2-Way Factorial ANOVA, a 3x5 Design

Orthogonal Polynomial Trend Analysis

Bruce Dudek

2025-04-08

Contents

1 Introduction

This document is intended for the use of students in the APSY511 statistics course at the University at Albany. However, all of the ideas and examples provide a template for approaching 2-way factorial designs, with a focus on trend analysis for a quantitative IV.

The document can be a good supplement to the primary tutorial document on 2-way ANOVA that is also available on bcdudek.net. It also builds on the accompanying document that laid out trend analysis on a 2x7 design.

The user is expected to have a comfort level with the ANOVA terminology of effects: main effects and interactions; simple main effects; simple main effect contrasts; interaction contrasts; orthogonal contrast sets of analytical contrasts. Help with this terminology can be found in the 3-way tutorial document provided.

The document does not do a full analysis on the data set (very little EDA, no evaluation of assumptions, etc). The focus is on handling IVs that are quantitative by using orthogonal polynomial trend analysis and provision of code templates for analogous designs.

```
library(gt)
library(psych)
library(emmeans)
library(knitr)
library(phia)
library(ggplot2)
library(ggthemes)
library(Rmisc)
library(sjstats)
library(afex)
```

2 Data Import and Management

The data set is part of a larger study on disinhibitory effects of alcohol in several genetically defined stocks of mice (Dudek lab). Three of the mouse stocks from that study are included here. For purposes of choosing an orthogonal set of contrasts involving the 3-level mouse factor, it is assumed that two of them (AU and CBY) are genetically more closely related to each other than to a third strain, C57/BL6. This relatedness claim may not be fully accurate,

but for purposes of this template document, assume that it is true, since it drives the contrast choice.

Mice were treated with one of five doses of ethanol and tested for fifteen minutes in an automated activity monitor. The DV was simply the distance traveled.

```
data1 <- read.csv("etoh1_511class.csv", stringsAsFactors = TRUE)
```

For some analyses (perhaps **afex**) a subject number variable is required and it was imported in the .csv file. Here it is converted to a factor.

```
data1$snum <- as.factor(data1$snum)
```

Since the dose variable in the imported data frame is a string variable, the order of its levels are alphabetical. This needs to be reordered to reflect the fact that the control condition (Zero) was given zero alcohol and should be first. Dose is now an ordered factor.

```
  data1$dose<- ordered(data1$dose,
    levels=c("Zero","1","1.5","2","2.5"))
levels(data1$dose)
```

```
[1] "Zero" "1"    "1.5"  "2"    "2.5"
```

In order to draw a line graph with the proper scaling of the dose variable and placement of the groups at their correct dose value, a new numeric variable is created for dose. It is used only in the graphing functions.

```
data1[which(data1$dose == "Zero"),"edose"] <- 0
data1[which(data1$dose == "1"),"edose"] <- 1
data1[which(data1$dose == "1.5"),"edose"] <- 1.5
data1[which(data1$dose == "2"),"edose"] <- 2
data1[which(data1$dose == "2.5"),"edose"] <- 2.5
class(data1$edose)
```

```
[1] "numeric"
```

3 Graph of the dose response curves

A **ggplot2** graph can produce a line graph with std error bars (SEM). This requires first extracting descriptive information from the raw data frame into a format that **ggplot2** can handle.

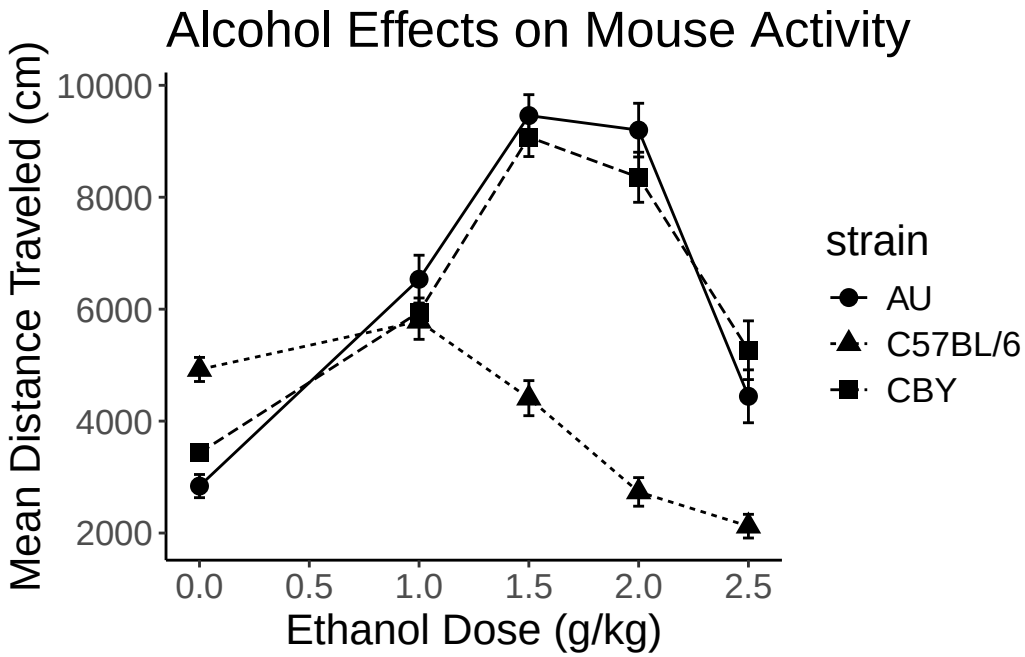
```
mouse_summ <- Rmisc::summarySE(data1,measurevar="dist15", groupvars=c("edose","strain"))
#str(mouse_summ)
# rename the column that contains the mean to something less confusing
colnames(mouse_summ) <- c("ethanoldose", "strain","N", "mean", "sd", "se", "95%ci" )
gt::gt(mouse_summ)
```

ethanoldose	strain	N	mean	sd	se	95%ci
0.0	AU	27	2839.333	1075.7231	207.02300	425.5419
0.0	C57BL/6	28	4922.821	1139.0655	215.26315	441.6835
0.0	CBY	31	3432.645	435.5221	78.22208	159.7508
1.0	AU	26	6534.577	2187.8689	429.07640	883.6994
1.0	C57BL/6	28	5784.143	1709.9253	323.14551	663.0398
1.0	CBY	32	5947.188	1429.8204	252.75892	515.5052
1.5	AU	28	9459.071	1975.6674	373.36604	766.0838
1.5	C57BL/6	28	4410.179	1657.7237	313.28033	642.7981
1.5	CBY	33	9075.364	1999.8239	348.12465	709.1067
2.0	AU	27	9199.074	2492.0609	479.59735	985.8265
2.0	C57BL/6	27	2735.111	1326.8037	255.34349	524.8661
2.0	CBY	31	8356.613	2490.4440	447.29695	913.5022
2.5	AU	27	4443.222	2456.4900	472.75172	971.7551
2.5	C57BL/6	27	2122.963	1098.9411	211.49131	434.7266
2.5	CBY	32	5266.656	2967.8770	524.65149	1070.0338

The user is best served by keeping this graph visible while attempting to interpret the patterns of results coming from the various analyses.

```
p1 <-ggplot(mouse_summ, aes(x=ethanoldose, y=mean, group=strain)) +
  geom_line(aes(linetype=strain)) +
  geom_point(aes(shape=strain),size=3) +
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), colour="black", width=.05)+
  xlab("Ethanol Dose (g/kg)")+
  ylab("Mean Distance Traveled (cm)")+
  ggtitle("Alcohol Effects on Mouse Activity")+
```

```
  theme_classic()+
  theme(text = element_text(size=16))
p1
```



The visual impression of the plot leaves one with a fairly simple impression of what the outcome was. The dose response curve shapes for the two "related" strains (AU and CBY) were very similar. But those shapes were different from the shape of the B6 curve. Some subtle differences between the AU and CBY curves may exist, but we need to test for that.

4 Perform the Omnibus ANOVA

An omnibus analysis does not require that factors have orthogonal contrast coding schemes assigned. Dummy coding (`contr.treatment`) can work and effect coding (`contr.sum`) can be preferred (as in the **afex** package). However we have followed a recommended logic of evaluating omnibus ANOVAs with analytical/orthogonal contrasts in place for the omnibus analyses; some followup analyses can the be more direct.

In factorial designs it becomes very important to understand these coding schemes and set the correct contrast sets. One particular issue here is that when the dose factor was changed above to an ordered factor, its contrasts were automatically set to orthogonal polynomials (this is an R "feature").

```
#|results: hold
class(data1$dose)
```

```
[1] "ordered" "factor"
```

```
gt::gt(round(as.data.frame(contrasts(data1$dose)),4))
```

.L	.Q	.C	^4
-0.6325	0.5345	-0.3162	0.1195
-0.3162	-0.2673	0.6325	-0.4781
0.0000	-0.5345	0.0000	0.7171
0.3162	-0.2673	-0.6325	-0.4781
0.6325	0.5345	0.3162	0.1195

However, there is a problem here. That automatic choice of orthogonal polynomials for the dose variable assumes that the levels of the factor (the doses) are equally spaced. In this study, they were not. So, we need to re-specify the set of trend coefficients using the `contr.poly` function as we saw with the 1-way trend tutorial.

```
dosecontrasts <- contr.poly(5, scores=c(0,1,1.5,2,2.5))
contrasts(data1$dose) <- dosecontrasts
contrasts(data1$dose)
```

```
              .L         .Q         .C          ^4
Zero -0.72782534  0.4907292 -0.1676525  0.03671115
1    -0.20795010 -0.4728845  0.6311625 -0.36711155
1.5   0.05198752 -0.4595010 -0.2169621  0.73422310
2     0.31192515 -0.1159905 -0.6213006 -0.55066732
2.5   0.57186277  0.5576469  0.3747527  0.14684462
```

We also need contrasts for the strain variable. The following set makes sense based on the logic from the introductory description of the data set and the mouse strains.

```
contrastsstrain <- cbind(
                      ac1=c(-1,2,-1),
                      ac2=c(1,0,-1))
contrasts(data1$strain) <- contrastsstrain
contrasts(data1$strain)
```

```
        ac1 ac2
AU       -1   1
C57BL/6   2   0
CBY      -1  -1
```

Now fit the base `aov` model:

```
fitbase.aov <- aov(dist15~dose*strain, data=data1)
Anova(fitbase.aov,type="III")
```

```
Anova Table (Type III tests)

Response: dist15
                Sum Sq  Df  F value    Pr(>F)
(Intercept) 1.3640e+10   1 3804.658 < 2.2e-16 ***
dose        1.0387e+09   4   72.427 < 2.2e-16 ***
strain      5.6724e+08   2   79.109 < 2.2e-16 ***
dose:strain 7.9054e+08   8   27.563 < 2.2e-16 ***
Residuals   1.4950e+09 417
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5 Interaction Contrasts and Main Effect Contrasts with `summary.lm`

As we have seen with the previous 1 way and 2 way tutorials as well as the tutorial documents on coding schemes, the most efficient way of obtaining tests on main effect contrasts and interaction contrasts is the use of the `summary.lm` on an `aov` object. However, this efficiency only exists because we have taken the approach assigning orthogonal sets of contrasts to all factors prior to execution of the omnibus ANOVA using `aov` or `lm`. When this is done (as was the case here, above), the interpretation of the main effect contrasts and interaction contrasts produced by `summary.lm` is straight forward. We have been accustomed to seeing contrasts tested with F tests in other software, but the fact of having t-tests here is not an issue since all of these effects are 1 df effects in the ANOVA sense. So, t-square=F…..

These tests are equivalent to F tests using Type III SS.

```
summary.lm(fitbase.aov)
```

```
Call:
```

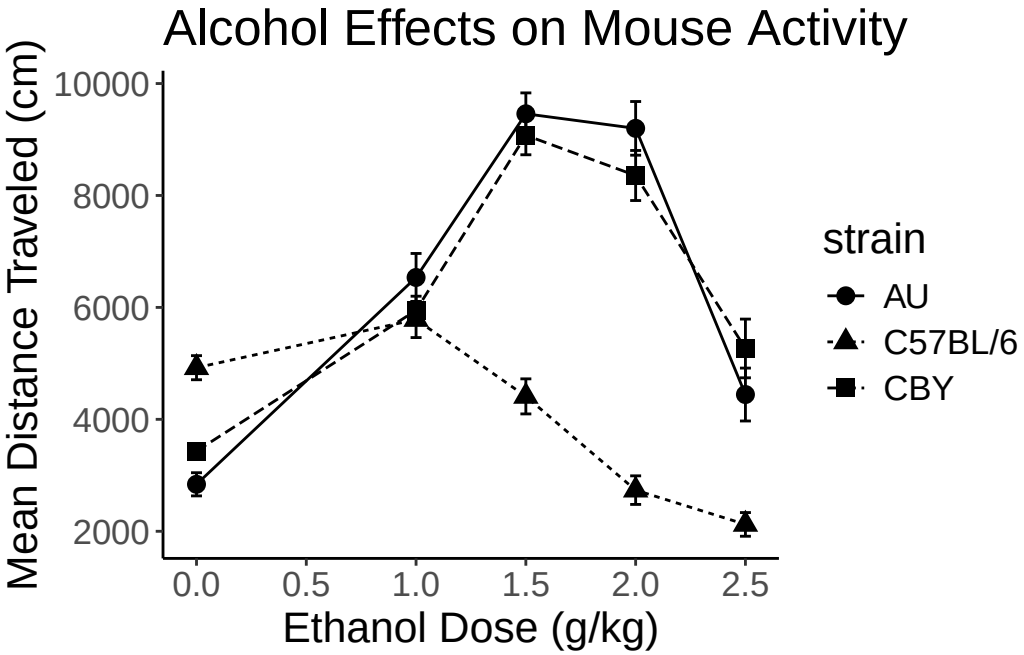
```
aov(formula = dist15 ~ dose * strain, data = data1)

Residuals:
    Min      1Q  Median      3Q     Max
-6015.6 -1033.1    21.4  1082.5  8210.3

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       5635.26      91.36  61.682  < 2e-16 ***
dose.L             780.85     204.76   3.813 0.000158 ***
dose.Q           -3147.37     204.05 -15.424  < 2e-16 ***
dose.C           -1166.16     205.08  -5.686 2.44e-08 ***
dose^4             371.97     203.25   1.830 0.067948 .
strainac1         -820.11      65.21 -12.576  < 2e-16 ***
strainac2           39.68     110.83   0.358 0.720494
dose.L:strainac1 -1635.07     146.01 -11.198  < 2e-16 ***
dose.Q:strainac1   834.02     145.63   5.727 1.96e-08 ***
dose.C:strainac1  1065.50     146.26   7.285 1.62e-12 ***
dose^4:strainac1  -135.51     145.35  -0.932 0.351730
dose.L:strainac2    60.76     248.64   0.244 0.807063
dose.Q:strainac2  -651.07     247.57  -2.630 0.008859 **
dose.C:strainac2  -222.52     248.98  -0.894 0.371981
dose^4:strainac2  -270.26     246.08  -1.098 0.272723
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1893 on 417 degrees of freedom
Multiple R-squared:  0.6178,    Adjusted R-squared:  0.605
F-statistic: 48.14 on 14 and 417 DF,  p-value: < 2.2e-16
```

The `summary.lm` approach to obtaining main effect and interaction contrasts is by far the easiest. The downside is that if one wants F tests, one has to square the t values. But these tests are equivalent to type III F tests.

6 Follow up analyses using emmeans and phia

It is helpful to have the graph of the data visible as the interpretation of these analyses are performed, so it is reproduced here.

```
p1
```



Both **emmeans** and **phia** can provide numerous followup analyses including simple effects and contrasts. For each of them, the relevant orthogonal polynomial and strain contrasts have to be provided separately. They cannot be drawn from the contrasts that were established above for the two factors.

The overall logic is that the interaction term is only fully understood when interaction contrasts, simple main effects, and simple main effect contrasts are evaluated. At times, it is also useful to perform *post hoc* multiple comparison tests to evaluate dose effects within a single strain.

All of these things are demonstrated here as well as main effect contrasts. The focus is more on providing the template for all of the kinds of analyses rather than providing the best analysis of this particular data set.

6.1 Contrasts and Simple effects with emmeans

This section was initially thought to be a set of analyses that **emmeans** could not perform (trend contrasts with unequal intervals). However…..

The approach taken here is not available in the CRAN release of **emmeans** as of the writing of this document. The release version has a built in function to handle orthogonal polynomials ("poly"), but it does not use coefficients produced by `contrast.poly` and cannot handle factors where the quantitative levels of the IV are unequally spaced. Communications with the **emmeans** author (Russ Lenth) resulted in his creation of another function ("opoly") that permits specification of the values of the quantitative IV with the `scores` argument in the

same way that `contr.poly` does. This new function is available in the Github development version of **emmeans** and it should make it to CRAN in an upcoming release. Without this amazing responsiveness of Lenth to my inquiry, performing trend analysis on this particular unequally spaced dose IV would have been a major task.

See the Github page where you can find install instructions and look at the "issues" tab to see the request and conversation.

emmeans on GitHub

The use of **emmeans** here follows the basic logic we developed in earlier tutorials, with a few new capabilities added.

6.1.1 Main effect contrasts on dose: trend

For the dose/strain example data set, these main effect contrasts would not be of interest since dose had a sizable interaction with strain. But for template purposes of this document it is demonstrated.

First extract the grid of descriptive stats required - collapsing on strain.

```
dose.emm <- emmeans::emmeans(fitbase.aov, ~dose)
```

```
NOTE: Results may be misleading due to involvement in interactions
```

```
dose.emm
```

```
 dose emmean  SE  df lower.CL upper.CL
 Zero   3732 205 417     3330     4134
 1      6089 205 417     5686     6491
 1.5    7648 201 417     7252     8044
 2      6764 206 417     6359     7168
 2.5    3944 205 417     3542     4347

Results are averaged over the levels of: strain
Confidence level used: 0.95
```

Passing the "opoly" argument to the `contrast` function, along with the "scores" argument produces tests of the trend components on the marginal means for the dose factor (collapsed on strain and thus not of actual interest in this data set)

```
emmeans::contrast(dose.emm, "opoly", scores=c(0,1,1.5,2,2.5))
```

```
 contrast  estimate  SE  df t.ratio p.value
 linear         781 205 417   3.813  0.0002
 quadratic    -3147 204 417 -15.424  <.0001
 cubic        -1166 205 417  -5.686  <.0001
 quartic        372 203 417   1.830  0.0679

Results are averaged over the levels of: strain
```

```
#show the coefficients used.
knitr::kable(coef(.Last.value))
```

6.1.2 Simple Main Effects of Dose

The character of an interaction can be partially addressed with examination of simple main effects. Here, we examine the effect of dose at each level of strain.

First extract the grid of desctriptive statistics.

```
d.s.emm <- emmeans::emmeans(fitbase.aov, ~dose, by="strain")
d.s.emm
```

```
strain = AU:
 dose emmean  SE  df lower.CL upper.CL
 Zero   2839 364 417     2123     3556
 1      6535 371 417     5805     7265
 1.5    9459 358 417     8756    10162
 2      9199 364 417     8483     9915
 2.5    4443 364 417     3727     5160

strain = C57BL/6:
 dose emmean  SE  df lower.CL upper.CL
 Zero   4923 358 417     4219     5626
 1      5784 358 417     5081     6488
 1.5    4410 358 417     3707     5114
 2      2735 364 417     2019     3451
 2.5    2123 364 417     1407     2839

strain = CBY:
```

```
 dose emmean  SE  df lower.CL upper.CL
 Zero   3433 340 417     2764     4101
 1      5947 335 417     5289     6605
 1.5    9075 330 417     8427     9723
 2      8357 340 417     7688     9025
 2.5    5267 335 417     4609     5925

Confidence level used: 0.95
```

Use the `contrast` and `test` functions to obtain the three 4 df simple main effect tests. The "eff" argument is an efficient way of providing a set of coding vectors and it is "effect" coding - doesn't really matter since we are not asking for contrasts. The `joint` argument ensures that the overall simple effects are tested rather than contrasts. Unsurprisingly, the effect of dose significant in each of the three strains.

```
emmeans::test(emmeans::contrast(d.s.emm, "eff"), joint=TRUE)
```

```
 strain  df1 df2 F.ratio p.value note
 AU        4 417  64.006  <.0001  d
 C57BL/6   4 417  17.849  <.0001  d
 CBY       4 417  47.087  <.0001  d

d: df1 reduced due to linear dependence
```

6.1.3 Pairwise comparisons to follow up Simple Main Effects

In the following section, we will apply trend analysis to the dose effect for the set of simple main effects. But at times, it may be useful to do post hoc multiple comparison tests on sets of means such as the dose groups within each strain, separately.

This section shows how easily **emmeans** handles that problem.

```
pairs(d.s.emm, adjust="tukey")
```

```
strain = AU:
 contrast   estimate  SE  df t.ratio p.value
 Zero - 1      -3695 520 417  -7.103  <.0001
 Zero - 1.5    -6620 511 417 -12.962  <.0001
 Zero - 2      -6360 515 417 -12.341  <.0001
 Zero - 2.5    -1604 515 417  -3.112  0.0169
 1 - 1.5       -2924 516 417  -5.671  <.0001
```

```
 1 - 2         -2664 520 417  -5.121  <.0001
 1 - 2.5        2091 520 417   4.020  0.0007
 1.5 - 2         260 511 417   0.509  0.9865
 1.5 - 2.5      5016 511 417   9.821  <.0001
 2 - 2.5        4756 515 417   9.229  <.0001

strain = C57BL/6:
 contrast   estimate  SE  df t.ratio p.value
 Zero - 1       -861 506 417  -1.702  0.4338
 Zero - 1.5      513 506 417   1.013  0.8493
 Zero - 2       2188 511 417   4.284  0.0002
 Zero - 2.5     2800 511 417   5.482  <.0001
 1 - 1.5        1374 506 417   2.715  0.0534
 1 - 2          3049 511 417   5.970  <.0001
 1 - 2.5        3661 511 417   7.169  <.0001
 1.5 - 2        1675 511 417   3.280  0.0099
 1.5 - 2.5      2287 511 417   4.478  0.0001
 2 - 2.5         612 515 417   1.188  0.7584

strain = CBY:
 contrast   estimate  SE  df t.ratio p.value
 Zero - 1      -2515 477 417  -5.270  <.0001
 Zero - 1.5    -5643 474 417 -11.915  <.0001
 Zero - 2      -4924 481 417 -10.238  <.0001
 Zero - 2.5    -1834 477 417  -3.844  0.0013
 1 - 1.5       -3128 470 417  -6.659  <.0001
 1 - 2         -2409 477 417  -5.049  <.0001
 1 - 2.5         681 473 417   1.438  0.6036
 1.5 - 2         719 474 417   1.518  0.5515
 1.5 - 2.5      3809 470 417   8.108  <.0001
 2 - 2.5        3090 477 417   6.476  <.0001

P value adjustment: tukey method for comparing a family of 5 estimates
```

6.1.4 Simple main effects of dose: trend contrasts

As discussed above, the "opoly" argument provides the appropriate tests of trend on the dose factor. Here this is done separately in each strain, as a function of how the d.s.emm object was created.

```
emmeans::contrast(d.s.emm, "opoly", scores=c(0,1,1.5,2,2.5))
```

```
strain = AU:
 contrast  estimate  SE  df t.ratio p.value
 linear        2477 365 417   6.791  <.0001
 quadratic    -4632 365 417 -12.706  <.0001
 cubic        -2454 367 417  -6.689  <.0001
 quartic        237 362 417   0.656  0.5124

strain = C57BL/6:
 contrast  estimate  SE  df t.ratio p.value
 linear       -2489 361 417  -6.903  <.0001
 quadratic    -1479 360 417  -4.110  <.0001
 cubic          965 361 417   2.670  0.0079
 quartic        101 360 417   0.280  0.7793

strain = CBY:
 contrast  estimate  SE  df t.ratio p.value
 linear        2355 338 417   6.966  <.0001
 quadratic    -3330 335 417  -9.941  <.0001
 cubic        -2009 337 417  -5.967  <.0001
 quartic        778 334 417   2.331  0.0202
```

```
# show the coefficient matrix
knitr::kable(coef(.Last.value))
```

6.1.5 Two-way interaction contrasts

These single df contrasts are somewhat difficult to obtain in **emmeans**. The simplest way is to use the `summary.lm` function applied to the `aov` fit as seen above. But it is now possible to obtain the full set of eight orthogonal interaction contrasts available with trend on the dose factor and a specially created orthogonal contrast set which makes sense for the strain factor in this study, given the nature of the three strains described in the introduction.

First we have to create the grid of means on the full design - cell means (these are the "estimated marginal means" in regression terminology - although here they are cell means rather than actual marginals).

```
#full.emm <- emmeans::emmeans(fitbase.aov, ~dose*strain)
full.emm = emmeans(fitbase.aov, ~dose*strain)
full.emm
```

```
 dose strain  emmean  SE  df lower.CL upper.CL
 Zero AU        2839 364 417     2123     3556
```

```
 1    AU        6535 371 417     5805     7265
 1.5  AU        9459 358 417     8756    10162
 2    AU        9199 364 417     8483     9915
 2.5  AU        4443 364 417     3727     5160
 Zero C57BL/6   4923 358 417     4219     5626
 1    C57BL/6   5784 358 417     5081     6488
 1.5  C57BL/6   4410 358 417     3707     5114
 2    C57BL/6   2735 364 417     2019     3451
 2.5  C57BL/6   2123 364 417     1407     2839
 Zero CBY       3433 340 417     2764     4101
 1    CBY       5947 335 417     5289     6605
 1.5  CBY       9075 330 417     8427     9723
 2    CBY       8357 340 417     7688     9025
 2.5  CBY       5267 335 417     4609     5925

Confidence level used: 0.95
```

Now we set a custom function for the creation of the orthogonal set of contrasts to be employed. They are the same that we used above for the omnibus analysis, but **cannot** use the contrasts already assigned to the strain factor - we need to create them a different way. In this custom function you can find the two contrasts that I have placed into a data frame called "ocon". This function will be called by the **emmeans** `contrast` function to perform the contrast task on the full.emm grid of means - the omnibus `aov` fit is not needed. This .emmc custom function can be renamed and set up to use any set of contrasts - they don't have to be orthogonal, although the value of orthogonal contrasts has been recognized.

```
### Setting up a custom contrast function
orthstrain.emmc <- function(...) {
    ocon <- as.data.frame(cbind(strcon1 <- c(-1,2,-1),
                                strcon2 <- c(1,0,-1)))
    names(ocon) <- c("strcon1","strcon2")
    attr(ocon, "desc") <- "Strain Contrasts"
    ocon
}
```

emmeans has a custom function built in for creating/using orthogonal polynomial contrasts called `poly.emmc` but we cannot use that one because it doesn't permit unequal spacing of the levels of the dose IV. So, an alternative function is now in the development version of **emmeans**. It is called `opoly.emmc`. It can be used directly here along with our custom `orthstrain` function to pass the relevant contrasts to the interaction argument in `contrast`. The scores argument is needed for the unequal spacing that `opoly` needs to know about.

With this syntatical structure, we have tests of each of the eight interaction contrasts, and they match the product of the `summary.lm` function above - reassuring.

I have also asked **emmeans** to show the coding vectors that were created for the analyses. They are in the same order as the list in the table from the `contrast` function.

```
# note that the scores argument is used for opoly
emmeans::contrast(full.emm,
                  interaction=c("opoly","orthstrain"),
                  scores=c(0,1,1.5,2,2.5))
```

```
 dose_opoly strain_orthstrain estimate  SE  df t.ratio p.value
 linear     strcon1              -9810 876 417 -11.198  <.0001
 quadratic  strcon1               5004 874 417   5.727  <.0001
 cubic      strcon1               6393 878 417   7.285  <.0001
 quartic    strcon1               -813 872 417  -0.932  0.3517
 linear     strcon2                122 497 417   0.244  0.8071
 quadratic  strcon2              -1302 495 417  -2.630  0.0089
 cubic      strcon2               -445 498 417  -0.894  0.3720
 quartic    strcon2               -541 492 417  -1.098  0.2727
```

```
#coef(.Last.value)
```

6.1.6 Pairwise approach to interaction contrasts

One additional possibility is an alternative to examination of the "orthogonal contrasts on orthogonal contrasts" idea reflected in the interaction contrasts above.

We might want to as a more *post hoc* set of questions. Can we take pairs of strains at a time and ask do they differ in the trend components. Thus each contrast would pit one strain against another and examine each of the four trend components. *E.g.*, DoseLin x AUvsCBY.

For this, we need to write another .emmc custom function and I simply called it "nonorthstrain".

Note that the AU vs CBy contrast suggested here was already a member of the orthogonal set examined above.

```
### Setting up a custom contrast function
nonorthstrain.emmc <- function(...) {
    nonorth <- as.data.frame(cbind(AUvsB6 <- c(1,-1,0),
                                   AUvsCBy <- c(1,0,-1),
                                   B6vsCBy <- c(0,1,-1)))
```

```
    names(nonorth) <- c("AUvsB6","AUvsCBy","B6vsCBy")
    attr(nonorth, "desc") <- "Strain Pairs"
    nonorth
}
```

And we use that custom function for strain contrasts along with the `opoly` function again. A Holm adjustment might be warrented here since these pairwise comparisons were likely *post hoc*.

```
# note that the scores argument is used for opoly
emmeans::contrast(full.emm,
                  interaction=c("opoly","nonorthstrain"),
                              scores=c(0,1,1.5,2,2.5),
                  adjust="holm")
#knitr::kable(coef(.Last.value))
```

```
 dose_opoly strain_nonorthstrain estimate  SE  df t.ratio p.value
 linear     AUvsB6                   4966 513 417   9.683  <.0001
 quadratic  AUvsB6                  -3153 512 417  -6.154  <.0001
 cubic      AUvsB6                  -3419 515 417  -6.640  <.0001
 quartic    AUvsB6                    136 510 417   0.267  1.0000
 linear     AUvsCBy                   122 497 417   0.244  1.0000
 quadratic  AUvsCBy                 -1302 495 417  -2.630  0.0532
 cubic      AUvsCBy                  -445 498 417  -0.894  1.0000
 quartic    AUvsCBy                  -541 492 417  -1.098  1.0000
 linear     B6vsCBy                 -4844 494 417  -9.800  <.0001
 quadratic  B6vsCBy                  1851 492 417   3.764  0.0013
 cubic      B6vsCBy                  2974 494 417   6.022  <.0001
 quartic    B6vsCBy                  -677 491 417  -1.379  0.8433

P value adjustment: holm method for 12 tests
```

6.1.7 Interaction Comparisons with emmeans

At times, it is useful to conceptualize and analyze interaction comparisons, rather than, or in addition to interaction contrasts. These effects involve contrasts on one factor but not the other. For example we might as about *dose-quadratic x strain*. That question would ask: "is the quadratic shape in the three strains the same?", permitting any variation in the quadratic component across the strains to enter the interaction comparison term. This is in contrast to interaction contrasts which will have contrasts on both factors.

Interaction contrasts can be seen as decompositions of Interaction comparisons. Thus dose-quadratic x straincontrast1 and dose-quadratic x straincontrast2 are both 1 df contrasts. They could be pooled to give the 2 df dose-quadratic by strain interaction comparison.

Here, all four of the dose trend components are interacted with strain for four interaction comparisons:

- dose-linear x strain
- dose-quadratic x strain
- dose-cubic x strain
- dose-quartic x strain

The way that **emmeans** accomplished this is not immediately obvious when looking at the code. We can see that the `contrast` statement for the interaction contrasts is repeated here and assigned to an object. Then that object is passed to the `test` function. The arguments on the `test` function are critical. "joint=TRUE" tells test to jointly consider the contrasts on strain - essentially pool the contrasts involving straincontrast1 and straincontrast2. This yeilds the 2 df interaction comparisons. But how can `test` know to still ask about the effect of the dose contrasts? This is with the `by="dose_opoly"` argument. This says to interact each polynomial contrast with the remaining variable in the .emm object (strain). You can see how the `constrast` function labels the dose and strain contrasts in the table of eht intcontrasts object, and the dose_opoly label is found there.

```
intcontrasts <- emmeans::contrast(full.emm,
                  interaction=c("opoly","orthstrain"),
                  scores=c(0,1,1.5,2,2.5))
intcontrasts # repeats the table seen above for interaction contrasts
```

```
 dose_opoly strain_orthstrain estimate  SE  df t.ratio p.value
 linear     strcon1             -9810 876 417 -11.198  <.0001
 quadratic  strcon1              5004 874 417   5.727  <.0001
 cubic      strcon1              6393 878 417   7.285  <.0001
 quartic    strcon1              -813 872 417  -0.932  0.3517
 linear     strcon2               122 497 417   0.244  0.8071
 quadratic  strcon2             -1302 495 417  -2.630  0.0089
 cubic      strcon2              -445 498 417  -0.894  0.3720
 quartic    strcon2              -541 492 417  -1.098  0.2727
```

```
test(intcontrasts, joint=TRUE, by="dose_opoly")
```

```
 dose_opoly df1 df2 F.ratio p.value
 linear       2 417  62.728  <.0001
```

```
quadratic    2 417  19.181  <.0001
cubic        2 417  26.681  <.0001
quartic      2 417   1.087  0.3383
```

The conclusion is that we reject the three null hypotheses that linear, quadratic, and cubic shapes of dose are each the same in the three strains. Examination of the graphs helps "see" these varying shapes and with some experience, it is possible to see why the subtle cubic shape differs. It is similar in strains AU and CBy, but different in B6. This is why the dose_quadratic x straincontrast1 interaction contrast is significant.

6.1.8 Note on using emmeans

There are features of using **emmeans** that are very attractive to the analyst. However, obtaining effect sizes for the simple effects and contrasts is not something that it looks like can be accomplished. And since the tables produced by `test` and `contrast` provide t-tests rather than F tests, there are no SS calculated for the effects. This means that effect sizes such as eta-squared and partial eta-squared cannot be manually computed. Still working on this issue….

6.2 Contrasts and Simple effects with phia

The **phia** capabilities also permit examination of the simple effects and contrasts in a design such as this where orthogonal polynomials are appropriate for one factor.

One distinction in using **phia** compared to **emmeans** is that it can work with contrast vectors produced by `contr.poly` or custom created matrices of contrasts. Another distinction is that each contrast requires its own `testInteractions` function - a good bit less efficient when evaluation a large number of contrasts.

In order to work with contrasts, `testInteractions` needs them submitted as a list rather than as a numeric vector. I have done this by first, creating an object that has the coefficients. Note that this is not associated with the dose factor in the data1 data frame. It is an independent object at this point. Initially I have done this for a five-level variable such as dose. Strain follows.

```
dosetrend <- contr.poly(5, scores=c(0,1,1.5,2,2.5))
dosetrend
```

```
              .L         .Q         .C          ^4
[1,] -0.72782534  0.4907292 -0.1676525  0.03671115
[2,] -0.20795010 -0.4728845  0.6311625 -0.36711155
```

```
[3,]  0.05198752 -0.4595010 -0.2169621  0.73422310
[4,]  0.31192515 -0.1159905 -0.6213006 -0.55066732
[5,]  0.57186277  0.5576469  0.3747527  0.14684462
```

Next, we can extract individual columns of trend coefficients and assign them to objects that are lists. I did this for the linear, quadratic, and cubic but not quartic since we are not going to be very interested in quartic for this data set - and it saves time/space in this template document.

```
doselin <- list(dose=dosetrend[,1])
doselin
```

```
$dose
[1] -0.72782534 -0.20795010  0.05198752  0.31192515  0.57186277
```

```
dosequad <- list(dose=dosetrend[,2])
dosequad
```

```
$dose
[1]  0.4907292 -0.4728845 -0.4595010 -0.1159905  0.5576469
```

```
dosecubic <- list(dose=dosetrend[,3])
dosecubic
```

```
$dose
[1] -0.1676525  0.6311625 -0.2169621 -0.6213006  0.3747527
```

Now we do the same thing for an orthogonal set on the strain variable - the same set used above.

```
strainorth <- cbind(strain1=c(-1,2,-1),
                    strain2=c(1,0,-1))
strainorth
```

```
     strain1 strain2
[1,]      -1       1
[2,]       2       0
[3,]      -1      -1
```

Now extract the vectors as lists.

```
strainc1 <- list(strain=strainorth[,1])
strainc1
```

```
$strain
[1] -1  2 -1
```

```
strainc2 <- list(strain=strainorth[,2])
strainc2
```

```
$strain
[1]  1  0 -1
```

One thing to keep in mind is that the contrasts employed do not have to be orthogonal. The `testInteractions` function takes one at a time anyway.

An important caveat:

In all of these **phia** tables the residual df and SS are placed in the wrong columns in the table the df for residual should be 417 throughout and the SS is 1.495e+09. This is great example of how one should know what to expect for df and not just trust the software. This is just an outputting formatting error on the part of the programmer.

6.2.1 Main effect Contrasts

Evaluation of main effect contrasts is possible by simply passing the custom dose contrast list and leaving out any `fixed` or `across` arguments.

Linear is evaluated first:

```
testInteractions(fitbase.aov, custom=doselin, adjustment="none")
```

```
F Test:
P-value adjustment method: none
           Value     SE        Df Sum of Sq      F    Pr(>F)
dose1     780.85 204.76 1.000e+00  52138035 14.543 0.0001577 ***
Residuals        417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And then quadratic:

```
testInteractions(fitbase.aov, custom=dosequad, adjustment="none")
```

```
F Test:
P-value adjustment method: none
            Value     SE        Df Sum of Sq     F    Pr(>F)
dose1     -3147.4 204.05 1.000e+00 852931245 237.9 < 2.2e-16 ***
Residuals         417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then cubic:

```
testInteractions(fitbase.aov, custom=dosecubic, adjustment="none")
```

```
F Test:
P-value adjustment method: none
            Value     SE        Df Sum of Sq      F    Pr(>F)
dose1     -1166.2 205.08 1.000e+00 115930145 32.336 2.443e-08 ***
Residuals         417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

These tests match the output from **emmeans**, although I did not obtain the cubic and quartic contrasts to save space in the document. Analysts can generalize from what is shown here.

6.2.2 Simple Main Effects with phia

This section focuses on only one set of simple main effects, the effect of dose at levels of strain. Thus there are three tests. The table is a bit awkward and since it is wide it wraps the last several columns where the most relevant information is. The values in the first section of table are the estimates and std errors of each of the component contrast effects - but we only care about the three 4 df simple main effects here. We see that dose is significant in each of the three strains.

```
testInteractions(fitbase.aov, across="dose", fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
            dose1   dose2    dose3  dose4    SE1       SE2    SE3    SE4 Df
```

```
     AU    2476.7 -4632.5 -2454.18 237.22 364.68 3.650e+02 366.87 361.82  4
C57BL/6   -2489.3 -1479.3   964.83 100.96 360.63 3.600e+02 361.30 359.98  4
    CBY    2355.2 -3330.3 -2009.13 777.74 338.07 3.350e+02 336.71 333.63  4
Residuals                                417.00 1.495e+09
          Sum of Sq      F    Pr(>F)
     AU   917891291 64.006 < 2.2e-16 ***
C57BL/6   255967621 17.849  1.54e-13 ***
    CBY   675261926 47.087 < 2.2e-16 ***
Residuals
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.2.3 Simple Main Effect Contrasts with phia

By using the same "listed" contrast vectors from above, obtaining SME contrasts is a direct outcome produced by passing those custom vectors and using the `fixed` argument to specify the variable that is held at one level or the other for the two simple effects.

First, dose-linear at the three strains.

Once again, note the incorrect placement of the df and SS for the residual term.

```
testInteractions(fitbase.aov, custom=doselin, fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
                  Value     SE        Df Sum of Sq      F    Pr(>F)
     AU : dose1  2476.7 364.68 1.000e+00 165358081 46.123 3.840e-11 ***
C57BL/6 : dose1 -2489.3 360.63 1.000e+00 170820580 47.646 1.909e-11 ***
    CBY : dose1  2355.2 338.07 1.000e+00 173992429 48.531 1.274e-11 ***
Residuals               417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The dose-quadratic at those two levels of strain.

```
testInteractions(fitbase.aov, custom=dosequad, fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
                  Value     SE        Df Sum of Sq       F    Pr(>F)
```

```
     AU : dose1 -4632.5 364.59 1.000e+00 578799526 161.442 < 2.2e-16 ***
C57BL/6 : dose1 -1479.3 359.97 1.000e+00  60547684  16.888 4.775e-05 ***
    CBY : dose1 -3330.3 335.02 1.000e+00 354275222  98.817 < 2.2e-16 ***
Residuals               417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Then, dose-cubic at those twolevels of strain.

```
testInteractions(fitbase.aov, custom=dosecubic, fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
                   Value     SE        Df Sum of Sq       F    Pr(>F)
     AU : dose1 -2454.18 366.87 1.000e+00 160432847 44.7489 7.228e-11 ***
C57BL/6 : dose1   964.83 361.30 1.000e+00  25566551  7.1312  0.007872 **
    CBY : dose1 -2009.13 336.71 1.000e+00 127648750 35.6046 5.168e-09 ***
Residuals                417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

I am uncertain why `testInteractions` creates tables here with so many decimal places for df (and the exponential notation) It would be possible to adjust that with use of the `gt` tabling function since the output of the `testInteractions function is a data frame and`gt‘ works on data frames or tibbles. E.g, ….

```
cubicsme <- testInteractions(fitbase.aov, custom=dosecubic, fixed="strain", adjustment="none"
gt::gt(cubicsme)
```

Value	SE	Df	Sum of Sq	F	Pr(>F)
-2454.1820	366.8728	1	160432847	44.748909	7.227532e-11
964.8289	361.3014	1	25566551	7.131179	7.871724e-03
-2009.1346	336.7101	1	127648750	35.604569	5.167675e-09
NA	417.0000	1495019620	NA	NA	NA

Once again, note the incorrect placement of the df and SS for the residual term.

We see that both linear, quadratic, and cubic components of the dose function are all significant in each of the three strains. It remains to test which of those trend components interact with the contrast set on strain. Next section.

6.2.4 Interaction Contrasts using Trend Analysis in phia

Using the named contrasts created above, we can extract 2-way interaction contrasts by passing the relevant contrasts to the `custom` argument in `testInteractions`. But this is tedious since we have to test one contrast at a time. The values match what was seen above in the output from `summary.lm` and **emmeans** (squaring those t's to match the F's here)

Note that it doesn't matter which order the custom contrasts are named since the list objects carry the factor name.

```
testInteractions(fitbase.aov, custom=c(doselin,strainc1), adjustment="none")
```

```
F Test:
P-value adjustment method: none
                  Value     SE        Df Sum of Sq     F    Pr(>F)
dose1 : strain1 -9810.4 876.07 1.000e+00 449581044 125.4 < 2.2e-16 ***
Residuals                417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Next are the remaining three from the set of listed contrasts that I created above. The user can generalize to produce the contrasts involving cubic and quartic trend components.

```
testInteractions(fitbase.aov, custom=c(dosequad,strainc1), adjustment="none")
```

```
F Test:
P-value adjustment method: none
                 Value     SE        Df Sum of Sq      F    Pr(>F)
dose1 : strain1 5004.1 873.78 1.000e+00 117589625 32.799 1.959e-08 ***
Residuals               417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
testInteractions(fitbase.aov, custom=c(doselin,strainc2), adjustment="none")
```

```
F Test:
P-value adjustment method: none
                 Value     SE        Df Sum of Sq      F Pr(>F)
dose1 : strain1 121.52 497.28 1.000e+00    214099 0.0597 0.8071
Residuals               417.00 1.495e+09
```

```
testInteractions(fitbase.aov, custom=c(dosequad,strainc2), adjustment="none")
```

```
F Test:
P-value adjustment method: none
                  Value     SE        Df Sum of Sq      F   Pr(>F)
dose1 : strain1 -1302.1 495.14 1.000e+00  24795292 6.9161 0.008859 **
Residuals               417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The conclusion is that it is much easier just to use `summary.lm` on the `aov` fit object after the orthogonal contrasts are assigned to the factors as we did above.

6.2.5 Interaction comparisons with phia

At times, it is useful to conceptualize and analyze interaction comparisons, rather than, or in addition to interaction contrasts. These effects involve contrasts on one factor but not the other. Here I looked at Doselinear by Strain and Dosequadratic by Strain. Strain is an intact 2 df source here so the interactions have 2 df. The interpretation is that both the linear and quadratic shapes vary across levels of strain. The interaction contrasts can then be seen as the decomposition of these interaction comparisons into more specific patterns across strain. I didn't do cubic, just to save space here.

```
testInteractions(fitbase.aov, custom=doselin, across="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
          strain1 strain2    SE1       SE2 Df Sum of Sq      F    Pr(>F)
dose1     -9810.4  121.52 876.07 4.970e+02  2 449781544 62.728 < 2.2e-16 ***
Residuals                417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
testInteractions(fitbase.aov, custom=dosequad, across="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
          strain1 strain2    SE1       SE2 Df Sum of Sq      F    Pr(>F)
dose1      5004.1 -1302.1 873.78 4.950e+02  2 137535766 19.181 1.074e-08 ***
```

```
Residuals                  417.00 1.495e+09
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Additional discussion of interaction comparisons is in the **emmeans** section above.

6.3 Comparison of phia and emmeans

Both have their strengths - for `testInteractions` we have SS provided in the tables (even though some terms were misplaced in the tables). With **emmeans** we can more efficiently obtain each of the contrasts as part of a set of things simultaneously evaluated. In `testInteractions` each contrast has to be requested individually.

Manually creating contrasts is a bit of a pain in each with the listing necessity in **phia** and the custom contrast function in **emmeans**.

But once the learning curve is mastered, both packages provide very strong tools. I still conclude that they are much less efficient than SPSS MANOVA for these types of analyses.

7 Summary

The design has been fully evaluated using a framework of orthogonal contrasts for each factor. Additional evaluation of orthogonal polynomials has been the primary goal of this document in the context of a factorial design where the non-trend factor also has more than two levels. Some attention to post hoc multiple comparison tests was also provided.

The most difficult contrast to interpret is an interaction contrast when contrasts are on both factors (contrasts of contrasts). Experience with this is essential and it is probably easier to develop those skills when one factor has trend analysis associated with it.

Considerable effort went in to structuring the **phia** and **emmeans** approaches to interaction contrasts here. But truthfully, the easiest and most efficient way of obtaining the full set of them, within the orthogonal set context, is the use of `summary.lm` on an `aov` fit of the omnibus model as shown in an initial section. But classes of simple effects can only be obtained with **emmeans** and **phia** (and possibly `glht`).

8 Reproducibility

```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26100)

Matrix products: default


locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] afex_1.3-1         lme4_1.1-35.5      Matrix_1.7-0       sjstats_0.19.0
 [5] Rmisc_1.5.1        plyr_1.8.9         lattice_0.22-6     ggthemes_5.1.0
 [9] ggplot2_3.5.1      phia_0.3-1         car_3.1-2          carData_3.0-5
[13] knitr_1.48         emmeans_1.11.0-001 psych_2.4.6.26     gt_0.11.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.5        xfun_0.46           insight_0.20.2
 [4] numDeriv_2016.8-1.1 vctrs_0.6.5         tools_4.4.2
 [7] generics_0.1.3      parallel_4.4.2      datawizard_0.12.2
[10] sandwich_3.1-0      tibble_3.2.1        fansi_1.0.6
[13] pkgconfig_2.0.3     lifecycle_1.0.4     farver_2.1.2
[16] compiler_4.4.2      stringr_1.5.1       munsell_0.5.1
[19] mnormt_2.1.1        codetools_0.2-20    lmerTest_3.1-3
[22] htmltools_0.5.8.1   yaml_2.3.10         nloptr_2.1.1
[25] pillar_1.9.0        MASS_7.3-61         boot_1.3-30
[28] abind_1.4-5         multcomp_1.4-26     nlme_3.1-165
```

```
[31] tidyselect_1.2.1    digest_0.6.36       performance_0.12.2
[34] mvtnorm_1.2-5       stringi_1.8.4       reshape2_1.4.4
[37] dplyr_1.1.4         purrr_1.0.2         labeling_0.4.3
[40] splines_4.4.2       fastmap_1.2.0       grid_4.4.2
[43] colorspace_2.1-1    cli_3.6.3           magrittr_2.0.3
[46] survival_3.7-0      utf8_1.2.4          TH.data_1.1-2
[49] withr_3.0.1         scales_1.3.0        estimability_1.5.1
[52] rmarkdown_2.27      zoo_1.8-12          coda_0.19-4.1
[55] evaluate_0.24.0     rlang_1.1.4         Rcpp_1.0.13
[58] xtable_1.8-4        glue_1.7.0          xml2_1.3.6
[61] minqa_1.2.7         rstudioapi_0.16.0   jsonlite_1.8.8
[64] R6_2.5.1
```