ANOVA Models with a Block

Neil J. Hatfield

Last Updated: October 31, 2023

In this tutorial, we are going to explore looking at ANOVA models involving blocks. Specifically, we'll look at Randomized Complete Block Designs (RCBDs). Keep in mind that we can add a Block to pretty much any ANOVA model.

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. 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", "DescTools",
              "emmeans")
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 tutorial, we're going to look at two contexts: **Farming Barley** and **Dental Pain**. I'll give a brief description of each context, show how to load the data, and discuss whether an ANOVA + Block model is appropriate.

2.1 Farming Barley

A farmer wants to test out four varieties of barley and see if there is any difference in yield (bussels per acre). He has four fields in which he can plant the barley. However, the farmer is aware of differences between each field. For example,

- One field has a higher clay content in the soil than the others
- One field has rockier soil than the others
- Two fields are in wetter climates; two are in drier climates
- One field has loose soil while another field has much more compacted soil.
- Two fields are relatively flat, one has a hill in the middle, and the last has a valley.

Given that the fields will be our measurement units, there is quite a bit of variation between them. This variation could easily become confounded with the impact of barley variety on crop yield. Thus, we're in a perfect situation to make use a block of field.

We can access and clean the data through the following code:

```
# Load Barley Data ----
barleyData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/barley.dat",
  header = TRUE,
  sep = ","
```

```
)

# Glancing at the data frame, we can improve our column names (variables)
# as well as set our factor and block to be.

## I'm going to change "Treatment" to "Varietal"
names(barleyData)[which(names(barleyData) == "Treatment")] <- "Varietal"

## Set Varietal as a Factor
barleyData$Varietal <- as.factor(barleyData$Varietal)

## Tell R to consider our block, Field, as a "factor"
barleyData$Field <- as.factor(barleyData$Field)

## I'm also going to simplify "Planting.Harvesting.Order" to just "Order"
names(barleyData)[which(names(barleyData) == "Planting.Harvesting.Order")] <- "Order"
```

2.1.1 Checking Appropriateness

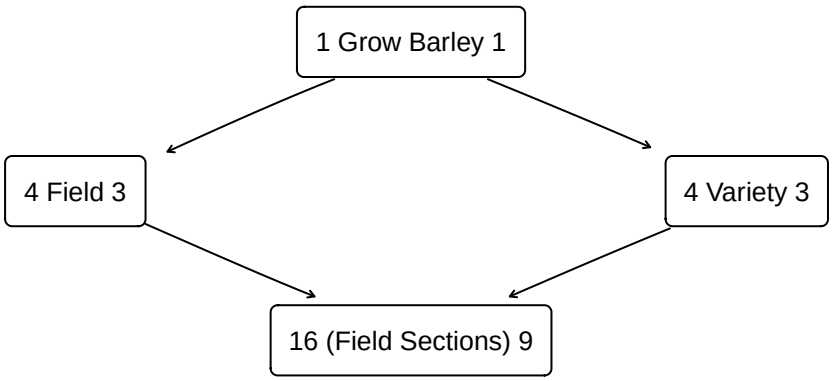


Figure 1: Hasse Diagram for Barley Crop Yield Study

From Figure 1 we can see that for our continuous response crop yield (bushels per acre) and our factor of variety of barley planted (4 types), we have an additive model with estimable effects and errors. Further, we've incorporated our four fields into the model to block any confounding between them and the type of barely.

2.2 Dental Pain

In a study reported by Kutner et al. (2005)[1], an anesthesiologist did a comparative study on the effects of acupuncture and codeine on the amount of pain experienced by male participants after a dental procedure. The two factors in the study were the drug (two levels–codeine and a placebo sugar pill) and acupuncture points (two levels–active points and inactive points). The 32 participants were divided into eight blocks based upon a pre-assessment of their pain tolerance. Within each of these eight blocks, the anesthesiologist randomly assigned treatments; double blinding was used in the study.

We may access this date with the following code:

```
# Load Dental Pain Data ----
dentalData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/dentalPain.txt",
  header = TRUE,
  sep = ""
)

## Tell R to treat our factors and our block as "factor"
dentalData$tolerance <- as.factor(dentalData$tolerance)

dentalData$drug <- dplyr::case_match(
```

[1]Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2005). Applied linear statistical models. McGraw-Hill Irwin.

```
  .x = dentalData$drug,
  1 ~ "placebo",
  2 ~ "codeine",
  .ptype = factor(levels = c("placebo", "codeine"))
)

dentalData$acupuncture <- dplyr::case_match(
  .x = dentalData$acupuncture,
  1 ~ "inactive",
  2 ~ "active",
  .ptype = factor(levels = c("inactive", "active"))
)
```

2.2.1 Checking Appropriateness

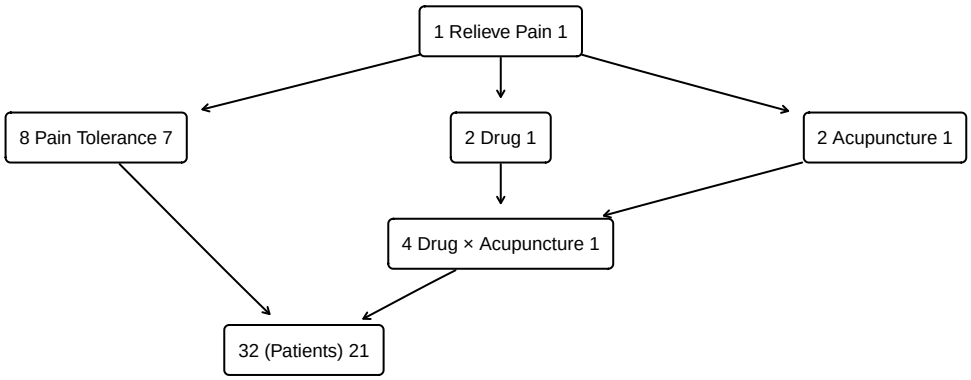


Figure 2: Hasse Diagram for Dental Pain Study

2.2.1.1 Your Turn Write a paragraph about whether or not we can use ANOVA models with the Dental Pain study.

3 Fit the Models

ANOVA models with a block are set up *almost* like you would set up a factorial model. There are two important differences:

1) We typically want to put the block into our model **first**, and then the rest of our factors.
2) Our block should not be interacting with any of our other factors so we should not include the block in any interactions.

```
# Demo code for fitting models ----

## Farming Barley Study
barleyModel <- aov(
  formula = Yield ~ Field + Varietal,
  data = barleyData
)

## Dental Pain Study
dentalModel <- aov(
  formula = relief ~ tolerance + drug*acupuncture,
  data = dentalData
)
```

4 Assessing Assumptions

In terms of assumptions, we still have our core three: Gaussian Residuals, Homoscedasticity, and Independence of Observations. With the block, we also want to check an interaction plot to make sure that there isn't anything strange/unexpected going on.

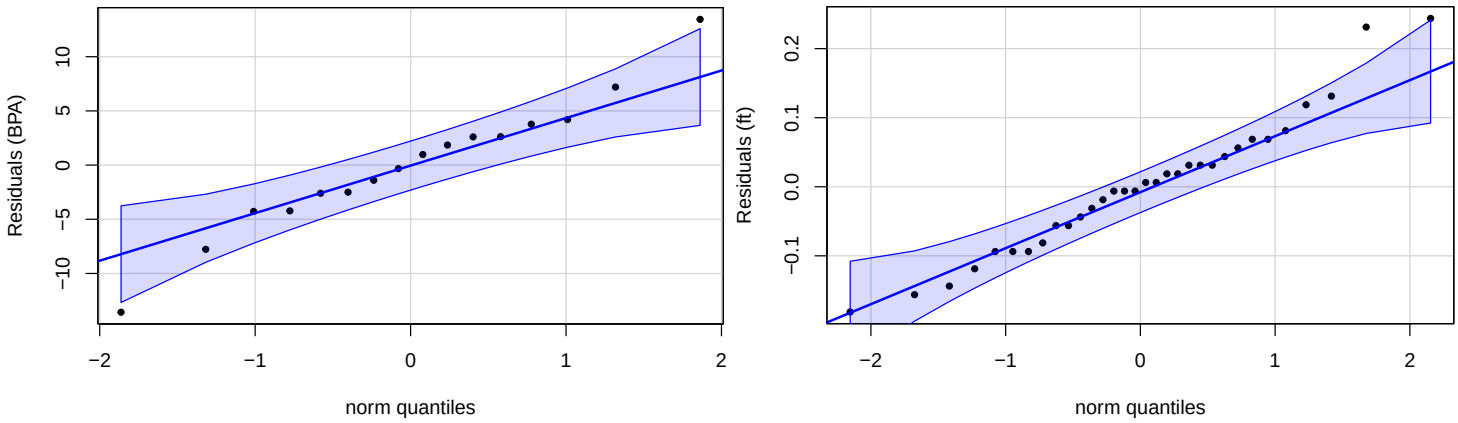
4.1 Gaussian Residuals

Just as before we will make use of QQ Plots.

```
# Demo Code for QQ Plots ----
## Chunk options for side-by-side
### fig.subcap=c("Farming Barley Study", "Dental Pain Study"), fig.ncol=2,
### out.width="50%"

## QQ Plot for Barley Model Residuals
car::qqPlot(
  x = residuals(barleyModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (BPA)"
)

## QQ Plot for Dental Study Residuals
car::qqPlot(
  x = residuals(dentalModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (ft)"
)
```



(a) Farming Barley Study

(b) Dental Pain Study

Figure 3: QQ Plots

For the Farming Barley study, we have two of 16 residuals beyond the envelope or 12.5%. Given the balanced design, we will proceed with saying that this assumption has been satisfied for the Farming Barley study.

4.1.1 Your Turn

How would you assess the Gaussian Residual assumption for the Dental Pain Study?

4.2 Assessing Homoscedasticity

When we account for the the block in our model, we won't get the nice strips that we're used to seeing. However, we can still use the basic idea of the strip chart to assess homoscedasticity. We can still look for patterns and we'll add a smoother to our plot, transforming the strip chart to a Tukey-Anscombe Plot.

4.2.1 Farming Barley

We don't see any patterns to the plot (Figure 4), which is a good sign. While there does appear to be more variation on the high end of the fitted values, the smoothed line is fairly flat. The upticks at the ends might be the result of the small sample size. In all, we will take the homoscedasticity assumption to be satisfied.

```
# Demo code for making Tukey-Anscombe plot ----
## Farming Barley Study
ggplot(
  data = data.frame(
    residuals = residuals(barleyModel),
    fitted = fitted.values(barleyModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline( ## Adds reference line at zero
    yintercept = 0,
    linetype = "dashed",
    color = "grey50"
  ) +
  geom_smooth( ## Adds the smoothed line
    formula = y ~ x,
    method = stats::loess,
    method.args = list(degree = 1),
    se = FALSE,
    linewidth = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (BPA)") +
  ylab("Residuals (BPA)")
```

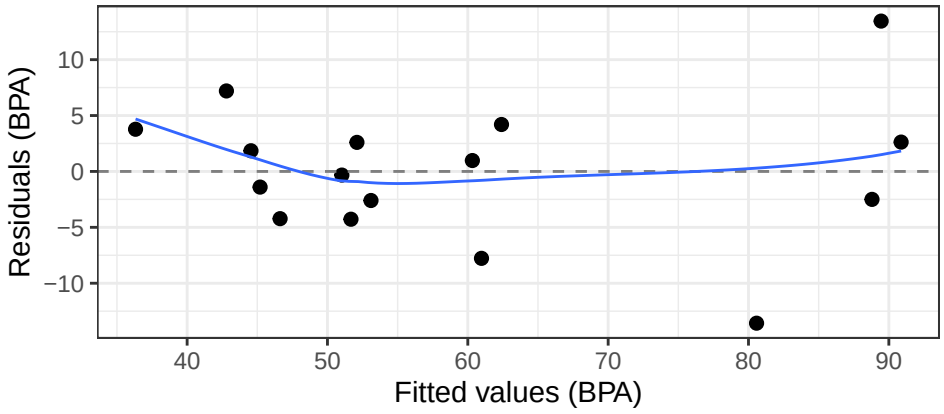


Figure 4: Tukey-Anscombe Plot for Barley Crop Yield Study

4.2.2 Dental Pain Study

Figure 5 is the Tukey-Anscombe plot for our Estimation Errors study. While the smoother line (solid blue) is flat, I do see an increase in the variation towards the right-hand side (fitted values at or above 110 ft). This might be the result of a potential outlier. We will want to thoroughly explore our data to better understand what might be happening.

```
# Demo code for making Tukey-Anscombe plot ----
## Dental Pain Study
ggplot(
  data = data.frame(
    residuals = residuals(dentalModel),
    fitted = fitted.values(dentalModel)
  ),
  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") +
  ylab("Residuals")
```

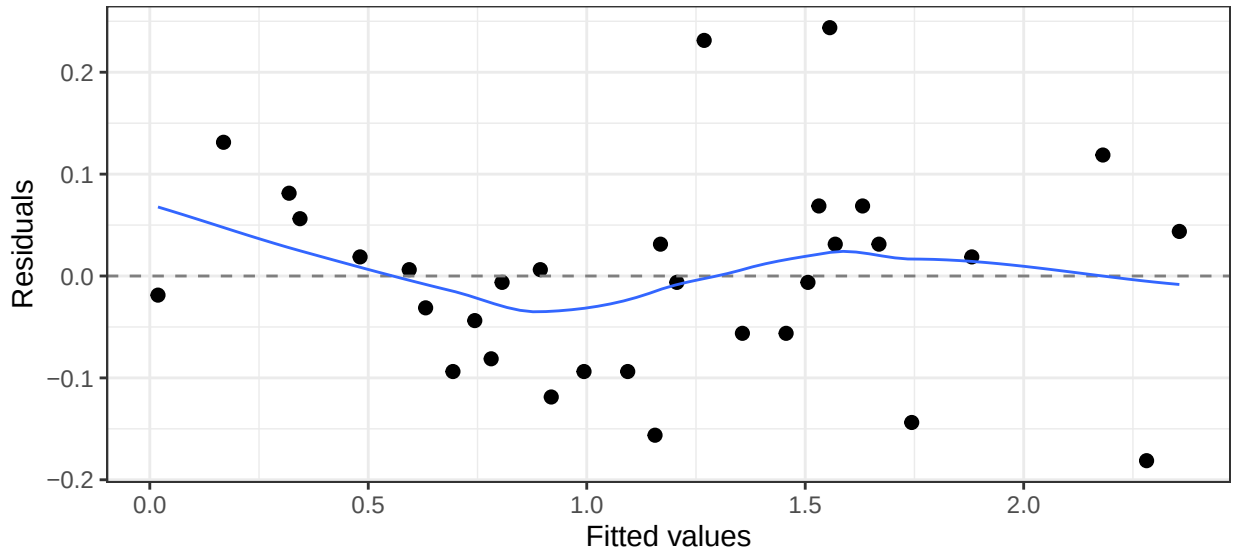


Figure 5: Tukey-Anscombe Plots for Dental Pain Study

4.3 Assessing Independence of Observations

Again, we can assess the Independence of Observations in the same ways as from Unit 3: making use of our knowledge of the study design, and, IF we know measurement order, index plots.

4.3.1 Dental Pain Study

We do not have an accurate accounting of measurement order for the Dental Pain study. Thus, we will only have to go off of our knowledge of the study design and how we carried out the study.

4.3.2 Farming Barley

In the Farming Barley study, we do know measurement order. Thus, we can make use of an Index plot.

```
# Demo code for index plots ----
## Farming Barley Study
## Using the residuals from the model which includes the block
ggplot(
  data = data.frame(
    residuals = barleyModel$residuals,
    index = 1:length(barleyModel$residuals)
  ),
```

```
  mapping = aes(x = index, y = residuals)
) +
  geom_point(size = 1.5) +
  geom_line() +
  theme_bw() +
  geom_hline(
    yintercept = 0,
    linetype = "dashed",
    color = "red"
  ) +
  xlab("Measurement order") +
  ylab("Residuals")
```

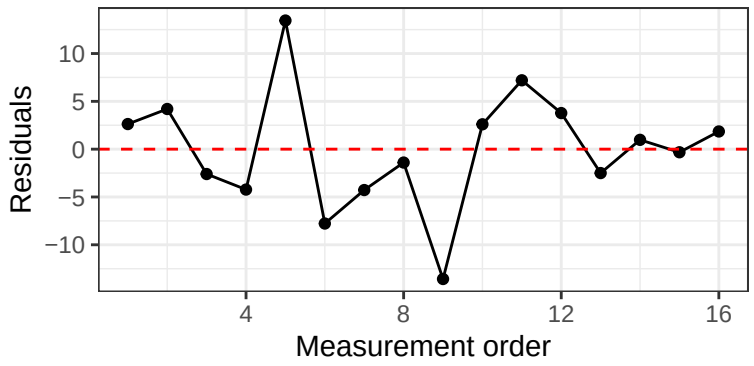


Figure 6: Index Plot for Farming Barley Residuals

From Figure 6, we don't necessarily see any patterns which would indicate a threat of the assumption of independent observations.

We can also form Index Plots in a different way: rather than looking at the residuals, we can also plot the actual response values. However, if we go this route, we will want to incorporate our block into the Index Plot.

```
# Demo code for Alternative Index Plot for Barley Yields ----

# Note: I'm using the boastUtils package for a different color palette.
# You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

ggplot(
  data = barleyData,
  mapping = aes(
    x = Order,
    y = Yield,
    color = Field,
    shape = Varietal,
    linetype = Field
  )
) +
  geom_point(size = 2) +
  geom_path(
    mapping = aes(group = Field)
  ) +
  ggplot2::theme_bw() +
  xlab("Planting/Havesting Order") +
  ylab("Yield (BPA)") +
  scale_color_manual(values = boastUtils::psuPalette)
```

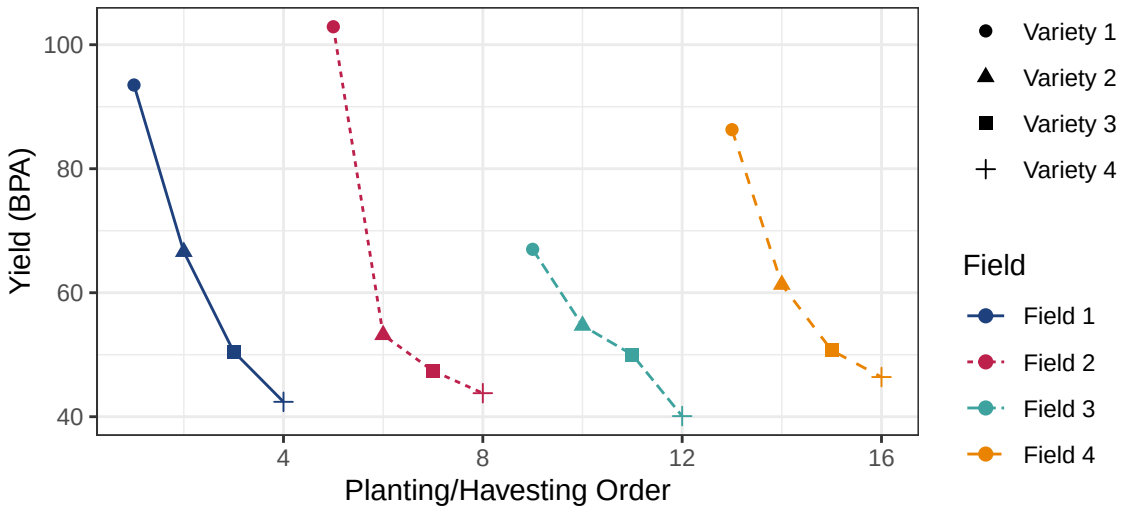


Figure 7: Index Plot for Barley Crop Yield Residuals

While at first glance, there appears to be a pattern to the response (Figure 7; a repeating downward line), we can see that this is an artifact of the block and treatments.

4.4 Assessing Interaction

The last aspect we need to check is whether there is a worrisome interaction between our block and our factor. If there is such an interaction, then our model (response ~ block + factor) is not valid. We will make an interaction plot to assess this issue.

For an interaction plot, we will plot the values of the *Sample Arithmetic Mean* across all combinations of blocks and factors look for consistency in performance. If there is no interaction, then we should see consistent behaviors (essentially parallel lines). If there is interaction, then we should switches in behavior such as non-parallel lines moving in opposite directions.

4.4.1 Farming Barley

```
# Demo code for an interaction plot ----

# Note: I'm using the boastUtils package for a different color palette.
#You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

# Interaction Plot for Field and Treatment
ggplot2::ggplot(
  data = barleyData,
  mapping = aes(
    x = Varietal,
    y = Yield,
    color = Field,
    shape = Field,
    linetype = Field,
    group = Field
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Variety") +
  ylab("Yield (BPA)") +
  labs(color = "Field") +
  theme(
    legend.position = "right"
  ) +
```

```
  scale_color_manual(values = boastUtils::psuPalette)
```

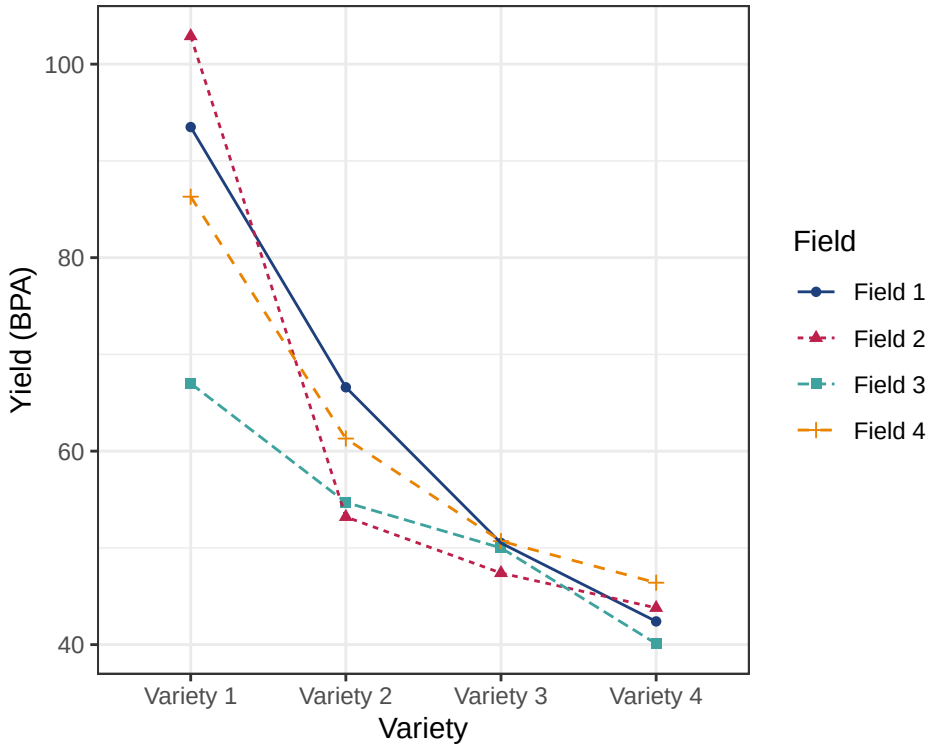


Figure 8: Interaction Plot for Barley Varietal and Field

In the interaction plot (Figure 8), we see essentially the same behavior of barley variety in each field. This indicates that there is not any type of interaction between field and barley varietal. While the lines are not perfectly parallel, they all reflect the same general behavior. An interaction to be concerned about would be if for Variety 2, Field 4 had a yield higher than Variety 1, Field 2's yield

4.4.2 Dental Pain Study

For the Dental Pain study, we can examine Figure 9.

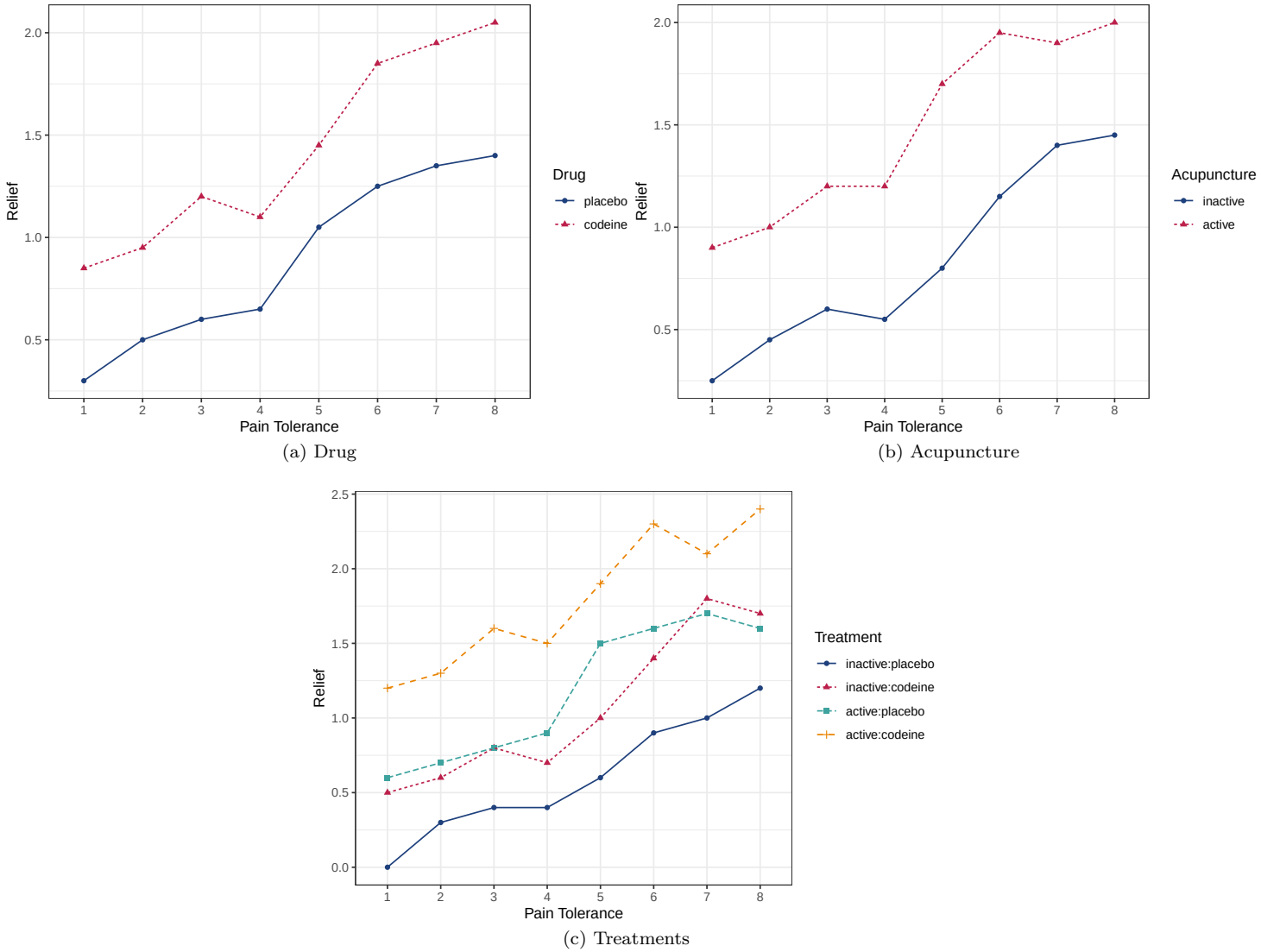
```
# Note: I'm using the boastUtils package for a different color palette.
# You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

# Demo code for an interaction plot ----
## Interaction Plot for Tolerance and Drug
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = drug,
    shape = drug,
    linetype = drug,
    group = drug
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
  ylab("Relief") +
```

```
  labs(color = "Drug", linetype = "Drug", shape = "Drug") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

## Interaction Plot for Tolerance and Acupuncture
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = acupuncture,
    shape = acupuncture,
    linetype = acupuncture,
    group = acupuncture
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
  ylab("Relief") +
  labs(color = "Acupuncture", linetype = "Acupuncture", shape = "Acupuncture") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

## Interaction Plot for Tolerance and Treatments
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = acupuncture:drug,
    shape = acupuncture:drug,
    linetype = acupuncture:drug,
    group = acupuncture:drug
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
  ylab("Relief") +
  labs(color = "Treatment", linetype = "Treatment", shape = "Treatment") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)
```



(a) Drug

(b) Acupuncture

(c) Treatments

Figure 9: Interaction Plot for Dental Pain Study

Notice that there are three separate plots in Figure 9; this is a reflection of the fact that we are fitting a Two-way Factorial + a Block ANOVA model. Thus, we will want to explore whether our block is interacting with *any* of our main effects and interaction terms.

When we look at Figure 9a, we can have a point of concern between levels 3 and 4 of pain tolerance when looking at the impact of Drug. The general pattern of behavior appears to change at these levels. Similarly, the 3-4 and 6-7 sections of Figure 9b are also somewhat concerning when we explore the impact of Acupuncture. When we look at the interaction term by pain tolerance (Figure 9c), sections 3-4 echo what we've previously seen, as well as for section 6-7. There is a new potential issue in section 7-8.

4.5 Assumption Decisions

For the Farming Barley study, I'm going to say that all four of the assumptions are satisfied.

For the most part, our data appear to satisfy the assumptions. However, before we make our final determination, we might want to explore whether we have any potential outliers that might be driving the concern we saw in the Tukey-Anscombe plot for assessing homoscedasticity.

4.5.1 Your Turn–Explore the Dental Pain Data

Decide whether there are any potential outliers that we might need to handle and then how to handle them. If you opt to remove or change any observations, re-check all of the assumptions.

5 Results

For the Results portion of this guide/tutorial, I'm going to focus on the Farming Barley study. I'll leave the Dental Pain study to you as an exercise.

Remember, for any results section for a parametric shortcut, we have several parts: the omnibus tests and effect sizes, point estimates, and Post Hoc (pairwise and/or contrasts).

5.1 Omnibus Results

We generate our omnibus results *almost* exactly like we would for a one-way ANOVA model. The only change is in the Modern ANOVA table for the effect size estimates. Rather than using `"raw"`, we will use `"partial"` to reflect the fact that we now have multiple terms in our model

```
# Omnibus Test/Modern ANOVA Table
parameters::model_parameters(
  model = barleyModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  type = 3, # You will want to use these, esp. for Imbalance
  drop = "(Intercept)",
  verbose = FALSE
) %>%
  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 Barley Crop Yield 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 1: ANOVA Table for Barley Crop Yield Study

	Source	SS	df	MS	F	p-value	Partial Eta Sq.	Partial Omega Sq.	Partial Epsilon Sq.
1	Field	259.265	3	86.4217	1.3443	0.3203	0.3094	0.0606	0.0793
2	Varietal	4573.105	3	1524.3683	23.7116	0.0001	0.8877	0.8098	0.8503
4	Residuals	578.590	9	64.2878					

We still interpret all values just as we did in Unit 3. The biggest catch is how we treat the row of Table 1 connected to our block, Field. That is to say, we don't actually care about this row. This means that we won't interpret the F ratio, the p-value, or the effect sizes for Field. After all, we didn't want to test the impact of field on the barley yield; we just didn't want field to become confounded with the kind of barley planted (varietal).

We would still point out that the varietal planted accounted for nearly 24 times as much variation as what was left unexplained/accounted for by our model. This translates to varietal explaining around 85% of the total variation in yield. Under the null hypothesis, we would only anticipate this extreme of a result 1/100th of a percent of the time.

5.2 Relative Efficiency

One of the things that Randomized Complete Block Designs can help with is design efficiency. By this we mean, how much we can save in terms of sample size by using a block design versus a completely randomized design. To help us get this measure, I've written the `block.RelEff` function to help us.

```
# Demo code for Relative Efficiency of the Block
## Farming Barley Study
block.RelEff(
```

```
  aov.obj = barleyModel,
  blockName = "Field",
  trtName = "Varietal"
)
```

```
## [1] "The relative efficiency of the block, Field, is 1.028."
```

You can also run this code inline to have the resulting sentence appear as part of your narrative. For example, "The relative efficiency of the block, Field, is 1.028."

We can interpret this relative efficiency as telling us how many times larger we would need the per group sample size to be if we didn't use the block. Sometimes using a block will yield a decent efficiency, sometimes not so much. My general recommendation is that if your block's relative efficiency is at least 1, that's good enough.

NOTE: The `block.RelEff` function currently only works for One-way ANOVA + Block models.

5.3 Point Estimates

If we want to get point estimates for our Grand Mean, Treatment Effects, and Block Effects, we can. We use the same methods as before. I recommend that you first run `dummy.coef(anovaModel)` in your console so you can see the order of terms.

```
# Point Estimates for Farming Barley
pEst <- dummy.coef(barleyModel)
pEst <- unlist(pEst)
names(pEst) <- c(
  "Grand Mean",
  levels(barleyData$Field), # I know that this is the correct order because
  levels(barleyData$Varietal) # I ran dummy.coef(barleyModel) in my console first
  )

data.frame("Estimate" = pEst) %>%
  knitr::kable(
  digits = 3,
  caption = "Point Estimates from the Barley Crop Yield Study",
  booktabs = TRUE,
  align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 2: Point Estimates from the Barley Crop Yield Study

	Estimate
Grand Mean	59.800
Field 1	3.450
Field 2	2.025
Field 3	-6.850
Field 4	1.375
Variety 1	27.625
Variety 2	-0.850
Variety 3	-10.150
Variety 4	-16.625

Again, we interpret these numbers as rates: "bushels per acre per test plot" or more simply, "yield per plot" (where we're using [test] plot to mean a quarter-subsection of a field).

If you don't want the estimates for the block, you can do the following:

```r
pEst <- dummy.coef(barleyModel)
pEst <- unlist(pEst[which(names(pEst) != "Field")])
names(pEst) <- c(
  "Grand Mean",
  levels(barleyData$Varietal) # Using levels here will help stop the accidental
  # mislabeling of estimates
  )

data.frame("Estimate" = pEst) %>%
  knitr::kable(
  digits = 3,
  caption = "Point Estimates from the Barley Crop Yield Study",
  booktabs = TRUE,
  align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 3: Point Estimates from the Barley Crop Yield Study

	Estimate
Grand Mean	59.800
Variety 1	27.625
Variety 2	-0.850
Variety 3	-10.150
Variety 4	-16.625

5.4 Post Hoc–Pairwise

Just as with One-way and Factorial Designs, we can do Post Hoc analyses in several ways. Generally speaking, we will **only** do pairwise comparisons on main effects and/or interaction terms. Remember, our goal for the block is to use up/explain variation, *we are not interested in inferences about the block.*

5.4.1 `emmeans` Approach

We can use the `emmeans` package with out block models. This package gives some some pretty decent flexibility. Let's look at a testing family for the pairwise comparisons of varietal, controlling SCI at 0.1 via Tukey's HSD.

```r
# Demo code for post hoc pairwise via emmeans ----
## Barley Study
barleyPostHoc1 <- emmeans::emmeans(
  object = barleyModel,
  specs = pairwise ~ Varietal,
  adjust = "tukey",
  level = 0.9
)

barleyEffects1 <- as.data.frame(
  eff_size(
    object = barleyPostHoc1,
    sigma = sigma(barleyModel),
    edf = df.residual(barleyModel)
  )
) %>%
  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(barleyPostHoc1$contrasts) %>%
  left_join(
    y = barleyEffects1,
    by = join_by(contrast == contrast)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Barley Varietal",
  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 4: Post Hoc Comparisons for Barley Varietal

Pair	Difference	SE	DF	t	p-value	Cohen's d	Prob. of Superiority
Variety 1 - Variety 2	28.475	5.67	9	5.022	0.003	3.551	0.994
Variety 1 - Variety 3	37.775	5.67	9	6.663	0.000	4.711	1.000
Variety 1 - Variety 4	44.250	5.67	9	7.805	0.000	5.519	1.000
Variety 2 - Variety 3	9.300	5.67	9	1.640	0.405	1.160	0.794
Variety 2 - Variety 4	15.775	5.67	9	2.782	0.083	1.967	0.918
Variety 3 - Variety 4	6.475	5.67	9	1.142	0.675	0.808	0.716

Table 4 provides the pairwise comparisons for the different varietals in our Barley Study along with estimates of effect sizes. Even though we used a One-way ANOVA + Block model, we still interpret all of these values in the same ways that we have been.

5.4.2 `tukeyHSD` Approach

One catch with the `emmeans` approach is that you don't necessarily get confidence intervals. If you want intervals (and don't want to write the code to construct them yourself), you can turn to the `tukeyHSD` function instead. Since we are using a model beyond a classic One-way ANOVA, we need to add an additional argument to our call, `which`.

```
# Demo Code for Post Hoc Pairwise Comparisons via tukeyHSD ----
barleyPH <- TukeyHSD(
  x = barleyModel,
  which = "Varietal", # We need to specify which model term we want
  conf.level = 0.9
)

knitr::kable(
  barleyPH$Varietal,
  digits = 4,
  caption = "Post Hoc Tukey HSD Comparisons",
  col.names = c("Difference", "Lower Bound",
```

```
                "Upper Bound", "Adj. p-Value"),
  align = 'cccc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 5: Post Hoc Tukey HSD Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
Variety 2-Variety 1	-28.475	-43.5533	-13.3967	0.0033
Variety 3-Variety 1	-37.775	-52.8533	-22.6967	0.0004
Variety 4-Variety 1	-44.250	-59.3283	-29.1717	0.0001
Variety 3-Variety 2	-9.300	-24.3783	5.7783	0.4053
Variety 4-Variety 2	-15.775	-30.8533	-0.6967	0.0833
Variety 4-Variety 3	-6.475	-21.5533	8.6033	0.6747

Notice that we added on the `which` argument to the `TukeyHSD` call. This isolates the appropriate portion of the Tukey HSD output related to our factor of interest. If you omit this, you'll get Tukey HSD reports for every term in the model. In the Farming Barley situation, one for Field and one for Varietal.

We still interpret the results in the exact same way.

5.4.3 `DescTools` Approach

You can make similar adjustments in the `DescTools::PostHocTest` if you are wanting to control a different Type I Error Rate and/or use a different method. Table 6 provides an example using Bonferroni's method for controlling SCI.

```
## DescTools Pairwise Method
dtPH <- DescTools::PostHocTest(
  x = barleyModel, # Your aov/lm object
  which = "Varietal", # Specify which factor
  method = "bonf", # Your chosen method
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
)

## Kable Code for DescTools
knitr::kable(
  x = dtPH$Varietal, # Notice the use of the factor name
  digits = 3,
  caption = paste( # Creates a nice title; copy at will
    "Post Hoc",
    attr(dtPH, "method"),
    "Comparisons"
  ),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 6: Post Hoc Bonferroni Comparisons

	Difference	Lower Bound	Upper Bound	Adj. p-Value
Variety 2-Variety 1	-28.475	-45.106	-11.844	0.004
Variety 3-Variety 1	-37.775	-54.406	-21.144	0.001
Variety 4-Variety 1	-44.250	-60.881	-27.619	0.000
Variety 3-Variety 2	-9.300	-25.931	7.331	0.812
Variety 4-Variety 2	-15.775	-32.406	0.856	0.128
Variety 4-Variety 3	-6.475	-23.106	10.156	1.000

Note: Dunnett's Test does not currently allow for a block design.

5.4.4 Effect Sizes

Regardless of which method you use for pairwise comparisons (`emmeans`, `tukeyHSD`, or `DescTools`), you can opt to give your pairwise effect sizes as a separate table. To get the effect sizes for our desired pairwise comparisons, we will turn to the `anova.PostHoc` function. However, we need to take care with the arguments of this function:

- The `aov.obj` is the primary argument and is where we pass the `aov` (or `lm`) output to.
- The `response` argument takes a character string as the name of the response attribute.
- The `mainEffect` argument takes a character string that names the factor you are wanting to do pairwise comparisons on.

```
anova.PostHoc(
  aov.obj = barleyModel, # Our aov output
  response = "Yield", # Our response variable
  mainEffect = "Varietal" # Our factor variable
) %>%
  knitr::kable(
    digits = 3,
    caption = "Post Hoc Comparison Effect Sizes",
    col.names = c("Pairwise Comparison","Cohen's d", "Hedge's g",
                  "Prob. of Superiority"),
    align = 'lccc',
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 7: Post Hoc Comparison Effect Sizes

Pairwise Comparison	Cohen's d	Hedge's g	Prob. of Superiority
Variety.1 vs. Variety.2	2.451	2.131	0.958
Variety.1 vs. Variety.3	3.493	3.037	0.993
Variety.1 vs. Variety.4	4.052	3.523	0.998
Variety.2 vs. Variety.3	2.061	1.792	0.927
Variety.2 vs. Variety.4	3.313	2.881	0.990
Variety.3 vs. Variety.4	3.005	2.613	0.983

We need to make an important note here: suppose that our overall Unusualness Threshold was 0.1. From the Tukey HSD results, Variety 3 vs 2, 4 vs 2, and 4 vs 3 would NOT be statistically significant. From the effect size table, we would say that there are rather large effects as Cohen's d and Hedge's g are all quite large. Further, the probability of superiority for each

pairing is far from 0.5 (no practical effect). Just as effect sizes temper statistical significance, statistical significance moderates effect sizes. In these three cases, while there appears to be a large effect, there is enough variation in those groups that the effect is not statistically large enough to escape through the noise of the group.

5.5 Post Hoc–Contrasts

Even with a block, we can still use make use of the idea of contrasts. For example, let's say that barley varieties 1 and 2 are from one company while 3 and 4 are from a second company. We can test the contrast of companies, even in this blocking design.

5.5.1 `emmeans` Approach

The `emmeans` approach will let us incorporate effect sizes into our results (Table 8).

```
# Demo Code for Contrasts using emmeans ----
## Get the means
barleyMeans <- emmeans::emmeans(
  object = barleyModel,
  specs = ~ Varietal # Nothing goes on the left of ~; list what term you want
)

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

## Apply the contrasts
barleyContrasts1 <- emmeans::contrast(
  object = barleyMeans, # Notice that this is the means object
  method = list(
    "Company A vs. Company B" = c(1/2, 1/2, -1/2, -1/2)
  ),
  adjust = "none" # No MC/SI adjustment
)

## Add effect sizes and make a nice looking table
as.data.frame(barleyContrasts1) %>%
  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 = "Barley Study Main Effects Contrast on Varietal Company",
    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 8: Barley Study Main Effects Contrast on Varietal Company

Contrast	Difference	SE	DF	t Statistic	p-value	Cohen's d	Prob. of Superiority
Company A vs. Company B	26.775	4.009	9	6.679	0	4.453	0.999

5.5.2 Base `R` Approach

We can also use the base `R` approach will let us incorporate our contrast directly into a classical ANOVA table (Table 9).

```r
# Demo Code for Contrasts via base R ----
## Define the contrast
company <- c(1/2, 1/2, -1/2, -1/2)

## Bind the contrast to our factor
contrasts(barleyData$Varietal) <- company

## Refit our model so that our contrast gets tested
barleyContrast <- aov(
  formula = Yield ~ Varietal + Field,
  data = barleyData
)

## Get the updated ANOVA Table
### Remember, you could also use the DescTools package for Scheffé here
conOut <- summary.aov(
  object = barleyContrast,
  split = list(
    Varietal = list(
      "Company A vs. Company B" = 1
    )
  )
)

## Make a nice table
knitr::kable(
  x = conOut[[1]],
  digits = 4,
  col.names = c(
    "DF", "SS", "MS", "F", "p-value"),
  caption = "ANOVA Table for Barley Crop Yield Contrasts",
  booktabs = TRUE,
  align = rep("c", 5)
) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```

Table 9: ANOVA Table for Barley Crop Yield Contrasts

	DF	SS	MS	F	p-value
Varietal	3	4573.105	1524.3683	23.7116	0.0001
Varietal: Company A vs. Company B	1	2867.602	2867.6025	44.6057	0.0001
Field	3	259.265	86.4217	1.3443	0.3203
Residuals	9	578.590	64.2878		

6 Your Turn

Using your transformed Dental Pain data, attempt to create the omnibus ANOVA table, get the relative efficiency for the block design, point estimates, and the pairwise comparisons.

7 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", "DescTools",
              "emmeans")
lapply(packages, library, character.only = TRUE)

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

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
# Demo code to set up R ----
## Load packages
packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "hasseDiagram", "DescTools",
              "emmeans")
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")

# Load Barley Data ----
barleyData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/barley.dat",
  header = TRUE,
  sep = ","
)

# Glancing at the data frame, we can improve our column names (variables)
# as well as set our factor and block to be.

## I'm going to change "Treatment" to "Varietal"
names(barleyData)[which(names(barleyData) == "Treatment")] <- "Varietal"

## Set Varietal as a Factor
barleyData$Varietal <- as.factor(barleyData$Varietal)

## Tell R to consider our block, Field, as a "factor"
barleyData$Field <- as.factor(barleyData$Field)

## I'm also going to simplify "Planting.Harvesting.Order" to just "Order"
names(barleyData)[which(names(barleyData) == "Planting.Harvesting.Order")] <- "Order"

# Demo code of Barley Hasse Diagram ----
modelLabels <- c("1 Grow Barley 1", "4 Field 3", "4 Variety 3", "16 (Field Sections) 9")
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
)

# Load Dental Pain Data ----
dentalData <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/dentalPain.txt",
  header = TRUE,
  sep = ""
)

## Tell R to treat our factors and our block as "factor"
dentalData$tolerance <- as.factor(dentalData$tolerance)

dentalData$drug <- dplyr::case_match(
  .x = dentalData$drug,
  1 ~ "placebo",
  2 ~ "codeine",
  .ptype = factor(levels = c("placebo", "codeine"))
)

dentalData$acupuncture <- dplyr::case_match(
  .x = dentalData$acupuncture,
  1 ~ "inactive",
  2 ~ "active",
  .ptype = factor(levels = c("inactive", "active"))
)

# Demo code for Dental Pain Hasse Diagram ----
modelLabels <- c("1 Relieve Pain 1", "8 Pain Tolerance 7", "2 Drug 1", "2 Acupuncture 1",
                 "4 Drug × Acupuncture 1", "32 (Patients) 21")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, FALSE,
           FALSE, FALSE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE, TRUE, FALSE,
           FALSE, FALSE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE,
           TRUE, TRUE, TRUE, TRUE, TRUE, FALSE),
  nrow = 6,
  ncol = 6,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Demo code for fitting models ----

## Farming Barley Study
barleyModel <- aov(
  formula = Yield ~ Field + Varietal,
  data = barleyData
)

## Dental Pain Study
```

```r
dentalModel <- aov(
  formula = relief ~ tolerance + drug*acupuncture,
  data = dentalData
)

# Demo Code for QQ Plots ----
## Chunk options for side-by-side
### fig.subcap=c("Farming Barley Study", "Dental Pain Study"), fig.ncol=2,
### out.width="50%"

## QQ Plot for Barley Model Residuals
car::qqPlot(
  x = residuals(barleyModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (BPA)"
)

## QQ Plot for Dental Study Residuals
car::qqPlot(
  x = residuals(dentalModel),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals (ft)"
)

# Demo code for making Tukey-Anscombe plot ----
## Farming Barley Study
ggplot(
  data = data.frame(
    residuals = residuals(barleyModel),
    fitted = fitted.values(barleyModel)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 2) +
  geom_hline( ## Adds reference line at zero
    yintercept = 0,
    linetype = "dashed",
    color = "grey50"
  ) +
  geom_smooth( ## Adds the smoothed line
    formula = y ~ x,
    method = stats::loess,
    method.args = list(degree = 1),
    se = FALSE,
    linewidth = 0.5
  ) +
  theme_bw() +
  xlab("Fitted values (BPA)") +
  ylab("Residuals (BPA)")

# Demo code for making Tukey-Anscombe plot ----
## Dental Pain Study
ggplot(
```

```r
  data = data.frame(
    residuals = residuals(dentalModel),
    fitted = fitted.values(dentalModel)
  ),
  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") +
  ylab("Residuals")

# Demo code for index plots ----
## Farming Barley Study
## Using the residuals from the model which includes the block
ggplot(
  data = data.frame(
    residuals = barleyModel$residuals,
    index = 1:length(barleyModel$residuals)
  ),
  mapping = aes(x = index, y = residuals)
) +
  geom_point(size = 1.5) +
  geom_line() +
  theme_bw() +
  geom_hline(
    yintercept = 0,
    linetype = "dashed",
    color = "red"
  ) +
  xlab("Measurement order") +
  ylab("Residuals")
# Demo code for Alternative Index Plot for Barley Yields ----

# Note: I'm using the boastUtils package for a different color palette.
# You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

ggplot(
  data = barleyData,
  mapping = aes(
    x = Order,
    y = Yield,
    color = Field,
    shape = Varietal,
    linetype = Field
  )
) +
```

```
  geom_point(size = 2) +
  geom_path(
    mapping = aes(group = Field)
  ) +
  ggplot2::theme_bw() +
  xlab("Planting/Havesting Order") +
  ylab("Yield (BPA)") +
  scale_color_manual(values = boastUtils::psuPalette)

# Demo code for an interaction plot ----

# Note: I'm using the boastUtils package for a different color palette.
#You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

# Interaction Plot for Field and Treatment
ggplot2::ggplot(
  data = barleyData,
  mapping = aes(
    x = Varietal,
    y = Yield,
    color = Field,
    shape = Field,
    linetype = Field,
    group = Field
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Variety") +
  ylab("Yield (BPA)") +
  labs(color = "Field") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

# Note: I'm using the boastUtils package for a different color palette.
# You would need to install this package before using
# devtools::install_github("EducationShinyAppTeam/boastUtils")

# Demo code for an interaction plot ----
## Interaction Plot for Tolerance and Drug
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = drug,
    shape = drug,
    linetype = drug,
    group = drug
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
```

```
  ylab("Relief") +
  labs(color = "Drug", linetype = "Drug", shape = "Drug") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

## Interaction Plot for Tolerance and Acupuncture
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = acupuncture,
    shape = acupuncture,
    linetype = acupuncture,
    group = acupuncture
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
  ylab("Relief") +
  labs(color = "Acupuncture", linetype = "Acupuncture", shape = "Acupuncture") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

## Interaction Plot for Tolerance and Treatments
ggplot2::ggplot(
  data = dentalData,
  mapping = aes(
    x = tolerance,
    y = relief,
    color = acupuncture:drug,
    shape = acupuncture:drug,
    linetype = acupuncture:drug,
    group = acupuncture:drug
  )
) +
  stat_summary(fun = "mean", geom = "point") +
  stat_summary(fun = "mean", geom = "line") +
  ggplot2::theme_bw() +
  xlab("Pain Tolerance") +
  ylab("Relief") +
  labs(color = "Treatment", linetype = "Treatment", shape = "Treatment") +
  theme(
    legend.position = "right"
  ) +
  scale_color_manual(values = boastUtils::psuPalette)

# Omnibus Test/Modern ANOVA Table
parameters::model_parameters(
  model = barleyModel,
  effectsize_type = c("eta", "omega", "epsilon"),
  type = 3, # You will want to use these, esp. for Imbalance
  drop = "(Intercept)",
```

```
  verbose = FALSE
) %>%
  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 Barley Crop Yield 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 Relative Efficiency of the Block
## Farming Barley Study
block.RelEff(
  aov.obj = barleyModel,
  blockName = "Field",
  trtName = "Varietal"
)

# Point Estimates for Farming Barley
pEst <- dummy.coef(barleyModel)
pEst <- unlist(pEst)
names(pEst) <- c(
  "Grand Mean",
  levels(barleyData$Field), # I know that this is the correct order because
  levels(barleyData$Varietal) # I ran dummy.coef(barleyModel) in my console first
  )

data.frame("Estimate" = pEst) %>%
  knitr::kable(
  digits = 3,
  caption = "Point Estimates from the Barley Crop Yield Study",
  booktabs = TRUE,
  align = "c"
  ) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )

pEst <- dummy.coef(barleyModel)
pEst <- unlist(pEst[which(names(pEst) != "Field")])
names(pEst) <- c(
  "Grand Mean",
  levels(barleyData$Varietal) # Using levels here will help stop the accidental
  # mislabeling of estimates
  )

data.frame("Estimate" = pEst) %>%
  knitr::kable(
  digits = 3,
  caption = "Point Estimates from the Barley Crop Yield Study",
  booktabs = TRUE,
  align = "c"
```

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

# Demo code for post hoc pairwise via emmeans ----
## Barley Study
barleyPostHoc1 <- emmeans::emmeans(
  object = barleyModel,
  specs = pairwise ~ Varietal,
  adjust = "tukey",
  level = 0.9
)

barleyEffects1 <- as.data.frame(
  eff_size(
    object = barleyPostHoc1,
    sigma = sigma(barleyModel),
    edf = df.residual(barleyModel)
  )
) %>%
  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(barleyPostHoc1$contrasts) %>%
  left_join(
    y = barleyEffects1,
    by = join_by(contrast == contrast)
  ) %>%
  knitr::kable(
  digits = 3,
  caption = "Post Hoc Comparisons for Barley Varietal",
  col.names = c("Pair", "Difference", "SE", "DF", "t", "p-value", "Cohen's d",
                "Prob. of Superiority"),
  align = "lcccccccc",
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = c("HOLD_position")
  )

# Demo Code for Post Hoc Pairwise Comparisons via tukeyHSD ----
barleyPH <- TukeyHSD(
  x = barleyModel,
  which = "Varietal", # We need to specify which model term we want
  conf.level = 0.9
)

knitr::kable(
  barleyPH$Varietal,
  digits = 4,
```

```
  caption = "Post Hoc Tukey HSD Comparisons",
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'cccc',
  booktabs = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

## DescTools Pairwise Method
dtPH <- DescTools::PostHocTest(
  x = barleyModel, # Your aov/lm object
  which = "Varietal", # Specify which factor
  method = "bonf", # Your chosen method
  conf.level = 0.9 # 1 -- Your Overall Type I Error Rate
)

## Kable Code for DescTools
knitr::kable(
  x = dtPH$Varietal, # Notice the use of the factor name
  digits = 3,
  caption = paste( # Creates a nice title; copy at will
    "Post Hoc",
    attr(dtPH, "method"),
    "Comparisons"
  ),
  col.names = c("Difference", "Lower Bound",
                "Upper Bound", "Adj. p-Value"),
  align = 'lcccc',
  booktabs = TRUE,
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )

anova.PostHoc(
  aov.obj = barleyModel, # Our aov output
  response = "Yield", # Our response variable
  mainEffect = "Varietal" # Our factor variable
) %>%
  knitr::kable(
    digits = 3,
    caption = "Post Hoc Comparison Effect Sizes",
    col.names = c("Pairwise Comparison","Cohen's d", "Hedge's g",
                  "Prob. of Superiority"),
    align = 'lccc',
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("condensed", "boardered"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

```r
# Demo Code for Contrasts using emmeans ----
## Get the means
barleyMeans <- emmeans::emmeans(
  object = barleyModel,
  specs = ~ Varietal # Nothing goes on the left of ~; list what term you want
)

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

## Apply the contrasts
barleyContrasts1 <- emmeans::contrast(
  object = barleyMeans, # Notice that this is the means object
  method = list(
    "Company A vs. Company B" = c(1/2, 1/2, -1/2, -1/2)
  ),
  adjust = "none" # No MC/SI adjustment
)

## Add effect sizes and make a nice looking table
as.data.frame(barleyContrasts1) %>%
  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 = "Barley Study Main Effects Contrast on Varietal Company",
    col.names = c("Contrast", "Difference", "SE", "DF", "t Statistic",
                  "p-value", "Cohen's d", "Prob. of Superiority"),
    align = "lcccccccc",
    booktabs = TRUE
  ) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = c("HOLD_position", "scale_down")
  )

# Demo Code for Contrasts via base R ----
## Define the contrast
company <- c(1/2, 1/2, -1/2, -1/2)

## Bind the contrast to our factor
contrasts(barleyData$Varietal) <- company

## Refit our model so that our contrast gets tested
barleyContrast <- aov(
  formula = Yield ~ Varietal + Field,
  data = barleyData
)

## Get the updated ANOVA Table
### Remember, you could also use the DescTools package for Scheffé here
conOut <- summary.aov(
  object = barleyContrast,
  split = list(
    Varietal = list(
      "Company A vs. Company B" = 1
    )
```

```
  )
)

## Make a nice table
knitr::kable(
  x = conOut[[1]],
  digits = 4,
  col.names = c(
    "DF", "SS", "MS", "F", "p-value"),
  caption = "ANOVA Table for Barley Crop Yield Contrasts",
  booktabs = TRUE,
  align = rep("c", 5)
) %>%
  kableExtra::kable_styling(
    font_size = 12,
    latex_options = c("HOLD_position")
  )
```