

Repeated Measures

Neil J. Hatfield

4/22/2022

In this tutorial we will take a look at Repeated Measures ANOVA models. Keep in mind that there are **TWO** different types of Repeated Measures models: applying every treatment to each subject/measurement unit (a.k.a. “Within Subjects”) and measuring each subject/measurement unit multiple times but each only gets one treatment (a.k.a “Nested Repeated Measures”).

Make sure that you carefully examine the study design so that you correctly identify which type of Repeated Measures design you have. Here is the general structure of this guide/tutorial:

- Important Starting Considerations
 - Convention
 - Approaches
 - Identify Type of Repeated Measures
 - Wide and Long Formats of Data Frames
- Setting Up R
 - Loading Packages, Setting Options, and Additional Tools
 - Key Packages, `rstatix`, `lme4`, `nlme`
- Within Subjects Repeated Measures Design
 - Example Context
 - Fit the Model
 - * Appropriateness and Hasse Diagram
 - * Form the Models
 - Assessing Assumptions
 - * Gaussian Residuals
 - * Gaussian Subject Effects
 - * Homoscedasticity
 - * Independence of Observations up to Subject
 - * No Interaction between Subjects and Factor
 - * Sphericity
 - Results
 - * Omnibus
 - Relative Efficiency
 - Adjusting for Violation of Sphericity
 - * Point Estimates
 - * Post Hoc-Pairwise Comparisons
 - * Post Hoc Effect Sizes
- Nested Repeated Measures
 - Example Context
 - Fit the Model
 - * Appropriateness and Hasse Diagram
 - * Form the Models
 - Assessing Assumptions
 - * Gaussian Residuals

- * Gaussian Treatment Effects
- * Homoscedasticity
- * Independence of Subjects
- * Interaction of Subject and Factor
- * Sphericity
- Results
 - * Omnibus
 - * Point Estimates
 - * Post Hoc-Pairwise Comparisons
 - * Effect Sizes

Important Starting Considerations

There are a few important considerations you need to think through before you get too far into the process of analyzing Repeated Measures designs.

Conventions

In Repeated Measure designs, our measurement units are not as straightforward as in our other designs. To help signal this, we often use the term *subjects* to denote the living being (or object) who produces the multiple observations. This has two important consequences.

First, our last node in the Hasse diagram will no longer be the measurement units (i.e., objects/living beings) but rather the observations. Second, our subjects are almost always presumed to be randomly sampled from a broader population. Thus, they are *random effects* and need to be marked as such in our Hasse diagram.

Approaches

What I present here is **ONE** approach to using R to conduct analysis of Repeated Measures designs. This is one particular of analysis where there are dozens of different approaches which have their own champions. In working on putting together this resource for you, I easily came across dozens of different guides all using different formulations and different packages. This is my attempt to provide *one* set of approaches that is mostly consistent with all of our other approaches.

Identify Type of Repeated Measures

Repeated Measures Designs are almost exclusively used in experimental settings. To decide on which type of Repeated Measures Design you're facing, ask yourself the following questions:

- Who are the subjects?
- What are the treatments?
- How many different treatments does each subject get?
 - Answer: All of them → **Within Subjects Repeated Measures**
 - Answer: Only one → **Nested Repeated Measures**

Wide and Long Formats of Data Frames

In both types of Repeated Measures Designs, we will need the data to be arranged in two formats: “wide” and “long”. These terms refer to the construction of the data frame.

A “long data frame” is what we're most used to working with. Here, each row represents a unique combination of Subject & Treatment or Subject & Time Point. If we have n subjects, and g treatments (or t time points of measurement), we should have a total of $n \cdot g$ rows (alternatively, $n \cdot t$ rows). Our response is a single column. Visually, imagine your data frame as a rectangle that is taller than it is wide.

A “wide data frame” is a re-arrangement. Here, each subject gets one and only one row. Instead of a single response column, we have multiple response columns. In fact, we’ll have a separate response column for each of the g treatments (or t time points of measurement). Now imagine your data frame as a rectangle that is wider than it is tall.

One of the first challenges you must tackle in analyzing Repeated Measures data is identifying which of these formats your data is currently in. Then creating a new data frame that is in the other format. In these situations, I tend to not include **Data** on the end of my object name; rather, I use either **Long** or **Wide** so that I have a reminder of which format I’m calling.

Transforming Data Frame Formats

Thankfully, we have some useful functions to help us. As part of the **tidyverse**, the package **tidyr** gives us the functions **pivot_wider** and **pivot_longer**. Given the imagery of the rectangles, you can imagine turning (pivoting) the long rectangle into the wide rectangle and vice versa. This imagery can help you keep in mind that **pivot_wider** takes a long data frame and makes a wide data frame. The **pivot_longer** function starts with a wide data frame and returns a long data frame.

```
# Generic Demo Code for creating a wide data frame
# Note: this code assumes you already read in a long format data frame

#IMPORTANT: you will need to the code to match the names in your actual situation

dataWide <- pivot_wider(
  data = dataLong, # "dataLong" is the name of the long format data frame
  names_from = group, #"group" is the name of the column that contains your treatments
  values_from = response #"response" is the name of the column with the response values
)
```

```
# Generic Demo Code for creating a long data frame
# Note: this code assumes you already read in a wide format data frame

#IMPORTANT: you will need to the code to match the names in your actual situation

dataLong <- pivot_longer(
  data = dataWide, # "dataWide" is the name of the wide format data frame
  cols = !subject, # Says to not use the "subject" column
  names_to = "group", # This is the new column you want the treatments to go to
  names_transform = list(group = as.factor), # Makes the treatment column a factor
  values_to = "response" # A new column that will contain all of the response values
)
```

You’ll be able to see the above code examples in action in the examples below.

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.

The core set of packages to load in include the following:

- **tidyverse**—for data cleaning/wrangling and the pipe, `%>%`
- **knitr** & **kableExtra**—for making professional looking tables
- **parameters**—to construct modern ANOVA tables, get omnibus effect sizes, and to switch out the type of *Sums of Squares*
- **hasseDiagram**—to construct Hasse diagrams
- **car** for nice QQ Plots
- **psych**—for easy descriptive statistics by group
- **emmeans**—for getting point estimates which attend to our models as well as doing post hoc analyses
- **rstatix**—to help with assessing sphericity
- **lme4** and **nlme**—to help assess random effects

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", "emmeans", "rstatix", "lme4", "nlme")
lapply(packages, library, character.only = TRUE)

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

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

Within Subjects Repeated Measures Design

The Within Subjects Repeated Measures Design has the hallmark that each subject will be given each and every treatment. To prevent an order effect, we should randomize the order in which each subject gets the treatments.

Example Context—Taste Testing Beer

Beer is big business; the craft brewing industry contributed \$79.1 billion to the US Economy in 2018 and 550,000+ jobs (PA: \$6.335 billion).

Getting a craft beer scored can be quite the achievement. In a single blind tasting, judges are given a chilled, properly poured beer and told the style category. They then judge the beer on Aroma (24 pts), Appearance (6 pts), Flavor (40 pts), Mouthfeel (10 pts), and Overall Impression (20 pts).

We have decided to put several State College beers to the test:

- Barnstormer (IPA, Happy Valley Brewing Company)
- Craftsman (Brown, HVBC)
- Red Mo (Red, Otto's)
- King Richard Red (Amber, Robin Hood)

For this study, we have used a lottery to select six individuals to act as judges. Each judge will be presented with samples of the four beers and they will score each beer. The order in which each judge samples/scores the beers will be determined by the research team drawing labeled tokens without replacement.

Fit the Model

- Example Context
- Fit the Model
 - Appropriateness and Hasse Diagram
 - Form the Models
- Assessing Assumptions
 - Gaussian Residuals
 - Gaussian Subject Effects
 - Homoscedasticity
 - Independence of Observations up to Subject
 - No Interaction between Subjects and Factor
 - Sphericity
- Results
 - Omnibus

- * Relative Efficiency
 - * Adjusting for Violation of Sphericity
- Point Estimates
- Post Hoc-Pairwise Comparisons
- Post Hoc Effect Sizes
- Nested Repeated Measures
 - Example Context
 - Fit the Model
 - * Appropriateness and Hasse Diagram
 - * Form the Models
 - Assessing Assumptions
 - * Gaussian Residuals
 - * Gaussian Treatment Effects
 - * Homoscedasticity
 - * Independence of Subjects
 - * Interaction of Subject and Factor
 - * Sphericity
 - Results
 - * Omnibus
 - * Point Estimates
 - * Post Hoc-Pairwise Comparisons
 - * Effect Sizes

For this particular study, we can express the Repeated Measures-Within Subjects design with the following Hasse diagram (Figure 1).

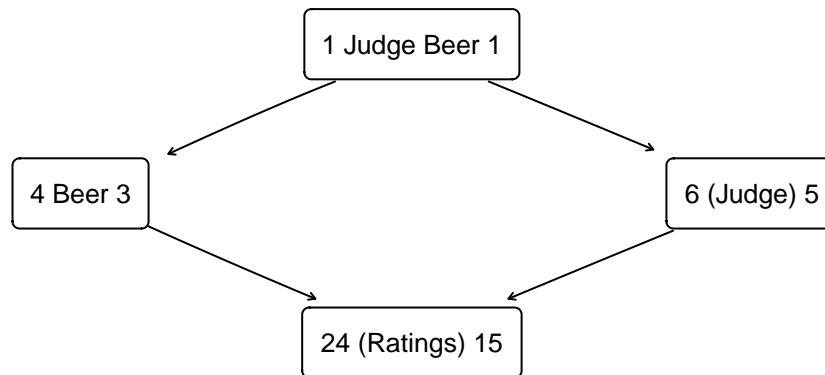


Figure 1: Hasse Diagram for Beer Judging

Notice that this layout looks a lot like a Randomized Complete Block Design (RCBD). For a one-way (single factor) experiment with repeated measures on all treatments, we end up with an essentially identical approach.

Data and Format of the Data Table

The following are the data for this situation:

```

# Beer Data
beer <- data.frame(
  judge = as.factor(rep(x = LETTERS[1:6], each = 4)),
  beer = as.factor(rep(
    x = c("Barnstormer", "King Richard Red", "Craftsman", "Red Mo"),
    times = 6)),
  score = c(
    50, 60, 70, 70,
    38, 45, 58, 60,
  )

```

```

45, 48, 60, 58,
65, 65, 75, 75,
55, 60, 70, 65,
48, 53, 68, 63
)
)

```

For the methods we're going to look at, we need our data in the long format: each row is unique observation. Fortunately, our data are already in this format.

If our data were in the wide format (i.e., each row refers to a single subject and *all* of their observations), we would see something like the following for a data table:

Long to Wide Format Transformation

For assessing Compound Symmetry/Sphericity, we will need our data in the wide format. We will use the `pivot_wider` function from the `tidyr` package to transform our data into a wide format.

```

beerWide <- pivot_wider(
  data = beer,
  names_from = beer,
  values_from = score
)

```

```
beerWide
```

```

## # A tibble: 6 x 5
##   judge Barnstormer 'King Richard Red' Craftsman 'Red Mo'
##   <fct>      <dbl>          <dbl>      <dbl>      <dbl>
## 1 A          50            60          70          70
## 2 B          38            45          58          60
## 3 C          45            48          60          58
## 4 D          65            65          75          75
## 5 E          55            60          70          65
## 6 F          48            53          68          63

```

Wide to Long Format Transformation

If your data happen to be in the wide format, we can switch to the long format via the `pivot_longer` function from the `tidyr` package (part of the `tidyverse`).

```

beerLong <- pivot_longer(
  data = beerWide,
  cols = !judge, # This says to use all except the judge column
  names_to = "beer", # what column to place the beer names in
  names_transform = list(beer = as.factor), # turn beer to a factor
  values_to = "score" # what column to put the values in
)

```

```
head(beerLong, n = 5) # Display the first 5 rows.
```

```

## # A tibble: 5 x 3
##   judge beer      score
##   <fct> <fct>      <dbl>
## 1 A    Barnstormer    50
## 2 A    King Richard Red  60
## 3 A    Craftsman      70
## 4 A    Red Mo         70
## 5 B    Barnstormer    38

```

Explore the Data

Just as with all of our other models, you should explore the data through data visualizations and descriptive statistics.

Fit the Models

We are going to need to fit three (3) models for Within Subjects designs:

- 1) We will fit a One-way ANOVA + Block (RCBD) model for our ANOVA table.
- 2) We will fit a Mixed Effects model for estimating judge effects and assess assumptions about the judge factor.
- 3) We will use a Nested Approach via the `rstatix` package for assessing Compound Symmetry/Sphericity assumption.

```
# Omnibus Model (for our ANOVA table)
beerOmni <- aov(
  formula = score ~ judge + beer,
  data = beerLong
)

# Random Effect Model (Assumption Check and Point Estimation)
beerMixed <- lme4::lmer(
  formula = score ~ (1|judge) + beer,
  data = beerLong
)

# Compound Symmetry/Sphericity Assessment
beerSphere <- rstatix::anova_test(
  data = beerLong,
  formula = score ~ beer + Error(judge %in% beer)
)

## The %in% tells R that judge should be treated as nested in beer
```

Assess the Assumptions

For Within Subjects Repeated Measures designs, we have the following assumptions:

- 1) Our residuals follow a Gaussian distribution,
- 2) Subject effects follow a Gaussian distribution (just like a Random Effect),
- 3) Homoscedasticity around the model,
- 4) Independence of Observations up to Subject; that is, observations should be independent between subjects much like for RCBDs,
- 5) No interaction between Subjects and Factor (just like RCBDs), and
- 6) We have Sphericity.

Gaussian Residuals

Use a QQ plot:

```
# QQ plot for residuals
car::qqPlot(
  x = residuals(beerMixed),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (score)"
)
```

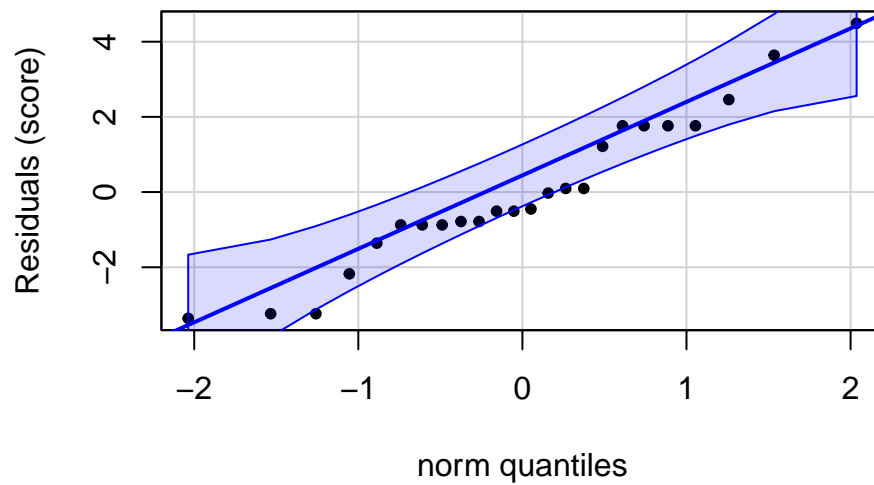


Figure 2: QQ Plot of Beer Judging Residuals

Gaussian Treatment Effects

Again, use a QQ plot.

```
# Judge Effects
car::qqPlot(
  x = unlist(
    lme4::ranef(
      object = beerMixed,
      whichel = c("judge")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Judge Effects"
)
```

Homoscedasticity

Use a Tukey-Anscombe Plot

```
# Tukey-Anscombe Plot
ggplot(
  data = data.frame(
    residuals = residuals(beerMixed),
    fitted = fitted.values(beerMixed)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline(
    yintercept = 0,

```

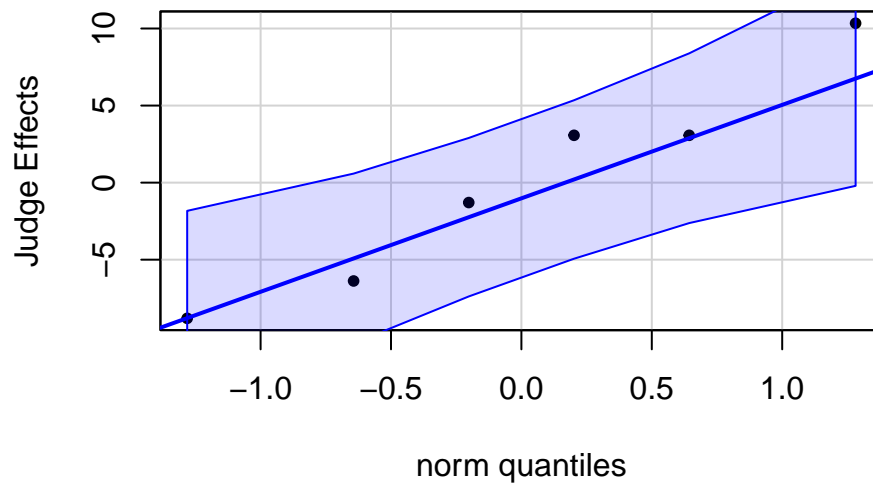



Figure 3: QQ Plot for Judge Effects

```
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 (score)") +
ylab("Residuals (score)")
```

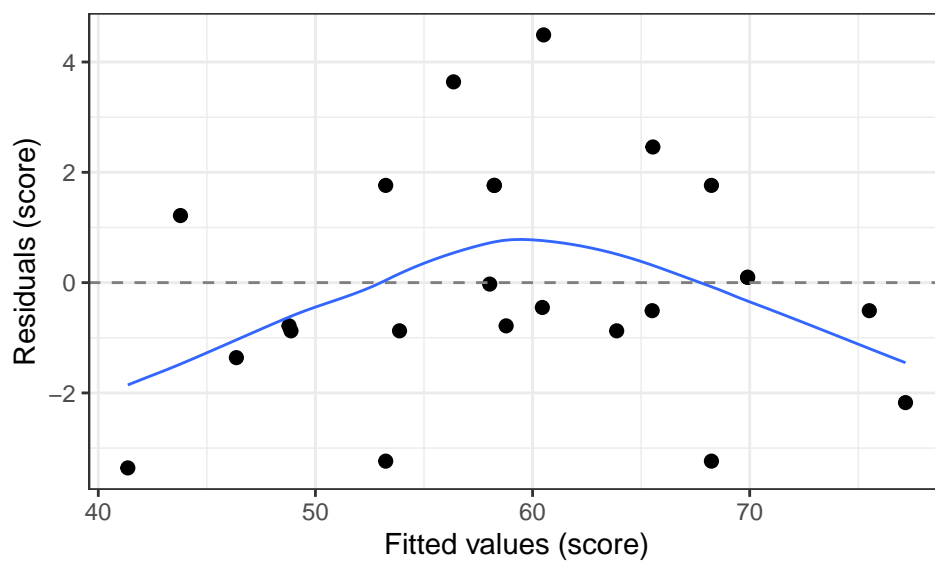


Figure 4: Tukey-Anscombe Plot for Beer Judging Study

Independence of Subjects

If you happen to know measurement order, you can look at an Index plot. However, be sure that you incorporate the subject attribute into the plot (in this situation, the judge). This can help highlight that whether any patterns are due to our subjects (and therefore anticipated) or if we have any additional patterns threats. Treat this situation just like checking for independence of subjects in a RCBD.

Interaction of Subject and Factor

We will want to make an interaction plot to look for consistency between the judges and the beers; again, think about what we would do with a RCBD.

```
# Interaction Plot
ggplot(
  data = beer,
  mapping = aes(
    x = beer,
    y = score,
    color = judge,
    group = judge
  )
) +
  geom_point(size = 2) +
  geom_line() +
  ggplot2::theme_bw() +
  xlab("Beer") +
  ylab("Score") +
  labs(
    color = "Judge"
  )
)
```

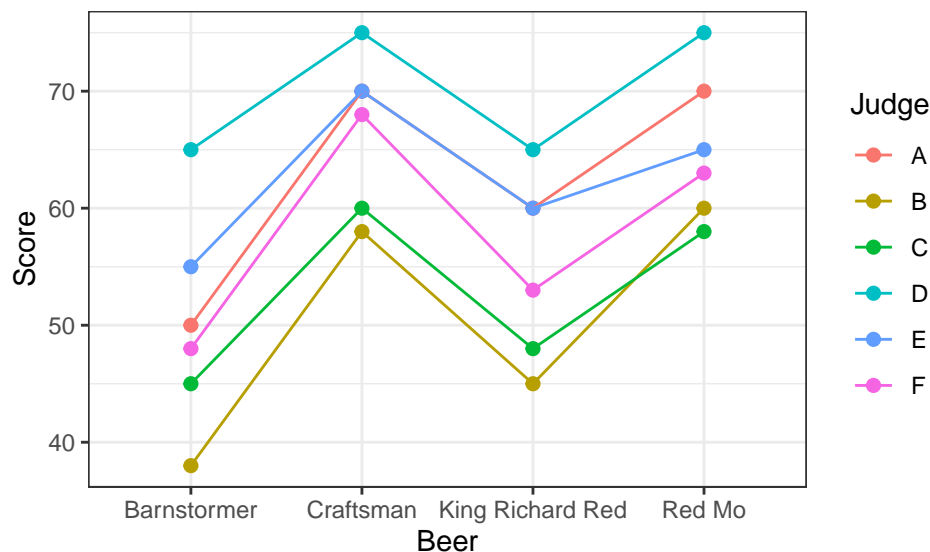


Figure 5: Interaction Plot for Beer and Judge

Sphericity

The idea behind sphericity is that we have essentially the same levels of variation for the *differences between treatments*. While there is a visual method we can use here, we do need to do so with some caution. Much like looking for homoscedasticity, we'll want to see if any difference has *excessively different* variation than another difference. Unlike homoscedasticity **there is no**

rule of thumb/guideline (e.g., more than twice) for sphericity. Thus, this is one assumption where we will supplement with a formal test: Mauchly's Test of Sphericity.

```
# Sphericity Plot
sphericityPlot(
  dataWide = beerWide, # Data needs to be in wide format
  subjectID = "judge" # character name of the subject column
)
```

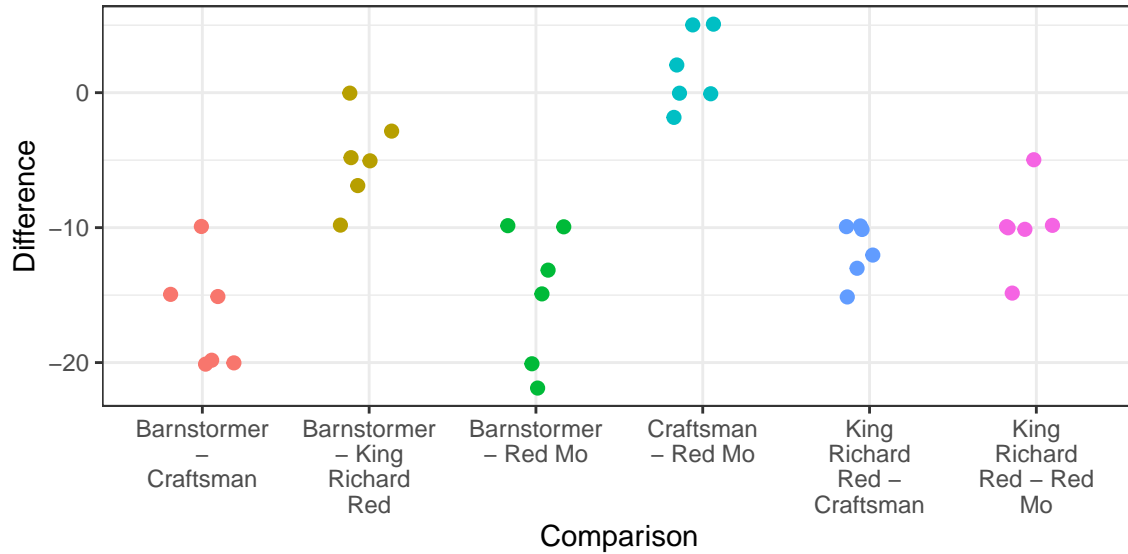


Figure 6: Sphericity Plot for Beer Judging

```
## Look for any groups that have very different amounts of variation
```

The null hypothesis for Mauchly's Test is that there is **no** violation of Sphericity (Compound Symmetry); under this hypothesis, Mauchly's Test Statistic, W follow a χ^2 with 2 *degrees of freedom*. To see the results of Mauchly's test, we will use the following code:

```
# Mauchly's Test
beerSphere$`Mauchly's Test for Sphericity` %>%
  dplyr::select(Effect, W, p) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Effect", "Mauchly's W", "p"),
    caption = "Mauchly's Sphericity Test",
    align = c("l", "c", "c"),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 1: Mauchly's Sphericity Test

| Effect | Mauchly's W | p |
|--------|-------------|-------|
| beer | 0.365 | 0.597 |

Thus, we have $W = 0.365$ with a p -value of 0.597 (use your overall Type I Error Risk to set the Unusualness Threshold here). This is one of those strange cases where we *want to fail to reject* the null hypothesis.

Results

Omnibus Test

For our omnibus test, we will use our RCBD equivalent design:

```
beerTable1 <- parameters::model_parameters(
  model = beerOmni, # the RCBD with all fixed effects
  omega_squared = "partial",
  eta_squared = "partial",
  epsilon_squared = "partial"
)
beerTable1$p <- lapply(
  X = beerTable1$p,
  FUN = pvalRound
)

knitr::kable(
  x = beerTable1,
  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 Beer Judging 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 2: ANOVA Table for Beer Judging Study

| Source | SS | df | MS | F | p-value | Partial Omega Sq. | Partial Eta Sq. | Partial Epsilon Sq. |
|-----------|----------|----|----------|---------|----------|-------------------|-----------------|---------------------|
| judge | 1045.833 | 5 | 209.1667 | 32.8534 | < 0.0001 | 0.869 | 0.9163 | 0.8884 |
| beer | 1150.000 | 3 | 383.3333 | 60.2094 | < 0.0001 | 0.881 | 0.9233 | 0.9080 |
| Residuals | 95.500 | 15 | 6.3667 | | | | | |

Notice that I slipped the `pvalRound` function in to take care of the fact that the p -values are excessively small and were coming out as 0 after rounding. We interpret the terms in this table exactly as we have been all semester.

Efficiency of Repeated Measures

Since the One-way Within Subjects subjects is like a RCBD, we can get a measure of the efficiency of such a design versus a CRD.

```
# Use the block relative efficiency function
block.RelEff(
  aov.obj = beerOmni,
  blockName = "judge",
  trtName = "beer"
)
```

```
## [1] "The relative efficiency of the block, judge, is 7.715."
```

Thus, we would need 8 times as many ratings for each beer as what we used in order to get the same level of information. That would mean that we would need around 48 scores for each beer.

What if Sphericity is Violated?

If Sphericity is violated (i.e., Mauchly's Test leads you to reject the null hypothesis), we are not out of luck. When we fit the model to check for Sphericity, we also automatically got two corrected tests: the Greenhouse-Geisser and the Huynh-Feldt corrections:

```
correctedTable <- beerSphere$`Sphericity Corrections` %>%
  dplyr::select(GGe, `p[GG]`, HFe, `p[HF]`)
correctedTable$`p[GG]` <- lapply(
  X = correctedTable$`p[GG]`,
  FUN = pvalRound
)
correctedTable$`p[HF]` <- lapply(
  X = correctedTable$`p[HF]`,
  FUN = pvalRound
)
knitr::kable(
  x = correctedTable,
  digits = 4,
  col.names = c("Greenhouse-Geisser", "p-value", "Huynh-Feldt", "p-value"),
  caption = "Sphericity Corrections",
  align = "c",
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 3: Sphericity Corrections

| Greenhouse-Geisser | p-value | Huynh-Feldt | p-value |
|--------------------|----------|-------------|----------|
| 0.613 | < 0.0001 | 0.952 | < 0.0001 |

For the Corrections, the p -values are the adjusted p -values for the **omnibus** test. Thus, we would say that there is an effect due to our repeated measures model (i.e., there is a difference in the scores of the beer).

Post Hoc Analysis

For Post Hoc Analysis, we will want to make use of the **emmeans** package and make sure that we're looking at the correct aspect of our model. We already assume that there's some difference in the judges, thus we are really just after the marginals of our other factor(s). In this situation, the type of beer.

```
# Using emmeans
beerPH <- emmeans::emmeans(
  object = beerMixed, # Notice the use of the mixed model here
  specs = pairwise ~ beer,
  adjust = "tukey",
  level = 0.92
)
```

Point Estimates You can get point estimates for the Beer effects:

```
## Point Estimates
as.data.frame(beerPH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Type of Beer", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 92\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 4: Marginal Means-Tukey 92% Adjustment

| Type of Beer | Marginal Mean | SE | DF | Lower Bound | Upper Bound |
|------------------|---------------|-------|--------|-------------|-------------|
| Barnstormer | 50.1667 | 3.084 | 5.9383 | 43.6635 | 56.6698 |
| Craftsman | 66.8333 | 3.084 | 5.9383 | 60.3302 | 73.3365 |
| King Richard Red | 55.1667 | 3.084 | 5.9383 | 48.6635 | 61.6698 |
| Red Mo | 65.1667 | 3.084 | 5.9383 | 58.6635 | 71.6698 |

Pairwise Comparisons You can also do the standard pairwise comparisons of the beer types.

```
# Pairwise Comparisons
as.data.frame(beerPH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Marginal Means-Tukey 92\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 5: Marginal Means-Tukey 92% Adjustment

| Comparison | Difference | SE | DF | t Statistic | p-value |
|--------------------------------|------------|--------|----|-------------|---------|
| Barnstormer - Craftsman | -16.6667 | 1.4568 | 15 | -11.4407 | 0.0000 |
| Barnstormer - King Richard Red | -5.0000 | 1.4568 | 15 | -3.4322 | 0.0174 |
| Barnstormer - Red Mo | -15.0000 | 1.4568 | 15 | -10.2966 | 0.0000 |
| Craftsman - King Richard Red | 11.6667 | 1.4568 | 15 | 8.0085 | 0.0000 |
| Craftsman - Red Mo | 1.6667 | 1.4568 | 15 | 1.1441 | 0.6692 |
| King Richard Red - Red Mo | -10.0000 | 1.4568 | 15 | -6.8644 | 0.0000 |

Effect Sizes

For Post Hoc Effect Sizes, you'll need to use the `emmeans` package and my `probSup` function:

```
tempEMM <- emmeans::emmeans(  
  object = beerMixed,  
  specs = "beer"  
)  
  
# Pass the stored marginals into the effect size function  
cohenTemp <- emmeans::eff_size(  
  object = tempEMM,  
  sigma = sigma(beerMixed),  
  edf = df.residual(beerMixed)  
)  
  
# Create a data frame, add on the probability of superiority  
# Send that data frame into a nice table  
as.data.frame(cohenTemp) %>%  
  dplyr::mutate(  
    ps = probSup(effect.size),  
    .after = effect.size  
  ) %>%  
  dplyr::select(contrast, effect.size, ps) %>%  
  knitr::kable(  
    digits = 3,  
    col.names = c("Comparison", "Cohen's d", "Probability of Superiority"),  
    align = "lcc",  
    caption = "Effect Sizes for Beer",  
    booktab = TRUE  
  ) %>%  
  kableExtra::kable_styling(  
    bootstrap_options = c("striped", "condensed"),  
    font_size = 12,  
    latex_options = "HOLD_position"  
  )
```

Table 6: Effect Sizes for Beer

| Comparison | Cohen's d | Probability of Superiority |
|--------------------------------|-----------|----------------------------|
| Barnstormer - Craftsman | -6.605 | 0.000 |
| Barnstormer - King Richard Red | -1.982 | 0.081 |
| Barnstormer - Red Mo | -5.945 | 0.000 |
| Craftsman - King Richard Red | 4.624 | 0.999 |
| Craftsman - Red Mo | 0.661 | 0.680 |
| King Richard Red - Red Mo | -3.963 | 0.003 |

Nested Repeated Measures Context

The second kind of Repeated Measures ANOVA design deals with taking multiple measurements from our measurement units on the same attribute over time. Unlike the Within Subjects design, each measurement unit here only gets **ONE** treatment. A somewhat handy way to help you decide if you're in a Nested Repeated Measures design is to see if you can think about the situation as being like a Pre-/Post-Testing situation. This classic situation involves testing/measuring everyone before we apply treatments, then apply the treatments (each person only gets one), and then testing/measuring everyone again afterwards. If you can fit the situation in to the pre/post design, you're a Nested Repeated Measures design.

For this example, we are going to look at the impact of two advertising campaigns on the volume of sales of athletic shoes over time. Ten similar test markets were selected at random to participate in this study. The two advertising campaigns were similar in all respects except that a different national sports personality was used in each. Sales data were collected for three two-week periods (before-t1, during-t2, and after-t3)

Hasse Diagram

If you use the Hasse Diagram app, you'll need to watch out for a couple of things:

- 1) You'll need to remove the interaction of Time Point X Market Nested Campaign. (Use Markets X Time as your measurement unit.)
- 2) The *degrees of freedom* will be off for the Markets. The app isn't subtracting the *degrees of freedom* for Campaign. Thus, you'll have to manually adjust the code until I can get a fix in place.
- 3) Remember to carry your *degrees of freedom* fix through to the final node.

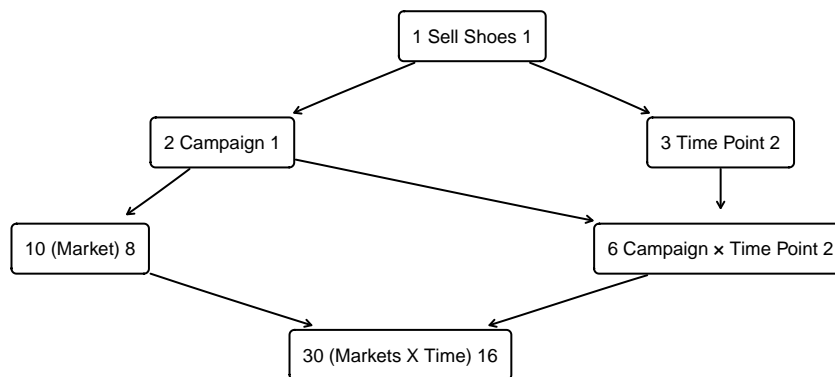


Figure 7: Hasse Diagram for Shoe Advertisement Study

Data

The shoe sales data comes to use in the wide format. We will want to make a long format version as well.

```

shoesWide <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/master/dataFiles/shoes.csv",
  header = TRUE,
  sep = ",",
)

shoesWide$campaign <- as.factor(shoesWide$campaign)
shoesWide$market <- as.factor(shoesWide$market)

# Make a long version of the data
shoesLong <- tidyr::pivot_longer(
  data = shoesWide,
  cols = dplyr::starts_with("t"),
  names_to = "time",
  names_ptypes = list("time" = factor()),
  values_to = "sales"
)

# Recode times to before, during, after and puts an ordering
shoesLong$time <- dplyr::recode_factor(
  shoesLong$time,
  "t1" = "before",
  "t2" = "during",

```



```
"t3" = "after",
  .ordered = TRUE
)
```

Explore the Data

As usual, you'll want to explore the data. With Repeated Measures data, remember that can the explore the data by looking across subjects at a particular time point or across time points (following a particular subject) or both.

Fit the Models

We will fit two models: one that we'll use for the omnibus test and one we'll use for sphericity checking.

Let's begin with our omnibus testing model. Unfortunately, there is not a nice, clean way to get a well organized table here without doing some manipulation; more on this in the results section.

The key here is that you will want to make sure that you listen/watch for a particular warning message: **Error() model is singular**. This is because our final interaction term (subject x time) uses up all remaining *degrees of freedom* so we will not have a traditional Residuals/Error term. That is the crux of this particular message.

```
# Omnibus Model
shoesModel <- aov(
  formula = sales ~ campaign * time + Error(market %in% campaign),
  data = shoesLong
)
```

Due to the lack of a typical Residuals/Error term, we will need to use an alternative route for getting things like residuals or fitted values for our assumption checking. For our second model, we will turn to the **nlme** package.

```
# Use the nlme package to fit a model that we can use for assumption checking
shoesAssumptions <- nlme::lme(
  data = shoesLong,
  fixed = sales ~ campaign * time,
  random = ~ 1|market
)
```

Assessing the Assumptions

The methods here are the same as for the Within Subjects Repeated Measures design.

Gaussian Residuals

Use a QQ plot:

```
# QQ plot for residuals
car::qqPlot(
  x = residuals(shoesAssumptions),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (sales)"
)
```

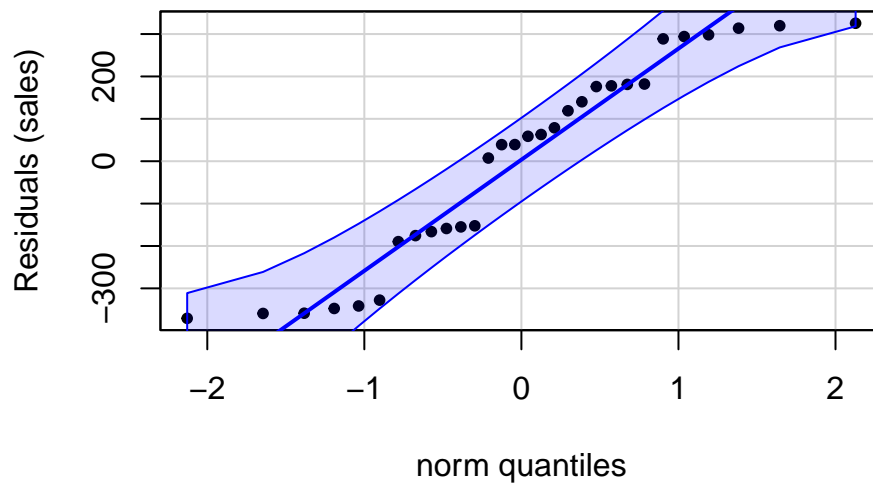


Figure 8: QQ Plot of Shoe Sales Study

Gaussian Treatment Effects

Again, use a QQ plot.

```
# Market Effects
car::qqPlot(
  x = unlist(
    lme4::ranef(
      object = shoesAssumptions,
      whichel = c("market")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Market"
)
```

Homoscedasticity

Use a Tukey-Anscombe Plot

```
# Tukey-Anscombe Plot
ggplot(
  data = data.frame(
    residuals = residuals(shoesAssumptions),
    fitted = fitted.values(shoesAssumptions)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline(
    yintercept = 0,

```

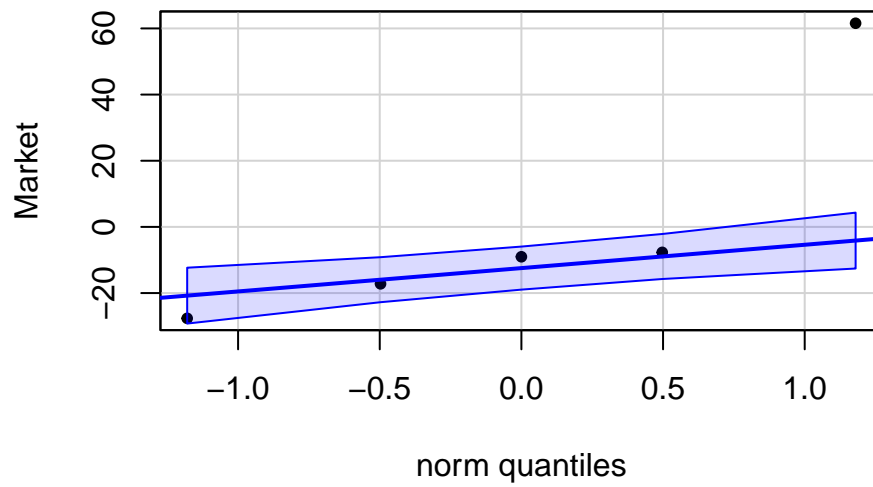


Figure 9: QQ Plot for Market Effects

```
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 (sales)") +
ylab("Residuals (sales)")
```

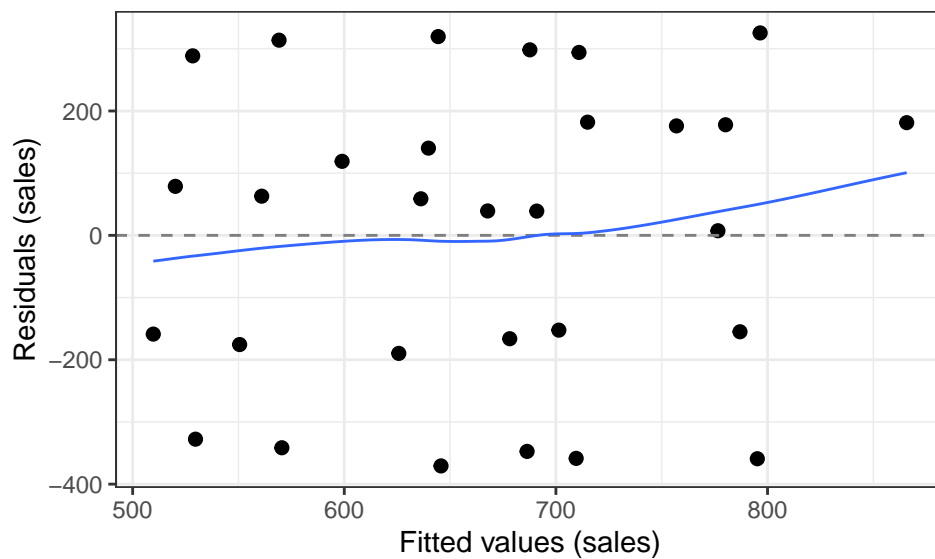


Figure 10: Tukey-Anscombe Plot for Shoe Sales Study

Independence of Subjects

One of the interesting things about Nested Repeated Measures is that we inherently know at least *some* of the the measurement order. We might not know which store got measured before which other store, but we know the sequence of measurements for each store. Thus, we can make the following plot:

```
ggplot(  
  data = shoesLong,  
  mapping = aes(  
    x = time,  
    y = sales,  
    color = paste(campaign, market, sep = ":"),  
    group = paste(campaign, market, sep = ":")  
  )  
) +  
  geom_point() +  
  geom_line() +  
  theme_bw() +  
  xlab("Time Point (relative to campaign)") +  
  ylab("Sales (coded)") +  
  labs(  
    color = "Campagin:Market"  
  )  
)
```

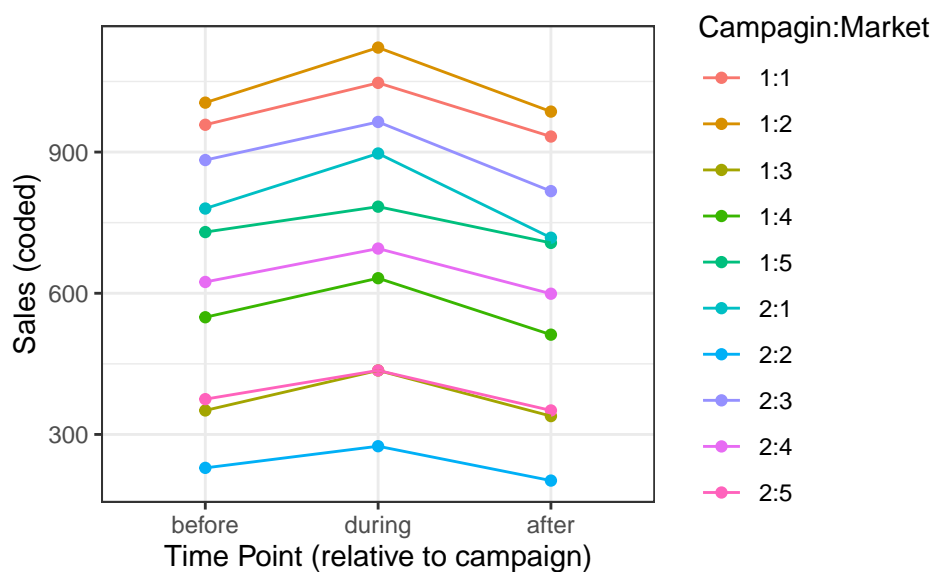


Figure 11: Shoe Sales by Time and Store

Now, this plot is great for letting us compare the effects over time. This is not necessarily the greatest for letting us see if our subjects are independent of one another. The consistency of the effects over time do indicate that the inherent dependency of each subject's measurements *is* consistent across time. If a particular market had a huge increase in sales after the campaign, that would suggest something strange happened.

For Nested Repeated Measures we will ultimately want to fall back on our study design to make a solid justification. This is why I've been pressing you all course long on including details/being specific.

Interaction of Subject and Factor

A similar plot to the previous Time Series plot is to construct a plot known as "Growth Curves". In essence, we want to see how the response changes for each subject over time... but we're going to separate the data a bit more cleanly so we can see if and how the factor might interact with our subjects.

```
# Growth Curves
ggplot(
  data = shoesLong,
  mapping = aes(
    x = time,
    y = sales,
    color = market,
    group = market
  )
) +
  geom_point(size = 2) +
  geom_line() +
  facet_wrap(facets = ~campaign) +
  ggplot2::theme_bw() +
  xlab("Time Period") +
  ylab("Sales (coded)") +
  labs(
    color = "Market"
  )
)
```

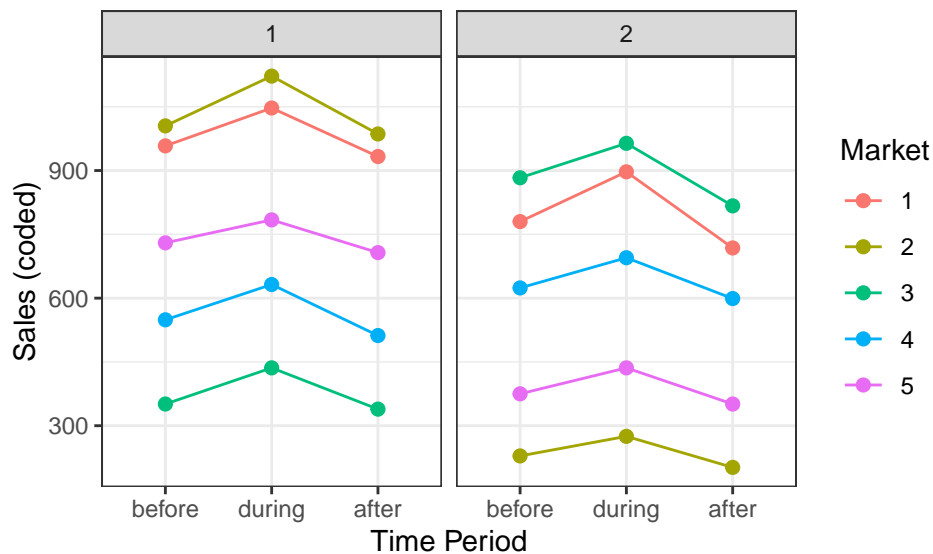


Figure 12: Growth Curves for Shoe Sales Study

Notice that we used the `facet_wrap` on campaign to split the time series plot into separate panels/facets for each campaign. If we see the same behaviors in both facets, then there is no worrisome interaction between subjects and our factor.

Keep in mind that Market 3 in Campaign 1 is *not* the same market as Market 3 in Campaign 2. They just happen to be the *third* market inside each campaign.

Sphericity

The idea behind sphericity is that we have essentially the same levels of variation for the *differences between treatments*. While there is a visual method we can use here, we do need to do so with some caution. Much like looking for homoscedasticity, we'll want to see if any difference has *excessively different* variation than another difference. Unlike homoscedasticity **there is no rule of thumb/guideline** (e.g., more than twice) for sphericity. Thus, this is one assumption where we will supplement with a formal test: Mauchly's Test of Sphericity.

```
# Sphericity Plot
sphericityPlot(
  dataWide = shoesWide,
  subjectID = c("market", "campaign"),

```

```
colsIgnore = NULL
)
```

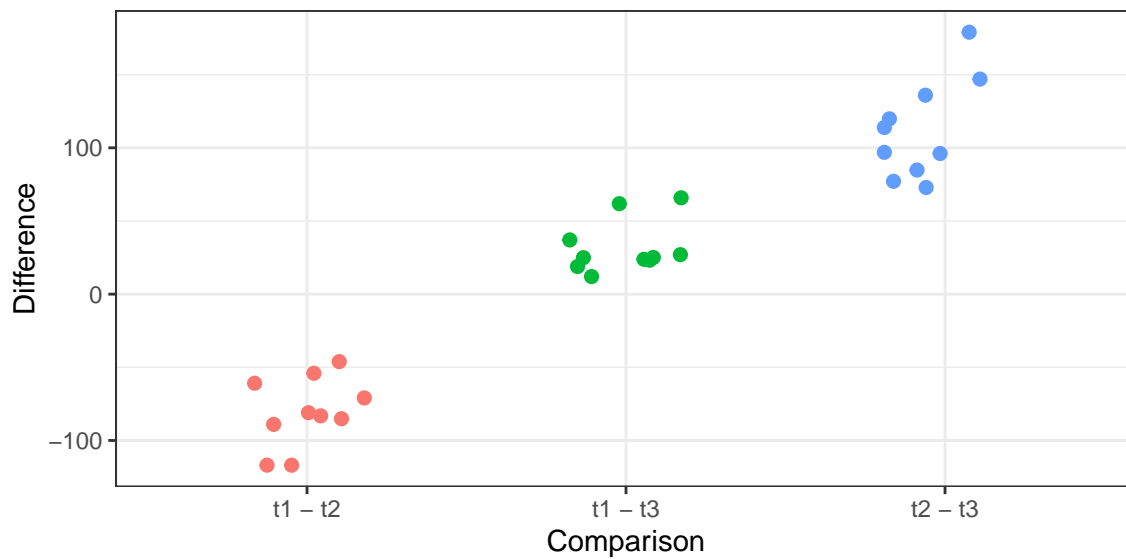


Figure 13: Sphericity Plot for Shoe Sales Study

```
## Look for any groups that have very different amounts of variation
```

The nlme package does not support Mauchly's Test for Sphericity.

Results

Omnibus Test

As mentioned, getting a nice looking table is not as straightforward with the Nested Repeated Measures ANOVA problems.

```
shoesTemp <- summary(shoesModel)
shoesOmni <- rbind(
  shoesTemp$`Error: market:campaign`[[1]],
  shoesTemp$`Error: Within`[[1]]
)

row.names(shoesOmni) <- c("campaign", "market", "time", "campaign:time", "market:time")

shoesOmni["market", "F value"] <- shoesOmni["market", "Mean Sq"] /
  shoesOmni["market:time", "Mean Sq"]
shoesOmni["market", "Pr(>F)"] <- pf(
  q = shoesOmni["market", "F value"],
  df1 = shoesOmni["market", "Df"],
  df2 = shoesOmni["market:time", "Df"],
  lower.tail = FALSE
)

shoesOmni %>%
  tibble::rownames_to_column(
    var = "Source"
  ) %>%
  knitr::kable(
    digits = 4,
```

```

col.names = c("Source", "df", "SS", "MS", "F", "p-value"),
caption = "ANOVA Table for Athletic Shoes Study",
align = c('l',rep('c',5)),
booktab = TRUE
) %>%
kableExtra::kable_styling(
  bootstrap_options = c("striped", "condensed"),
  font_size = 12,
  latex_options = c("HOLD_position")
)

```

Table 7: ANOVA Table for Athletic Shoes Study

| Source | df | SS | MS | F | p-value |
|---------------|----|--------------|-------------|----------|---------|
| campaign | 1 | 168150.5333 | 168150.5333 | 0.7336 | 0.4166 |
| market | 8 | 1833680.9333 | 229210.1167 | 640.3113 | 0.0000 |
| time | 2 | 67073.0667 | 33536.5333 | 93.6862 | 0.0000 |
| campaign:time | 2 | 391.4667 | 195.7333 | 0.5468 | 0.5892 |
| market:time | 16 | 5727.4667 | 357.9667 | | |

Post Hoc

Here you might want to look at both Campaign and Time Effects:

```

# Using emmeans
shoesCampaignPH <- emmeans::emmeans(
  object = shoesModel,
  specs = pairwise ~ campaign,
  adjust = "tukey",
  level = 0.99
)

shoesTimePH <- emmeans::emmeans(
  object = shoesModel,
  specs = pairwise ~ time,
  adjust = "tukey",
  level = 0.99
)

```

Point Estimates You can get point estimates for both effects:

```

## Point Estimates
as.data.frame(shoesCampaignPH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Campaign", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```

Table 8: Marginal Means-Tukey 99% Adjustment

| Campaign | Marginal Mean | SE | DF | Lower Bound | Upper Bound |
|----------|---------------|---------|----|-------------|-------------|
| 1 | 739.4000 | 123.615 | 8 | 324.6237 | 1154.176 |
| 2 | 589.6667 | 123.615 | 8 | 174.8904 | 1004.443 |

```
as.data.frame(shoesTimePH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Time Point", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 9: Marginal Means-Tukey 99% Adjustment

| Time Point | Marginal Mean | SE | DF | Lower Bound | Upper Bound |
|------------|---------------|---------|------|-------------|-------------|
| before | 648.4 | 87.5454 | 8.05 | 355.1807 | 941.6193 |
| during | 728.8 | 87.5454 | 8.05 | 435.5807 | 1022.0193 |
| after | 616.4 | 87.5454 | 8.05 | 323.1807 | 909.6193 |

Pairwise Comparisons You can also do the standard pairwise comparisons

```
# Pairwise Comparisons
as.data.frame(shoesCampaignPH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Campaign Comparison-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 10: Campaign Comparison-Tukey 99% Adjustment

| Comparison | Difference | SE | DF | t Statistic | p-value |
|------------|------------|---------|----|-------------|---------|
| 1 - 2 | 149.7333 | 174.818 | 8 | 0.8565 | 0.4166 |


```

as.data.frame(shoesTimePH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Time Point-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```

Table 11: Time Point-Tukey 99% Adjustment

| Comparison | Difference | SE | DF | t Statistic | p-value |
|-----------------|------------|--------|----|-------------|---------|
| before - during | -80.4 | 8.4613 | 16 | -9.5021 | 0.0000 |
| before - after | 32.0 | 8.4613 | 16 | 3.7819 | 0.0044 |
| during - after | 112.4 | 8.4613 | 16 | 13.2840 | 0.0000 |

Effect Sizes

We will not worry about effect sizes here.

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", "lme4", "nlme", "emmeans", "rstatix")
lapply(packages, library, character.only = TRUE)

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

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

# Generic Demo Code for creating a wide data frame
# Note: this code assumes you already read in a long format data frame

#IMPORTANT: you will need to the code to match the names in your actual situation

dataWide <- pivot_wider(
  data = dataLong, # "dataLong" is the name of the long format data frame
  names_from = group, #"group" is the name of the column that contains your treatments
  values_from = response #"response" is the name of the column with the response values
)

# Generic Demo Code for creating a long data frame
# Note: this code assumes you already read in a wide format data frame

#IMPORTANT: you will need to the code to match the names in your actual situation

dataLong <- pivot_longer(
  data = dataWide, # "dataWide" is the name of the wide format data frame
  cols = !subject, # Says to not use the "subject" column
  names_to = "group", # This is the new column you want the treatments to go to
  names_transform = list(group = as.factor), # Makes the treatment column a factor
  values_to = "response" # A new column that will contain all of the response values
)

# Demo code to set up R
## Load packages
packages <- c("tidyverse", "knitr", "kableExtra",
             "parameters", "hasseDiagram", "car",
             "psych", "emmeans", "rstatix", "lme4", "nlme")
lapply(packages, library, character.only = TRUE)

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

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

# Hasse Diagram for Beer Judging
modellabels <- c("1 Judge Beer 1", "4 Beer 3", "6 (Judge) 5", "24 (Ratings) 15")
modelMatrix <- matrix(
```

```

data = c(FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE,
        FALSE, TRUE, TRUE, TRUE, FALSE),
nrow = 4,
ncol = 4,
byrow = FALSE
)
hasseDiagram::hasse(
  data = modelMatrix,
  labels = modelLabels
)

# Beer Data
beer <- data.frame(
  judge = as.factor(rep(x = LETTERS[1:6], each = 4)),
  beer = as.factor(rep(
    x = c("Barnstormer", "King Richard Red", "Craftsman", "Red Mo"),
    times = 6)),
  score = c(
    50, 60, 70, 70,
    38, 45, 58, 60,
    45, 48, 60, 58,
    65, 65, 75, 75,
    55, 60, 70, 65,
    48, 53, 68, 63
  )
)

beerWide <- pivot_wider(
  data = beer,
  names_from = beer,
  values_from = score
)

beerWide

beerLong <- pivot_longer(
  data = beerWide,
  cols = !judge, # This says to use all except the judge column
  names_to = "beer", # what column to place the beer names in
  names_transform = list(beer = as.factor), # turn beer to a factor
  values_to = "score" # what column to put the values in
)

head(beerLong, n = 5) # Display the first 5 rows.

# Omnibus Model (for our ANOVA table)
beerOmni <- aov(
  formula = score ~ judge + beer,
  data = beerLong
)

# Random Effect Model (Assumption Check and Point Estimation)
beerMixed <- lme4::lmer(
  formula = score ~ (1|judge) + beer,
  data = beerLong
)

# Compound Symmetry/Sphericity Assessment
beerSphere <- rstatix::anova_test(

```

```

data = beerLong,
formula = score ~ beer + Error(judge %in% beer)
)

## The %in% tells R that judge should be treated as nested in beer

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

# Judge Effects
car::qqPlot(
  x = unlist(
    lme4::ranef(
      object = beerMixed,
      whichel = c("judge")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Judge Effects"
)

# Tukey-Anscombe Plot
ggplot(
  data = data.frame(
    residuals = residuals(beerMixed),
    fitted = fitted.values(beerMixed)
  ),
  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 (score)") +
  ylab("Residuals (score)")

# Interaction Plot
ggplot(
  data = beer,

```

```

mapping = aes(
  x = beer,
  y = score,
  color = judge,
  group = judge
)
) +
  geom_point(size = 2) +
  geom_line() +
  ggplot2::theme_bw() +
  xlab("Beer") +
  ylab("Score") +
  labs(
    color = "Judge"
  )

# Sphericity Plot
sphericityPlot(
  dataWide = beerWide, # Data needs to be in wide format
  subjectID = "judge" # character name of the subject column
)

## Look for any groups that have very different amounts of variation

# Mauchly's Test
beerSphere$`Mauchly's Test for Sphericity` %>%
  dplyr::select(Effect, W, p) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Effect", "Mauchly's W", "p"),
    caption = "Mauchly's Sphericity Test",
    align = c('l', "c", "c"),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

beerTable1 <- parameters::model_parameters(
  model = beerOmni, # the RCBD with all fixed effects
  omega_squared = "partial",
  eta_squared = "partial",
  epsilon_squared = "partial"
)

beerTable1$p <- lapply(
  X = beerTable1$p,
  FUN = pvalRound
)

knitr::kable(
  x = beerTable1,
  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 Beer Judging 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")
)

# Use the block relative efficiency function
block.RelEff(
  aov.obj = beerOmni,
  blockName = "judge",
  trtName = "beer"
)

correctedTable <- beerSphere$`Sphericity Corrections` %>%
  dplyr::select(GGe, `p[GG]`, HFe, `p[HF]`)
correctedTable$`p[GG]` <- lapply(
  X = correctedTable$`p[GG]`,
  FUN = pvalRound
)
correctedTable$`p[HF]` <- lapply(
  X = correctedTable$`p[HF]`,
  FUN = pvalRound
)
knitr::kable(
  x = correctedTable,
  digits = 4,
  col.names = c("Greenhouse-Geisser", "p-value", "Huynh-Feldt", "p-value"),
  caption = "Sphericity Corrections",
  align = "c",
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Using emmeans
beerPH <- emmeans::emmeans(
  object = beerMixed, # Notice the use of the mixed model here
  specs = pairwise ~ beer,
  adjust = "tukey",
  level = 0.92
)

## Point Estimates
as.data.frame(beerPH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Type of Beer", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 92\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```

```

)

# Pairwise Comparisons
as.data.frame(beerPH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Marginal Means-Tukey 92\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

tempEMM <- emmeans::emmeans(
  object = beerMixed,
  specs = "beer"
)

# Pass the stored marginals into the effect size function
cohenTemp <- emmeans::eff_size(
  object = tempEMM,
  sigma = sigma(beerMixed),
  edf = df.residual(beerMixed)
)

# Create a data frame, add on the probability of superiority
# Send that data frame into a nice table
as.data.frame(cohenTemp) %>%
  dplyr::mutate(
    ps = probSup(effect.size),
    .after = effect.size
  ) %>%
  dplyr::select(contrast, effect.size, ps) %>%
  knitr::kable(
    digits = 3,
    col.names = c("Comparison", "Cohen's d", "Probability of Superiority"),
    align = "lcc",
    caption = "Effect Sizes for Beer",
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

# Hasse Diagram for Shoe Advertisement Study
modellLabels <- c("1 Sell Shoes 1", "2 Campaign 1", "10 (Market) 8", "3 Time Point 2",
                 "6 Campaign x Time Point 2", "30 (Markets X Time) 16")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE,
            FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE,
            FALSE, FALSE, TRUE, TRUE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE,
            TRUE, TRUE, FALSE),

```

```

nrow = 6,
ncol = 6,
byrow = FALSE
)
hasseDiagram::hasse(
  data = modelMatrix,
  labels = modelLabels
)

shoesWide <- read.table(
  file = "https://raw.githubusercontent.com/neilhatfield/STAT461/master/dataFiles/shoes.csv",
  header = TRUE,
  sep = ",",
)

shoesWide$campaign <- as.factor(shoesWide$campaign)
shoesWide$market <- as.factor(shoesWide$market)

# Make a long version of the data
shoesLong <- tidyr::pivot_longer(
  data = shoesWide,
  cols = dplyr::starts_with("t"),
  names_to = "time",
  names_ptypes = list("time" = factor()),
  values_to = "sales"
)

# Recode times to before, during, after and puts an ordering
shoesLong$time <- dplyr::recode_factor(
  shoesLong$time,
  "t1" = "before",
  "t2" = "during",
  "t3" = "after",
  .ordered = TRUE
)

# Omnibus Model
shoesModel <- aov(
  formula = sales ~ campaign * time + Error(market %in% campaign),
  data = shoesLong
)

# Use the nlme package to fit a model that we can use for assumption checking
shoesAssumptions <- nlme::lme(
  data = shoesLong,
  fixed = sales ~ campaign * time,
  random = ~ 1|market
)

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

```



```

# Market Effects
car::qqPlot(
  x = unlist(
    lme4::ranef(
      object = shoesAssumptions,
      whichel = c("market")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Market"
)

# Tukey-Anscombe Plot
ggplot(
  data = data.frame(
    residuals = residuals(shoesAssumptions),
    fitted = fitted.values(shoesAssumptions)
  ),
  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 (sales)") +
  ylab("Residuals (sales)")

ggplot(
  data = shoesLong,
  mapping = aes(
    x = time,
    y = sales,
    color = paste(campaign, market, sep = ":"),
    group = paste(campaign, market, sep = ":")
  )
) +
  geom_point() +
  geom_line() +
  theme_bw() +
  xlab("Time Point (relative to campaign)") +
  ylab("Sales (coded)") +
  labs(
    color = "Campagin:Market"
  )
)

# Growth Curves

```

```

ggplot(
  data = shoesLong,
  mapping = aes(
    x = time,
    y = sales,
    color = market,
    group = market
  )
) +
  geom_point(size = 2) +
  geom_line() +
  facet_wrap(facets = ~campaign) +
  ggplot2::theme_bw() +
  xlab("Time Period") +
  ylab("Sales (coded)") +
  labs(
    color = "Market"
  )

# Sphericity Plot
sphericityPlot(
  dataWide = shoesWide,
  subjectID = c("market", "campaign"),
  colsIgnore = NULL
)

## Look for any groups that have very different amounts of variation

shoesTemp <- summary(shoesModel)
shoesOmni <- rbind(
  shoesTemp$`Error: market:campaign`[[1]],
  shoesTemp$`Error: Within`[[1]]
)

row.names(shoesOmni) <- c("campaign", "market", "time", "campaign:time", "market:time")

shoesOmni["market", "F value"] <- shoesOmni["market", "Mean Sq"] /
  shoesOmni["market:time", "Mean Sq"]
shoesOmni["market", "Pr(>F)"] <- pf(
  q = shoesOmni["market", "F value"],
  df1 = shoesOmni["market", "Df"],
  df2 = shoesOmni["market:time", "Df"],
  lower.tail = FALSE
)

shoesOmni %>%
  tibble::rownames_to_column(
    var = "Source"
  ) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Source", "df", "SS", "MS", "F", "p-value"),
    caption = "ANOVA Table for Athletic Shoes Study",
    align = c('l', rep('c', 5)),
    booktab = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```

```

)

# Using emmeans
shoesCampaignPH <- emmeans::emmeans(
  object = shoesModel,
  specs = pairwise ~ campaign,
  adjust = "tukey",
  level = 0.99
)

shoesTimePH <- emmeans::emmeans(
  object = shoesModel,
  specs = pairwise ~ time,
  adjust = "tukey",
  level = 0.99
)

## Point Estimates
as.data.frame(shoesCampaignPH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Campaign", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

as.data.frame(shoesTimePH$emmeans) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Time Point", "Marginal Mean", "SE", "DF",
                  "Lower Bound", "Upper Bound"),
    caption = "Marginal Means-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Pairwise Comparisons
as.data.frame(shoesCampaignPH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Campaign Comparison-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(

```

```

    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
)

as.data.frame(shoesTimePH$contrasts) %>%
  knitr::kable(
    digits = 4,
    col.names = c("Comparison", "Difference", "SE", "DF",
                  "t Statistic", "p-value"),
    caption = "Time Point-Tukey 99\\% Adjustment",
    align = c("l", rep("c", 5)),
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

```