Introduction to Factorial Designs

Henry Scharf

MATH/STAT 571B

Module Goals:

Ch. 5 [DAE]: Introduction to Factorial Designs

Students will be able to:

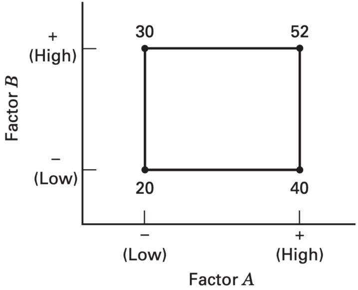
1. Explain the benefits of using a factorial design over alternatives.
2. Implement interaction models for two-factor factorial designs using R.
3. Explain interactions in the context of linear regression.
4. Explain methods for blocking nuisance variables with factorial designs.

Basic Definitions and Principles

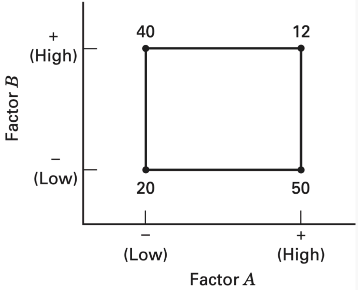
Basic Definitions and Principles

- **factorial design**: All possible combinations of factor levels; factors are **crossed**.
- **main effects**: Change in response for one ‘unit’ change in factor.
- **interaction**: When the effect of one factor depends on the level of another.
  - When an interaction is present, main effects *cannot be meaningfully studied in isolation*.

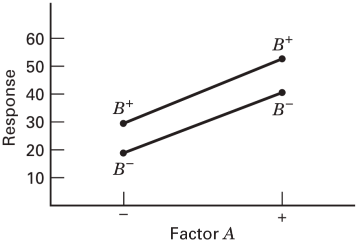
Design Diagrams

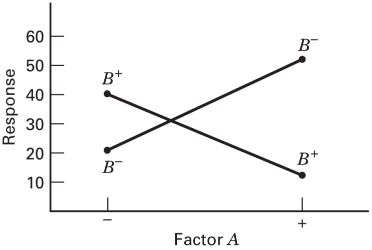


■ **FIGURE 5.1** **A two-factor factorial experiment, with the response (*y*) shown at the corners**

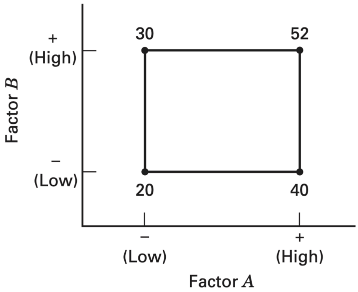


■ **FIGURE 5.2** **A two-factor factorial experiment with interaction**

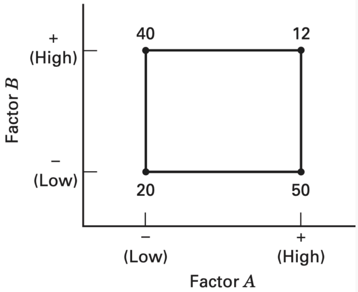




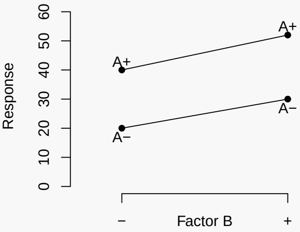
Design Diagrams

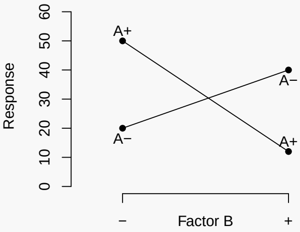


■ **FIGURE 5.1 A two-factor factorial experiment, with the response (y) shown at the corners**



■ **FIGURE 5.2 A two-factor factorial experiment with interaction**





Linear Regression Representation

$$
\begin{aligned}
y_{ij} &= \mu + \tau_i + \theta_j + (\tau\theta)_{ij} + \epsilon_{ij}, & i = 1, \ldots, a,\ j = 1, \ldots, b \\
y_k &= \beta_0 + \beta_1 x_{k1} + \beta_2 x_{k2} + \beta_{12} x_{k1} x_{k2} + \epsilon_k, & k = 1, \ldots, ab
\end{aligned}
$$

- Coding "Low" and "High" as $x = -1$ and $x = 1$ is consistent with the constraints $\sum_{i=1}^{a} \tau_i = 0$ and $\sum_{j=1}^{b} \theta_j = 0$.[1]

```
dat <- data.frame(A = c(-1, 1, -1, 1), B = c(-1, -1, 1, 1), y = c(20, 40, 30, 52))
dat$AB <- dat$A * dat$B
dat$AB
## [1]  1 -1 -1  1
coef(aov(y ~ A + B + AB, data = dat))
## (Intercept)           A           B          AB
##        35.5        10.5         5.5         0.5
```

- E.g., $\hat{y}_{-+} = 35.5 + 10.5(-1) + 5.5(1) + 0.5(-1 \cdot 1) = 30$

[1] Compare with DAE p. 185.

Linear Regression Representation

$$
\begin{aligned}
y_{ij} &= \mu + \tau_i + \theta_j + (\tau\theta)_{ij} + \epsilon_{ij}, && i = 1, \ldots, a,\ j = 1, \ldots, b \\
y_k &= \tilde{\beta}_0 + \tilde{\beta}_1 x_{k1} + \tilde{\beta}_2 x_{k2} + \tilde{\beta}_{12} x_{k1} x_{k2} + \epsilon_k, && k = 1, \ldots, ab
\end{aligned}
$$

- Coding "Low" and "High" as $x = 0$ and $x = 1$ is consistent with the constraints $\tau_1 = 0$ and $\theta_1 = 0$.[2]

```r
dat <- data.frame(A = c(0, 1, 0, 1), B = c(0, 0, 1, 1), y = c(20, 40, 30, 52))
dat$AB <- dat$A * dat$B
dat$AB
## [1] 0 0 0 1
coef(aov(y ~ A + B + AB, data = dat))
## (Intercept)           A           B          AB
##          20          20          10           2
```

- E.g., $\hat{y}_{-+} = 20 + 20(0) + 10(1) + 2(0 \cdot 1) = 30$

(1) What is $\hat{y}_{++}$?

[2] Default in R.

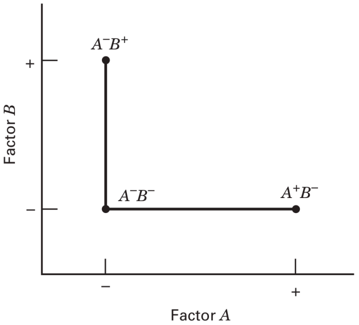
The Advantage of Factorials

An Alternative: One-factor-at-a-time

```r
dat_Ftrl <- data.frame(A = rep(c("Low", "High", "Low", "High"), 3),
                       B = rep(c("Low", "Low", "High", "High"), 3))
dat_OaaT <- data.frame(A = rep(c("Low", "Low", "High"), 4),
                       B = rep(c("High", "Low", "Low"), 4))
```

- Consider two designs with the same total number of observations.
- Factorial design has 3 replicates of 4 factor combinations.
- "One-factor-at-a-time" design has 4 replicates of 3 factor combinations.

(2) How can you tell that the one-factor-at-a-time design does not have orthogonal factors?



■ **FIGURE 5.7** A one-factor-at-a-time experiment

An Alternative: One-factor-at-a-time

```
set.seed(2024)
tau <- c(Low = -1, High = 1)
theta <- c(Low = 0, High = 1.5)
sigma <- 1
epsilon <- rnorm(12, sd = sigma)
dat_Ftrl$y <- tau[dat_Ftrl$A] + theta[dat_Ftrl$B] + epsilon
dat_OaaT$y <- tau[dat_OaaT$A] + theta[dat_OaaT$B] + epsilon
head(dat_OaaT)
##      A    B          y
## 1  Low High  1.4819694
## 2  Low  Low -0.5312850
## 3 High  Low  0.8920287
## 4  Low High  0.2871218
## 5  Low  Low  0.1580985
## 6 High  Low  2.2923548
```

Orthogonality

```
anova(aov(y ~ A + B, dat = dat_Ftrl))
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## A          1 14.2982 14.2982  13.382 0.00525 **
## B          1  3.3379  3.3379   3.124 0.11094
## Residuals  9  9.6160  1.0684
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
anova(aov(y ~ B + A, dat = dat_Ftrl))
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## B          1  3.3379  3.3379   3.124 0.11094
## A          1 14.2982 14.2982  13.382 0.00525 **
## Residuals  9  9.6160  1.0684
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Orthogonality

```
anova(aov(y ~ A + B, dat = dat_OaaT))
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## A          1  4.9020  4.9020  4.2977 0.06802 .
## B          1  5.0475  5.0475  4.4252 0.06474 .
## Residuals  9 10.2656  1.1406
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
anova(aov(y ~ B + A, dat = dat_OaaT))
## Analysis of Variance Table
##
## Response: y
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## B          1  0.7033  0.7033  0.6166 0.45248
## A          1  9.2462  9.2462  8.1063 0.01918 *
## Residuals  9 10.2656  1.1406
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Efficiency

- As we've seen with past designs, orthogonality leads to efficiency in detecting factor effects.

```
p_values <- sapply(1:1000, function(rep){
  epsilon <- rnorm(12, sd = sigma)
  dat_Ftrl$y <- tau[dat_Ftrl$A] + theta[dat_Ftrl$B] + epsilon
  dat_OaaT$y <- tau[dat_OaaT$A] + theta[dat_OaaT$B] + epsilon
  cbind(Ftrl = anova(aov(y ~ A + B, dat = dat_Ftrl))[c("A", "B"), "Pr(>F)"],
        OaaT = c(anova(aov(y ~ B + A, dat = dat_OaaT))["A", "Pr(>F)"],
                 anova(aov(y ~ A + B, dat = dat_OaaT))["B", "Pr(>F)"]))
})
mean_p_values <- matrix(rowMeans(p_values), 2, 2)
colnames(mean_p_values) <- c("Ftrl", "Oaat")
rownames(mean_p_values) <- c("A", "B")
mean_p_values
##         Ftrl       Oaat
## A 0.02671223 0.05991287
## B 0.08409549 0.14590490
```

Interactions

```
tau
##  Low High
##   -1    1
theta
##  Low High
##  0.0  1.5
tau_theta <- c("HighHigh" = 2)
```

- Effect of changing Factor A from “Low” to “High” when Factor B is “High”: $\tau_H - \tau_L + (\tau\theta)_{H,H} - (\tau\theta)_{L,H} = 1 - (-1) + 2 - 0 = 4$

(3) What is the effect of changing Factor B from “High” to “Low” when Factor A is “Low”?

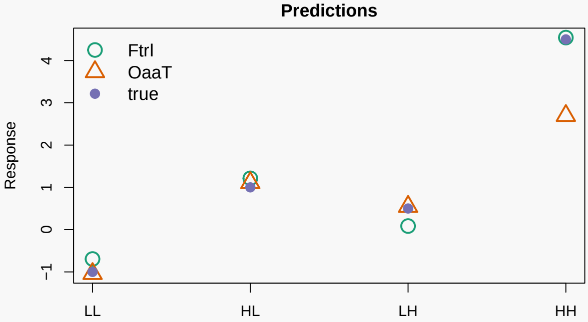
Interactions

- Perhaps the *most important limitation* of one-factor-at-a-time relative to a factorial design is the inability to estimate interactions.
- In general, a factorial design implies investigation of an interaction.

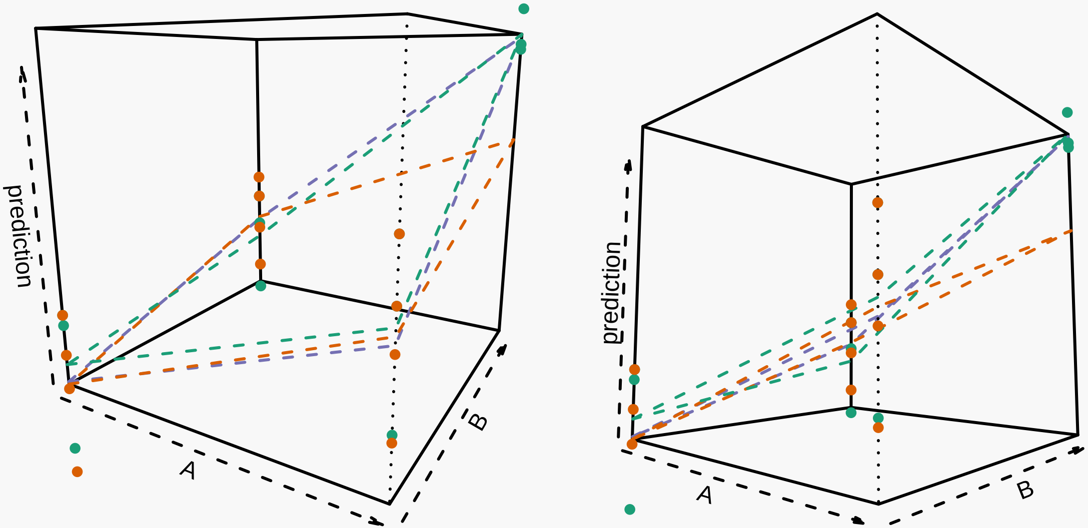
```
dat_Ftrl$y <- tau[dat_Ftrl$A] + theta[dat_Ftrl$B] +
  tau_theta * (dat_Ftrl$A == "High" & dat_Ftrl$B == "High") + epsilon
dat_OaaT$y <- tau[dat_OaaT$A] + theta[dat_OaaT$B] +
  tau_theta * (dat_OaaT$A == "High" & dat_OaaT$B == "High") + epsilon
Ftrl_aov <- aov(y ~ A * B, dat = dat_Ftrl)
OaaT_aov <- aov(y ~ A * B, dat = dat_OaaT)
coef(Ftrl_aov)
## (Intercept)        ALow        BLow   ALow:BLow
##    4.543002   -4.458085   -3.329762    2.549909
coef(OaaT_aov)
## (Intercept)        ALow        BLow
##    2.695740   -2.150143   -1.588633
```

Interactions

```
predictions <- expand.grid(A = c("Low", "High"), B = c("Low", "High"))
predictions$Ftrl <- predict(Ftrl_aov, predictions)
predictions$OaaT <- predict(OaaT_aov, predictions)
## Warning in predict.lm(OaaT_aov, predictions): prediction from rank-deficient
## fit; attr(*, "non-estim") has doubtful cases
predictions$true <- tau[predictions$A] + theta[predictions$B] +
  tau_theta * (predictions$A == "High" & predictions$B == "High")
```



Interactions



Example: Battery Life

$$SS_T = SS_{\text{Material}} + SS_{\text{Temp.}} + SS_{\text{Material}\times\text{Temp.}} + SS_E$$

- R formula syntax: `y ~ factorA * factorB` includes both main effects and interaction.
- Equivalent to: `y ~ factorA + factorB + factorA:factorB`

```
battery <- read.csv("battery.csv")
battery_aov <- aov(life ~ as.factor(material) * as.factor(temp), data = battery)
anova(battery_aov)
## Analysis of Variance Table
##
## Response: life
##                                     Df Sum Sq Mean Sq F value    Pr(>F)
## as.factor(material)                  2  10684  5341.9  7.9114  0.001976 **
## as.factor(temp)                      2  39119 19559.4 28.9677 1.909e-07 ***
## as.factor(material):as.factor(temp)  4   9614  2403.4  3.5595  0.018611 *
## Residuals                           27  18231   675.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tukey's HSD

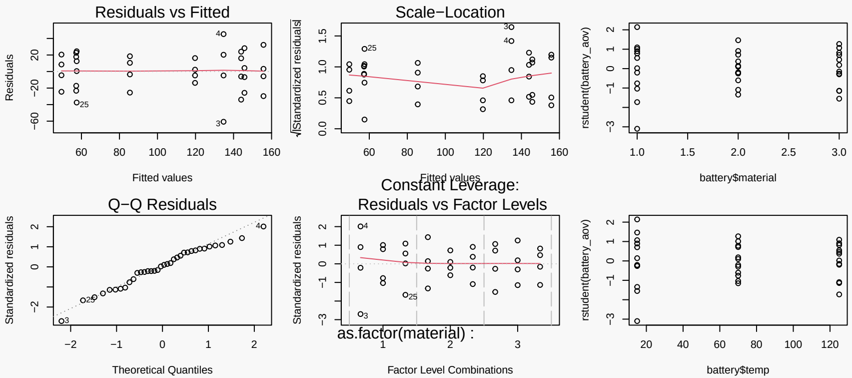
- Following example in DAE p. 195, suppose we want to estimate the differences in Material effects for the case when Temperature is 70°F.

```
TukeyHSD(battery_aov, which = "as.factor(material):as.factor(temp)")
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## $`as.factor(material):as.factor(temp)`
##                  diff         lwr        upr     p adj
## 3:125-3:15   -58.50 -120.323184   3.323184 0.0742711
## 2:70-1:70     62.50    0.676816 124.323184 0.0460388
## 3:70-1:70     88.50   26.676816 150.323184 0.0014173
## 1:125-1:70     0.25  -61.573184  62.073184 1.0000000
## 2:125-1:70    -7.75  -69.573184  54.073184 0.9999614
## 3:125-1:70    28.25  -33.573184  90.073184 0.8281938
## 3:70-2:70     26.00  -35.823184  87.823184 0.8822881
## 1:125-2:70   -62.25 -124.073184  -0.426816 0.0474675
```

(4) How would you group the materials' effects at 70°F?

Model Adequacy Checking

```r
layout(matrix(1:6, 2, 3)); par(mar = c(4, 4, 2.5, 1.5))
plot(battery_aov)
plot(battery$material, rstudent(battery_aov))
plot(battery$temp, rstudent(battery_aov))
```



(5) Does anything strike you as concerning?

Model Adequacy Checking

```r
par(mar = c(4, 4, 0.5, 0.5))
interaction_plot(battery_aov, lwd = 2, cex = 1.5, ylim = range(battery$life))
points(life ~ jitter(material, 0.5), data = battery, pch = (1:3)[as.factor(temp)],
       col = RColorBrewer::brewer.pal(3, "Dark2")[as.factor(temp)])
```