A 2x7 Factorial Dose Response Study

Orthogonal Polynomial Trend Analysis

Bruce Dudek

2025-04-06

Contents

1 Introduction

Mice were treated with one of seven doses of cocaine. Cocaine, as a stimulant drug, is expected to produce increases in activity levels. Mice were measured in an automated activity monitoring apparatus for 30 min. The primary outcome variable examined here is simply total distance traveled.

Two closely related mouse strains were tested. The working hypothesis is that any differences observed in cocaine response between the two strains might be due to a very small number of mutations differentiating the two strains in the short amount of generational time that their lineages were separated.

This template document demonstrates only a few of the 2-way ANOVA basics about executing trend analysis in a 2 factor design.

Several Packages are used:

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

2 Import and Manage Data

Read the data:

```
data1 <- read.csv("mousecoc_small.csv", stringsAsFactors=TRUE)
gt::gt(psych::headTail(data1))
```

snum	cdist30	dose	strain
1	3288	control(0)	B6J
2	4076	5mg/kg	B6J
3	24130	5mg/kg	BR
4	39543	15mg/kg	BR
...	...	NA	NA

294	13490	15mg/kg	B6J
295	7961	control(0)	BR
296	33977	30mg/kg	BR
297	8892	control(0)	BR

First, I will change the levels of the dose variable to the proper order, not alphabetical (initially examine the levels in the order they exist after the data frame is first created.) This is important both for graphing purposes and for the use of orthogonal polynomial contrasts.

```
levels(data1$dose)
```

```
[1] "10mg/kg"    "15mg/kg"    "20mg/kg"    "25mg/kg"    "30mg/kg"
[6] "5mg/kg"     "control(0)"
```

```
data1$dose<- ordered(data1$dose,
    levels=c("control(0)","5mg/kg","10mg/kg","15mg/kg","20mg/kg",
             "25mg/kg","30mg/kg"))
levels(data1$dose)
```

```
[1] "control(0)" "5mg/kg"     "10mg/kg"    "15mg/kg"    "20mg/kg"
[6] "25mg/kg"    "30mg/kg"
```

In order to properly draw a line graph, a new variable must be created for dose, since an X axis variable on a plot needs to be numeric to be properly spaced.

```
data1[which(data1$dose == "control(0)"),"cocainedose"] <- 0
data1[which(data1$dose == "5mg/kg"),"cocainedose"] <- 5
data1[which(data1$dose == "10mg/kg"),"cocainedose"] <- 10
data1[which(data1$dose == "15mg/kg"),"cocainedose"] <- 15
data1[which(data1$dose == "20mg/kg"),"cocainedose"] <- 20
data1[which(data1$dose == "25mg/kg"),"cocainedose"] <- 25
data1[which(data1$dose == "30mg/kg"),"cocainedose"] <- 30
class(data1$cocainedose)
```

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

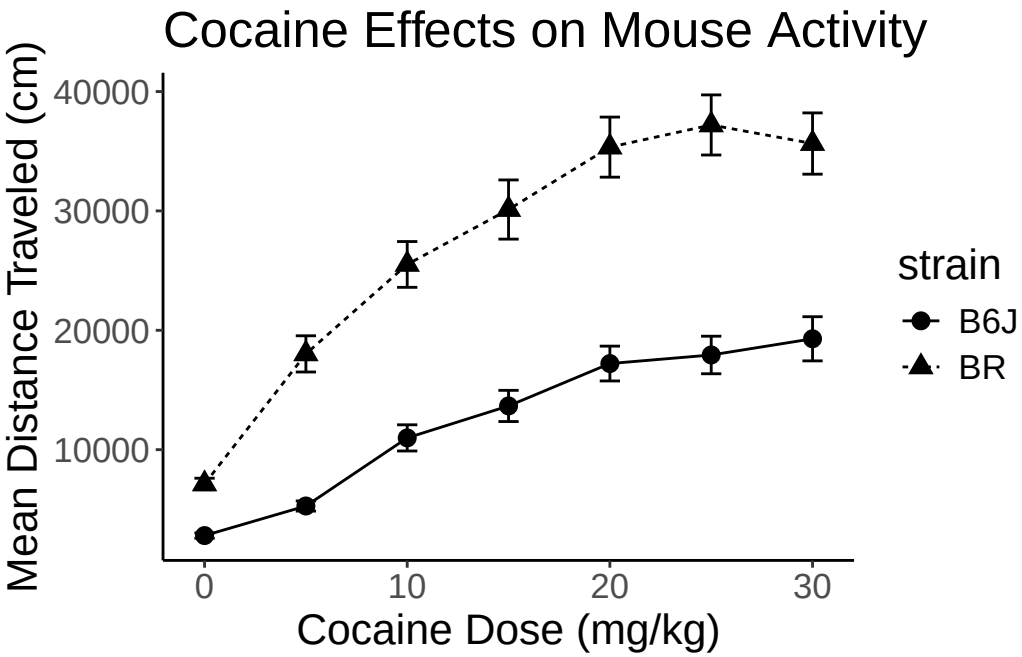
3 Graph of the dose response curves

A line graph drawn with `ggplot` requires a data frame that has the summary statistics for each group. `ggplot` won't work directly on the data frame for this kind of plot. So, first, create the summary data frame.

```
mouse_summ <- Rmisc::summarySE(data1,measurevar="cdist30", groupvars=c("cocainedose","strain"
#str(mouse_summ)
# rename the column that contains the mean to something less confusing
colnames(mouse_summ) <- c("cocainedose", "strain","N", "mean", "sd", "se", "95%ci" )
gt::gt(mouse_summ)
```

cocainedose	strain	N	mean	sd	se	95%ci
0	B6J	21	2803.762	1020.190	222.6238	464.3850
0	BR	17	7127.647	1984.033	481.1986	1020.0954
5	B6J	24	5280.750	2083.401	425.2724	879.7431
5	BR	18	18022.833	6436.944	1517.2022	3201.0168
10	B6J	24	10985.333	5375.119	1097.1915	2269.7136
10	BR	20	25515.450	8577.518	1917.9914	4014.4022
15	B6J	26	13662.269	6665.400	1307.1924	2692.2132
15	BR	18	30115.833	10513.587	2478.0762	5228.2838
20	B6J	25	17213.080	7292.433	1458.4865	3010.1682
20	BR	18	35342.222	10674.949	2516.1096	5308.5272
25	B6J	27	17931.370	8173.143	1572.9221	3233.1876
25	BR	18	37200.000	10678.226	2516.8821	5310.1571
30	B6J	24	19287.083	9059.427	1849.2479	3825.4607
30	BR	17	35642.765	10580.484	2566.1442	5439.9827

```
ggplot(mouse_summ, aes(x=cocainedose, 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=1)+
  xlab("Cocaine Dose (mg/kg)")+
  ylab("Mean Distance Traveled (cm)")+
  ggtitle("Cocaine Effects on Mouse Activity")+
  theme_classic()+
  theme(text = element_text(size=16))
```



4 Perform the Omnibus ANOVA

In factorial designs it becomes very important to understand 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	^5	^6
-0.5669	0.5455	-0.4082	0.2417	-0.1091	0.0329
-0.3780	0.0000	0.4082	-0.5641	0.4364	-0.1974
-0.1890	-0.3273	0.4082	0.0806	-0.5455	0.4935
0.0000	-0.4364	0.0000	0.4835	0.0000	-0.6580

0.1890	-0.3273	-0.4082	0.0806	0.5455	0.4935
0.3780	0.0000	-0.4082	-0.5641	-0.4364	-0.1974
0.5669	0.5455	0.4082	0.2417	0.1091	0.0329

The change to orthogonal contrasts for dose is ok, in the context of producing the omnibus F table, but there is an issue if the second factor is left at its default dummy (treatment coding). The Omnibus model would be flawed. So the strain variable should also be switched to an analytical contrast in order to produce an overall design matrix that produces the correct results.

```
#look at default contrast for strain
contrasts(data1$strain)
```

```
    BR
B6J  0
BR   1
```

```
#change coding for strain to an analytical contrast
straincontrast <- cbind(strain=c(1,-1))
contrasts(data1$strain) <- straincontrast
gt::gt(round(as.data.frame(contrasts(data1$strain)),1))
```

strain
1
-1

Since the sample sizes were unequal, Type III SS is requested for testing the omnibus effects. We find that the interaction results in rejection of the null hypothesis and therefore understand the need for examination of simple effects, and contrasts.

```
fit1.aov <- aov(cdist30~dose*strain, data=data1)
car::Anova(fit1.aov, type="III")
```

```
Anova Table (Type III tests)

Response: cdist30
                Sum Sq  Df   F value    Pr(>F)
(Intercept) 1.1246e+11   1 1927.9867 < 2.2e-16 ***
```

```
dose        1.7986e+10   6   51.3908 < 2.2e-16 ***
strain      1.5286e+10   1  262.0581 < 2.2e-16 ***
dose:strain 1.4597e+09   6    4.1706 0.0004908 ***
Residuals   1.6508e+10 283
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Effect sizes on the omnibus effect are obtained here:

```
# the gt function permits nicer formatting of the table
# anova_stats(fit_base.aov)
gt(anova_stats(fit1.aov)[1:3,1:6],rownames_to_stub = T)
```

	etasq	partial.etasq	omegasq	partial.omegasq	epsilonsq	cohens.f
dose	0.325	0.496	0.318	0.478	0.318	0.992
strain	0.314	0.487	0.313	0.474	0.313	0.975
dose:strain	0.029	0.081	0.022	0.060	0.022	0.297

Another way to perform the ANOVA is with AFEX which provides the ges effect size. Note that `aov_car` changes contrasts to effect coding but this is not permanent for the two factors - it is just internal to the `aov_car` modeling.

```
fit_base.afex <-  aov_car(cdist30~dose*strain + Error(1|snum), type=3, observed="strain", da
```

```
Contrasts set to contr.sum for the following variables: dose, strain
```

```
gt::gt(nice(fit_base.afex))
```

Effect	df	MSE	F	ges	p.value
dose	6, 283	58331494.02	51.39 ***	.351	<.001
strain	1, 283	58331494.02	262.06 ***	.460	<.001
dose:strain	6, 283	58331494.02	4.17 ***	.044	<.001

5 Trend analysis for omnibus effects

First, use `aov` to refit the model employing the trend coefficient vectors, which are set for the "dose" factor here.

Although the contrasts for dose and strain were set to our desired orthogonal contrasts, it is useful to repeat the process here to ensure that they are in place (since other analyses were done in between). Since strain is only a two-level factor it could be set to `contr.sum`, but I manually created an orthogonal contrast (which would be equivalent to `contr.sum` for this 2-level factor). This is an important conceptual component of the analysis that would then generalize to designs where the second independent variable has more than two levels.

```
# reset dose to orth polynomial coding
contrasts(data1$dose) <- contr.poly(7)
gt::gt(round(as.data.frame(contrasts(data1$dose)),4))
```

.L	.Q	.C	^4	^5	^6
-0.5669	0.5455	-0.4082	0.2417	-0.1091	0.0329
-0.3780	0.0000	0.4082	-0.5641	0.4364	-0.1974
-0.1890	-0.3273	0.4082	0.0806	-0.5455	0.4935
0.0000	-0.4364	0.0000	0.4835	0.0000	-0.6580
0.1890	-0.3273	-0.4082	0.0806	0.5455	0.4935
0.3780	0.0000	-0.4082	-0.5641	-0.4364	-0.1974
0.5669	0.5455	0.4082	0.2417	0.1091	0.0329

```
#reset strain to analytical contrast coding
straincontrast <- cbind(strain=c(1,-1))
contrasts(data1$strain) <- straincontrast
gt::gt(round(as.data.frame(contrasts(data1$strain)),1))
```

strain
1
-1

The same omnibus outcome is repeated here for the sake of completeness in this section.

```
trend1.aov  <- aov(cdist30~dose*strain, data=data1)
car::Anova(trend1.aov,type=3)
```

```
Anova Table (Type III tests)

Response: cdist30
                Sum Sq  Df   F value    Pr(>F)
(Intercept) 1.1246e+11   1 1927.9867 < 2.2e-16 ***
dose        1.7986e+10   6   51.3908 < 2.2e-16 ***
strain      1.5286e+10   1  262.0581 < 2.2e-16 ***
dose:strain 1.4597e+09   6    4.1706 0.0004908 ***
Residuals   1.6508e+10 283
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can't use the "split" argument in `anova` since that would give us Type I SS. So, we use summary.lm to obtain the tests of the individual vectors. Main effect contrasts on dose are provided, as well as the six interaction contrast tests. Since strain is only a two-level factor the test of its regression coefficient here is the same as for the omnibus model main effect

```
trendtests <- summary.lm(trend1.aov)
trendtests
```

```
Call:
aov(formula = cdist30 ~ dose * strain, data = data1)

Residuals:
     Min       1Q   Median       3Q      Max
-29216.0  -2479.6    345.2   4407.7  17448.2

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         19723.6      449.2  43.909  < 2e-16 ***
dose.L              20287.8     1209.5  16.774  < 2e-16 ***
dose.Q              -6436.0     1204.9  -5.342 1.90e-07 ***
dose.C               -588.7     1191.4  -0.494  0.62161
dose^4               -110.2     1181.2  -0.093  0.92571
dose^5               -111.3     1172.9  -0.095  0.92447
dose^6                897.0     1170.2   0.766  0.44403
strainstrain        -7271.7      449.2 -16.188  < 2e-16 ***
dose.L:strainstrain -4984.2     1209.5  -4.121 4.96e-05 ***
dose.Q:strainstrain  3294.8     1204.9   2.734  0.00664 **
dose.C:strainstrain  -389.1     1191.4  -0.327  0.74421
dose^4:strainstrain  1235.2     1181.2   1.046  0.29662
```

```
dose^5:strainstrain   -213.9     1172.9  -0.182  0.85543
dose^6:strainstrain    173.8     1170.2   0.149  0.88203
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7638 on 283 degrees of freedom
Multiple R-squared:  0.6691,    Adjusted R-squared:  0.6539
F-statistic: 44.01 on 13 and 283 DF,  p-value: < 2.2e-16
```

But to make certain that the results are as expected, I have converted the t values to F values. I need to extract the t values and square them. The resulting F's match what we have seen before in SPSS MANOVA and below with `emmeans`.

```
str(trendtests$coefficients)
```

```
 num [1:14, 1:4] 19724 20288 -6436 -589 -110 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:14] "(Intercept)" "dose.L" "dose.Q" "dose.C" ...
  ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
```

```
teststats <- as.data.frame(trendtests$coefficients[2:14,3])
colnames(teststats) <- "tvalues"
teststats$Fvalues <- round(teststats$tvalues**2,4)
teststats$tvalues <- round(teststats$tvalues,4)
teststats
```

```
                   tvalues  Fvalues
dose.L             16.7736 281.3522
dose.Q             -5.3415  28.5319
dose.C             -0.4941   0.2442
dose^4             -0.0933   0.0087
dose^5             -0.0949   0.0090
dose^6              0.7665   0.5875
strainstrain      -16.1882 262.0581
dose.L:strainstrain -4.1208  16.9812
dose.Q:strainstrain  2.7345   7.4773
dose.C:strainstrain -0.3266   0.1067
dose^4:strainstrain  1.0456   1.0934
dose^5:strainstrain -0.1824   0.0333
dose^6:strainstrain  0.1485   0.0221
```

6 Follow up with simple effects and contrasts using emmeans

Recall that **emmeans** provides capability for obtaining simple effects and contrasts on all three multiple df effects: Main effect contrasts, Simple Main effect contrasts, and interaction contrasts.

In the data set used here, the omnibus interaction was present, so we need to examine interaction contrasts, simple main effects, and simple main effect contrasts. But I will also show how to obtain main effect contrasts as part of the template motivation for this document.

6.1 Interaction Contrasts with emmeans

This first section begins with the 2-way dose by strain interaction contrasts, duplicating the outcome found with `summary.lm` above.

Initially, extract the grid of means from the fit object. `emmeans` does not require that the fit object was created with orthogonal contrasts, but I've use the orthogonal contrast fit object here.

```
factorial.emm <- emmeans::emmeans(trend1.aov, ~dose*strain)
factorial.emm
```

```
 dose       strain emmean   SE  df lower.CL upper.CL
 control(0) B6J      2804 1670 283     -477     6084
 5mg/kg     B6J      5281 1560 283     2212     8349
 10mg/kg    B6J     10985 1560 283     7917    14054
 15mg/kg    B6J     13662 1500 283    10714    16611
 20mg/kg    B6J     17213 1530 283    14206    20220
 25mg/kg    B6J     17931 1470 283    15038    20825
 30mg/kg    B6J     19287 1560 283    16218    22356
 control(0) BR       7128 1850 283     3481    10774
 5mg/kg     BR      18023 1800 283    14479    21566
 10mg/kg    BR      25515 1710 283    22154    28877
 15mg/kg    BR      30116 1800 283    26572    33659
 20mg/kg    BR      35342 1800 283    31799    38886
 25mg/kg    BR      37200 1800 283    33657    40743
 30mg/kg    BR      35643 1850 283    31997    39289

Confidence level used: 0.95
```

I've obtained the 2-way interaction contrasts in a different way that was demonstrated in the larger 2-way factorial tutorial document using **emmeans**. This is a bit simpler.

Caution: With this approach, implementing the "poly" argument assumes equal interval spacing between the IV levels. If the spacing is not equal, then building a matrix of coefficients and passing them would be required as shown in the larger 2-way factorial tutorial document. (Addendum: R. Lenth has added another function in emmeans that provides coefficients when the spacing of the quantitative IV levels is unequal. See the 3x5 tutorial document for explanation.)

Use of the "consec" argument here is probably initially unclear. "Consec" does pairwise comparisons of successive pairs of groups. But with only two groups there is only one comparison: one strain vs the other. But other contrasts could be used for designs when the second IV has more than two levels. Each of these interaction contrasts can be thought of as Dose_trend by Strain. The "consec" argument was a bit of a kludge to get ther.

```
contrast(factorial.emm, interaction=c(dose="poly",strain="consec"))
```

```
 dose_poly strain_consec estimate    SE  df t.ratio p.value
 linear    BR - B6J         52748 12800 283   4.121  <.0001
 quadratic BR - B6J        -60394 22100 283  -2.734  0.0066
 cubic     BR - B6J          1906  5840 283   0.327  0.7442
 quartic   BR - B6J        -30656 29300 283  -1.046  0.2966
 degree 5  BR - B6J          3921 21500 283   0.182  0.8554
 degree 6  BR - B6J        -10567 71100 283  -0.149  0.8820
```

6.2 Main effect contrasts emmeans

Only the dose factor has more than one df, so it is the only one of the two main effects that could be decomposed into contrasts.

First, extract the grid of marginal means to be used in the contrasts. Note that these are the unweighted marginal means.

```
dose.emm <- emmeans::emmeans(trend1.aov, ~dose)
```

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

```
dose.emm
```

```
 dose       emmean   SE  df lower.CL upper.CL
 control(0)   4966 1250 283     2513     7418
 5mg/kg      11652 1190 283     9308    13996
 10mg/kg     18250 1160 283    15975    20526
```

```
 15mg/kg     21889 1170 283    19584    24194
 20mg/kg     26278 1180 283    23954    28601
 25mg/kg     27566 1160 283    25278    29853
 30mg/kg     27465 1210 283    25082    29848

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

The `contrast` function provides an easy way to simply request orthogonal polynomials on the dose factor.

Caution: I believe that with this approach, implementing the "poly" argument assumes equal interval spacing between the IV levels. If the spacing is not equal, then building a matrix of coefficients and passing them would be required as shown in the larger 2-way factorial tutorial document.

```
emmeans::contrast(dose.emm, "poly")
```

```
 contrast  estimate    SE  df t.ratio p.value
 linear      107353  6400 283  16.774  <.0001
 quadratic   -58987 11000 283  -5.342  <.0001
 cubic        -1442  2920 283  -0.494  0.6216
 quartic      -1368 14700 283  -0.093  0.9257
 degree 5     -1020 10700 283  -0.095  0.9245
 degree 6     27265 35600 283   0.766  0.4440

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

6.3 Simple Main Effects of Dose emmeans

Dose effects at levels of strain can be found by creating the `emmeans` object using the `by="strain"` argument. Dose is the factor of interest, and the grid of descriptive stats gives the means for dose separated into two tables, one for each strain.

```
d.s.emm <- emmeans::emmeans(trend1.aov, "dose", by="strain")
d.s.emm
```

```
strain = B6J:
 dose       emmean   SE  df lower.CL upper.CL
 control(0)   2804 1670 283     -477     6084
 5mg/kg       5281 1560 283     2212     8349
```

```
 10mg/kg     10985 1560 283     7917    14054
 15mg/kg     13662 1500 283    10714    16611
 20mg/kg     17213 1530 283    14206    20220
 25mg/kg     17931 1470 283    15038    20825
 30mg/kg     19287 1560 283    16218    22356

strain = BR:
 dose       emmean   SE  df lower.CL upper.CL
 control(0)   7128 1850 283     3481    10774
 5mg/kg      18023 1800 283    14479    21566
 10mg/kg     25515 1710 283    22154    28877
 15mg/kg     30116 1800 283    26572    33659
 20mg/kg     35342 1800 283    31799    38886
 25mg/kg     37200 1800 283    33657    40743
 30mg/kg     35643 1850 283    31997    39289

Confidence level used: 0.95
```

With a non-obvious syntactical structure we then obtain tests of the two simple main effects. This combined use of the `test` and `contrast` functions produce the expected 6df simple main effects of dose at each of the two strains, and the denominator df indicate that the Omnibus MSwg error term is being used. The "d" note can be ignored as it is generated with the use of the "eff" argument in a manner that won't be addressed here. The "joint" argument essentially means that the underlying contrasts for each effect are being jointly combined to obtain the more "omnibus" 6df simple main effects.

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

```
 strain df1 df2 F.ratio p.value note
 B6J      6 283  16.461  <.0001  d
 BR       6 283  36.509  <.0001  d

d: df1 reduced due to linear dependence
```

6.4 Simple Main Effect Contrasts emmeans

Now, we can break down the SME of dose into their trend components, separately at each level of strain

The same grid that we used to obtain the SME can be passed to the **emmeans** `contrast` function with the "poly" specification. It is implied that the trend vectors will be applied to

the "dose" factor since that is how the grid is created. The squares of these t values match the F values we obtained in other approaches to this question in SPSS, and the omnibus MSwg error term was used for each, appropriately so (seen with the df specification for each t value)

```
contrast(d.s.emm, "poly")
```

```
strain = B6J:
 contrast  estimate    SE  df t.ratio p.value
 linear       80979  8370 283   9.679  <.0001
 quadratic   -28790 14500 283  -1.992  0.0474
 cubic        -2395  3820 283  -0.628  0.5308
 quartic      13960 18900 283   0.738  0.4608
 degree 5     -2980 14100 283  -0.212  0.8323
 degree 6     32549 46300 283   0.704  0.4822

strain = BR:
 contrast  estimate    SE  df t.ratio p.value
 linear      133726  9690 283  13.804  <.0001
 quadratic   -89184 16700 283  -5.341  <.0001
 cubic          -489  4420 283  -0.111  0.9119
 quartic     -16696 22400 283  -0.745  0.4569
 degree 5       940 16300 283   0.058  0.9539
 degree 6     21982 54100 283   0.407  0.6846
```

See: https://cran.r-project.org/web/packages/emmeans/vignettes/interactions.html#contrasts for partial help on this coding approach.

7 Using phia for trend analysis in a 2x7 factorial

Some of the same effects extracted from use of **emmeans** will be obtained here, using **phia**.

Advantages:

- For many things, I find **phia** a bit easier to use
- It provides SS for effects so that effect sizes for contrasts might be manually computed.
- For Orthogonal Polynomials, it is simple to use coefficients that are appropriate for unequally spaced intervals of the quantitative independent variable - the **emmeans** approach to that is slightly more involved since it does not use `contr.poly`. See the 3x5 trend tutorial document for illustration of how to handle a quantitative IV with unequal spacing of factor levels.

Disadvantages:

- The way of handling contrasts can be a challenge when there are many of them. The `testInteractions` function requires work on one contrast at a time and this slows us down if there are many contrasts.

7.1 Interaction Contrasts with phia

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.

```
dosetrend <- contr.poly(7)
dosetrend
```

```
                .L            .Q            .C         ^4            ^5
[1,] -5.669467e-01  5.455447e-01 -4.082483e-01  0.2417469 -1.091089e-01
[2,] -3.779645e-01  9.690821e-17  4.082483e-01 -0.5640761  4.364358e-01
[3,] -1.889822e-01 -3.273268e-01  4.082483e-01  0.0805823 -5.455447e-01
[4,]  2.098124e-17 -4.364358e-01  4.532467e-17  0.4834938  5.342065e-16
[5,]  1.889822e-01 -3.273268e-01 -4.082483e-01  0.0805823  5.455447e-01
[6,]  3.779645e-01  0.000000e+00 -4.082483e-01 -0.5640761 -4.364358e-01
[7,]  5.669467e-01  5.455447e-01  4.082483e-01  0.2417469  1.091089e-01
              ^6
[1,]  0.03289758
[2,] -0.19738551
[3,]  0.49346377
[4,] -0.65795169
[5,]  0.49346377
[6,] -0.19738551
[7,]  0.03289758
```

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

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

```
$dose
[1] -5.669467e-01 -3.779645e-01 -1.889822e-01  2.098124e-17  1.889822e-01
[6]  3.779645e-01  5.669467e-01
```

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

```
$dose
[1]  5.455447e-01  9.690821e-17 -3.273268e-01 -4.364358e-01 -3.273268e-01
[6]  0.000000e+00  5.455447e-01
```

Now, we can pass those two contrasts for dose to the `testInteractions` function and create the interaction with strain by using the `across` argument.

This F test matches what we have seen above for the dose-linear by strain interaction contrast.

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 283 throughout and the SS is 1.6508e+10. A 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.

```
testInteractions(fit1.aov, custom=doselin, across="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
            Value   SE         Df Sum of Sq      F    Pr(>F)
dose1     -9968.3 2419 1.0000e+00 990538222 16.981 4.963e-05 ***
Residuals          283 1.6508e+10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Analogously for the quadratic component - dose-quadratic by strain:

```
testInteractions(fit1.aov, custom=dosequad, across="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
           Value     SE         Df Sum of Sq      F   Pr(>F)
dose1     6589.5 2409.8 1.0000e+00 436163702 7.4773 0.006642 **
Residuals         283.0 1.6508e+10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.2 Simple Main Effects with phia

This approach provides an unwieldy table. It first gives all of the parameters/estimates of the contrasts involved in the simple main effects before finally getting to the two 6-df simple main effect F values and p values. These SME tests match what we found above with `emmeans` and with SPSS.

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

```
gt::gt(testInteractions(fit1.aov, across="dose", fixed="strain", adjustment="none"))
```

dose1	dose2	dose3	dose4	dose5	dose6	SE1	SE2	
-16483.32	-14006.33	-8301.75	-5624.814	-2074.0033	-1355.713	2282.141	2.204758e+03	220
-28515.12	-17619.93	-10127.31	-5526.931	-300.5425	1557.235	2619.643	2.583003e+03	251
NA	NA	NA	NA	NA	NA	283.000	1.650781e+10	

7.3 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 strain B6 and BR.

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

```
testInteractions(fit1.aov, custom=doselin, fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
            Value     SE         Df  Sum of Sq       F    Pr(>F)
B6J : dose1 15304 1581.2 1.0000e+00 5.4644e+09  93.679 < 2.2e-16 ***
 BR : dose1 25272 1830.7 1.0000e+00 1.1115e+10 190.556 < 2.2e-16 ***
Residuals          283.0 1.6508e+10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The dose-quadratic at those two levels of strain.

```
testInteractions(fit1.aov, custom=dosequad, fixed="strain", adjustment="none")
```

```
F Test:
P-value adjustment method: none
              Value     SE         Df  Sum of Sq       F    Pr(>F)
B6J : dose1 -3141.3 1577.3 1.0000e+00  231364401  3.9664   0.04738 *
 BR : dose1 -9730.8 1821.9 1.0000e+00 1663960050 28.5259 1.902e-07 ***
Residuals            283.0 1.6508e+10
---
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, ….

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

```
# retesting the quadratic simple main effects
lin1 <- testInteractions(fit1.aov, custom=doselin, fixed="strain", adjustment="none")
str(lin1)
```

```
Classes 'anova' and 'data.frame':   3 obs. of  6 variables:
 $ Value    : num  15304 25272 NA
 $ SE       : num  1581 1831 283
 $ Df       : num  1.00 1.00 1.65e+10
 $ Sum of Sq: num  5.46e+09 1.11e+10 NA
 $ F        : num  93.7 190.6 NA
 $ Pr(>F)   : num  2.52e-19 1.71e-33 NA
 - attr(*, "heading")= chr "F Test: \nP-value adjustment method: none"
```

```
gt::gt(lin1)
```

Value	SE	Df	Sum of Sq	F	Pr(>F)
15303.58	1581.149	1	5464417912	93.67869	2.518935e-19
25271.93	1830.740	1	11115430235	190.55624	1.713196e-33
NA	283.000	16507812807	NA	NA	NA

7.4 Main Effect Contrasts with phia

This final analysis is possible by simply passing the custom dose contrast list and leaving out any `fixed` or `across` arguments.

Once again, linear first:

```
testInteractions(fit1.aov, custom=doselin, adjustment="none")
```

```
F Test:
P-value adjustment method: none
          Value     SE         Df  Sum of Sq      F    Pr(>F)
dose1     20288 1209.5 1.0000e+00 1.6412e+10 281.35 < 2.2e-16 ***
Residuals         283.0 1.6508e+10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And then quadratic:

```
testInteractions(fit1.aov, custom=dosequad, adjustment="none")
```

```
F Test:
P-value adjustment method: none
          Value     SE         Df  Sum of Sq      F    Pr(>F)
dose1     -6436 1204.9 1.0000e+00 1664310946 28.532 1.896e-07 ***
Residuals         283.0 1.6508e+10
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

8 Conclusions

It is reassuring to see the convergence of test values produced by `summary.lm` the various `emmeans` analyses, and the various `testInteractions` analyses. There is no obstacle in obtaining these full decompositons of the overall BG variation into omnibus, simple, and contrast effects. There are some idiosyncracies of each methodology to keep aware of

9 Caveat on comparison to SPSS

The student who assiduously compares values produced here to values found in our work with SPSS MANOVA may notice some slight differences in F values for the six interaction contrasts (and main effect contrasts) using the three different approaches here in R. The distinction is that for one major analysis to produce interaction contrasts in SPSS MANOVA, my example used the "singldf" approach to producing tests of contrasts. If that MANOVA work had not used "singldf" and had instead named all contrasts individually on the design statement, then the matching would have been perfect. The discrepancy is small, but something to be aware of. The SPSS "singldf" method uses a combination Type I and III SS, but everything with `summary.lm`, `emmeans` and `testInteractions and contrasts` here is based fully on Type III SS.

10 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] 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        bayestestR_0.14.0   xfun_0.46
 [4] insight_0.20.2      numDeriv_2016.8-1.1 vctrs_0.6.5
 [7] tools_4.4.2         generics_0.1.3      parallel_4.4.2
[10] datawizard_0.12.2   sandwich_3.1-0      tibble_3.2.1
[13] fansi_1.0.6         pkgconfig_2.0.3     lifecycle_1.0.4
[16] farver_2.1.2        compiler_4.4.2      stringr_1.5.1
[19] munsell_0.5.1       mnormt_2.1.1        codetools_0.2-20
[22] lmerTest_3.1-3      htmltools_0.5.8.1   yaml_2.3.10
[25] nloptr_2.1.1        pillar_1.9.0        MASS_7.3-61
[28] boot_1.3-30         abind_1.4-5         multcomp_1.4-26
[31] nlme_3.1-165        tidyselect_1.2.1    digest_0.6.36
[34] performance_0.12.2  mvtnorm_1.2-5       stringi_1.8.4
[37] reshape2_1.4.4      dplyr_1.1.4         purrr_1.0.2
[40] labeling_0.4.3      splines_4.4.2       fastmap_1.2.0
[43] grid_4.4.2          colorspace_2.1-1    cli_3.6.3
[46] magrittr_2.0.3      survival_3.7-0      utf8_1.2.4
[49] TH.data_1.1-2       withr_3.0.1         scales_1.3.0
[52] estimability_1.5.1  rmarkdown_2.27      zoo_1.8-12
[55] coda_0.19-4.1       evaluate_0.24.0     knitr_1.48
[58] parameters_0.22.1   rlang_1.1.4         Rcpp_1.0.13
[61] xtable_1.8-4        glue_1.7.0          xml2_1.3.6
[64] effectsize_0.8.9    minqa_1.2.7         rstudioapi_0.16.0
[67] jsonlite_1.8.8      R6_2.5.1
```