Randomized Blocks, Latin Squares, and Related Designs

Henry Scharf

MATH/STAT 571B

Module Goals:

Ch. 4 [DAE]:
Students will be able to:

1. Explain how and why blocking is implemented in experimental design.
2. Recognize specified designs from context of application.
3. Implement an appropriate analysis for the specified design using R.
4. Explain how blocking can be implemented for multiple sources of extraneous variability using Latin and Graeco-Latin square designs.

Randomized Complete Block Design

Vascular Example[1]

- Artificial veins produced by manufacturer through extrusion process.
- Rate of defects ("flicks") thought to depend on extrusion pressure, and also possibly the batch of material used.
- Four pressure levels randomly assigned within blocks defined by material batch.

```
vascular <- read.csv("vascular.csv")
head(vascular)
##   batch pressure flicks
## 1     1     8500   90.3
## 2     1     8700   92.5
## 3     1     8900   85.5
## 4     1     9100   82.5
## 5     2     8500   89.2
## 6     2     8700   89.5
```

[1] DAE Example 4.1 p.144

Vascular Example

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \;\; \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

- Interested in studying τ_i.
- Include β_j to hopefully reduce error variance σ^2 and therefore increase our power to detected small differences across τ_i.
- The R formula syntax for this model is `y ~ treatment + blocks`

Vascular Example: Analysis

```
vascular_aov <- aov(flicks ~ as.factor(pressure) + as.factor(batch), data = vascular)
summary(vascular_aov)
##                     Df Sum Sq Mean Sq F value  Pr(>F)
## as.factor(pressure)  3  178.2   59.39   8.107 0.00192 **
## as.factor(batch)     5  192.2   38.45   5.249 0.00553 **
## Residuals           15  109.9    7.33
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

(1) Recall from previous lectures: Why do we need the `as.factor()` around each predictor variable in the formula?

Vascular Example: Analysis

- Notice that changing R formula re-ordered rows in ANOVA output.
- p-value for `pressure` is overall test for treatment effect **under RCBD**.
- p-value for `batch` is overall test for block effect,[2] but this should only be used as a rough summary of evidence against $H_0 : \beta_1 = \cdots = \beta_b$

```
vascular_aov <- aov(flicks ~ as.factor(batch) + as.factor(pressure), data = vascular)
summary(vascular_aov)
##                      Df Sum Sq Mean Sq F value  Pr(>F)
## as.factor(batch)      5  192.2   38.45   5.249 0.00553 **
## as.factor(pressure)   3  178.2   59.39   8.107 0.00192 **
## Residuals            15  109.9    7.33
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

[2] see discussion on DAE p.143

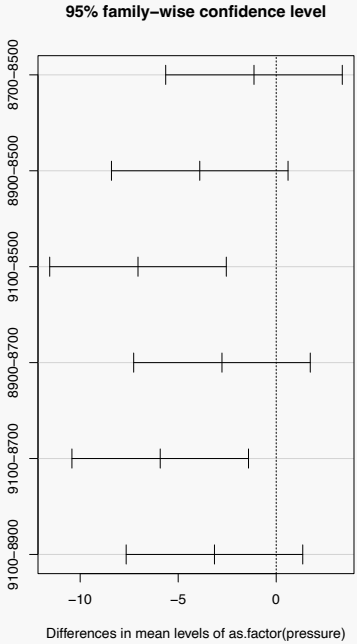
Vascular Example: Analysis

```
summary(lm(flicks ~ as.factor(pressure) + as.factor(batch), data = vascular))
## Call:
## lm(formula = flicks ~ as.factor(pressure) + as.factor(batch),
##     data = vascular)
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                90.721      1.657  54.735  < 2e-16 ***
## as.factor(pressure)8700    -1.133      1.563  -0.725 0.479457
## as.factor(pressure)8900    -3.900      1.563  -2.496 0.024713 *
## as.factor(pressure)9100    -7.050      1.563  -4.512 0.000414 ***
## as.factor(batch)2           2.050      1.914   1.071 0.301043
## as.factor(batch)3           3.300      1.914   1.724 0.105201
## as.factor(batch)4           2.850      1.914   1.489 0.157175
## as.factor(batch)5          -2.375      1.914  -1.241 0.233684
## as.factor(batch)6           6.750      1.914   3.527 0.003050 **
```

Vascular Example: Analysis

```
vascular_HSD <- TukeyHSD(vascular_aov,
                         which = "as.factor(pressure)")
plot(vascular_HSD)
```

(2) If lower rates of flicks are better, is there a best pressure? Are any clearly *not* the best?



Why this design?

- If we accept that this is a good design, then we can follow a recipe to provide a useful analysis.
- But, *why* is this a good design?

Controllable Nuisance Factors

- $y_{ij} = \mu + \tau_i + \underbrace{\beta_j}_{\text{nuisance}} + \epsilon_{ij}, \;\; \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$
- By **controllable** we mean we can control the assignment of treatments levels, not necessarily the level of the nuisance factor (if we could, it would probably be another treatment factor).
- Nuisance factor could drive variation in the response so we need to address it.
  - Randomization can prevent it from confounding our analysis, but it may be less powerful than other designs/analyses where we explicitly account for it.
- Want to *disentangle* variation due to the treatment from variation due to the nuisance factor.

Balance

- Balancing the number of observations for each treatment level is intuitively appealing for good reason.
- In HW1 and Ch. 2: Balanced sample sizes for each of two populations maximized power to detect a difference in means.
- This result extends to more than two groups. Balance leads to maximum power for comparing all pairwise means.
- Balanced designs tend to be more robust to violations of model assumptions (especially heteroskedasticity).

Two Alternatives to RCBD

- Aligned blocks and treatments
  - Easy to balance.
  - Could be convenient/inexpensive in context of experiment.
  - Maybe we can keep track of blocks and adjust for them in model?
- Completely Randomized Design (CRD) under balanced restriction
  - Protects against confounders and preserves causal interpretation.
  - Maybe we can keep track of blocks and adjust for them in model?

Two Alternatives to RCBD: Aligned

$$y_{ij} = \underbrace{\mu + \tau_i}_{\mu_i} + \beta_j + \epsilon_{ij}, \quad \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

Block A	Block B	Block C
Trt X	Trt Y	Trt Z
Trt X	Trt Y	Trt Z
Trt X	Trt Y	Trt Z

```
mus <- c(x = -1, y = 0, z = 1)
betas <- c(a = 1, b = 2, c = 3)
sigma <- 1/2
n <- length(mus) * length(betas)
treatments_aligned <- rep(c("x", "y", "z"), c(3, 3, 3))
treatments_aligned
## [1] "x" "x" "x" "y" "y" "y" "z" "z" "z"
blocks <- rep(c("a", "b", "c"), c(3, 3, 3))
blocks
## [1] "a" "a" "a" "b" "b" "b" "c" "c" "c"
```

Two Alternatives to RCBD: Aligned

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \;\; \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

```
y <- mus[treatments_aligned] + betas[blocks] + rnorm(n, sd = sigma)
data_aligned <- data.frame(block = blocks, treatment = treatments_aligned, y = y)
data_aligned
##   block treatment           y
## 1     a         x  0.007015276
## 2     a         x -0.541485849
## 3     a         x  0.074063542
## 4     b         y  1.880609431
## 5     b         y  2.207108523
## 6     b         y  1.707576693
## 7     c         z  3.853880465
## 8     c         z  4.457793226
## 9     c         z  4.162036987
```

Two Alternatives to RCBD: Aligned

- Changing the syntax in a simple ANOVA implementation reveals something problematic.

```
summary(aov(y ~ treatment + block, data = data_aligned))
##             Df Sum Sq Mean Sq F value   Pr(>F)
## treatment    2 27.892   13.95   155.2 6.82e-06 ***
## Residuals    6  0.539    0.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(aov(y ~ block + treatment, data = data_aligned))
##             Df Sum Sq Mean Sq F value   Pr(>F)
## block        2 27.892   13.95   155.2 6.82e-06 ***
## Residuals    6  0.539    0.09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two Alternatives to RCBD: Aligned

- Least-squares estimates aren't computable for all factors.

```
summary(lm(y ~ treatment + block, data = data_aligned))
## Call:
## lm(formula = y ~ treatment + block, data = data_aligned)
##
## Coefficients: (2 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1535     0.1731  -0.887 0.409348
## treatmenty    2.0852     0.2448   8.520 0.000143 ***
## treatmentz    4.3114     0.2448  17.615 2.15e-06 ***
## blockb            NA         NA      NA       NA
## blockc            NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two Alternatives to RCBD: Aligned

- Given the design, the predictors **treatment** and **block** are arranged so that if we know one, we know the other with certainty.
- In context of linear regression, our predictors are *perfectly collinear*.
- Effects of treatments and blocks are not jointly estimable. We can't tell which is responsible for change in response.
- Essentially zero power to detect non-zero treatment effect.

Two Alternatives to RCBD: CRD

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \;\; \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

Block A	Block B	Block C
Trt Y	Trt X	Trt X
Trt Y	Trt Z	Trt Z
Trt X	Trt Y	Trt Z

```
set.seed(2023)
treatments_CRD <- sample(rep(c("x", "y", "z"), 3), 9)
y <- mus[treatments_CRD] + betas[blocks] + rnorm(n, sd = sigma)
data_CRD <- data.frame(block = blocks, treatment = treatments_CRD, y = y)
```

- Still balanced in treatment assignments.

Two Alternatives to RCBD: CRD

- Changing the syntax in a simple ANOVA implementation still seems to matter.

```
summary(aov(y ~ treatment + block, data = data_CRD))
##             Df Sum Sq Mean Sq F value Pr(>F)
## treatment    2  9.392   4.696   4.918 0.0836 .
## block        2  4.543   2.272   2.379 0.2086
## Residuals    4  3.820   0.955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(aov(y ~ block + treatment, data = data_CRD))
##             Df Sum Sq Mean Sq F value Pr(>F)
## block        2  10.10   5.048   5.286 0.0753 .
## treatment    2   3.84   1.920   2.010 0.2487
## Residuals    4   3.82   0.955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Only defensible test yields p-value of 0.2487.

Two Alternatives to RCBD: CRD

```
summary(lm(y ~ treatment + block, data = data_CRD))
## Call:
## lm(formula = y ~ treatment + block, data = data_CRD)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06963    0.81108   0.086   0.9357
## treatmenty   1.16407    0.87405   1.332   0.2537
## treatmentz   1.55963    0.87405   1.784   0.1489
## blockb       1.29237    0.87405   1.479   0.2133
## blockc       2.32636    1.07049   2.173   0.0955 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two Alternatives to RCBD: CRD

- Given the design, knowing the value of one predictor *could* tell us something about the level of the other.[3]
- Effects of treatments and blocks are jointly estimable, but they *overlap* in the variation they can explain. Some portion of the overall variation is attributable to *either* treatment or blocks.
- Power of overall test is reduced as collinearity between treatments and blocks increases.

[3] How much depends on the particular realized design.

Randomized Complete Block Design

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \quad \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

Block A	Block B	Block C
Trt Y	Trt Z	Trt X
Trt X	Trt X	Trt Y
Trt Z	Trt Y	Trt Z

```
set.seed(2024)
treatments_RCBD <- c(sapply(1:3, function(blk) sample(c("x", "y", "z"), 3)))
y <- mus[treatments_RCBD] + betas[blocks] + rnorm(n, sd = sigma)
data_RCBD <- data.frame(block = blocks, treatment = treatments_RCBD, y = y)
```

Randomized Complete Block Design

- Changing the syntax in a simple ANOVA implementation *does not matter*.

```
summary(aov(y ~ treatment + block, data = data_RCBD))
##             Df Sum Sq Mean Sq F value  Pr(>F)
## treatment    2  8.902   4.451   18.07 0.00993 **
## block        2  5.152   2.576   10.46 0.02577 *
## Residuals    4  0.985   0.246
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(aov(y ~ block + treatment, data = data_RCBD))
##             Df Sum Sq Mean Sq F value  Pr(>F)
## block        2  5.152   2.576   10.46 0.02577 *
## treatment    2  8.902   4.451   18.07 0.00993 **
## Residuals    4  0.985   0.246
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Randomized Complete Block Design

```
summary(lm(y ~ treatment + block, data = data_RCBD))
## Call:
## lm(formula = y ~ treatment + block, data = data_RCBD)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3513     0.3699   0.950  0.39602
## treatmenty    0.8958     0.4052   2.211  0.09155 .
## treatmentz    2.4098     0.4052   5.947  0.00401 **
## blockb        0.4370     0.4052   1.079  0.34147
## blockc        1.7782     0.4052   4.389  0.01180 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Note that standard errors for effects are smaller than for CRD.

Randomized Complete Block Design

- Given the design, knowing the value of one predictor tells us *nothing* about the level of the other.
- This design ensures orthogonal decomposition of sums of squares:[4]

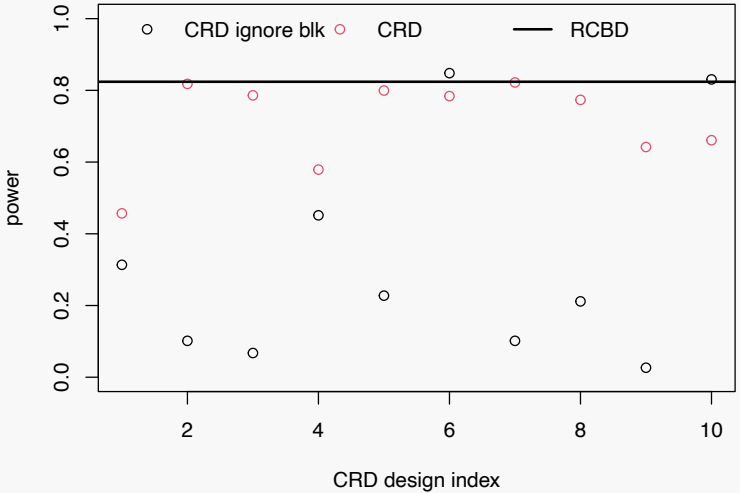
$$SS_T = SS_{\text{Treatments}} + SS_{\text{Blocks}} + SS_E$$

  - . . . "fundamental ANOVA equation for RCBD. . . " –DAE p. 142
- Variation in response explained by treatments and blocks is non-overlapping.

[4] In general: $SS_T = SS_{\text{Blocks}} + SS_{\text{Treatments|Blocks}} + SS_E$

Comparing power for CRD and RCBD

- Power calculations based on simulated data show that RCBD generally outperforms both CRD tests.



Latin Squares

Latin Squares

- Randomized Complete Block Designs allow us to optimally account for a single controllable nuisance factor.
- When we have multiple nuisance factors, we can extend this idea to more dimensions.
- To maintain optimality, find an "orthogonal" design.
  - Heuristic: Given a design, knowing the treatment level provides no information about an observation's group level (and vice versa).

Example: Rocket Propellant

- Primary factor of interest: propellant used in aircrew escape system on the observed burn rate.
- Nuisance factors:
  - Batch of material
  - Individual operator
- 5 different propellants will be tested using materials from 5 batches prepared by 5 operators.

$$y_{ijk} = \mu + \alpha_i + \tau_j + \beta_k + \epsilon_{ijk}, \quad \epsilon_{ij}k \sim \mathrm{N}(0, \sigma^2), \quad i, j, k = 1, \ldots, p$$

■ **TABLE 4.9**
Latin Square Design for the Rocket Propellant Problem

Batches of Raw Material	Operators				
	1	2	3	4	5
1	$A = 24$	$B = 20$	$C = 19$	$D = 24$	$E = 24$
2	$B = 17$	$C = 24$	$D = 30$	$E = 27$	$A = 36$
3	$C = 18$	$D = 38$	$E = 26$	$A = 27$	$B = 21$
4	$D = 26$	$E = 31$	$A = 26$	$B = 23$	$C = 22$
5	$E = 22$	$A = 30$	$B = 20$	$C = 29$	$D = 31$

Latin Squares

- All treatments occur once within each blocking variable.
- Knowing the row gives no information about column or treatment levels.
- Knowing the column...
- Knowing the treatment...

		Factor 1			
		1	2	3	4
Factor 2	1	A	B	D	C
	2	B	C	A	D
	3	C	D	B	A
	4	D	A	C	B

Finish the Latin Square

Factor 2 \ Factor 1	1	2	3	4
1	A	B	C	
2				A
3			B	
4				

Latin Squares

- For p treatment factors, p levels of blocking factor 1, and p levels of blocking factor 2, Latin Square designs preserve orthogonality across treatments and both nuisance factors.
- Orthogonality ensures the decomposition of sums of squares

$$SS_T = SS_{\text{Rows}} + SS_{\text{Columns}} + SS_{\text{Treatments}} + SS_E$$

$$SS_T = SS_{\text{Block 1}} + SS_{\text{Block 2}} + SS_{\text{Treatments}} + SS_E$$

$$SS_T = SS_\alpha + SS_\beta + SS_\tau + SS_E$$

Question... [5]

- But wait, how often will it really happen that we have *exactly* the same number of levels for treatments and TWO blocking variables?!

[5] At least, this is what *I* thought when I first saw Latin Squares...

Latin Squares

- In practice, we will usually specify treatment levels of interest (based on expert information).
- Blocks often correspond to variables we can't choose the levels for, but which we can assign.
- The *number* of levels of each blocking factor can be picked out of convenience, so this situation really can happen reasonably often.
- For instance, there is no reason to prefer to examine 5 batches of material and 5 operators for rocket propellant example beyond its usefulness in the design.

Example: Rocket Propellant

```
rocket <- read.csv("rocket.csv")
head(rocket, 4)
##   batch operator propellant assembly burn
## 1     1        1          A        a   24
## 2     2        1          B        b   17
## 3     3        1          C        c   18
## 4     4        1          D        d   26
rocket_aov <- aov(burn ~ propellant + as.factor(batch) + as.factor(operator),
                  data = rocket)
summary(rocket_aov)
##                     Df Sum Sq Mean Sq F value  Pr(>F)
## propellant           4    330   82.50   7.734 0.00254 **
## as.factor(batch)     4     68   17.00   1.594 0.23906
## as.factor(operator)  4    150   37.50   3.516 0.04037 *
## Residuals           12    128   10.67
```

(3) How much evidence is there that the propellant formulations have different effects on the burn rate? Batch of material? Operator?

Example: Rocket Propellant

```
summary(aov(burn ~ as.factor(batch) + as.factor(operator) + propellant,
            data = rocket))
```

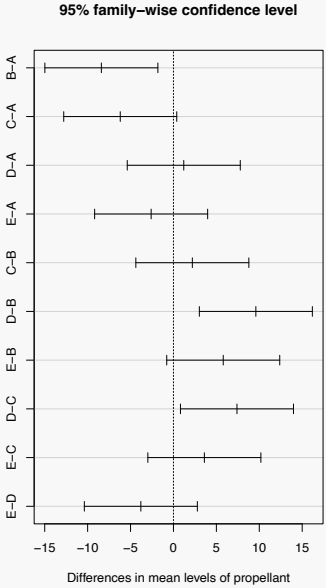
(4) How would the ANOVA table change if the order of predictor variables in the R formula were changed?

Example: Rocket Propellant

```
plot(TukeyHSD(rocket_aov, which = "propellant"))
```

(5) Which propellant has the most positive estimated effect on burn rate? Least positive?

(6) How could you "collect" these levels based on the 95% CIs?



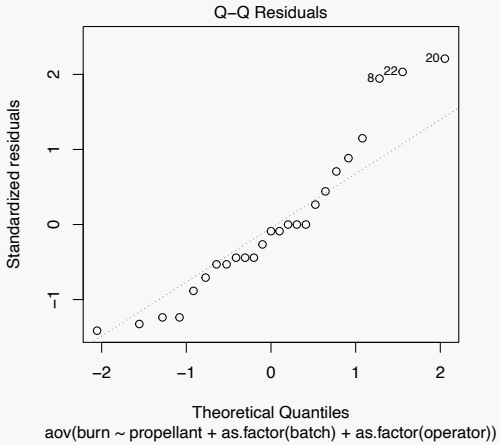
Model Adequacy Checking

Model Adequacy Checking

1. Normality
   - QQ-plot, histogram of residuals
2. Homoskedasticity: Residual variances could be constant across factor combinations
   - Residuals vs. fitted values, predictors should **not** show heteroskedasticity
3. Independence of ϵ_{ij} + Additivity of predictors / factors
   - Residual plots should be "unstructured"
   - Interaction plots

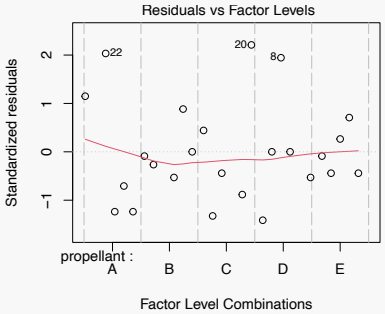
Normality

```
plot(rocket_aov, which = 2)
```



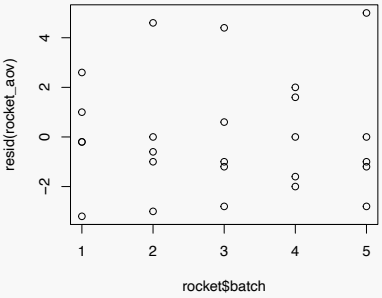
Homoskedasticity

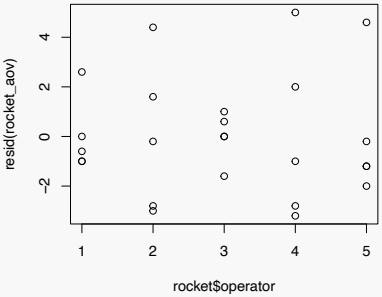
```
plot(rocket_aov, which = 5)
plot(rocket$batch, resid(rocket_aov))
plot(rocket$operator, resid(rocket_aov))
```



(7) Do you see any evidence of heteroskedasticity? If so, for which group/treatment level?

Use `rstudent()` in lieu of `resid()` for std. residuals.





Additivity

- To begin, return to treatment factor and one additional blocking variable.

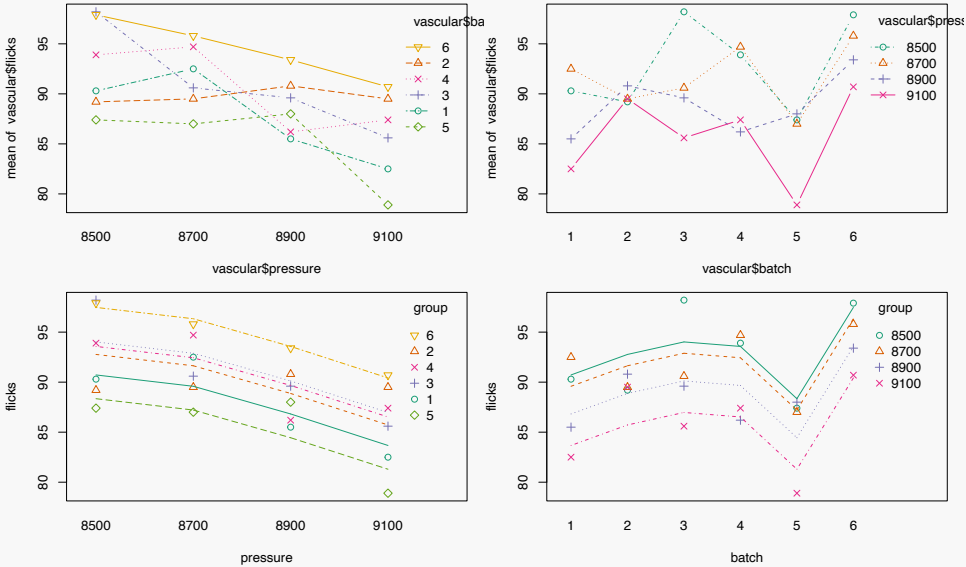
$$y_{ij} = \mu + \underbrace{\tau_i + \beta_j}_{\text{additive}} + \epsilon_{ij}, \quad \epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

- Changes in the response caused by treatment *are the same for all levels* of the group variable.
- Changes in the response caused by group *are the same for all levels* of the treatment variable.

Put another way...

- Whichever treatment is optimal for group A is also optimal for every other group.
- Whichever group had the best expected outcome for treatment level X also had the best expected outcome for all other treatment levels.

Interaction Plots: Two Versions



Interaction Plots: Two Versions

- Base R (`stats` package): `interaction.plot()`

```
interaction.plot(vascular$pressure, vascular$batch, vascular$flicks, type = "b",
                 pch = 1:6, col = RColorBrewer::brewer.pal(6, "Dark2"))
```

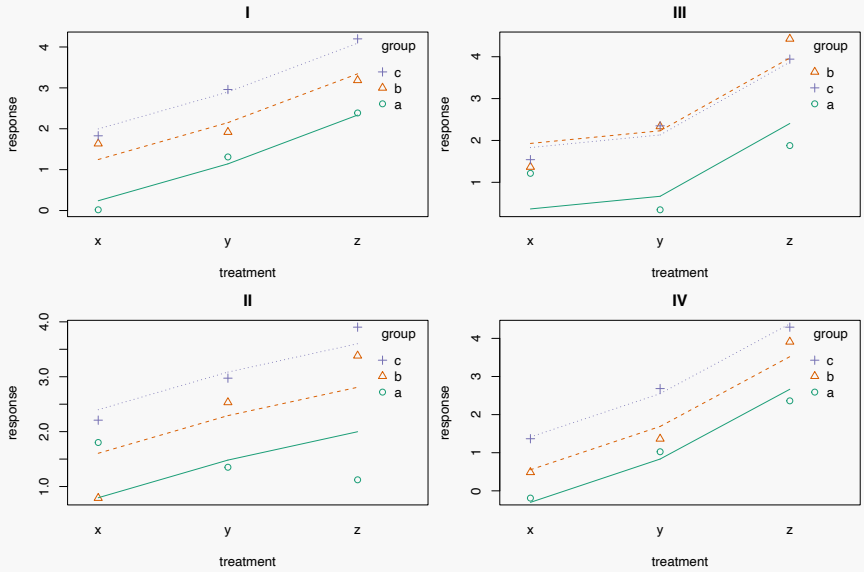
- Henry[6] (on D2L): `interaction_plot()`

```
source('interaction_plot.R')
interaction_plot(aov(flicks ~ as.factor(pressure) + as.factor(batch), data = vascular))
```

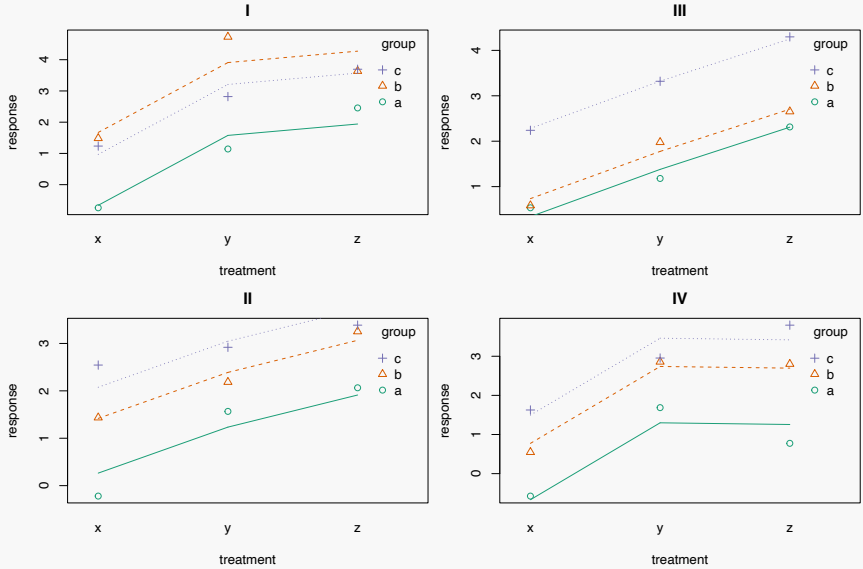
- To switch roles of treatment and group, change argument order or formula.

[6] Stolen from SAS...

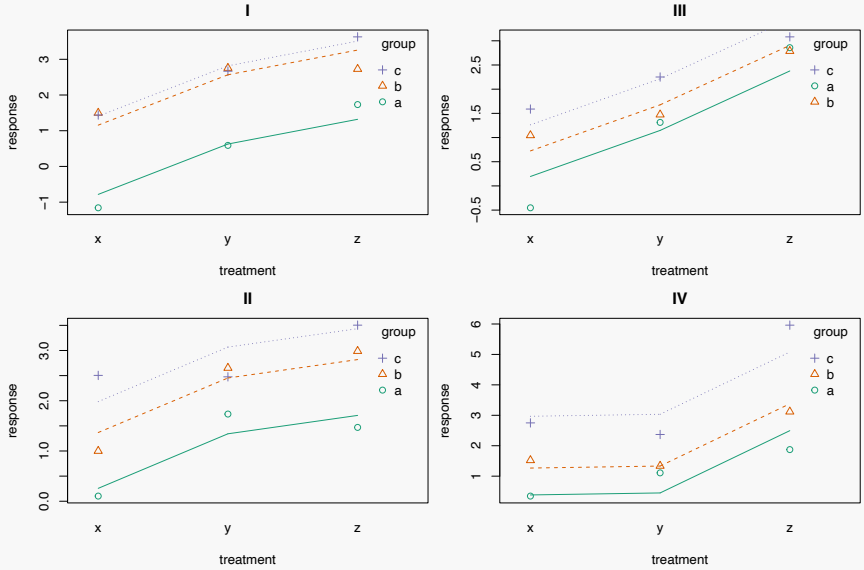
Sample Interaction Plots: Can You Spot the True Interaction?



Sample Interaction Plots: Can You Spot the True Interaction?



Sample Interaction Plots: Can You Spot the True Interaction?



Additivity: Three Predictors

- When there are more than two predictors, we can look at variables two at a time.
- `interaction_plot()` shows the first two variables and averages over the third.

```
layout(matrix(1:3, 1, 3)); par(mar = c(4, 4, 1, 1))
interaction_plot(aov(burn ~ propellant + as.factor(batch) + as.factor(operator),
                     data = rocket), main = "Propellant-Batch")
interaction_plot(aov(burn ~ propellant + as.factor(operator) + as.factor(batch),
                     data = rocket), main = "Propellant-Operator")
interaction_plot(aov = aov(burn ~ as.factor(operator) + as.factor(batch) + propellant,
                     data = rocket), main = "Operator-Batch")
```