Two-level factorial designs

What are two level factorial designs?

- These are designs where the factors can only take on two levels
- First we will see complete, balanced, and unreplicated (i.e. $r = 1$) design

Nomenclature

- Two-level full factorial designs are widely used in industrial statistics, and they come with their own nomenclature
- We usually refer to these designs as "2^k designs", where k is the number of factors. The name comes from the fact that if we have, say, 3 factors with 2 levels each, there are 2^3 combinations of 3 factors.
- Since these designs are unreplicated, the sample size of a 2^k the experiment is 2^k
- The levels of the factors are usually encoded as -1 and 1; this is not important (they are just labels)

What's special about these designs?

- These designs have a few peculiarities; we won't cover them in detail here, but you will see them if you take the course on industrial statistics next year
- Recall that, since the design is unreplicated, we can't do F-tests if we include all the interaction terms
- However, in 2^k designs, there's a clever strategy that allows us to identify "important" effects

Daniel plot

- If all the effects were insignificant, they would all be independent draws from a normal distribution centered at zero with the same variance
- This property is a consequence of working with a complete, balanced, two-level design; it isn't true in general
- Given this property, we can do a qq-plot of effects with a normal; if most effects are lined up with the exception of a few effects that stand out, the ones that stand out are the important effects
- The name of this qq-plot of effects is Daniel plot

Example: Spring

Response: number of compressions until a spring breaks

Factors:

- Length: 10 or 15cm
- Girth: 5 or 7mm
- Steel: Type A or B

2^3 design: 8 runs and 3 factors

Example: Read in data

```r
# read in data
molla = read.csv("http://vicpena.github.io/doe/molla.csv")

# create design matrix with FrF2
library(FrF2)
design = FrF2(nruns = 8,  nfactors = 3,
       factor.names = c("Longitud", "Gruix", "Tipus"),
       randomize = FALSE)
# add in response variable
design = add.response(design, molla$Comp)
```

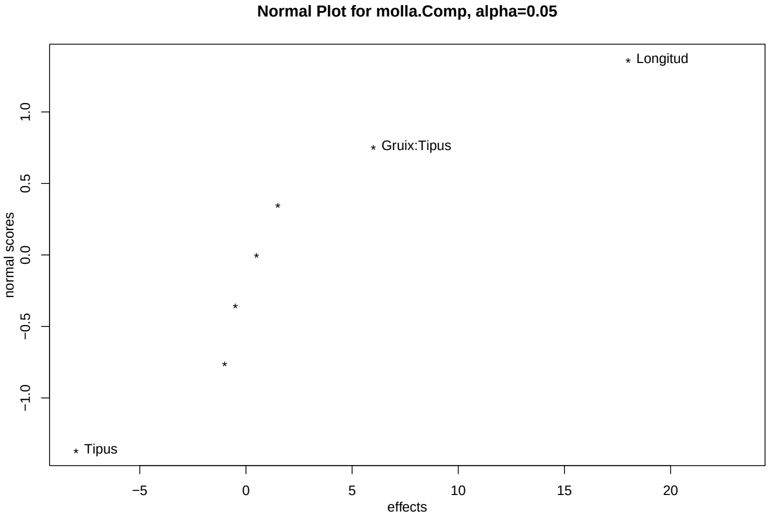
Example: ANOVA table

```r
# since r = 1, can't do any F-testing
mod = aov(molla.Comp ~ Longitud*Gruix*Tipus,
          data = design)
summary(mod)
```

```
##                      Df Sum Sq Mean Sq
## Longitud              1  648.0   648.0
## Gruix                 1    4.5     4.5
## Tipus                 1  128.0   128.0
## Longitud:Gruix        1    2.0     2.0
## Longitud:Tipus        1    0.5     0.5
## Gruix:Tipus           1   72.0    72.0
## Longitud:Gruix:Tipus  1    0.5     0.5
```

Example: Daniel plot

```
DanielPlot(design)
```



Example: Refitting model

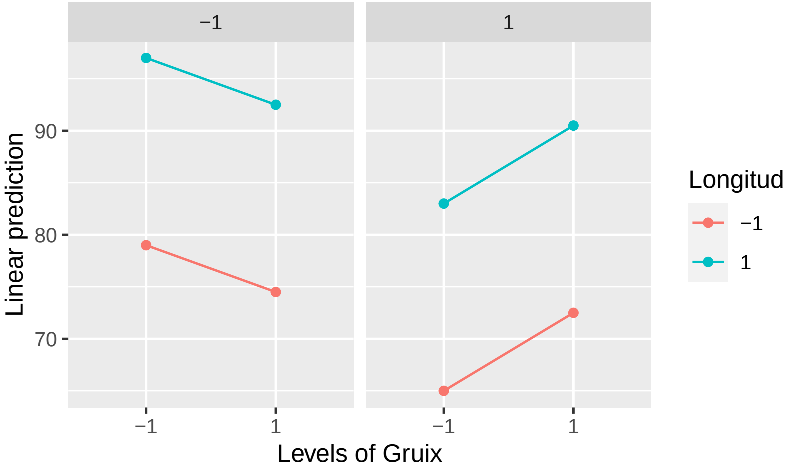
Refit model with “important” terms flagged by `DanielPlot`

```
mod2 = aov(molla.Comp ~ Longitud + Gruix*Tipus,
           data = design)
summary(mod2)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## Longitud      1  648.0   648.0   648.0 0.000133 ***
## Gruix         1    4.5     4.5     4.5 0.124027
## Tipus         1  128.0   128.0   128.0 0.001481 **
## Gruix:Tipus   1   72.0    72.0    72.0 0.003437 **
## Residuals     3    3.0     1.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
```

Example: `emmip`

```
library(emmeans)
emmip(mod2, Longitud ~ Gruix | Tipus)
```



Example: emmeans

```
library(emmeans)
emmeans(mod2, ~ Longitud + Gruix*Tipus)
```

```
##  Longitud Gruix Tipus emmean    SE df lower.CL upper.CL
##  -1       -1    -1      79.0 0.791  3     76.5     81.5
##  1        -1    -1      97.0 0.791  3     94.5     99.5
##  -1       1     -1      74.5 0.791  3     72.0     77.0
##  1        1     -1      92.5 0.791  3     90.0     95.0
##  -1       -1    1       65.0 0.791  3     62.5     67.5
##  1        -1    1       83.0 0.791  3     80.5     85.5
##  -1       1     1       72.5 0.791  3     70.0     75.0
##  1        1     1       90.5 0.791  3     88.0     93.0
##
## Confidence level used: 0.95
```

Fractional factorials

So far we have assumed that we can perform all 2^k experiments.

Unfortunately, we can't always afford to do that

In those circumstances, we can use 2^{k-p} designs

- k factors
- each factor has 2 levels
- we perform 2^{k-p} experiments

A price to pay

There is a price to pay for not observing all experimental conditions

- Some effects will be aliased (confounded)

However. . .

- We will be able to identify a subset of factors of models that seem important that can guide future experimentation

Example

- I go for a 5k run sometimes.
- Ignoring weather conditions and other factors, I want to test the effect of my shoes (A: Nike -1, Adidas 1) and whether I took coffee or not (B: No -1, Yes 1).
- If I try out all the combinations of shoes and coffee, it would be a full 2^2 factorial design.

Example

However, suppose that I only have 2 observations (a 2^{2-1} fractional design):

```
df = data.frame(A = c(-1, 1),
                B = c(-1, 1),
                time = c(18, 16))
df
```

```
##    A  B time
## 1 -1 -1   18
## 2  1  1   16
```

The effect of the shoes and the coffee is confounded; there's a 2 minute decrease, but it could be either to the change of shoes, the coffee, or both

Example

Now suppose that I had done these two runs instead

```r
df = data.frame(A = c(-1, 1),
                B = c(-1, -1),
                time = c(18, 16))
df
```

```
##    A  B time
## 1 -1 -1   18
## 2  1 -1   16
```

The effect of A and the interaction AB is confounded: there is a decrease of two minutes when I only changed shoes, but maybe that effect would've been different had I taken coffee. . .

Confounding. . .

In fractional designs, some effects will be impossible to disentangle

Fortunately, we can often use common sense to decide which of the confounded effects is most likely to be active

For example, if a two-way interaction is confounded with a third-order interaction, it seems safe to assume that the two-way interaction is the one that's active

Resolution of fractional designs

- Resolution III designs: Main effects are confounded with 2-way interactions
- Resolution IV designs: Main effects are confounded with 3-way interactions, 2-way interactions are confounded between them
- Resolution V designs: Main effects are confounded with four-way interactions and two-way interactions are confounded with three-way interactions

Resolution V designs are "almost complete", since we often ignore four-way interactions and, in practice, we almost always consider two-way interactions to be important before three-way interactions

Cookies: Cookies (Taback)

- Response: how good the taste is, on a scale from 0 to 10
- The factors are: butter (10g or 15g), sugar (0.5cup or 1cup), powder (1/2tsp or 1tsp), baking time (12min or 16min)
- 4 factors, but we can only run 8 experiments: 2^{4-3} design

Cookies: Creating design

Resolution IV: two-way interactions are confounded between them

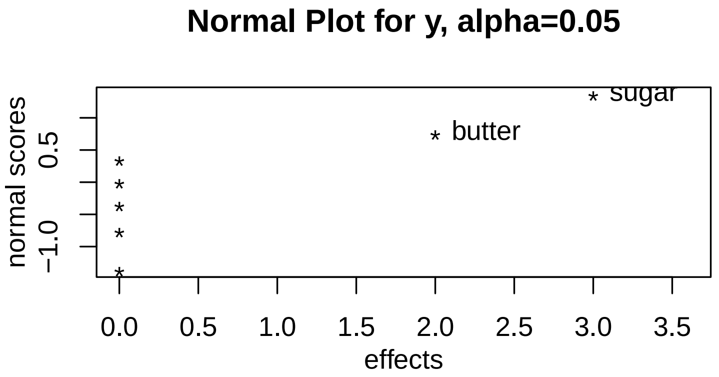
```
design = FrF2(nruns = 8, nfactors = 4,
         factor.names = c("butter", "sugar",
                          "powder", "baking"),
         randomize = FALSE)

aliasprint(design)
```

```
## $legend
## [1] A=butter B=sugar  C=powder D=baking
##
## $main
## character(0)
##
## $fi2
## [1] AB=CD AC=BD AD=BC
```

- Aliases: AB with CD, AC with BD, AD with BC

Cookies: add in response

```
y  =  c(2, 4, 5, 7, 2, 4, 5, 7)
design = add.response(design, y)
DanielPlot(design)
```



Cookies: re fit model

Fit model with only butter, sugar

```
mod = aov(y ~ sugar + butter, design)
summary(mod)
```

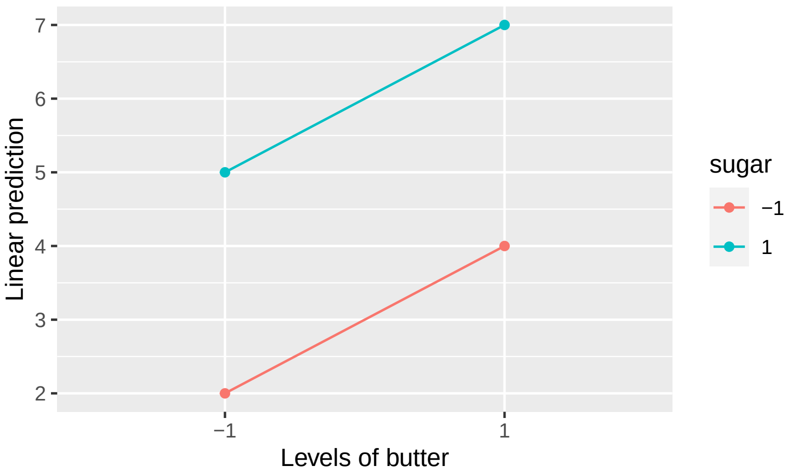
```
##             Df Sum Sq Mean Sq   F value Pr(>F)    
## sugar        1     18      18 1.678e+31 <2e-16 ***
## butter       1      8       8 7.456e+30 <2e-16 ***
## Residuals    5      0       0                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.
```

All positive and significant

Plotting effects

All positive effects in sugar, powder

```r
library(emmeans)
emmip(mod, sugar ~ butter)
```



Estimated effects

```
emmeans(mod,  ~ sugar + butter)
```

```
##  sugar butter emmean       SE df lower.CL upper.CL
##  -1    -1          2 6.34e-16  5        2        2
##  1     -1          5 6.34e-16  5        5        5
##  -1    1           4 6.34e-16  5        4        4
##  1     1           7 6.34e-16  5        7        7
##
## Confidence level used: 0.95
```

Cookies: Conclusions

- In the experiment, we only considered two levels of butter and sugar. For the high values of those variables, we expect our cookies to taste like a 7 out of 10
- If we were to keep on experimenting, it may be worth trying higher values of butter and sugar

Exhaust pipe (Wu)

7 factors can influence the diameter of an exhaust pipe

The company can only afford to run 8 experiments

They ran a 2^{7-3} design

```
design = FrF2(nruns = 8, nfactors = 7, randomize = FALSE)
y = c(34.6, 46.3, 48.6, 44.9, 49.7, 34.0, 46.5, 49.0)
design = add.response(design, y)
```

Exhaust pipe: Aliasing

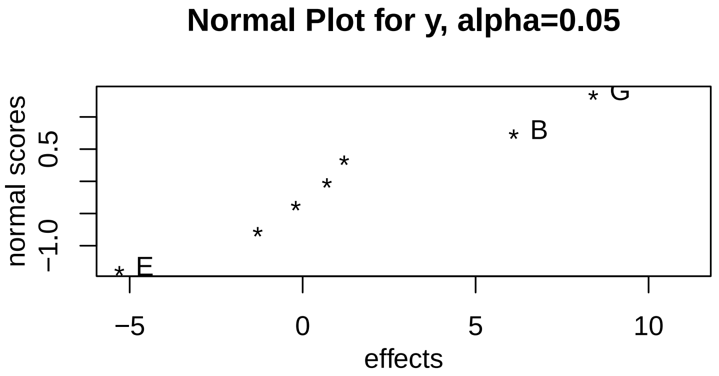
Main effects are aliased with interactions; for example A is confounded with BD, CE, FG:

```
aliasprint(design)
```

```
## $legend
## [1] A=A B=B C=C D=D E=E F=F G=G
##
## $main
## [1] A=BD=CE=FG B=AD=CF=EG C=AE=BF=DG D=AB=CG=EF E=AC=BG=
##
## $fi2
## character(0)
```

Exhaust pipe: Daniel Plot

```
DanielPlot(design)
```



Example: Back to aliasing

- B, E, and G marked as important
- B is aliased with AD, CF, **EG**
- E is aliased with AC, **BG**, DF
- G is aliased with AF, **BE**, CD

Example: What do we do?

- Which models seem likely to have generated the data?
- Candidates: B + E + G, B + E + BE, B + G + BG, E + G + BE
- Can't disentangle those!
- Next experiment? Do a full 2^3 experiment with B, E, and G
- Has this been useless? NO, we have identified 3 useful effects out of 7. A full 2^7 involves 128 experiments; a 2^{7-4} and a full 2^3 only involve 16 experiments