bear Study

6 Omnibus 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).

6.1 ANOVA Table

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 ---
## Battery Model
parameters::model_parameters(
  model = batteryModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  type = 3, # 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 Eta Sq.", "Partial Omega Sq.", "Partial Epsilon Sq."),
  caption = "ANOVA Table for Battery Manufacturing Study",
```

```r
  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 Eta Sq.	Partial Omega Sq.	Partial Epsilon Sq.
1	temperature	39118.722	2	19559.361	28.9677	< 0.0001	0.6821	0.6084	0.6586
2	material	10683.722	2	5341.861	7.9114	0.0020	0.3695	0.2774	0.3228
3	temperature:material	9613.778	4	2403.444	3.5595	0.0186	0.3453	0.2214	0.2483
5	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.

6.1.1 Your Turn

Make the modern ANOVA table for the Gummy Bears situation (pretending that we *don't* have the issues that we do). Write a paragraph to go with the table.

6.2 Point Estimates

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

```r
# 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 Battery Manufacturing Study",
```

Table 4: Point Estimates from the Battery Manufacturing Study

	Estimate
Grand Mean	105.53
15ºF	39.31
70ºF	2.06
125ºF	-41.36
Plate 1	-22.36
Plate 2	2.81
Plate 3	19.56
15ºF x Plate 1	12.28
70ºF x Plate 1	-27.97
125ºF x Plate 1	15.69
15ºF x Plate 2	8.11
70ºF x Plate 2	9.36
125ºF x Plate 2	-17.47
15ºF x Plate 3	-20.39
70ºF x Plate 3	18.61
125ºF x Plate 3	1.78

```
  booktabs = TRUE,
  align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    # latex_options = c("HOLD_position")
    # I commented out the above so the computer would find "best" placement
  )
```

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 70ºF, 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 15ºF. The performance for the batteries in this group starts with baseline (105.53 hrs/battery) and gets a bonus for 15ºF (+39.31 hrs/battery). However, this group also has a performance penalty for Plate 1 (-22.36 hrs/battery). However, the interaction of 15ºF 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.

6.2.1 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.

7 Pairwise Comparisons for Post Hoc Analysis

Post Hoc analysis in [full] factorial designs is a much richer ground. This is due to the fact that we simply have more going on to explore. In fact, we can conceptualize the following kinds of pairwise explorations:

- Pairwise Comparisons of Individual Main Effects
- Pairwise Comparisons of Conditioned Main Effects
- Pairwise Comparisons of Interactions

Each of these types of pairwise comparisons allow you to look at different aspects. **However,** you should specify what you are wanting to explore *prior* to doing your analysis so that you don't accidentally shift perspectives (EDA vs. CDA) and claim your work is for the other.

For all of these approaches, we will make use of the `emmeans` package for ease of coding and so that we can be sure to get effect sizes (practical significance).

7.1 Individual Main Effects Pairwise Comparisons

In this type of pairwise comparison, we are focusing on comparing the levels of a single factor (i.e., a main effect), ignoring all other factors in the model. This is equivalent to doing pairwise analysis in One-way ANOVA.

Given that the plate material came up as being an important contributor to how long the battery lasted, we might want to see which plate materials are statistically different from one another. We'll control SCI at 10% via Tukey's HSD.

```r
# Demo code for main effects pairwise comparisons ----
## Battery Life Example, Plate Material
platePostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material,
  adjust = "tukey",
  level = 0.9
)

knitr::kable(
  x = platePostHoc$contrasts,
  digits = 3,
  caption = "Post Hoc Comparisons for Main Effect Plate Material",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value"),
  align = "lccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 5: Post Hoc Comparisons for Main Effect Plate Material

Pair	Difference	SE	DF	t	p-value
Plate 1 - Plate 2	-25.167	10.608	27	-2.372	0.063
Plate 1 - Plate 3	-41.917	10.608	27	-3.951	0.001
Plate 2 - Plate 3	-16.750	10.608	27	-1.579	0.272

From Table 5, we can see that plate material 1 is statistically different from both plate materials 2 and 3, but plate materials 2 and 3 are not statistically different from each other. Given the order of comparison (the Pair column) and the values of the differences, the data suggest that plate material 1 leads to shorter battery life.

We can add measures of effect size to Table 5. In the following example, I'll show how we can get the effect size table, and then join that two the results table.

```r
# Demo code for main effects pairwise with effect sizes ----
## Battery Life Study, Plate Material
### Create effect size table
plateEffects <- as.data.frame(
  eff_size(
    object = platePostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, effect.size, ps)

### Build table
as.data.frame(platePostHoc$contrasts) %>%
  left_join(
    y = plateEffects,
    by = join_by(contrast == contrast)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Main Effect Plate Material",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value", "Cohen's d",
                "Prob. of Superiority"),
  align = "lccccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 6: Post Hoc Comparisons for Main Effect Plate Material

Pair	Difference	SE	DF	t	p-value	Cohen's d	Prob. of Superiority
Plate 1 - Plate 2	-25.167	10.608	27	-2.372	0.063	-0.969	0.247
Plate 1 - Plate 3	-41.917	10.608	27	-3.951	0.001	-1.613	0.127
Plate 2 - Plate 3	-16.750	10.608	27	-1.579	0.272	-0.645	0.324

We still interpret Cohen's d and the Probability of Superiority in the same was we did for pairwise comparisons in the One-way case.

7.2 Conditional Main Effects Pairwise Comparisons

In this type of pairwise comparison, we are still wanting to compare the levels of particular factor, but this time we want to incorporate information from another term in the model. That is, we want to make our pairwise comparisons of Factor A's levels when controlling/accounting for the levels of Factor B. These type of pairwise comparison isn't as common as the other two.

For this kind of pairwise comparison, we might pose the question of which plate materials are statistically different from one another when we account for the operating temperature. The `specs` argument to `emmeans` has a crucial change for this type of analysis. After we specify `pairwise ~ material` to indicate that we want to do pairwise comparisons between the levels of the plate material factor, we type `| temperature`. We read the vertical bar (|, a.k.a. "pipe") as "given" and then we list what we want to condition our results by. In this case, the operating temperature factor. For this testing family, we'll arbitrarily choose to control the False Discovery Rate via the Benjamini-Hochberg method at 8%.

```r
# Demo code for conditional pairwise comparisons ----
## Battery Life Example, Plate Material Conditioned on Temp
plateTempPostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material | temperature, # Notice the use of the |
  adjust = "BH",
  level = 0.92
)

knitr::kable(
  x = plateTempPostHoc$contrasts,
  digits = 3,
  caption = "Post Hoc Comparisons for Plate Material Conditioned by Temperature",
  col.names = c("Pair", "Temp.", "Difference", "SE", "DF", "t", "p-value"),
  align = "llccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 7: Post Hoc Comparisons for Plate Material Conditioned by Temperature

Pair	Temp.	Difference	SE	DF	t	p-value
Plate 1 - Plate 2	15ºF	-21.00	18.374	27	-1.143	0.619
Plate 1 - Plate 3	15ºF	-9.25	18.374	27	-0.503	0.619
Plate 2 - Plate 3	15ºF	11.75	18.374	27	0.639	0.619
Plate 1 - Plate 2	70ºF	-62.50	18.374	27	-3.402	0.003
Plate 1 - Plate 3	70ºF	-88.50	18.374	27	-4.817	0.000
Plate 2 - Plate 3	70ºF	-26.00	18.374	27	-1.415	0.168
Plate 1 - Plate 2	125ºF	8.00	18.374	27	0.435	0.667
Plate 1 - Plate 3	125ºF	-28.00	18.374	27	-1.524	0.209
Plate 2 - Plate 3	125ºF	-36.00	18.374	27	-1.959	0.181

Just as before, we can add on effect size estimates to Table 7 as shown in Table 8 and the following code.

```r
# Demo code for conditional pairwise comparisons with effect sizes ----
## Battery Life Study, Plate Material Conditioned on Temp
### Create effect size table
plateTempEffects <- as.data.frame(
  eff_size(
    object = plateTempPostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, temperature, effect.size, ps)

### Build table
as.data.frame(plateTempPostHoc$contrasts) %>%
  left_join(
    y = plateTempEffects,
    by = join_by(contrast == contrast, temperature == temperature)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Plate Material Conditioned by Temperature",
  col.names = c("Pair", "Temp.", "Difference", "SE", "DF", "t", "p-value",
                "Cohen's d", "Prob. of Superiority"),
  align = "llcccccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )
```

Table 8: Post Hoc Comparisons for Plate Material Conditioned by Temperature

Pair	Temp.	Difference	SE	DF	t	p-value	Cohen's d	Prob. of Superiority
Plate 1 - Plate 2	15ºF	-21.00	18.374	27	-1.143	0.619	-0.808	0.284
Plate 1 - Plate 3	15ºF	-9.25	18.374	27	-0.503	0.619	-0.356	0.401
Plate 2 - Plate 3	15ºF	11.75	18.374	27	0.639	0.619	0.452	0.625
Plate 1 - Plate 2	70ºF	-62.50	18.374	27	-3.402	0.003	-2.405	0.044
Plate 1 - Plate 3	70ºF	-88.50	18.374	27	-4.817	0.000	-3.406	0.008
Plate 2 - Plate 3	70ºF	-26.00	18.374	27	-1.415	0.168	-1.001	0.240
Plate 1 - Plate 2	125ºF	8.00	18.374	27	0.435	0.667	0.308	0.586
Plate 1 - Plate 3	125ºF	-28.00	18.374	27	-1.524	0.209	-1.078	0.223
Plate 2 - Plate 3	125ºF	-36.00	18.374	27	-1.959	0.181	-1.385	0.164

The biggest thing to keep in mind when you work with this table is that when you talk about the difference between plate material 1 and plate material 2, you need to specify at what operating temperature you're look at. From Table 8, plate material 1 is only statistically different from plate materials 2 and 3 at the 70ºF operating temperature. Notice how this is a more nuanced result than what we saw with just the [unconditional] main effect post hoc analysis.

7.3 Interaction Term Pairwise Comparisons

This last type of pairwise comparison focuses on interaction terms. Generally speaking, when we use this approach we jump straight to the treatment or grouping interaction term. That is, the highest order interaction term in the model that represents are experiment's treatments or what made the groups in a quasi-experiment.

In the Battery Life study, our treatments consisted of a pairing of a plate material and an operating temperature; e.g., plate material 1 and operating temperature of 15ºF. This means that there are nine different treatments which we could compare. From the omnibus results (Table 3), we saw that there is a statistically significant interaction term. Thus, we might want to see which of the treatments are statistically different from each other, beyond what we've already done.

For this example testing family, we'll control SCI at 10% using Tukey's adjustment. For this approach we need to place the interaction term that represents the treatment, i.e., `material:temperature` in the `specs` argument after `pairwise ~`.

```r
# Demo code for Pairwise Comparison of Interaction Term ----
## Battery Study, Treatments

## Generate pairwise comparisons
treatmentPostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material:temperature, # notice the interaction
  ajust = "tukey",
  level = 0.9
)

## Generate effect sizes
treatmentEffects <- as.data.frame(
  eff_size(
    object = treatmentPostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, effect.size, ps)

## Build table
as.data.frame(treatmentPostHoc$contrasts) %>%
  left_join(
    y = treatmentEffects,
    by = join_by(contrast == contrast)
  ) %>%
  kable(
    digits = 3,
    caption = "Post Hoc Comparisons of Battery Study Treatments",
    col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value",
                  "Cohen's d", "Prob. of Superiority"),
    align = "lccccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("scale_down") # let computer place table where it fits
  )
```

Notice that Table 9 is a rather large table. When working with factorial models, you will want to have care about what you do for Post Hoc analysis. In factorial settings our testing families can grow quite quickly thus we will want to think critically about what we actually want to do for Post Hoc analyses.

Table 9: Post Hoc Comparisons of Battery Study Treatments

Pair	Difference	SE	DF	t	p-value	Cohen's d	Prob. of Superiority
Plate 1 15ºF - Plate 2 15ºF	-21.00	18.374	27	-1.143	0.962	-0.808	0.284
Plate 1 15ºF - Plate 3 15ºF	-9.25	18.374	27	-0.503	1.000	-0.356	0.401
Plate 1 15ºF - Plate 1 70ºF	77.50	18.374	27	4.218	0.007	2.983	0.983
Plate 1 15ºF - Plate 2 70ºF	15.00	18.374	27	0.816	0.995	0.577	0.658
Plate 1 15ºF - Plate 3 70ºF	-11.00	18.374	27	-0.599	0.999	-0.423	0.382
Plate 1 15ºF - Plate 1 125ºF	77.25	18.374	27	4.204	0.007	2.973	0.982
Plate 1 15ºF - Plate 2 125ºF	85.25	18.374	27	4.640	0.002	3.281	0.990
Plate 1 15ºF - Plate 3 125ºF	49.25	18.374	27	2.680	0.202	1.895	0.910
Plate 2 15ºF - Plate 3 15ºF	11.75	18.374	27	0.639	0.999	0.452	0.625
Plate 2 15ºF - Plate 1 70ºF	98.50	18.374	27	5.361	0.000	3.791	0.996
Plate 2 15ºF - Plate 2 70ºF	36.00	18.374	27	1.959	0.582	1.385	0.836
Plate 2 15ºF - Plate 3 70ºF	10.00	18.374	27	0.544	1.000	0.385	0.607
Plate 2 15ºF - Plate 1 125ºF	98.25	18.374	27	5.347	0.000	3.781	0.996
Plate 2 15ºF - Plate 2 125ºF	106.25	18.374	27	5.783	0.000	4.089	0.998
Plate 2 15ºF - Plate 3 125ºF	70.25	18.374	27	3.823	0.017	2.703	0.972
Plate 3 15ºF - Plate 1 70ºF	86.75	18.374	27	4.721	0.002	3.338	0.991
Plate 3 15ºF - Plate 2 70ºF	24.25	18.374	27	1.320	0.917	0.933	0.745
Plate 3 15ºF - Plate 3 70ºF	-1.75	18.374	27	-0.095	1.000	-0.067	0.481
Plate 3 15ºF - Plate 1 125ºF	86.50	18.374	27	4.708	0.002	3.329	0.991
Plate 3 15ºF - Plate 2 125ºF	94.50	18.374	27	5.143	0.001	3.637	0.995
Plate 3 15ºF - Plate 3 125ºF	58.50	18.374	27	3.184	0.074	2.251	0.944
Plate 1 70ºF - Plate 2 70ºF	-62.50	18.374	27	-3.402	0.046	-2.405	0.044
Plate 1 70ºF - Plate 3 70ºF	-88.50	18.374	27	-4.817	0.001	-3.406	0.008
Plate 1 70ºF - Plate 1 125ºF	-0.25	18.374	27	-0.014	1.000	-0.010	0.497
Plate 1 70ºF - Plate 2 125ºF	7.75	18.374	27	0.422	1.000	0.298	0.584
Plate 1 70ºF - Plate 3 125ºF	-28.25	18.374	27	-1.537	0.828	-1.087	0.221
Plate 2 70ºF - Plate 3 70ºF	-26.00	18.374	27	-1.415	0.882	-1.001	0.240
Plate 2 70ºF - Plate 1 125ºF	62.25	18.374	27	3.388	0.047	2.396	0.955
Plate 2 70ºF - Plate 2 125ºF	70.25	18.374	27	3.823	0.017	2.703	0.972
Plate 2 70ºF - Plate 3 125ºF	34.25	18.374	27	1.864	0.642	1.318	0.824
Plate 3 70ºF - Plate 1 125ºF	88.25	18.374	27	4.803	0.001	3.396	0.992
Plate 3 70ºF - Plate 2 125ºF	96.25	18.374	27	5.238	0.000	3.704	0.996
Plate 3 70ºF - Plate 3 125ºF	60.25	18.374	27	3.279	0.060	2.319	0.949
Plate 1 125ºF - Plate 2 125ºF	8.00	18.374	27	0.435	1.000	0.308	0.586
Plate 1 125ºF - Plate 3 125ºF	-28.00	18.374	27	-1.524	0.835	-1.078	0.223
Plate 2 125ºF - Plate 3 125ºF	-36.00	18.374	27	-1.959	0.582	-1.385	0.164

8 Running Contrasts in Factorial Settings

One way to help limit the size of a testing family for post hoc analysis is to only include those comparisons that you want to test. (The `emmeans` package generates a warning message to this effect when you run the above commands for the treatment pairwise comparisons.)

The method for doing contrasts in factorial settings is the same as in the One-way setting. However, I strongly recommend that you use the `emmeans` package as opposed to other methods so that you can make full use of the factorial model.

8.1 Contrasts on Main Effect

Let's do a contrast on plate materials where we compare plate material 1 to the set of plate materials 2 and 3.

```r
# Demo Code for main effects contrasts ----
## Get the appropriate means from the model
plateMeans <- emmeans::emmeans(
  object = batteryModel,
  specs = ~ material # Nothing goes on the left of ~; list what term you want
)

# plateMeans # Look at the output object to double check the order of levels

## Apply the contrasts
plateContrasts1 <- emmeans::contrast(
  object = plateMeans, # Notice that this is the means object
  method = list(
    "Plate 1 vs. Plates 2 & 3" = c(1, -1/2, -1/2)
  ),
  adjust = "none" # No MC/SI adjustment
)

## Add effect sizes and make a nice looking table
as.data.frame(plateContrasts1) %>%
  dplyr::mutate(
    cohen = effectsize::t_to_d(t = t.ratio, df_error = df)$d, # Effect Sizes
    ps = probSup(cohen) # Effect sizes; this comes from Neil's ANOVA toolkit
  ) %>%
  kable(
    digits = 3,
    caption = "Battery Life Main Effects Contrast on Plate Material",
    col.names = c("Contrast", "Difference", "SE", "DF", "t Statistic",
                  "p-value", "Cohen's d", "Prob. of Superiority"),
    align = "lcccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )
```

Table 10: Battery Life Main Effects Contrast on Plate Material

Contrast	Difference	SE	DF	t Statistic	p-value	Cohen's d	Prob. of Superiority
Plate 1 vs. Plates 2 & 3	-33.542	9.187	27	-3.651	0.001	-1.405	0.16

All of our interpretations are the same as before.

8.2 Contrasts on Treatments

We can also run contrasts on our treatments. This include pairwise comparisons, thus we can do a subset instead of doing all of the $\binom{9}{2} = 36$. The following code shows how we might apply this approach.

```r
# Demo Code for main effects contrasts ----
## Get the appropriate means from the model
treatmentMeans <- emmeans::emmeans(
  object = batteryModel,
  specs = ~ material:temperature
)

# treatmentMeans #Be sure to look at the output so you can build your vectors
## Apply the contrasts
treatmentContrasts1 <- emmeans::contrast(
  object = treatmentMeans, # Notice that this is the means object
  method = list(
    "Plate 1 at 15ºF vs. Plate 2 at 70ºF" = c(1,0,0,0,-1,0,0,0,0),
    "Plate 1 vs Plates 2 & 3, all at 70ºF" = c(0,0,0,1,-1/2,-1/2,0,0,0),
    "Plate 1-all temps vs. Plate 2-all temps" = c(1/3,-1/3,0,1/3,-1/3,0,1/3,-1/3,0)
  ),
  adjust = "bonferroni"
)

## Add effect sizes and make a nice looking table
as.data.frame(treatmentContrasts1) %>%
  dplyr::mutate(
    cohen = effectsize::t_to_d(t = t.ratio, df_error = df)$d,
    ps = probSup(cohen)
  ) %>%
  kable(
    digits = 3,
    caption = "Treatment Contrasts for Battery Study--Bonferroni Adjusted",
    col.names = c("Contrast", "Difference", "SE", "DF", "t Statistic",
                  "p-value", "Cohen's d", "Prob. of Superiority"),
    align = "lccccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )
```

Table 11: Treatment Contrasts for Battery Study–Bonferroni Adjusted

Contrast	Difference	SE	DF	t Statistic	p-value	Cohen's d	Prob. of Superiority
Plate 1 at 15ºF vs. Plate 2 at 70ºF	15.000	18.374	27	0.816	1.000	0.314	0.588
Plate 1 vs Plates 2 & 3, all at 70ºF	-75.500	15.912	27	-4.745	0.000	-1.826	0.098
Plate 1-all temps vs. Plate 2-all temps	-25.167	10.608	27	-2.372	0.075	-0.913	0.259

9 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.

9.1 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 among 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.

9.2 Quick Example

For this example, I'm going to use a data set collected for the purpose of exploring the effects of different training methods (fixed, 2 levels) and energy drinks (fixed, 2 levels) on the time to run around a particular track.

Look at how the various values change across the tables.

```r
# 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
)
```

```r
# Demo Code
# Type I SSQs
parameters::model_parameters(
  model = runningModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  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 12: 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	0.7998	0.7896	0.7968
drink	455.2481	1	455.2481	59.3164	< 0.0001	0.4733	0.4545	0.4654
method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0543	0.0383	0.0400
Residuals	506.5440	66	7.6749					

```r
# Demo Code
# Type II SSQs
parameters::model_parameters(
  model = runningModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  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 13: 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	0.7247	0.7116	0.7205
drink	455.2481	1	455.2481	59.3164	< 0.0001	0.4733	0.4545	0.4654
method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0543	0.0383	0.0400
Residuals	506.5440	66	7.6749					

```r
# Demo Code
# Type III SSQs
parameters::model_parameters(
  model = runningModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  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 14: 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	0.7275	0.7145	0.7234
2	drink	484.2370	1	484.2370	63.0935	< 0.0001	0.4887	0.4701	0.4810
3	method:drink	29.0942	1	29.0942	3.7908	0.0558	0.0543	0.0383	0.0400
5	Residuals	506.5440	66	7.6749					

Notice how the various corresponding values in Tables 12, 13, and 14 change for our main effects (training method and drink type). While in this example we would be hard pressed to make different decisions about statistical significance, this will not always be the case. There are situations out there where you'll end up making different decisions based upon which type of Sums of Squares you use. Thus, you want to think carefully about what is most appropriate in your current context.

10 Code Appendix

```r
# 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, quietly = 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
gummyData <- openxlsx::readWorkbook(
  xlsxFile = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/gummyBears_Fall2023.xlsx",
  sheet = 1,
  colNames = TRUE,
  rowNames = FALSE
)

### Impose a logical ordering on the angle factor
gummyData$angle <- factor(
  x = gummyData$angle,
  levels = c("Flat", "Low", "High")
)

gummyData$position <- as.factor(gummyData$position)
gummyData$team <- as.factor(gummyData$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 = ","
)
```

```r
## Clean data
### If you are using dplyr version 1.1.0+
batteryData$temperature <- dplyr::case_match(
  .x = batteryData$temperature,
  15 ~ "15ºF",
  70 ~ "70ºF",
  125 ~ "125ºF",
  .ptype = factor(levels = c("15ºF", "70ºF", "125ºF"))
)

batteryData$material <- dplyr::case_match(
  .x = batteryData$material,
  1 ~ "Plate 1",
  2 ~ "Plate 2",
  3 ~ "Plate 3",
  .ptype = factor(levels = c("Plate 1", "Plate 2", "Plate 3"))
)

### If you are using dplyr version < 1.1.0
# 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 = gummyData,
  mapping = aes(
    x = angle,
    y = distance,
    fill = position
  )
) +
  geom_boxplot() +
  theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (cm)") +
  labs(
    fill = "Launch Position"
  ) +
  theme(
    legend.position = "bottom",
    text = element_text(size = 14)
  )

# Demo code for descriptive statistics for factorial treatment structure ----
## Using the psych package's describeBy function
## Battery Manufacturing
batteryStats <- psych::describeBy(
  life ~ temperature + material, # Notice that we use the + as an "AND"
  data = batteryData,
  na.rm = TRUE,
  skew = TRUE,
  ranges = TRUE,
  quant = c(0.25, 0.75),
  IQR = TRUE,
  mat = TRUE,
  digits = 4
)

batteryStats %>%
  tibble::remove_rownames() %>%
  dplyr::select(
    group1, group2, 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("l", "c"), times = c(2, 11)),
    col.names = c("Temperature", "Material", "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 ----
## Using dplyr's summarize function with custom choices of statistics
## Gummy Bears
gummyData %>%
  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),
    .groups = "drop"
  ) %>%
  knitr::kable(
```

```r
    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 For Battery Study ----
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, 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 = gummyData,
  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", linewidth = 1) +
  geom_jitter(width = 0.1, height = 0.1, alpha = 0.25, size = 1) +
  ggplot2::theme_bw() +
  xlab("Launch Angle") +
  ylab("Distance (cm)") +
  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
gummyModel <- aov(
  formula = distance ~ angle*position,
  data = gummyData
)

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

## Tukey-Anscombe Plot for Battery Study ----
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,
    linewidth = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (hours)") +
  ylab("Residuals (hours)")

# Index Plots for Gummy Bear Study ----
ggplot(
  data = gummyData,
  mapping = aes(
    x = order,
    y = distance
  )
) +
  geom_point(size = 0.5) +
  geom_line() +
  theme_bw() +
  xlab("Measurement order") +
  ylab("Distance (cm)") +
  facet_wrap(
    facets = vars(team),
    scales = "fixed"
  )

# Omnibus Test/Modern ANOVA Table ---
## Battery Model
parameters::model_parameters(
  model = batteryModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  type = 3, # 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 Eta Sq.", "Partial Omega 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 Battery Manufacturing Study",
  booktabs = TRUE,
  align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    # latex_options = c("HOLD_position")
    # I commented out the above so the computer would find "best" placement
  )

# Demo code for main effects pairwise comparisons ----
## Battery Life Example, Plate Material
platePostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material,
  adjust = "tukey",
  level = 0.9
)

knitr::kable(
  x = platePostHoc$contrasts,
  digits = 3,
  caption = "Post Hoc Comparisons for Main Effect Plate Material",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value"),
  align = "lccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo code for main effects pairwise with effect sizes ----
## Battery Life Study, Plate Material
### Create effect size table
plateEffects <- as.data.frame(
  eff_size(
    object = platePostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
```

```r
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, effect.size, ps)

### Build table
as.data.frame(platePostHoc$contrasts) %>%
  left_join(
    y = plateEffects,
    by = join_by(contrast == contrast)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Main Effect Plate Material",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value", "Cohen's d",
              "Prob. of Superiority"),
  align = "lccccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )


# Demo code for conditional pairwise comparisons ----
## Battery Life Example, Plate Material Conditioned on Temp
plateTempPostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material | temperature, # Notice the use of the |
  adjust = "BH",
  level = 0.92
)

knitr::kable(
  x = plateTempPostHoc$contrasts,
  digits = 3,
  caption = "Post Hoc Comparisons for Plate Material Conditioned by Temperature",
  col.names = c("Pair", "Temp.", "Difference", "SE", "DF", "t", "p-value"),
  align = "llccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo code for conditional pairwise comparisons with effect sizes ----
## Battery Life Study, Plate Material Conditioned on Temp
### Create effect size table
plateTempEffects <- as.data.frame(
  eff_size(
    object = plateTempPostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
```

```r
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, temperature, effect.size, ps)

### Build table
as.data.frame(plateTempPostHoc$contrasts) %>%
  left_join(
    y = plateTempEffects,
    by = join_by(contrast == contrast, temperature == temperature)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Plate Material Conditioned by Temperature",
  col.names = c("Pair", "Temp.", "Difference", "SE", "DF", "t", "p-value",
                "Cohen's d", "Prob. of Superiority"),
  align = "llcccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )


# Demo code for Pairwise Comparison of Interaction Term ----
## Battery Study, Treatments

## Generate pairwise comparisons
treatmentPostHoc <- emmeans::emmeans(
  object = batteryModel,
  specs = pairwise ~ material:temperature, # notice the interaction
  ajust = "tukey",
  level = 0.9
)

## Generate effect sizes
treatmentEffects <- as.data.frame(
  eff_size(
    object = treatmentPostHoc,
    sigma = sigma(batteryModel),
    edf = df.residual(batteryModel)
  )
) %>%
  dplyr::mutate( # The eff_size command places the pairs inside parentheses
    contrast = gsub(pattern = "[()]", replacement = "", x = contrast),
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, effect.size, ps)

## Build table
as.data.frame(treatmentPostHoc$contrasts) %>%
  left_join(
    y = treatmentEffects,
    by = join_by(contrast == contrast)
  ) %>%
  kable(
    digits = 3,
    caption = "Post Hoc Comparisons of Battery Study Treatments",
    col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value",
                  "Cohen's d", "Prob. of Superiority"),
    align = "lcccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("scale_down") # let computer place table where it fits
  )
# Demo Code for main effects contrasts ----
## Get the appropriate means from the model
plateMeans <- emmeans::emmeans(
  object = batteryModel,
  specs = ~ material # Nothing goes on the left of ~; list what term you want
)

# plateMeans # Look at the output object to double check the order of levels

## Apply the contrasts
plateContrasts1 <- emmeans::contrast(
  object = plateMeans, # Notice that this is the means object
  method = list(
    "Plate 1 vs. Plates 2 & 3" = c(1, -1/2, -1/2)
  ),
  adjust = "none" # No MC/SI adjustment
)

## Add effect sizes and make a nice looking table
as.data.frame(plateContrasts1) %>%
  dplyr::mutate(
    cohen = effectsize::t_to_d(t = t.ratio, df_error = df)$d, # Effect Sizes
    ps = probSup(cohen) # Effect sizes; this comes from Neil's ANOVA toolkit
  ) %>%
  kable(
    digits = 3,
    caption = "Battery Life Main Effects Contrast on Plate Material",
    col.names = c("Contrast", "Difference", "SE", "DF", "t Statistic",
                  "p-value", "Cohen's d", "Prob. of Superiority"),
    align = "lcccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )
# Demo Code for main effects contrasts ----
## Get the appropriate means from the model
treatmentMeans <- emmeans::emmeans(
  object = batteryModel,
  specs = ~ material:temperature
)
```

```r
# treatmentMeans #Be sure to look at the output so you can build your vectors
## Apply the contrasts
treatmentContrasts1 <- emmeans::contrast(
  object = treatmentMeans, # Notice that this is the means object
  method = list(
    "Plate 1 at 15ºF vs. Plate 2 at 70ºF" = c(1,0,0,0,-1,0,0,0,0),
    "Plate 1 vs Plates 2 & 3, all at 70ºF" = c(0,0,0,1,-1/2,-1/2,0,0,0),
    "Plate 1-all temps vs. Plate 2-all temps" = c(1/3,-1/3,0,1/3,-1/3,0,1/3,-1/3,0)
  ),
  adjust = "bonferroni"
)

## Add effect sizes and make a nice looking table
as.data.frame(treatmentContrasts1) %>%
  dplyr::mutate(
    cohen = effectsize::t_to_d(t = t.ratio, df_error = df)$d,
    ps = probSup(cohen)
  ) %>%
  kable(
    digits = 3,
    caption = "Treatment Contrasts for Battery Study--Bonferroni Adjusted",
    col.names = c("Contrast", "Difference", "SE", "DF", "t Statistic",
                  "p-value", "Cohen's d", "Prob. of Superiority"),
    align = "lcccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )

# 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,
  effectsize_type = c("eta", "omega", "epsilon"),
  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,
  effectsize_type = c("eta", "omega", "epsilon"),
  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,
  effectsize_type = c("eta", "omega", "epsilon"),
  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")
  )
```