Chapter 9: Multi-factor experiments

Introductory Statistics for Engineering Experimentation

Peter R. Nelson, Marie Coffin and Karen A.F. Copeland

Slides by Douglas Bates

Outline

Outline

Outline

Chapter 9: Multi-factor experiments

- We are not limited to using two factors in an experiment. Especially in screening experiments, where we are seeking to distinguish the "important few" factors from the "trivial many", it is useful to incorporate many factors.
- Incorporating many factors, each with several levels, results in a large number of combinations of levels. If we want replicates the number of observations required is even larger.
- One way to reduce the number of observations required is to use a small number of levels. We often use just two levels.
- At two levels, there is no distinction between a numeric covariate, a categorical covariate or an ordered category. For technical reasons we often represent such a factor as an ordered category.

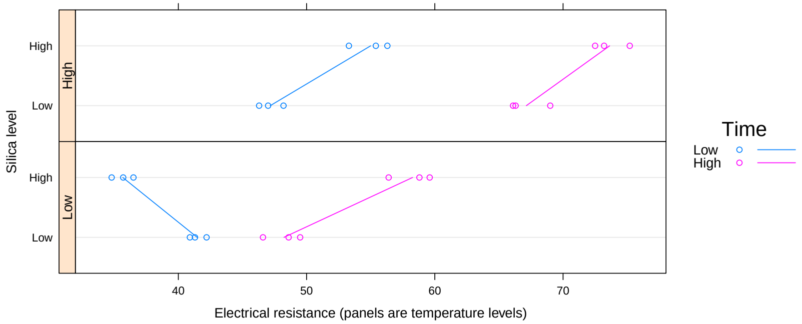
Section 9.1: ANOVA for multi-factor experiments

- For a replicated design we proceed as for two-factor experiments and fit a model with all possible interactions.
- We check the highest-order interaction first. If it is not significant we re-fit without that term and continue checking. In a balanced factorial design omitting certain interactions will not change the estimates of coefficients for other interactions, but it will change MS$_e$.
- The formula for model with all possible interactions has `*` between the factors.
- We can modify the formula by “subtracting” particular terms using the `update` function.

Example 9.1.1: Battery separator data

```
> str(separate)

'data.frame': 24 obs. of  4 variables:
 $ silica: Ord.factor w/ 2 levels "Low"<"High": 1 1 1 1 1 1 1 1 1 1 ...
 $ time  : Ord.factor w/ 2 levels "Low"<"High": 1 1 1 1 1 1 2 2 2 2 ...
 $ temp  : Ord.factor w/ 2 levels "Low"<"High": 1 1 1 2 2 2 1 1 1 2 ...
 $ y     : num  40.9 42.2 41.3 46.3 47 48.2 48.6 49.5 46.6 69 ...
```



Example 9.1.1 (cont'd)

```
> summary(fm1 <- aov(y ~ time * temp * silica, separate))
```

```
                  Df  Sum Sq Mean Sq F value    Pr(>F)
time               1 1732.30 1732.30 988.239 8.212e-16
temp               1 1318.68 1318.68 752.280 7.014e-15
silica             1  129.27  129.27  73.746 2.185e-07
time:temp          1   31.97   31.97  18.238 0.0005854
time:silica        1   78.84   78.84  44.979 5.041e-06
temp:silica        1   38.25   38.25  21.823 0.0002553
time:temp:silica   1  110.51  110.51  63.044 6.113e-07
Residuals         16   28.05    1.75
```

- We see that the highest-order interaction term is significant. In such cases we generally retain all the lower-level interactions and the main effects whether or not they have small p-values. (In this case they all do.)

Example 9.1.1 (cont'd)

- There are two approaches for follow-up. We can consider the 8 different combinations of factor levels as 8 levels in a one-factor model or we can condition on a level of one factor and consider the others.

```
> summary(fm1a <- aov(y ~ time * silica, separate,
+                        subset = temp == "High"))

            Df  Sum Sq Mean Sq  F value    Pr(>F)
time         1 1117.47 1117.47 567.2437 1.029e-08
silica       1  154.08  154.08  78.2149 2.107e-05
time:silica  1    1.33    1.33   0.6768    0.4345
Residuals    8   15.76    1.97
```

```
> summary(fm1b <- aov(y ~ time + silica, separate,
+                        subset = temp == "High"))

            Df  Sum Sq Mean Sq F value    Pr(>F)
time         1 1117.47 1117.47 588.371 1.646e-09
silica       1  154.08  154.08  81.128 8.483e-06
Residuals    9   17.09    1.90
```

Blocking with multiple experimental factors

- It is not uncommon to have blocking factors in a multi-factor design. These are known sources of variability that cannot (or should not) be held constant.
- As before, we list blocking factors first in any model formula. This is not important for balanced data but can be important with unbalanced data.

```
> str(defect <-
+     data.frame(num = c(9,9,5,5,7,8,2,3,8,7,4,4),
+                op = gl(3,4),
+                app = gl(2,2,12,labels = c("A","B")),
+                size = gl(2,1,12)))

'data.frame': 12 obs. of  4 variables:
 $ num : num  9 9 5 5 7 8 2 3 8 7 ...
 $ op  : Factor w/ 3 levels "1","2","3": 1 1 1 1 2 2 2 2 3 3 ...
 $ app : Factor w/ 2 levels "A","B": 1 1 2 2 1 1 2 2 1 1 ...
 $ size: Factor w/ 2 levels "1","2": 1 2 1 2 1 2 1 2 1 2 ...
```

Structure of defect data

```
> ftable(xtabs(num ~ op+size+app, defect)) # compare Table 9.7

        app A B
op size
1  1        9 5
   2        9 5
2  1        7 2
   2        8 3
3  1        8 4
   2        7 4

> summary(fm2 <- aov(num ~ op + size * app, defect))

            Df Sum Sq Mean Sq F value    Pr(>F)
op           2  8.167   4.083     9.8   0.01287
size         1  0.083   0.083     0.2   0.67041
app          1 52.083  52.083   125.0 3.056e-05
size:app     1  0.083   0.083     0.2   0.67041
Residuals    6  2.500   0.417
```

Reduced models

```
> summary(fm3 <- aov(num ~ op + size + app, defect))

            Df Sum Sq Mean Sq  F value    Pr(>F)
op           2  8.167   4.083  11.0645  0.006803
size         1  0.083   0.083   0.2258  0.649120
app          1 52.083  52.083 141.1290 6.803e-06
Residuals    7  2.583   0.369

> summary(fm4 <- aov(num ~ op + app, defect))

            Df Sum Sq Mean Sq F value    Pr(>F)
op           2  8.167   4.083   12.25  0.003671
app          1 52.083  52.083  156.25 1.570e-06
Residuals    8  2.667   0.333
```

- There is no need for multiple comparisons of levels of app (there are only two levels).
- Generally we don't compare levels of op because it is a blocking factor. In this case we may want to investigate why there is such a large variability between operators.

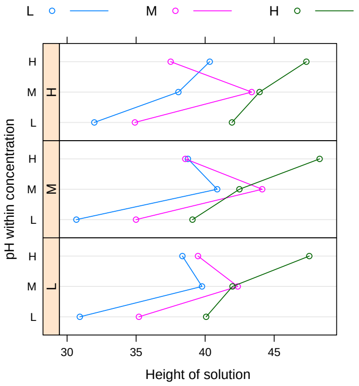
Unreplicated multi-factor designs

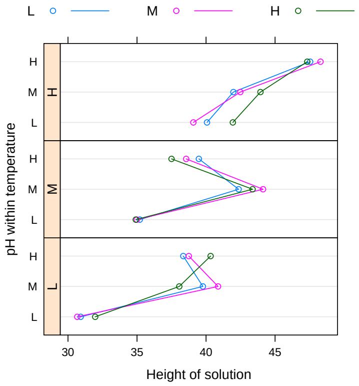
- If we allow the highest-order interaction term in an unreplicated factorial design we will have no degrees of freedom for error.
- We can remove the highest-order interaction by using "powers" in the formula to restrict to interactions up to a specific order.

```
> str(defoam)
```

```
'data.frame': 27 obs. of  4 variables:
 $ conc  : Ord.factor w/ 3 levels "L"<"M"<"H": 1 1 1 1 1 1 1 1 1 2 ...
 $ pH    : Ord.factor w/ 3 levels "L"<"M"<"H": 1 1 1 2 2 2 3 3 3 1 ...
 $ temp  : Ord.factor w/ 3 levels "L"<"M"<"H": 1 2 3 1 2 3 1 2 3 1 ...
 $ height: num  30.9 35.2 40.1 39.8 42.4 ...
```

Plots of the defoam data





- Note that there are effectively no differences due to concentration, at the levels examined.

Models for the defoam data

```
> summary(fm5 <- aov(height ~ (conc + pH + temp)^2, defoam))
            Df  Sum Sq Mean Sq F value    Pr(>F)
conc         2   0.759   0.380  0.3108 0.7413502
pH           2 238.977 119.488 97.7850 2.385e-06
temp         2 228.653 114.327 93.5608 2.826e-06
conc:pH      4   4.058   1.014  0.8301 0.5420108
conc:temp    4   2.949   0.737  0.6034 0.6712171
pH:temp      4  82.012  20.503 16.7789 0.0005885
Residuals    8   9.776   1.222
> summary(fm6 <- aov(height ~ (pH + temp)^2, defoam))
            Df  Sum Sq Mean Sq F value    Pr(>F)
pH           2 238.977 119.488 122.609 3.270e-11
temp         2 228.653 114.327 117.312 4.733e-11
pH:temp      4  82.012  20.503  21.038 1.378e-06
Residuals   18  17.542   0.975
```

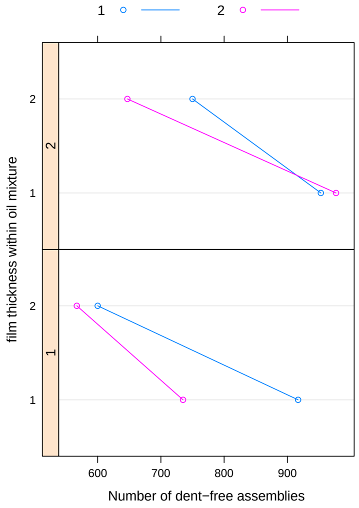
Section 9.2: 2^k factorial designs

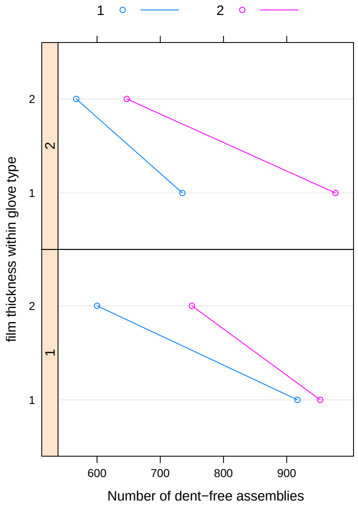
- A commonly used type of screening design is a 2^k factorial design in which each of k factors is examined at 2 levels only.
- Such a design will only detect linear trends and possible interactions. It cannot, for example, detect a quadratic relationship reliably (Fig. 9.9).
- One technical point, it can be an advantage to represent the 2-level factors as `ordered` and to use the "Helmert" contrasts for ordered factors. These provide a +/- coding in the model matrix.

```
> options(contrasts = c("contr.treatment", "contr.helmert"))
> str(dents <- data.frame(ok = c(917,600,953,750,735,567,977,647
+                         A = gl(2,1,8,ord=1), B = gl(2,2,8,ord=
+                         C = gl(2,4,ord=1)))

'data.frame': 8 obs. of  4 variables:
 $ ok: num  917 600 953 750 735 567 977 647
 $ A : Ord.factor w/ 2 levels "1"<"2": 1 2 1 2 1 2 1 2
 $ B : Ord.factor w/ 2 levels "1"<"2": 1 1 2 2 1 1 2 2
 $ C : Ord.factor w/ 2 levels "1"<"2": 1 1 1 1 2 2 2 2
```

Plots of the dents data





Model fits for the dents data

```
> summary(fm7 <- aov(ok ~ (A+B+C)^2, dents))

            Df Sum Sq Mean Sq F value Pr(>F)
A            1 129541  129541 13.6043 0.1685
B            1  32258   32258  3.3877 0.3168
C            1  10805   10805  1.1347 0.4799
A:B          1    288     288  0.0302 0.8904
A:C          1     61      61  0.0064 0.9494
B:C          1   2312    2312  0.2428 0.7085
Residuals    1   9522    9522
```

None of the two-factor interactions are significant (but it is very difficult to show significance with 1 degree of freedom for residuals).

```
> summary(fm8 <- aov(ok ~ A+B+C, dents))

            Df Sum Sq Mean Sq F value   Pr(>F)
A            1 129541  129541 42.5333 0.002854
B            1  32258   32258 10.5916 0.031243
C            1  10805   10805  3.5475 0.132744
Residuals    4  12183    3046
```

Model fits for the dents data (cont'd)

```
> summary(fm9 <- aov(ok ~ A+B, dents))

            Df Sum Sq Mean Sq F value   Pr(>F)
A            1 129541  129541 28.1769 0.003171
B            1  32258   32258  7.0166 0.045488
Residuals    5  22987    4597

> summary.lm(fm9)
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   768.25      23.97  32.047 5.56e-07
A1           -127.25      23.97  -5.308  0.00317
B1             63.50      23.97   2.649  0.04549
Residual standard error: 67.8 on 5 degrees of freedom
Multiple R-squared: 0.8756, Adjusted R-squared: 0.8258
F-statistic:  17.6 on 2 and 5 DF,  p-value: 0.005458
> fitted(fm9)

    1     2     3     4     5     6     7     8
832.0 577.5 959.0 704.5 832.0 577.5 959.0 704.5
```

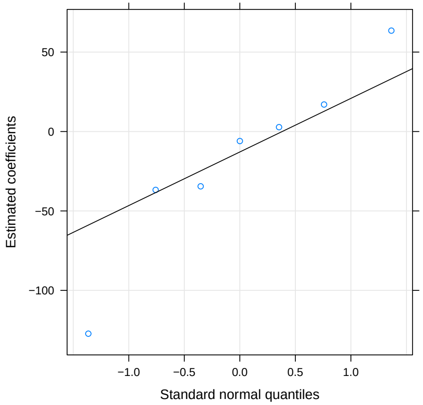
Interpretation of coefficients

- The coefficients shown in the `summary.lm` output are in the +/- encoding. The `(Intercept)` is the response for a "typical" setting; the `A1` coefficient is added to this for the high level of `A` and subtracted for the low level. Similarly for `B1`.
- That is, the fitted values are the `(Intercept)` plus the four possible combinations of ±`A1` and ±`B1`.
- *Yates' algorithm* is a method for determining these coefficient estimates by hand calculation. It is interesting but no longer needed.
- Notice also that the standard errors of all the coefficients are identical. This provides an alternative approach to evaluating the significance of coefficients in a *saturated* model (as many coefficients as there are responses). We examine a normal probability plot of the coefficient estimates (excluding the intercept) which we expect to have outliers. The non-outliers are the "trivial many". The outliers are the "important few".

QQ-plot of coefficients from the saturated model

```
> (cc <- coef(fm10 <- lm(ok ~ A * B * C, dents))[-1])
```

```
     A1       B1       C1    A1:B1    A1:C1    B1:C1 A1:B1:C1
-127.25    63.50   -36.75    -6.00     2.75    17.00   -34.50
```

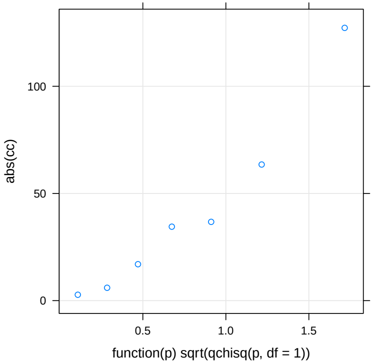


Interpretation of the QQ plot of coefficient estimates

- The plot on the previous slide indicates that 2 coefficients are the “important few”. Checking the listing of the coefficients we see that these are the main effects for A and for B — the same conclusion as before.
- Generally this approach is preferred to the approach of testing for two-factor interactions using F tests with only 1 denominator degree of freedom. F tests with very few denominator degrees of freedom are not at all powerful.
- One variation on this approach is to compare the absolute value of the coefficients to the theoretical quantiles. This is called a “half-normal” plot.
- It is not part of the course but the way to create such a plot is shown on the next slide. In this plot you are only interested in the large magnitudes. The “reference line” would be a line passing through the origin. You imagine the low magnitudes as falling near such a line with the high values above the line.

Half-normal plot of saturated model coefficients

```
> print(qqmath( ~ abs(cc), aspect = 1, type = c("g","p"),
+                 distribution = function(p) sqrt(qchisq(p, df = 1)
```



Here you could make a case for 5 values near the line (and 2 above) or for 6 values near the line and only one above.

Section 9.3 Fractional factorial designs

- When the number of factors in a design is large or the cost per observation is high, even an unreplicated 2^k factorial design may be too expensive. We can choose to run a fraction of the full factorial design.
- The most common such *fractional factorial design* is a $\frac{1}{2}$ fraction of the 2^k, sometimes written as a 2^{k-1} design.
- We must choose the fraction carefully. We will not be able to estimate all the main effects and interactions as they will be *aliased*. We don't want potentially important effects or interactions to be aliased with each other.
- When creating a 2^{k-1} design we confound the last factor with the highest order interaction of the preceeding factors.
- That is, we lay out a full factorial on $k-1$ two-level factors then determine the highest order interaction and use it to establish the low/high levels for the next factor.

A half-fraction of a 2^k design

- For a 2^{4-1} design (example 4.3.2) the first step is

```
> xmp932 <- data.frame(A = gl(2,1,8,ord=1), B = gl(2,2,8,ord=1),
+                      C = gl(2,4,ord=1))
> model.matrix(~ A * B * C, xmp932)

  (Intercept) A1 B1 C1 A1:B1 A1:C1 B1:C1 A1:B1:C1
1           1 -1 -1 -1     1     1     1       -1
2           1  1 -1 -1    -1    -1     1        1
3           1 -1  1 -1    -1     1    -1        1
4           1  1  1 -1     1    -1    -1       -1
5           1 -1 -1  1     1    -1    -1        1
6           1  1 -1  1    -1     1    -1       -1
7           1 -1  1  1    -1    -1     1       -1
8           1  1  1  1     1     1     1        1
attr(,"assign")
[1] 0 1 2 3 4 5 6 7
attr(,"contrasts")
attr(,"contrasts")$A
[1] "contr.helmert"
attr(,"contrasts")$B
[1] "contr.helmert"
```

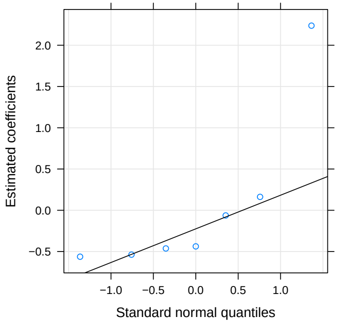
Example 9.3.2 (cont'd)

- We choose the low/high levels for factor `D` to be the same as the ABC interaction in the previous slide. That is we generate `D = ABC`.
- The *generator* relationship is `I = ABCD`. Not only are all the main effects confounded with a three-factor interaction but all the two-factor interactions are confounded with each other.
- The responses are observed and incorporated and we fit the saturated model on the first $k-1$ factors

```
> xmp932 <- within(xmp932, {
+     D <- ordered(c(1,2,2,1,2,1,1,2))
+     y <- c(3.6,10,8,3.2,7.6,3.2,3.7,6.0)
+ })
> (cc2 <- coef(fm11 <- lm(y ~ A * B * C, xmp932))[-1])

     A1       B1       C1    A1:B1    A1:C1    B1:C1 A1:B1:C1
-0.0625  -0.4375  -0.5375  -0.5625  -0.4625   0.1625   2.2375
```

Interpretation of coefficients



- The interpretation is that only the largest coefficient is important. This can be the three-factor interaction `ABC` or the main effect `D`. We assume it is the main effect that is important.

Resolution of fractional factorials

- The defining relation (`I = ABCD`, in our example) is composed of *words*. In this case there is only one word but for higher order fractions we use multiple words.
- The *resolution* of the design is the length of the shortest word in its defining relation. Our example is a *resolution IV* design.
- A *resolution III* design confounds some main effects with two-factor interactions. A *resolution IV* design confounds two-factor interactions with each other but not with main effects.
- Generally we prefer reolution IV or higher.