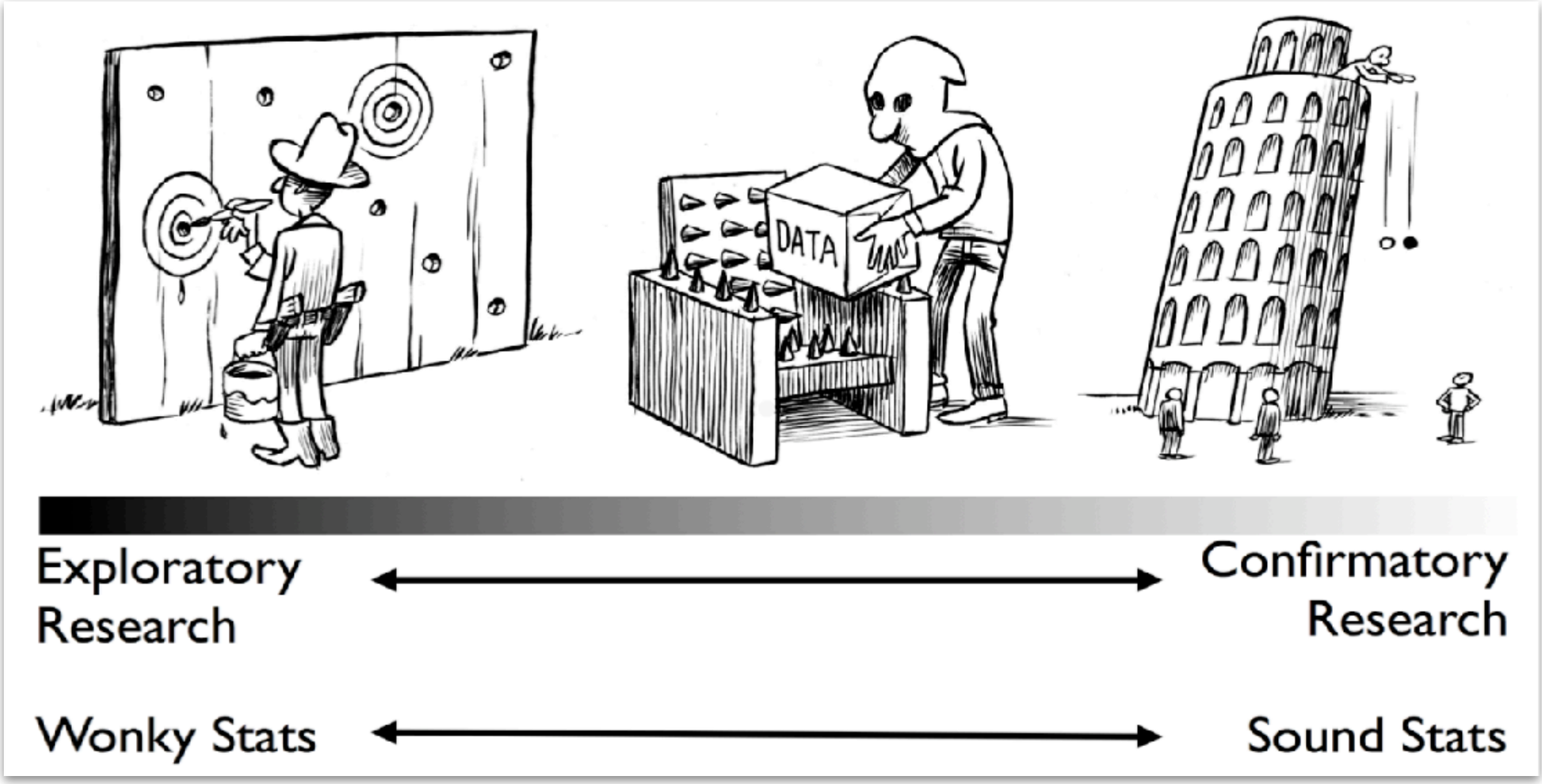
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

Power analysis



02/09/2024

Things that came up

Question Regarding MLR #37

Sherry Hu
50 minutes ago in General

PIN STAR WATCH 6 VIEWS

Dear Team,

When building multiple linear regression models and reporting, when should we use summary(model) and when should we use anova(model, naive horizontal model), as both would give p-values of each predictor, with summary(model) using the t distribution and the anova test using the F distribution. Is there a standard in deciding when to use what? When we are asked to explain the parameters of the model, should we include its significance/nonsignificance with the p-value from the summary(model) [t distribution] or should we include its significance/nonsignificance with the p-value from the anova test [F distribution]?

Thanks,

Comment Edit Delete Endorse •••

1 Answer

Tobi Gerstenberg STAFF
6 minutes ago

Thanks for posting Sherry! I'll address your question at the beginning of class on Friday.

Comment Edit Delete Endorse •••

Add comment

```
Analysis of Variance Table

Model 1: balance ~ 1
Model 2: balance ~ 1 + income
  Res.Df      RSS Df Sum of Sq      F    Pr(>F)
1    399 84339912
2    398 66208745  1  18131167 108.99 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05
‘.’ 0.1 ‘ ’ 1
```

`anova()` gives me *F*s

but `lm()` gives me *t*s

deterministic mapping between t and F

$$t^2 = F$$

$$10.44^2 = 108.99$$

```
Call:
lm(formula = balance ~ 1 + income, data = df.credit)

Residuals:
    Min      1Q  Median      3Q     Max
-803.64 -348.99  -54.42  331.75 1100.25

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 246.5148    33.1993   7.425  6.9e-13 ***
income        6.0484     0.5794  10.440  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 407.9 on 398 degrees of freedom
Multiple R-squared:  0.215,     Adjusted R-squared:  0.213
F-statistic:   109 on 1 and 398 DF,  p-value: < 2.2e-16
```

it doesn't really matter t-values can be negative

(which could be helpful for making it clear that the effect was negative)

```r
lm(formula = balance ~ hand * skill, data = df.poker) %>%
anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```r
fit1 = lm(formula = balance ~ 1, data = df.poker)
fit2 = lm(formula = balance ~ 1 + hand, data = df.poker)
fit3 = lm(formula = balance ~ 1 + skill, data = df.poker)
fit4 = lm(formula = balance ~ 1 + hand + skill, data = df.poker)
fit5 = lm(formula = balance ~ 1 + hand * skill, data = df.poker)
```

```r
anova(fit1, fit2)
```

```
Analysis of Variance Table

Model 1: balance ~ 1
Model 2: balance ~ 1 + hand
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    299 7580.0
2    297 5020.6  2    2559.4 75.703 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```r
lm(formula = balance ~ hand * skill, data = df.poker) %>%
anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

$$F(2,\ 294) = \frac{MSQ_{\text{Factor}}}{MSQ_{\text{Residuals}}}$$

$$F(2,\ 297) = \frac{\text{PRE}/(\text{PA} - \text{PC})}{(1 - \text{PRE})/(n - \text{PA})}$$

```
Analysis of Variance Table

Model 1: balance ~ 1
Model 2: balance ~ 1 + hand
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    299 7580.0
2    297 5020.6  2    2559.4 75.703 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```r
lm(formula = balance ~ hand * skill, data = df.poker) %>%
anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```r
fit1 = lm(formula = balance ~ 1, data = df.poker)
fit2 = lm(formula = balance ~ 1 + hand, data = df.poker)
fit3 = lm(formula = balance ~ 1 + skill, data = df.poker)
fit4 = lm(formula = balance ~ 1 + hand + skill, data = df.poker)
fit5 = lm(formula = balance ~ 1 + hand * skill, data = df.poker)
```

```r
anova(fit1, fit3)
```

```
Analysis of Variance Table

Model 1: balance ~ 1
Model 2: balance ~ 1 + skill
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    299 7580.0
2    298 7540.6  1    39.349 1.5551 0.2134
```

```
lm(formula = balance ~ hand * skill, data = df.poker) %>%
anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
fit1 = lm(formula = balance ~ 1, data = df.poker)
fit2 = lm(formula = balance ~ 1 + hand, data = df.poker)
fit3 = lm(formula = balance ~ 1 + skill, data = df.poker)
fit4 = lm(formula = balance ~ 1 + hand + skill, data = df.poker)
fit5 = lm(formula = balance ~ 1 + hand * skill, data = df.poker)
```

```
anova(fit4, fit5)
```

```
Analysis of Variance Table

Model 1: balance ~ 1 + hand + skill
Model 2: balance ~ 1 + hand * skill
  Res.Df    RSS Df Sum of Sq     F    Pr(>F)
1    296 4981.2
2    294 4752.3  2    228.98 7.083 0.0009901 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Plan for today

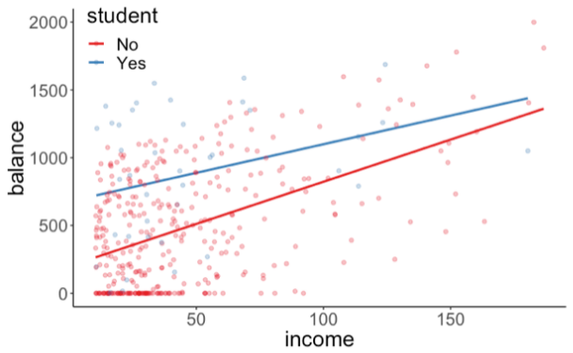
- Quick recap
- Unbalanced designs
- Linear contrasts
  - Testing specific hypotheses with linear contrasts
  - `emmeans` for handling linear contrasts in R
- Power analysis
  - Making decisions
  - Calculating power
  - Effect sizes
  - Determining sample size
- Power analysis via simulation
- Learn about more advanced simulation techniques in R
  - `map()`
  - list columns: `nest()`, `unnest()`

will share a 25 minute video lecture

Quick recap

Quick recap: Regressions with interactions

Interpretation



$$\widehat{\text{balance}}_i = 200.62 + 6.22 \cdot \text{income}_i + 476.68 \cdot \text{student}_i - 2.00 \cdot (\text{income}_i \times \text{student}_i)$$

if student = "No" $\widehat{\text{balance}}_i = 200.62 + 6.22 \cdot \text{income}_i$

if student = "Yes"

$$\begin{aligned}\widehat{\text{balance}}_i &= 200.62 + 6.22 \cdot \text{income}_i + 476.68 \cdot 1 - 2.00 \cdot (\text{income}_i \times 1) \\ &= 677.3 + 6.22 \cdot \text{income}_i - 2.00 \cdot \text{income}_i \\ &= 677.3 + 4.22 \cdot \text{income}_i\end{aligned}$$

`lm()` output

```
1 lm(formula = balance ~ income + student + income:student, data = df.credit) %>%
2   summary()
```

```
Call:
lm(formula = balance ~ income + student + income:student,
data = df.credit)

Residuals:
     Min       1Q   Median       3Q      Max
-773.39 -325.70  -41.13  321.65  814.04

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       200.6232    33.6984   5.953 5.79e-09 ***
income              6.2182     0.5921  10.502  < 2e-16 ***
studentYes        476.6758   104.3512   4.568 6.59e-06 ***
income:studentYes  -1.9992     1.7313  -1.155    0.249
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1

Residual standard error: 391.6 on 396 degrees of freedom
Multiple R-squared:  0.2799, Adjusted R-squared:  0.2744
F-statistic:  51.3 on 3 and 396 DF,  p-value: < 2.2e-16
```

?

```
1 fit_c = lm(formula = balance ~ student + income:student, data = df.credit)
2 fit_a = lm(formula = balance ~ income + student + income:student, data = df.credit)
3
4 anova(fit_c, fit_a)
```

```
1 fit_c = lm(formula = balance ~ income + student, data = df.credit)
2 fit_a = lm(formula = balance ~ income + student + income:student, data = df.credit)
3
4 anova(fit_c, fit_a)
```


`lm()` output

very important

```
Call:
lm(formula = balance ~ income + student + income:student,
data = df.credit)

Residuals:
     Min       1Q   Median       3Q      Max
-773.39 -325.70  -41.13  321.65  814.04

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)       200.6232    33.6984   5.953 5.79e-09 ***
income              6.2182     0.5921  10.502  < 2e-16 ***
studentYes        476.6758   104.3512   4.568 6.59e-06 ***
income:studentYes  -1.9992     1.7313  -1.155    0.249
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’
1

Residual standard error: 391.6 on 396 degrees of freedom
Multiple R-squared:  0.2799, Adjusted R-squared:  0.2744
F-statistic:  51.3 on 3 and 396 DF,  p-value: < 2.2e-16
```

what does this mean?

not the overall effect of income!

instead the predicted effect of income for non-students

we'll talk more about the difference between simple/conditional effects and main effects soon!

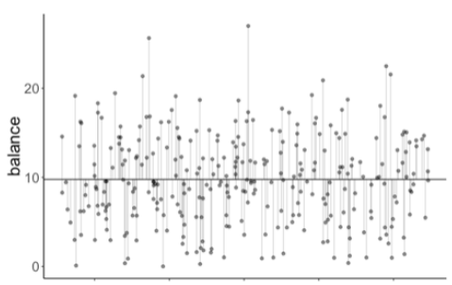
Quick recap: Categorial predictors

H_0: Card quality does not affect the final balance.

Model C

$$\text{balance}_i = \beta_0 + \epsilon_i$$

Model prediction



SSE(C) = 7580

Fitted model

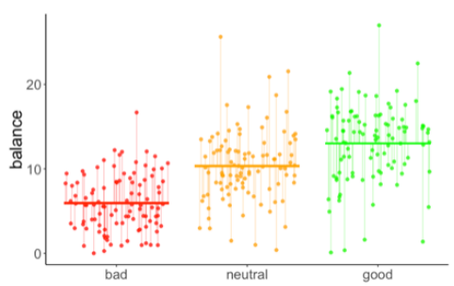
$$\widehat{\text{balance}}_i = 9.77$$

H_1: Card quality affects the final balance.

Model A

$$\text{balance}_i = \beta_0 + \beta_1 \text{hand\_neutral}_i + \beta_2 \text{hand\_good}_i + \epsilon$$

Model prediction



SSE(A) = 5021

Fitted model

$$\widehat{\text{balance}}_i = 5.94 + 4.41 \cdot \text{hand\_neutral}_i + 7.08 \cdot \text{hand\_good}_i$$

Analysis of variance

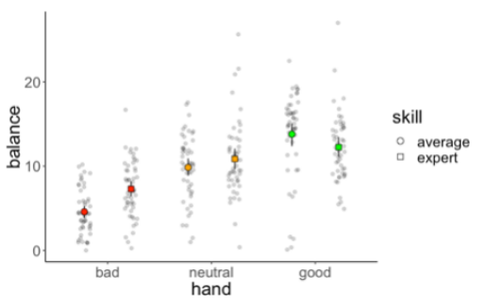
```
lm(formula = balance ~ hand * skill, data = df.poker) %>%
  anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

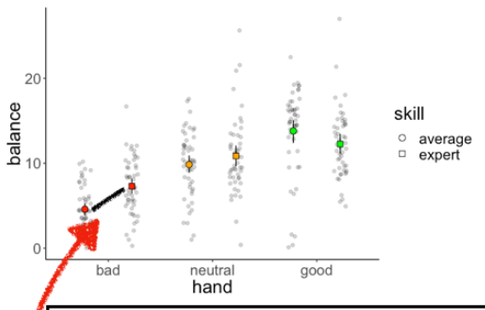
main effect of `hand`

no main effect of `skill`

interaction between `hand` and `skill`



Parameter interpretation

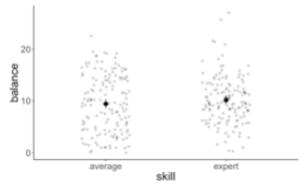



```
Call:
lm(formula = balance ~ hand * skill, data = df.poker)

Residuals:
     Min       1Q   Median       3Q      Max
-13.6976  -2.4740   0.0348   2.4644  14.7806

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)             4.5866     0.5686   8.067 1.85e-14 ***
handneutral             5.2572     0.8041   6.538 2.75e-10 ***
handgood                9.2110     0.8041  11.455  < 2e-16 ***
skillexpert             2.7098     0.8041   3.370 0.000852 ***
handneutral:skillexpert -1.7042    1.1372  -1.499 0.135038
handgood:skillexpert    -4.2522    1.1372  -3.739 0.000222 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.02 on 294 degrees of freedom
Multiple R-squared:  0.3731,  Adjusted R-squared:  0.3624
F-statistic: 34.99 on 5 and 294 DF,  p-value: < 2.2e-16
```

```
lm(formula = balance ~ hand * skill,
   data = df.poker) %>%
  anova()
```

```
Analysis of Variance Table

Response: balance
            Df Sum Sq Mean Sq F value    Pr(>F)
hand         2 2559.4 1279.70 79.1692 < 2.2e-16 ***
skill        1   39.3   39.35  2.4344 0.1197776
hand:skill   2  229.0  114.49  7.0830 0.0009901 ***
Residuals  294 4752.3   16.16
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

there was no main effect of skill!

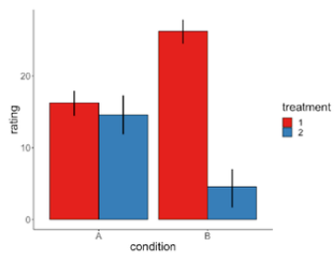


is this difference significantly different from 0?

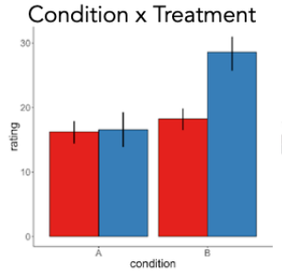
hand	average	expert	difference
bad	4.59	7.3	2.71

Solution

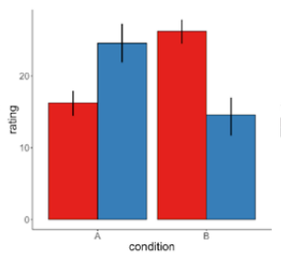
Treatment
Condition x Treatment



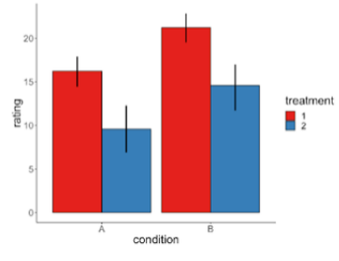
Condition,
Treatment,
Condition x Treatment



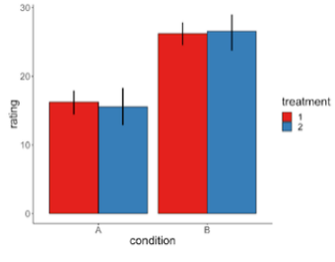
Condition x Treatment



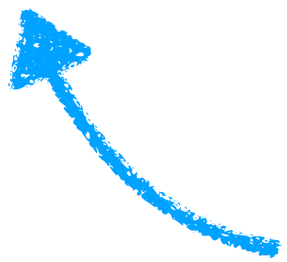
Condition
Treatment



Condition



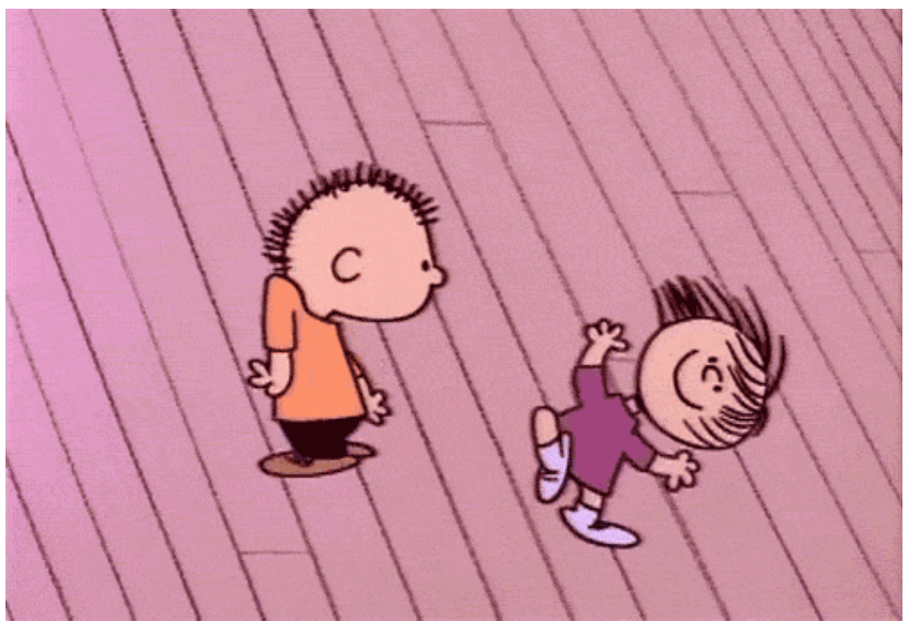
Unbalanced designs



not the same number of participants in each cell

ANOVA

- for all the examples so far, I've assumed a balanced design (i.e. the same number of observations in each of the different factor levels)
- things get funky when we have an unbalanced design



https://towardsdatascience.com/anovas-three-types-of-estimating-sums-of-squares-don-t-make-the-wrong-choice-91107c77a27a

Beware of unbalanced designs

```
1 lm(formula = balance ~ skill + hand, data = df.poker.unbalanced) %>%
2   anova()
```

```
Analysis of Variance Table

Response: balance
           Df Sum Sq Mean Sq F value  Pr(>F)
skill       1   74.3   74.28  4.2904 0.03922 *
hand        2 2385.1 1192.57 68.8827 < 2e-16 ***
Residuals 286 4951.5   17.31
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

flipped the order

```
1 lm(formula = balance ~ hand + skill, data = df.poker.unbalanced) %>%
2   anova()
```

```
Analysis of Variance Table

Response: balance
           Df Sum Sq Mean Sq F value Pr(>F)
hand        2 2419.8 1209.92 69.8845 <2e-16 ***
skill       1   39.6   39.59  2.2867 0.1316
Residuals 286 4951.5   17.31
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

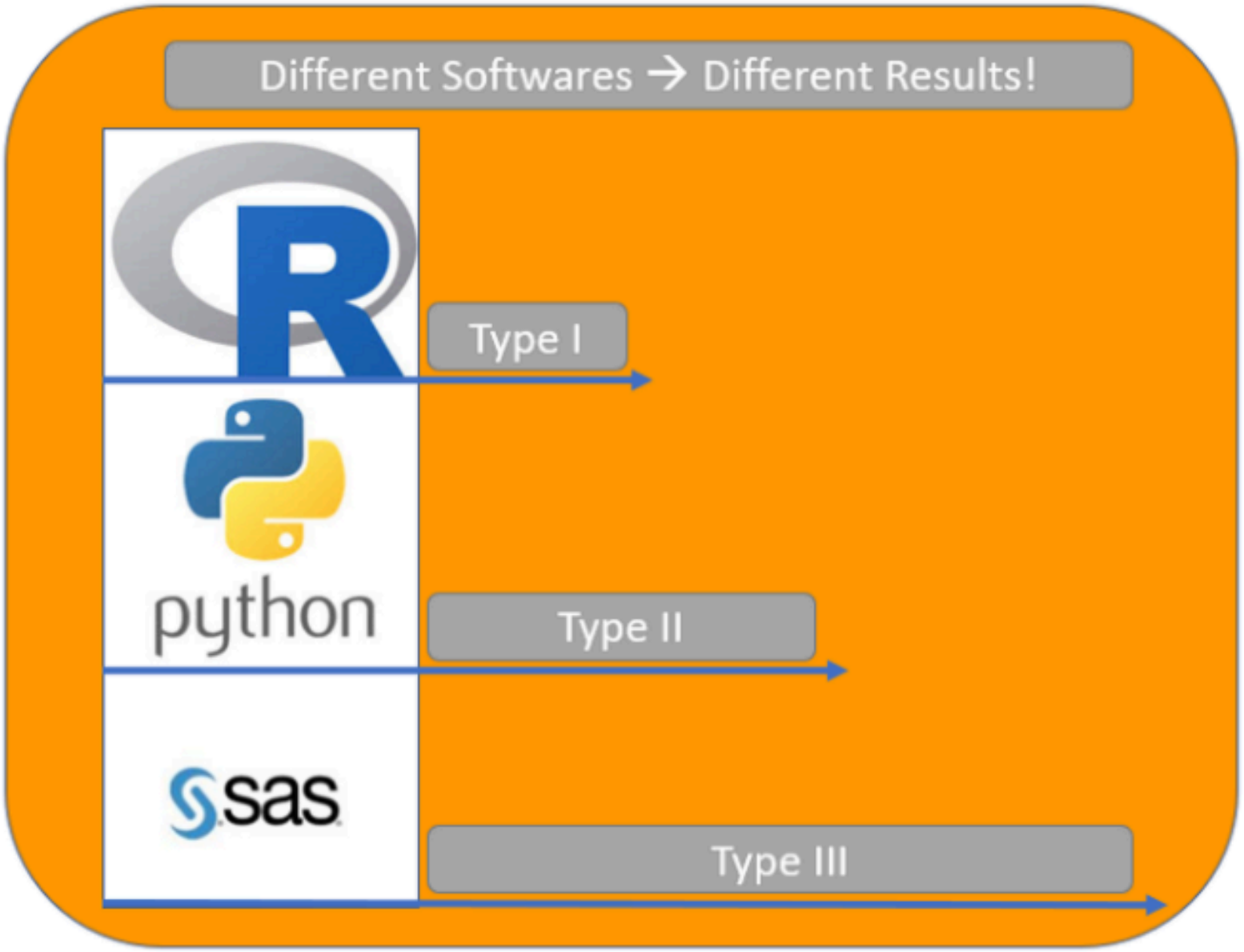
The different sums of squares

Three different methodologies for splitting variation exist: Type I, Type II and Type III Sums of Squares. They do not give the same result in case of unbalanced data.

Type I, Type II and Type III ANOVA have different outcomes!

https://towardsdatascience.com/anovas-three-types-of-estimating-sums-of-squares-don-t-make-the-wrong-choice-91107c77a27a

Default sums of squares ...



Default Types of Sums of Squares for different programming languages

not great for reproducibility ...

Type I Sums of Squares

Type I Sums of Squares are Sequential, so the order of variables in the models makes a difference. This is rarely what we want in practice!

Sums of Squares are Mathematically defined as:

- SS(A) for independent variable A
- SS(B | A) for independent variable B
- SS(AB | B, A) for the interaction effect

caution: this is what `anova()` uses by default

https://towardsdatascience.com/anovas-three-types-of-estimating-sums-of-squares-don-t-make-the-wrong-choice-91107c77a27a

Type II Sums of Squares

Type II Sums of Squares should be used if there is no interaction between the independent variables.

Sums of Squares are Mathematically defined as:

- SS(A | B) for independent variable A
- SS(B | A) for independent variable B
- No interaction effect

however, often not used in practice ...
(mostly because we are interested in interaction effects)

Type III Sums of Squares

The Type III Sums of Squares are also called partial sums of squares again another way of computing Sums of Squares:

- Like Type II, the Type III Sums of Squares are not sequential, so the order of specification does not matter.
- Unlike Type II, the Type III Sums of Squares do specify an interaction effect.

Sums of Squares are Mathematically defined as:

- SS(A | B, AB) for independent variable A
- SS(B | A, AB) for independent variable B

this is the default in the literature (e.g. SPSS uses it)

Route I: Using "afex"

```r
library("afex")

fit = aov_ez(id = "participant",
             dv = "balance",
             data = df.poker.unbalanced,
             between = c("hand", "skill"))
fit$Anova
```

```
Contrasts set to contr.sum for the following variables: hand, skill
Anova Table (Type III tests)

Response: dv
             Sum Sq  Df   F value    Pr(>F)
(Intercept) 27781.3   1 1676.9096 < 2.2e-16 ***
hand         2285.3   2   68.9729 < 2.2e-16 ***
skill          48.9   1    2.9540 0.0867525 .
hand:skill    246.5   2    7.4401 0.0007089 ***
Residuals    4705.0 284
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Route II: Using "emmeans"

preferred route!!

```
1 library("emmeans")
2
3 lm(formula = balance ~ hand + skill,
4     data = df.poker.unbalanced) %>%
5   joint_tests()
```

very handy function

model term	df1	df2	F.ratio	p.value
hand	2	284	68.973	<.0001
skill	1	284	2.954	0.0868
hand:skill	2	284	7.440	0.0007

Same same ...

Route I: Using "afex"

```
1 library("afex")
2
3 fit = aov_ez(id = "participant",
4               dv = "balance",
5               data = df.poker.unbalanced,
6               between = c("hand", "skill"))
7 fit$Anova
```

```
Contrasts set to contr.sum for the following variables: hand, skill
Anova Table (Type III tests)

Response: dv
             Sum Sq  Df   F value    Pr(>F)
(Intercept) 27781.3   1 1676.9096 < 2.2e-16 ***
hand         2285.3   2   68.9729 < 2.2e-16 ***
skill          48.9   1    2.9540 0.0867525 .
hand:skill    246.5   2    7.4401 0.0007089 ***
Residuals    4705.0 284
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Route II: Using "emmeans"

preferred route!!

```
1 library("emmeans")
2
3 lm(formula = balance ~ hand + skill,
4    data = df.poker.unbalanced) %>%
5   joint_tests()
```

very handy function

```
model term df1 df2 F.ratio p.value
 hand         2 284   68.973  <.0001
 skill        1 284    2.954  0.0868
 hand:skill   2 284    7.440  0.0007
```

... but different

can come apart when we deal with repeated observations, but we'll deal with that later!

Unbalanced design

- There are different kinds of ANOVAs, for which the sums of squares are calculated differently.
- This makes a difference when we have an unbalanced design (i.e. the number of participants is not the same for each cell in our design).

`joint_tests()` is your friend!

`anova()` function

- model comparison
- compare a compact with an augmented model to see whether the proportional reduction in error is worth it

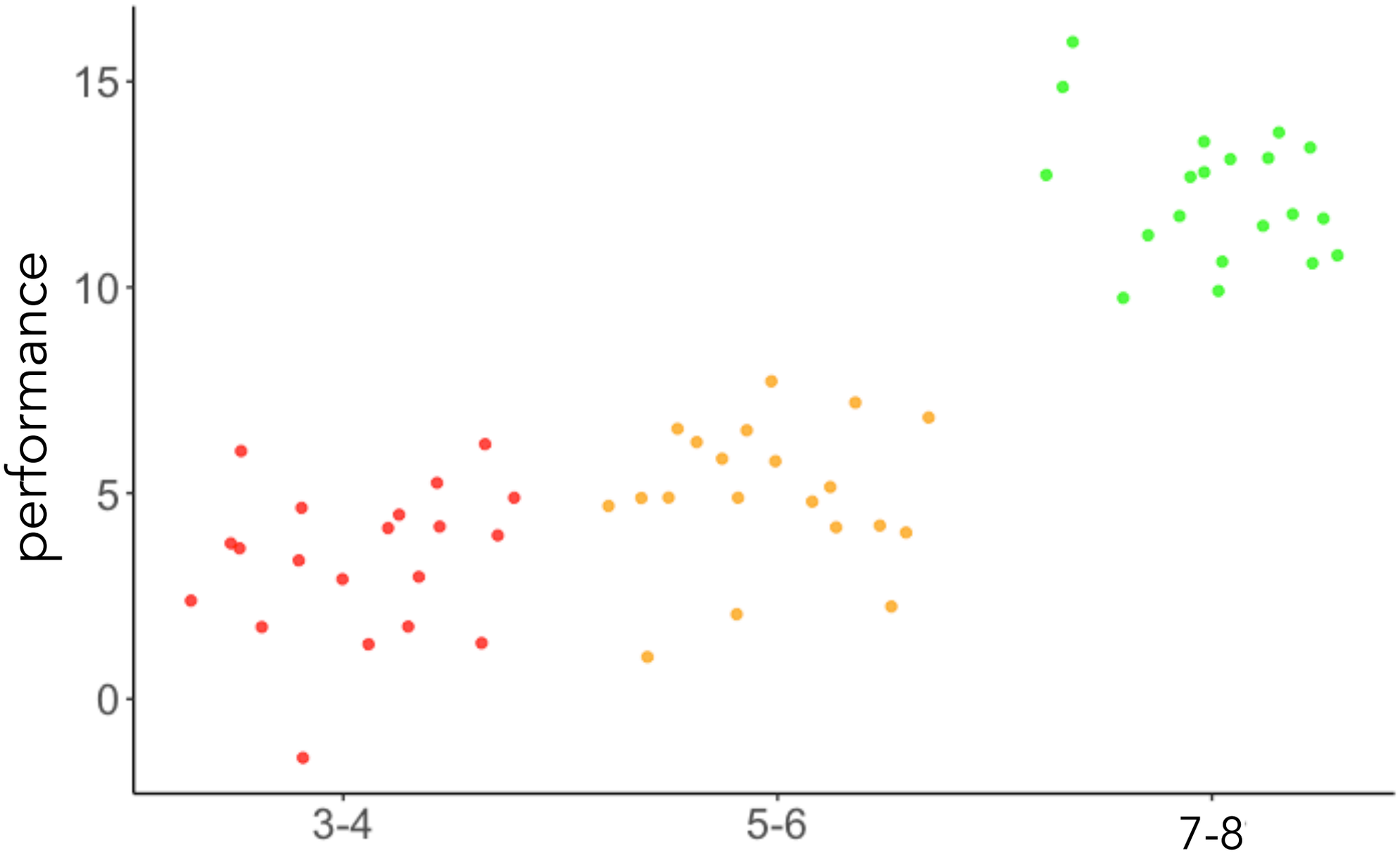
- run an ANOVA to get main effects and interactions effects when you have categorical predictors as variables

Linear contrasts

Testing (more) specific hypotheses with linear contrasts

Contrasts

Does performance increase with age?



Data from a hypothetical developmental study

Does performance increase with age?

ANOVA

Does performance differ between age groups?

post-hoc tests

3-4 vs. **5-6**

5-6 vs. **7-8**

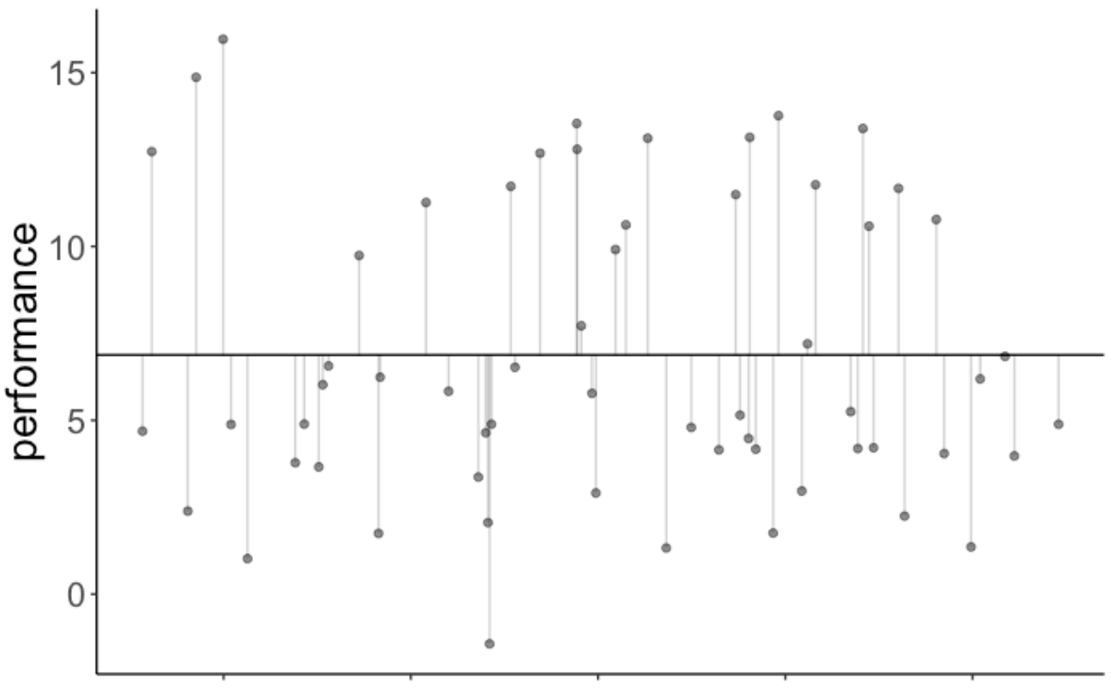
Is there are more direct way of asking this question with a statistical model?

Contrasts

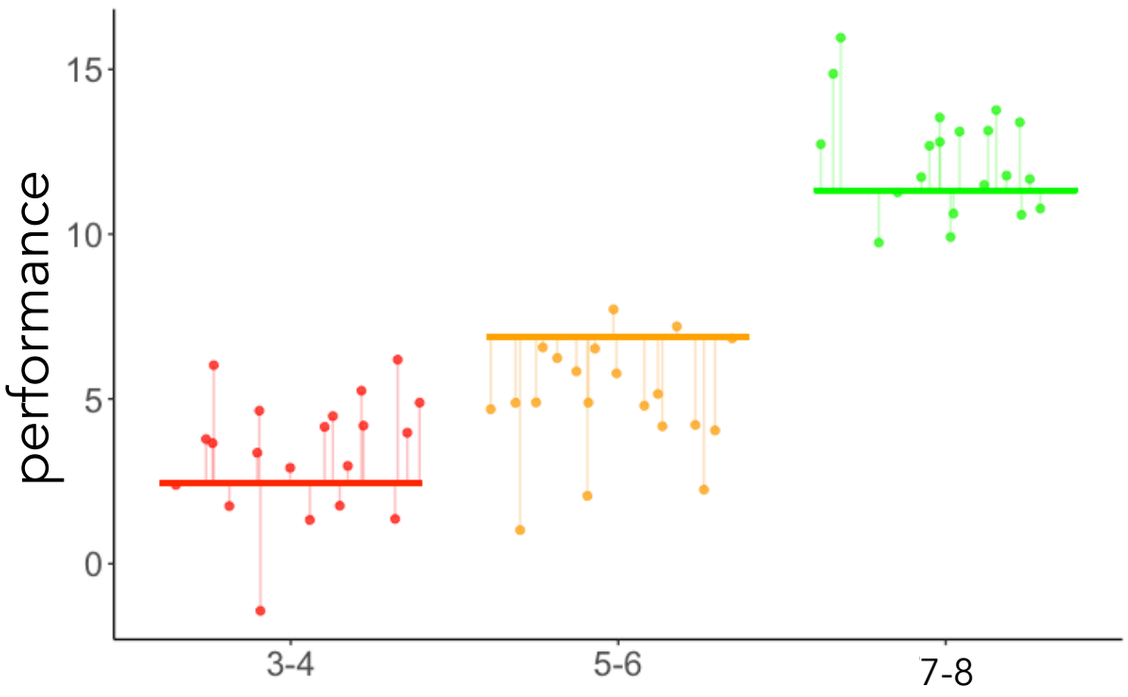
Does performance increase with age?

```
contrasts = c(-1, 0, 1)
```

Compact model



Augmented model

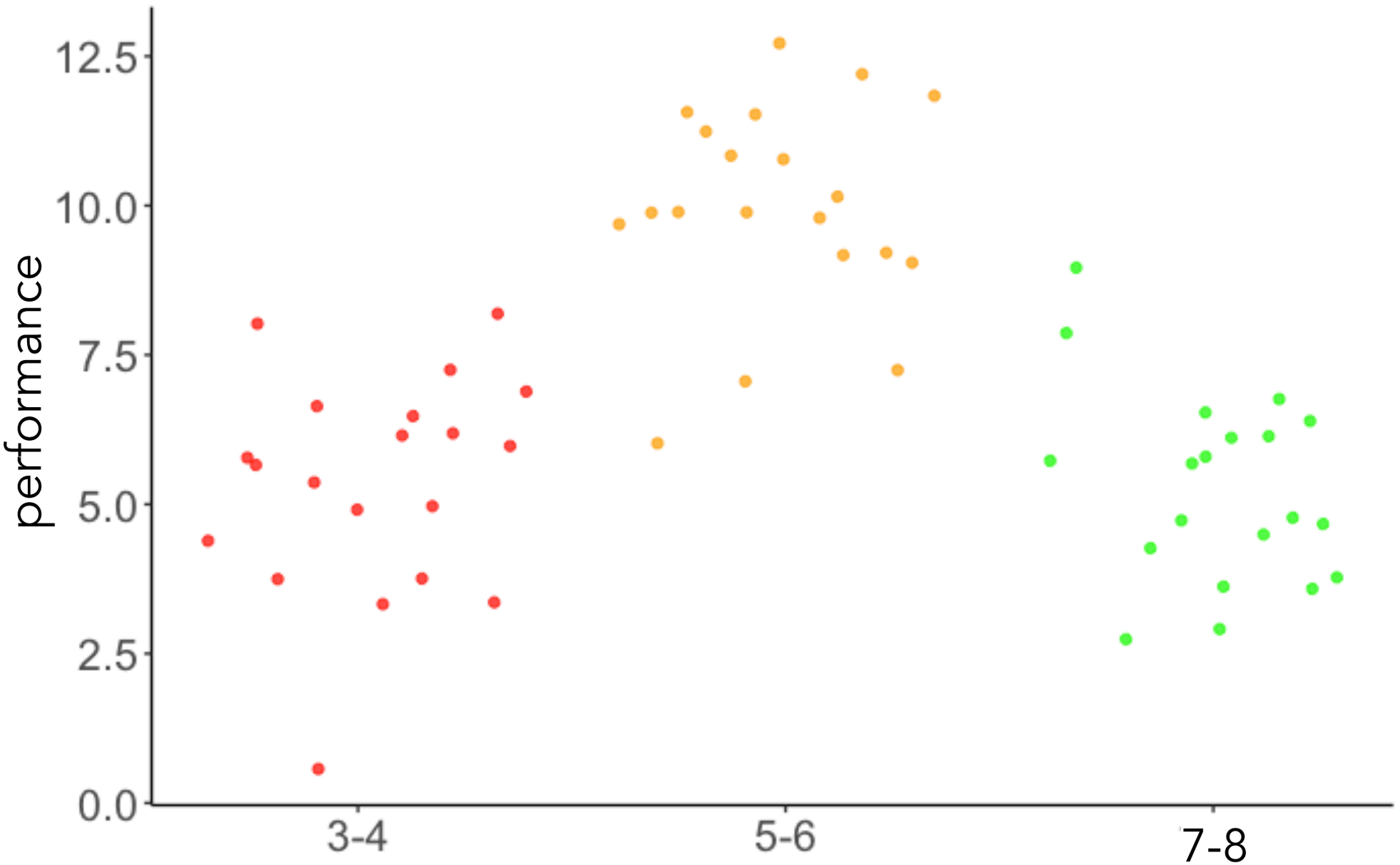


Model comparison

p < .001

Contrasts

Does performance increase with age?



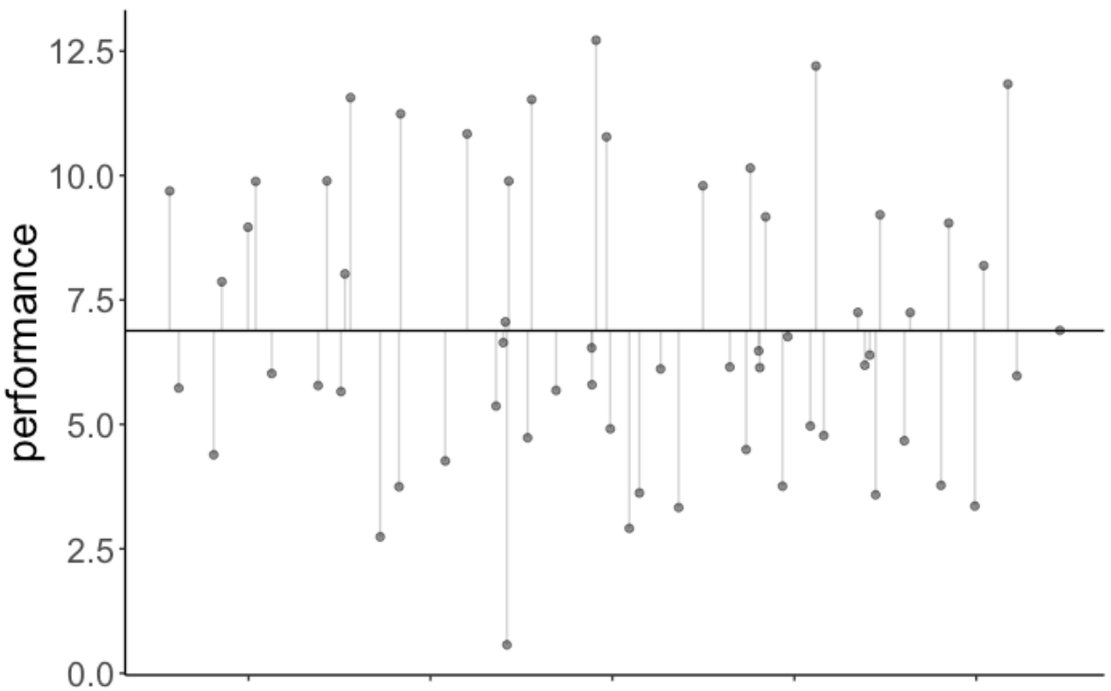
Data from another hypothetical developmental study

Contrasts

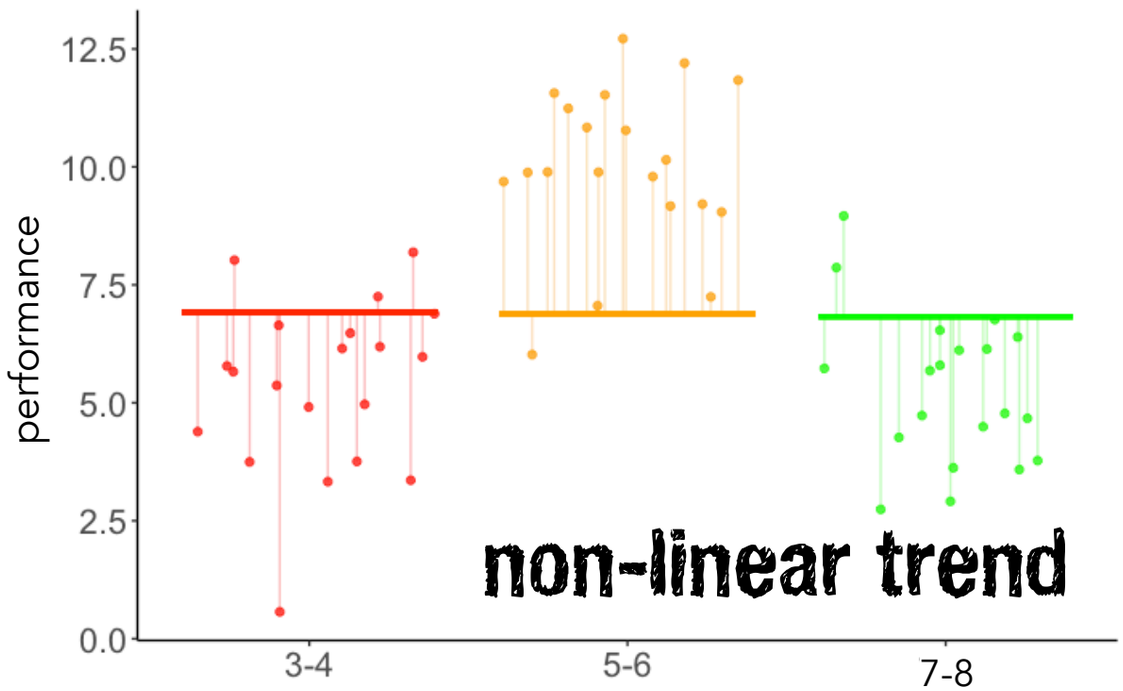
Does performance increase with age?

```
contrasts = c(-1, 0, 1)
```

Compact model



Augmented model



Model comparison

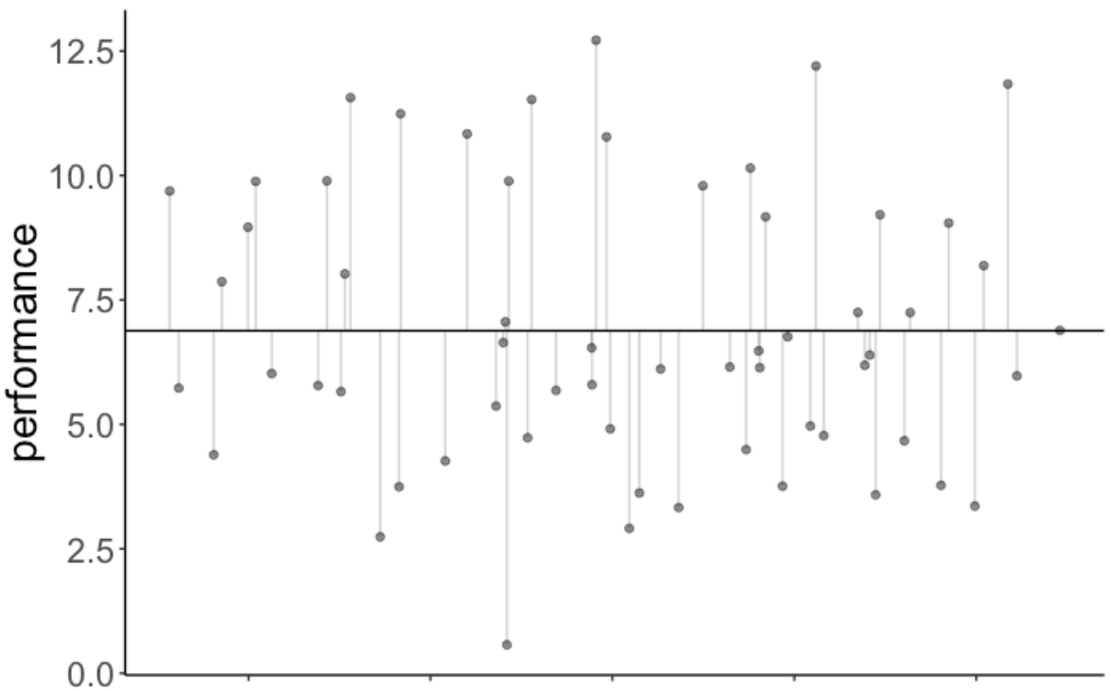
p = .8508

Contrasts

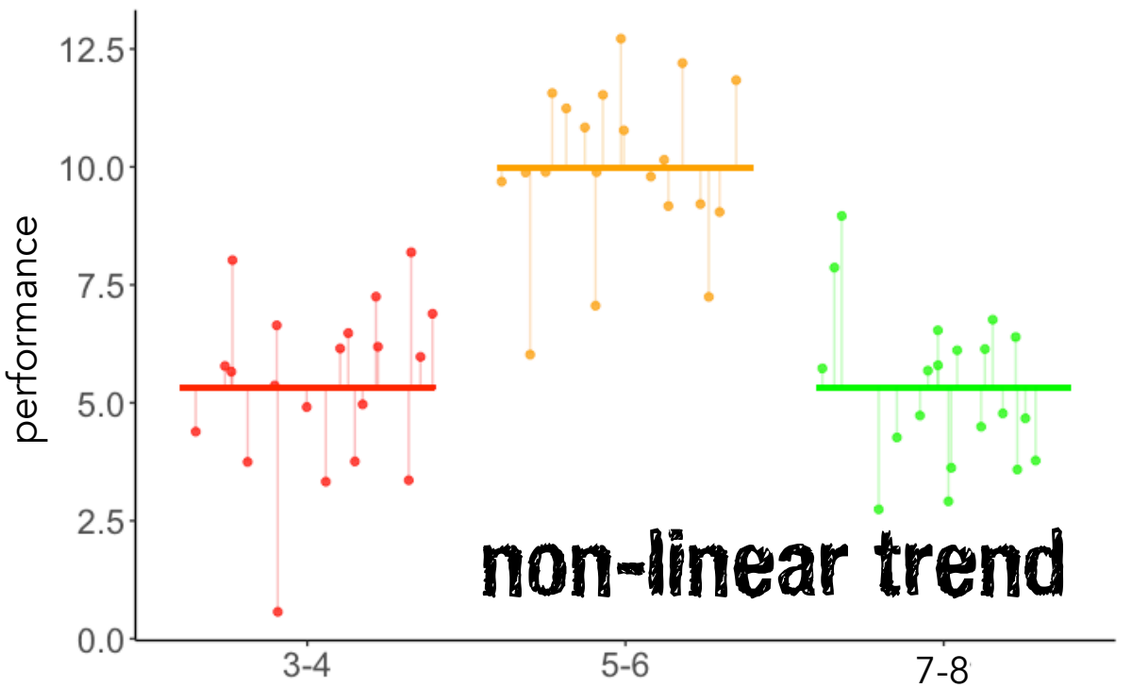
Does performance increase with age?

```
contrasts = c(-1, 2, -1)
```

Compact model



Augmented model



Model comparison

p < .001

`emmeans` for handling linear contrasts in R

Linear contrasts

~~How to use contrasts in R~~

In short: don't bother.[1]

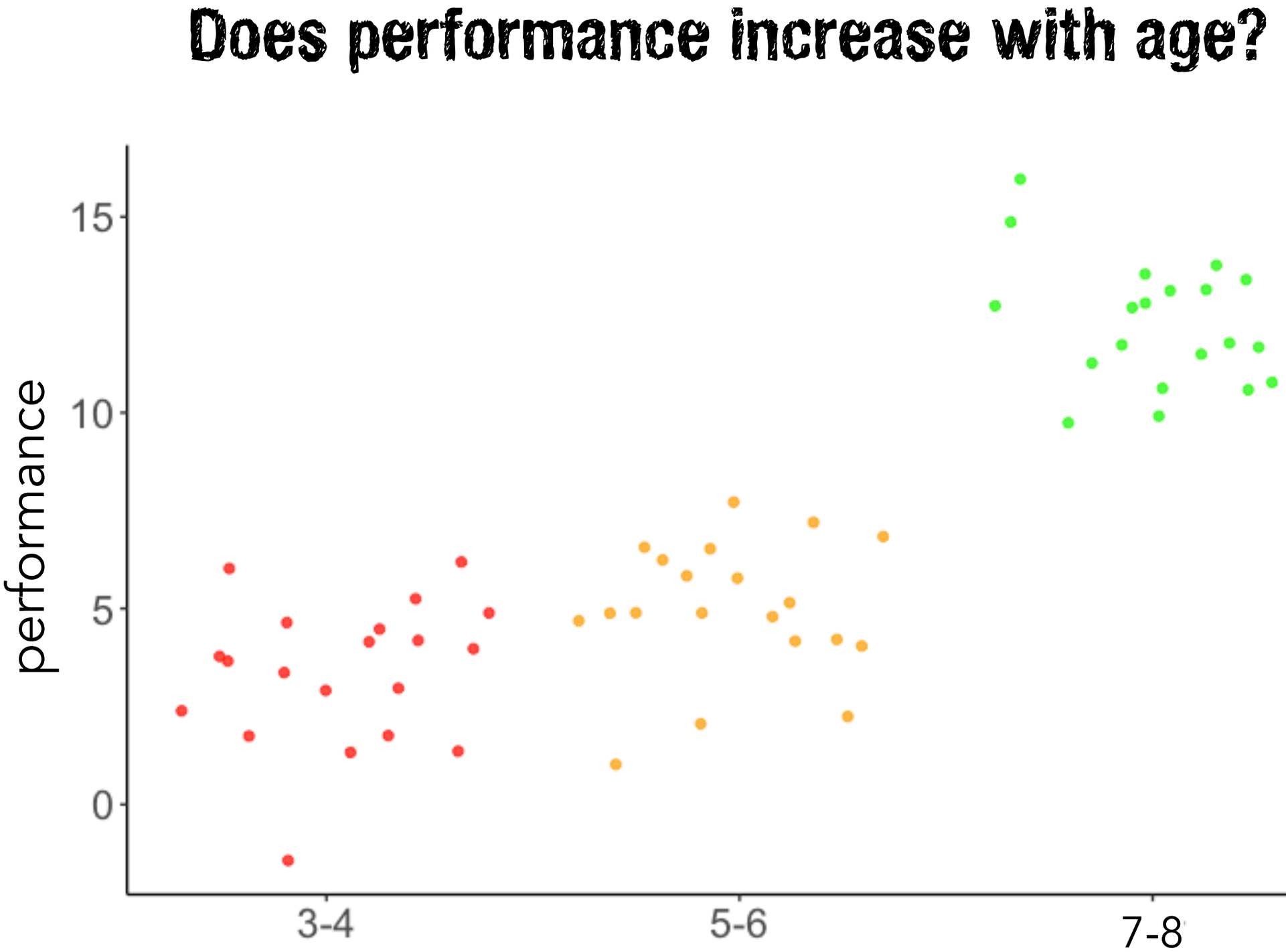
Like many before me, one of my stats classes technically "taught" me contrasts. But I didn't get the point and using them was cumbersome, so I promptly ignored them for years.

Luckily for me, someone came along and fixed the situation: `emmeans`. `emmeans` frames contrasts as a question you pose to a model: you can ask for all pairwise comparisons and get back that. `lm` and `summary` treat the same problem as fitting abstract coefficients, and you are left to answer your own question.

`emmeans` works with `lm`, `glm`, and the Bayesian friends in `brms` and `rstanarm`, so the process is applicable no matter the tool.

And you don't have to learn (much) about contrasts to take advantage of it.

Contrasts



Data from a hypothetical developmental study

Linear contrasts in R

```r
library("emmeans") # for calculating contrasts

#  fit the linear model
fit = lm(formula = performance ~ group,
         data = df.development)
```

fit linear model

Linear contrasts in R

```r
library("emmeans") # for calculating contrasts

#  fit the linear model
fit = lm(formula = performance ~ group,
         data = df.development)

# check factor levels
levels(df.development$group)
```

```
[1] "3-4" "5-6" "7-8"
```

check factor levels before defining contrasts

Linear contrasts in R

```r
library("emmeans") # for calculating contrasts

#  fit the linear model
fit = lm(formula = performance ~ group,
         data = df.development)

# check factor levels
levels(df.development$group) [1] "3-4" "5-6" "7-8"

# define the contrasts of interest
contrasts = list(young_vs_old = c(-0.5, -0.5, 1),
                 three_vs_five = c(-0.5, 0.5, 0))
```

set up linear contrasts

Linear contrasts in R

```r
library("emmeans") # for calculating contrasts

#  fit the linear model
fit = lm(formula = performance ~ group,
         data = df.development)

# check factor levels
levels(df.development$group) [1] "3-4" "5-6" "7-8"

# define the contrasts of interest
contrasts = list(young_vs_old = c(-0.5, -0.5, 1),
                 three_vs_five = c(-0.5, 0.5, 0))

# compute significance test on contrasts
fit %>%
  emmeans("group",
          contr = contrasts,
          adjust = "bonferroni") %>%
  pluck("contrasts")
```

compute the results

```
[1] "3-4" "5-6" "7-8"
 contrast       estimate        SE df t.ratio p.value
 young_vs_old  16.093541 0.4742322 57  33.936  <.0001
 three_vs_five  1.606009 0.5475962 57   2.933  0.0097

P value adjustment: bonferroni method for 2 tests
```

Linear contrasts in R

```r
library("emmeans") # for calculating contrasts

#  fit the linear model
fit = lm(formula = performance ~ group,
         data = df.development)

# check factor levels
levels(df.development$group)

# define the contrasts of interest
contrasts = list(young_vs_old = c(-1, -1, 2),
                 three_vs_five = c(-1, 1, 0))

# compute significance test on contrasts
fit %>%
  emmeans("group",
          contr = contrasts,
          adjust = "bonferroni") %>%
  pluck("contrasts")
```

```
[1] "3-4" "5-6" "7-8"
 contrast      estimate    SE df t.ratio p.value
 young_vs_old    32.187 0.948 57 33.936  <.0001
 three_vs_five    0.803 0.274 57  2.933  0.0097

P value adjustment: bonferroni method for 2 tests
```

hypothesis tests are the same!

Defining contrasts

- groups that we don't want to include in the comparison get a 0
- groups that we want to compare with one another should sum to 0
- this also means that all the contrasts together should sum to 0

Example:

```
contrasts = list(young_vs_old = c(-1, -1, 2),
                 three_vs_five = c(-1, 1, 0))
```

Post hoc tests

```r
fit = lm(formula = performance ~ group,
         data = df.development)

# pairwise differences between all the groups
fit %>%
  emmeans(pairwise ~ group) %>%
  pluck("contrasts")
```

all pairwise tests between groups

```
contrast    estimate        SE df t.ratio p.value
 3-4 - 5-6  -1.606009 0.5475962 57  -2.933  0.0145
 3-4 - 7-8 -16.896546 0.5475962 57 -30.856  <.0001
 5-6 - 7-8 -15.290537 0.5475962 57 -27.923  <.0001

P value adjustment: bonferroni method for 3 tests
```

Post hoc tests

```r
# fit the model
fit = lm(formula = balance ~ hand + skill,
         data = df.poker)

# post hoc tests
fit %>%
  emmeans(pairwise ~ hand + skill,
          adjust = "bonferroni") %>%
  pluck("contrasts")
```

the poker data

contrast	estimate	SE	df	t.ratio	p.value
bad,average - neutral,average	-4.381023	0.6051766	286	-7.239	<.0001
bad,average - good,average	-7.060823	0.6051766	286	-11.667	<.0001
bad,average - bad,expert	-0.740385	0.4896119	286	-1.512	1.0000
bad,average - neutral,expert	-5.121408	0.7611327	286	-6.729	<.0001
bad,average - good,expert	-7.801208	0.7611327	286	-10.249	<.0001
neutral,average - good,average	-2.679800	0.5884403	286	-4.554	0.0001
neutral,average - bad,expert	3.640638	0.7953578	286	4.577	0.0001
neutral,average - neutral,expert	-0.740385	0.4896119	286	-1.512	1.0000
neutral,average - good,expert	-3.420185	0.7654945	286	-4.468	0.0002
good,average - bad,expert	6.320438	0.7953578	286	7.947	<.0001
good,average - neutral,expert	1.939415	0.7654945	286	2.534	0.1774
good,average - good,expert	-0.740385	0.4896119	286	-1.512	1.0000
bad,expert - neutral,expert	-4.381023	0.6051766	286	-7.239	<.0001
bad,expert - good,expert	-7.060823	0.6051766	286	-11.667	<.0001
neutral,expert - good,expert	-2.679800	0.5884403	286	-4.554	0.0001

P value adjustment: bonferroni method for 15 tests

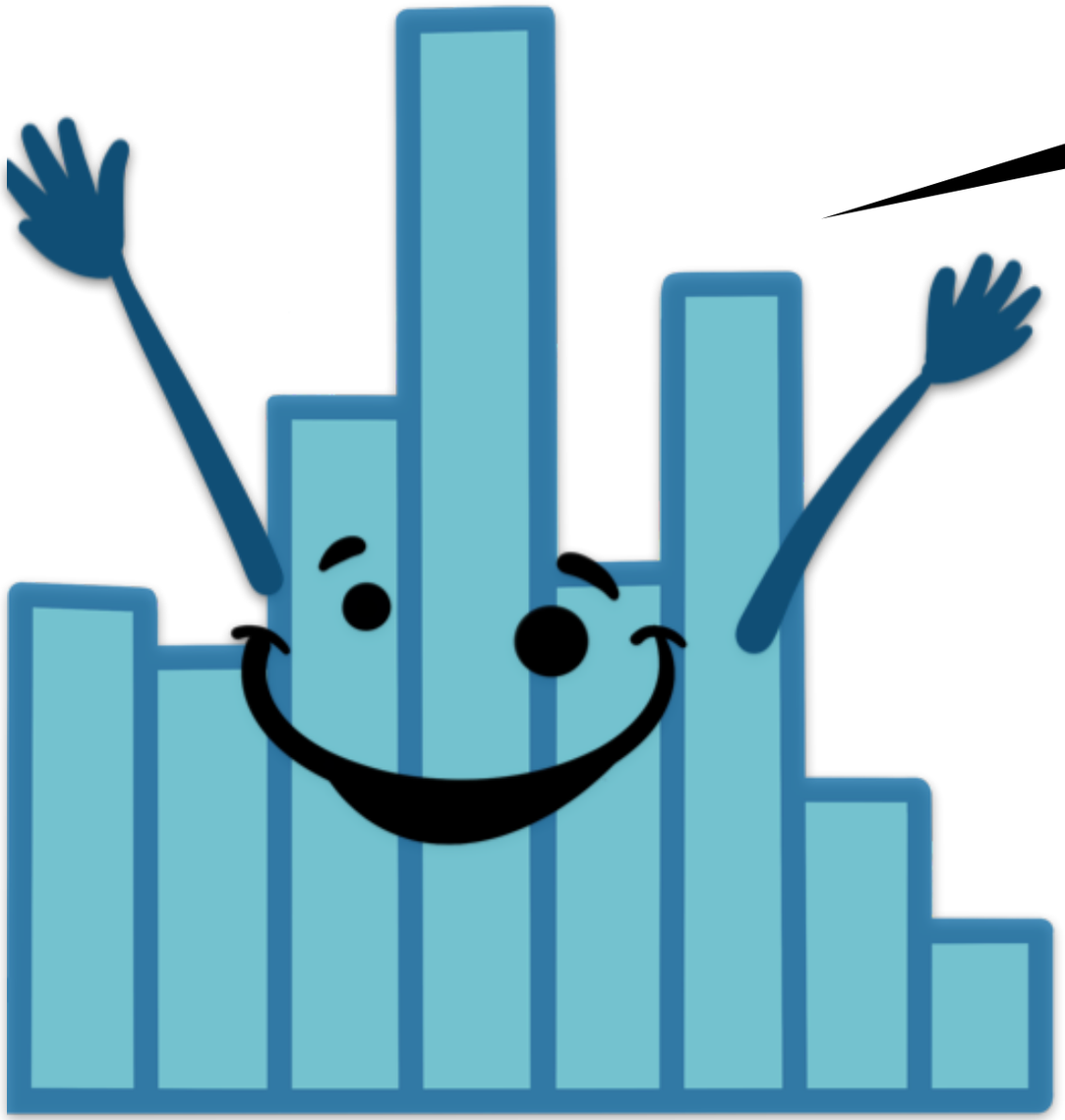
all pairwise tests between groups

that's a lot of tests!

... not

Contrasts

- linear contrasts allow us to ask more specific questions of our data
- rather than asking whether any of the group means are significantly different from each other (ANOVA), we can ask questions such as:
  - Does performance increase with age?
  - Is the overall performance in Condition B and C better from the performance in Condition A?



02:00

stretch break!

Power analysis

Making decisions

Type I Error

Type II Error

H_0: Not pregnant. H_1: Pregnant.

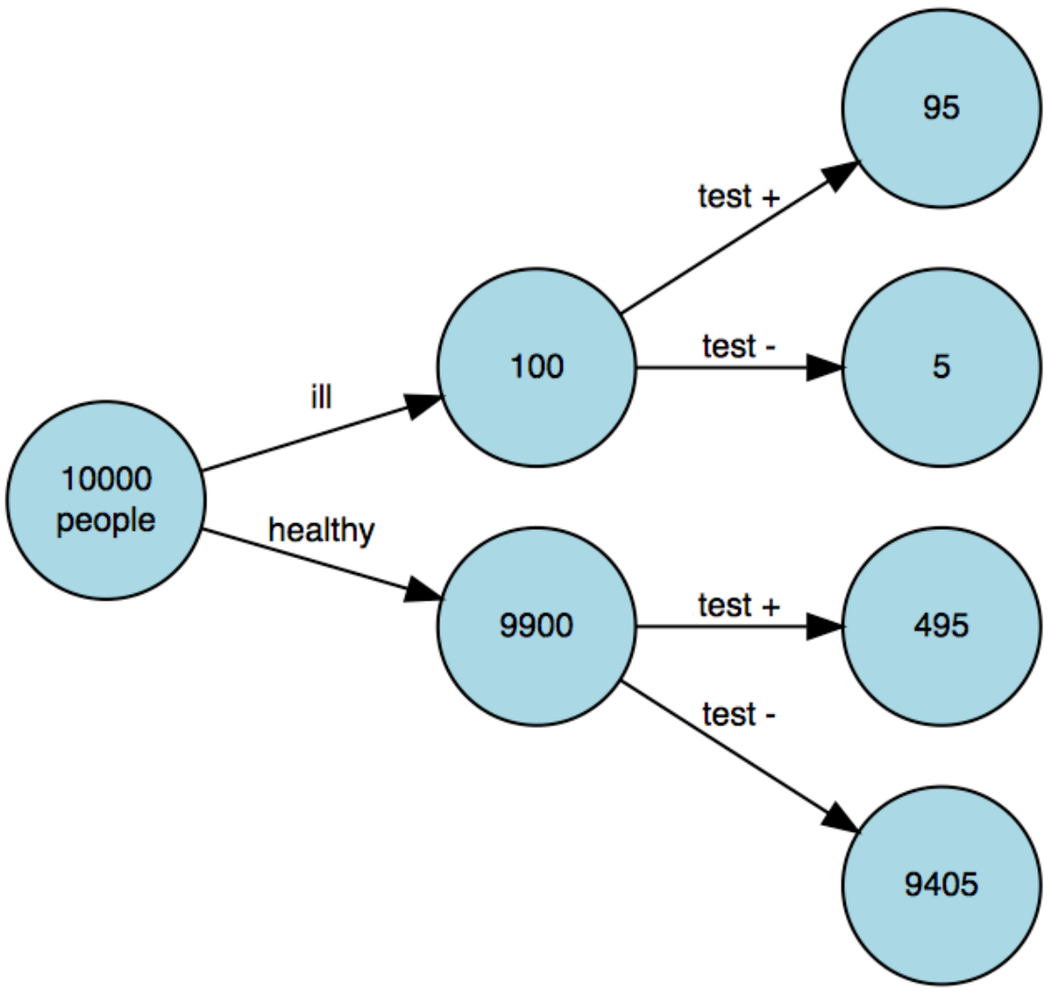
Type I Error: Falsely rejecting the null hypothesis (even though it is true).

Type II Error: Failing to reject the null hypothesis (even though it is false).

Clue guide to probability

H_0: The person is healthy.

H_1: The person is ill.



true positive

Sensitivity

$p(\text{reject } H_0 \mid H_1 \text{ is true})$

Power

$1 - \beta$

---

false negative

Type II error

$p(\text{not reject } H_0 \mid H_1 \text{ is true})$

β

---

false positive

Type I error

$p(\text{reject } H_0 \mid H_0 \text{ is true})$

α

---

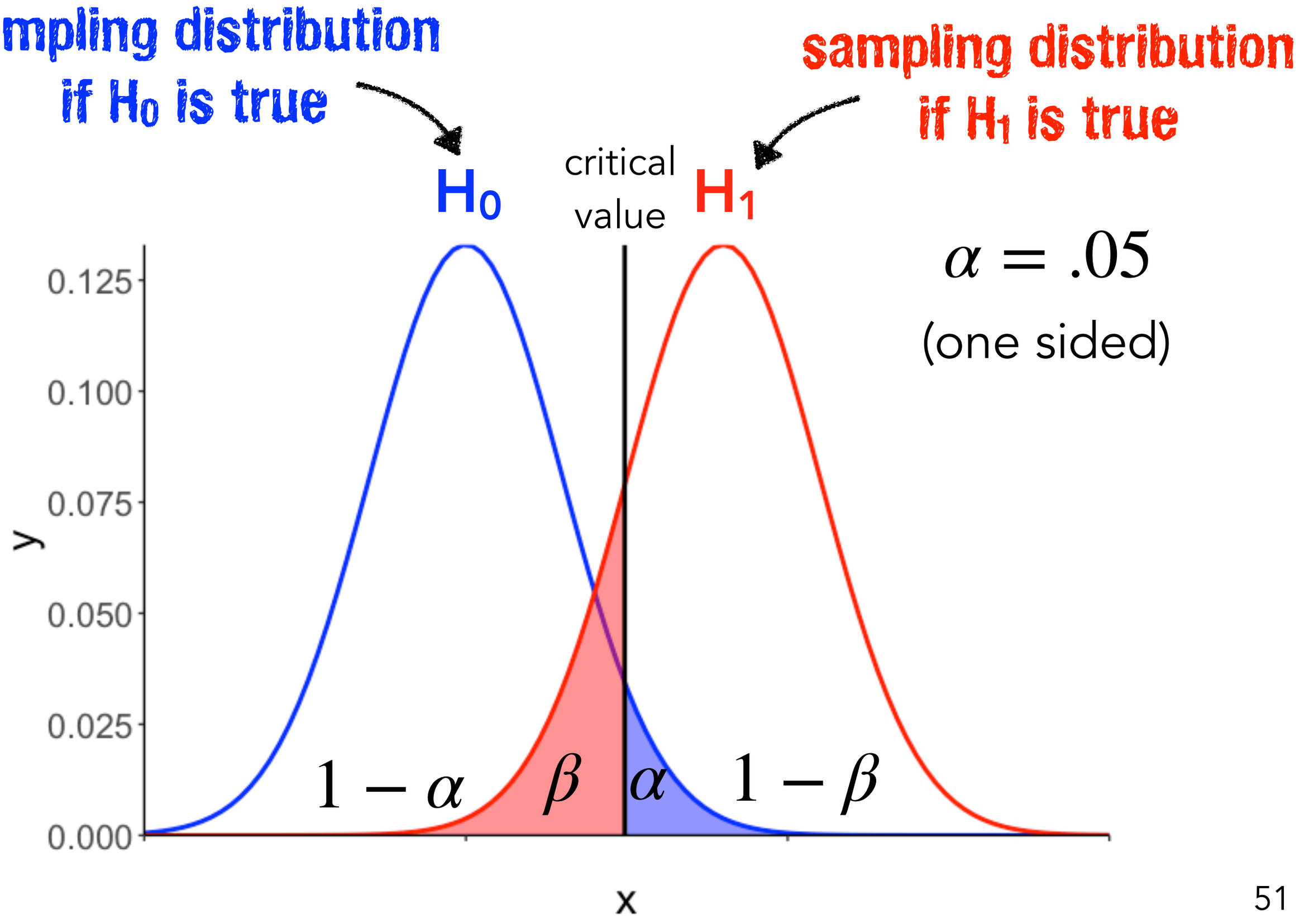
true negative

$p(\text{not reject } H_0 \mid H_0 \text{ is true})$

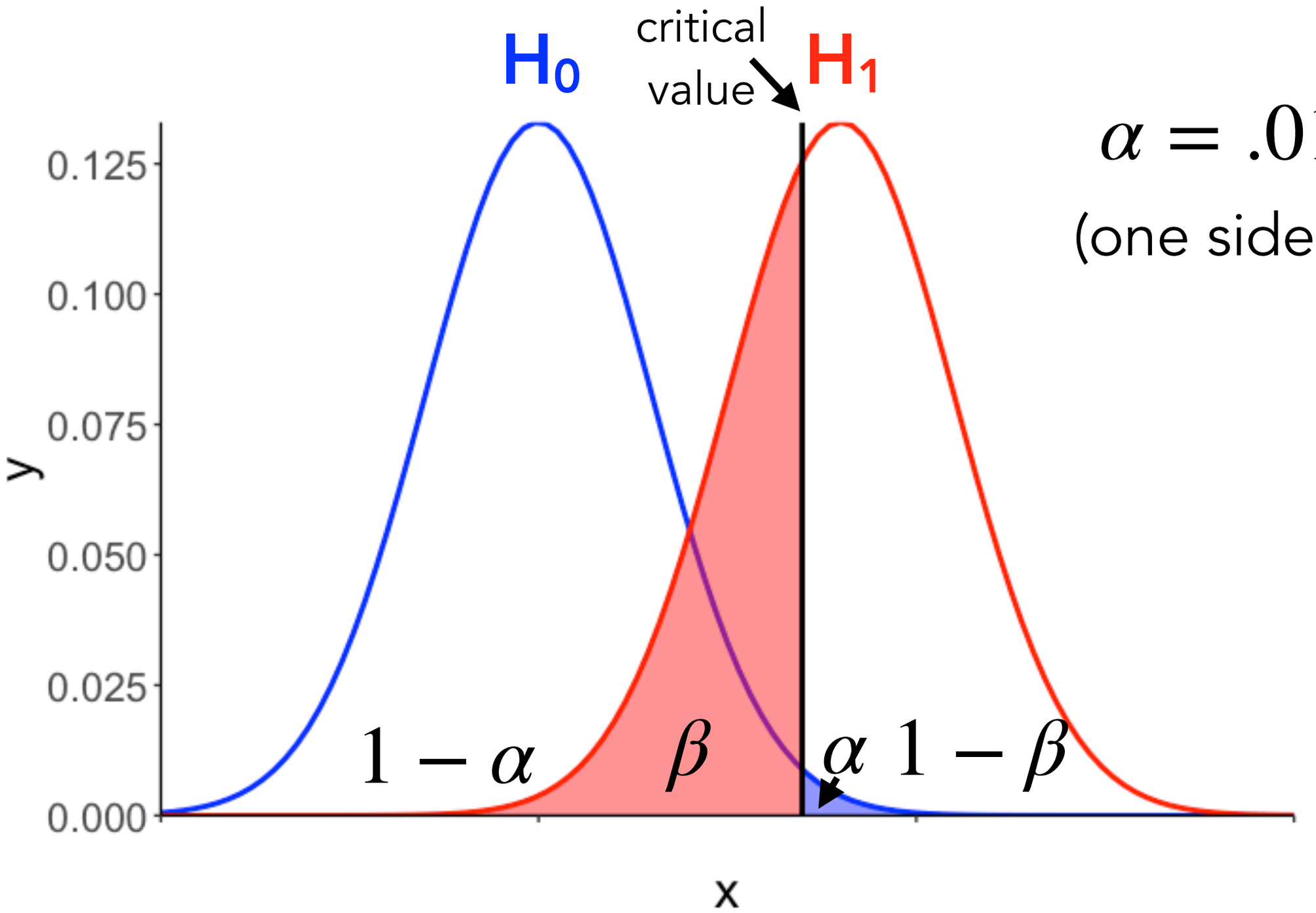
Specificity

$1 - \alpha$

What affects power?



What affects power?

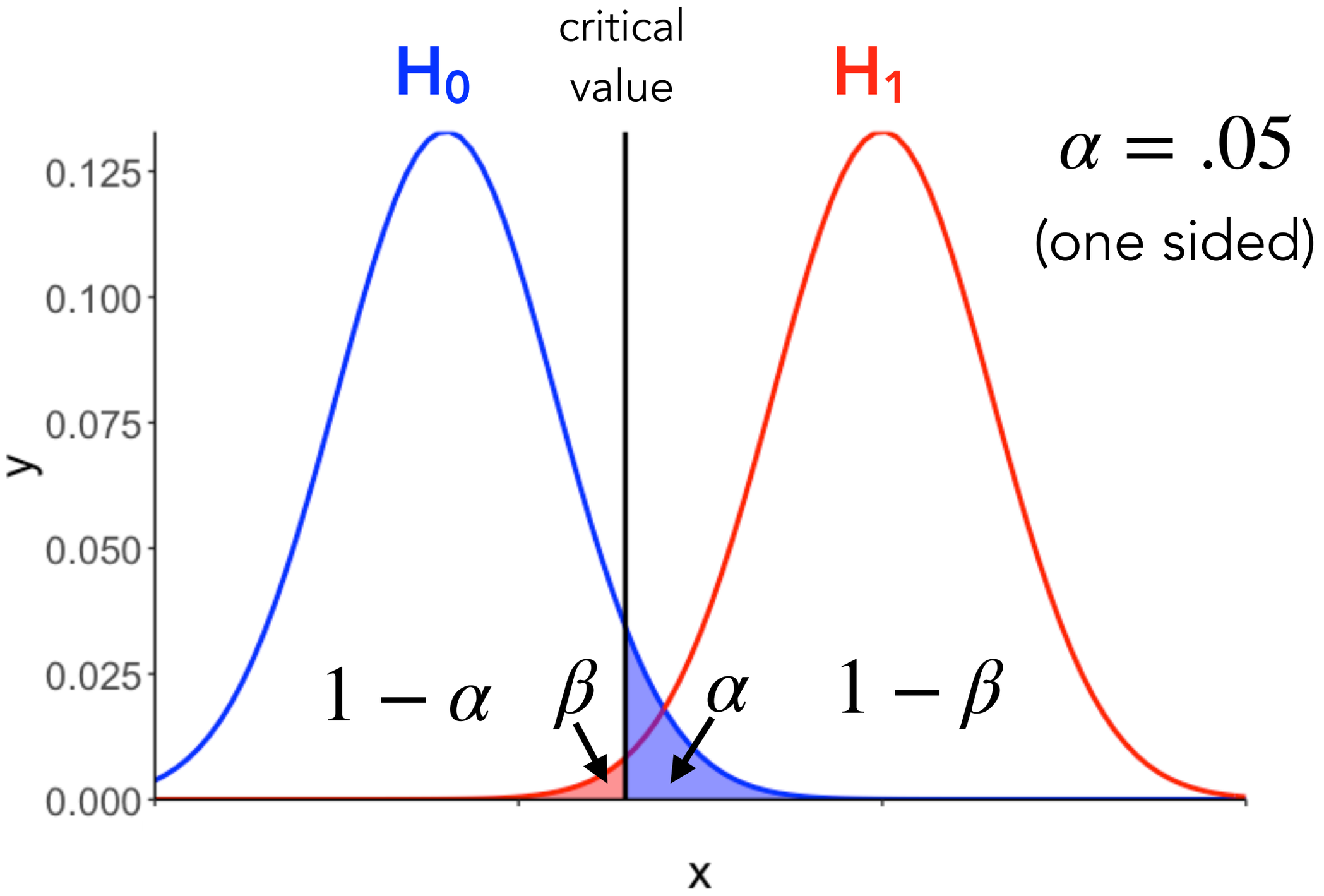


$\alpha = .01$

(one sided)

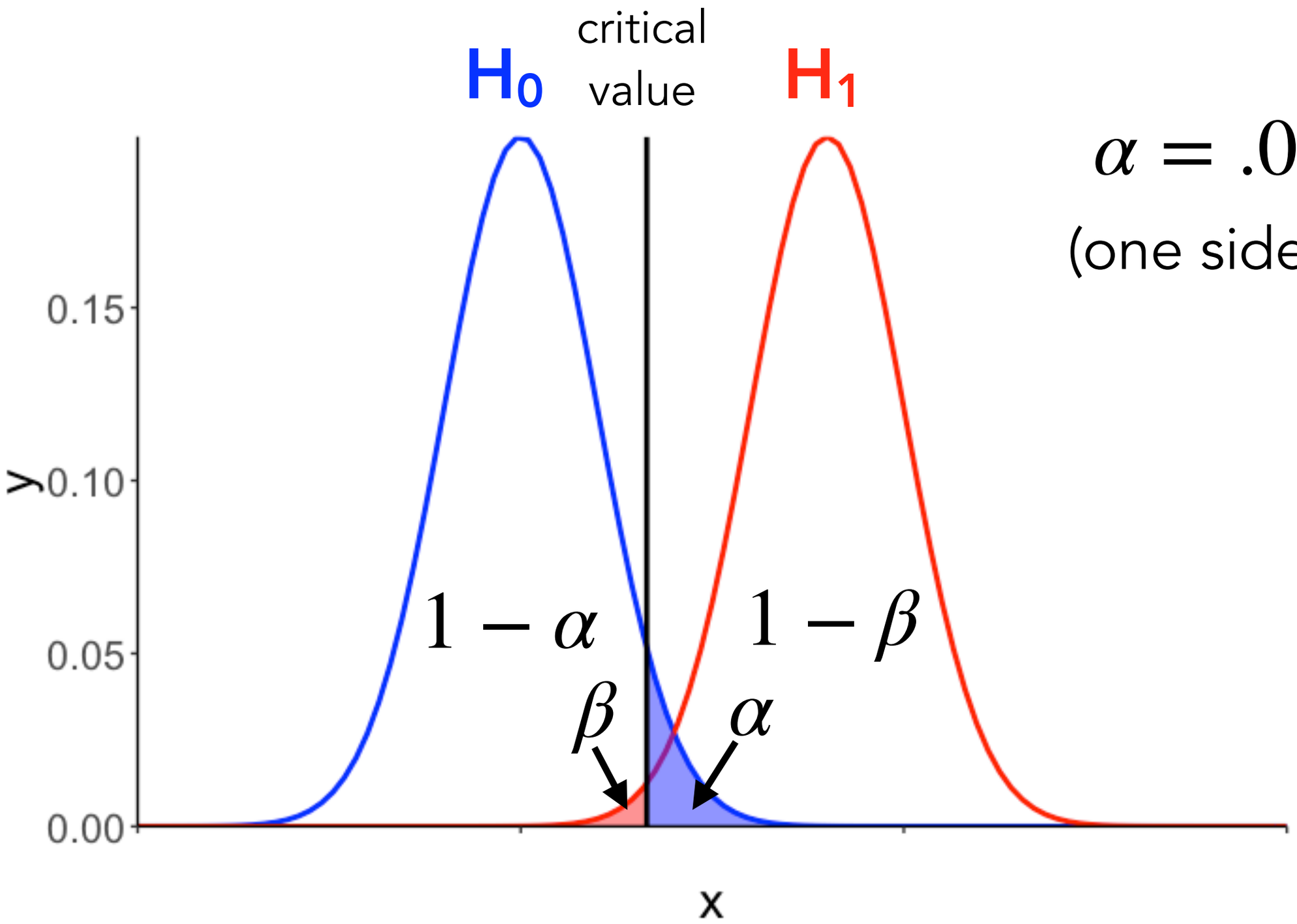
α affects power

What affects power?



distance between means affects power

What affects power?



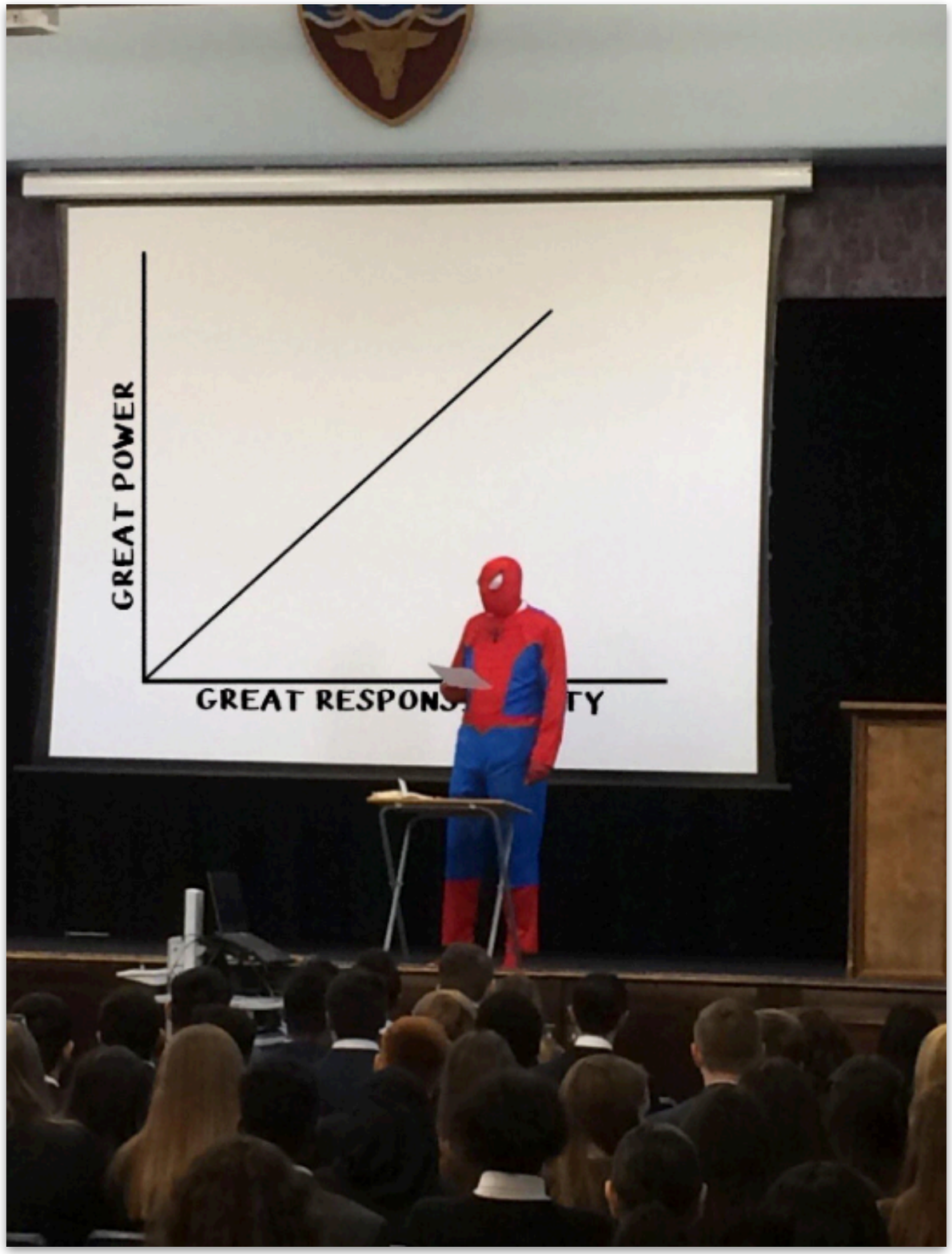
$\alpha = .05$

(one sided)

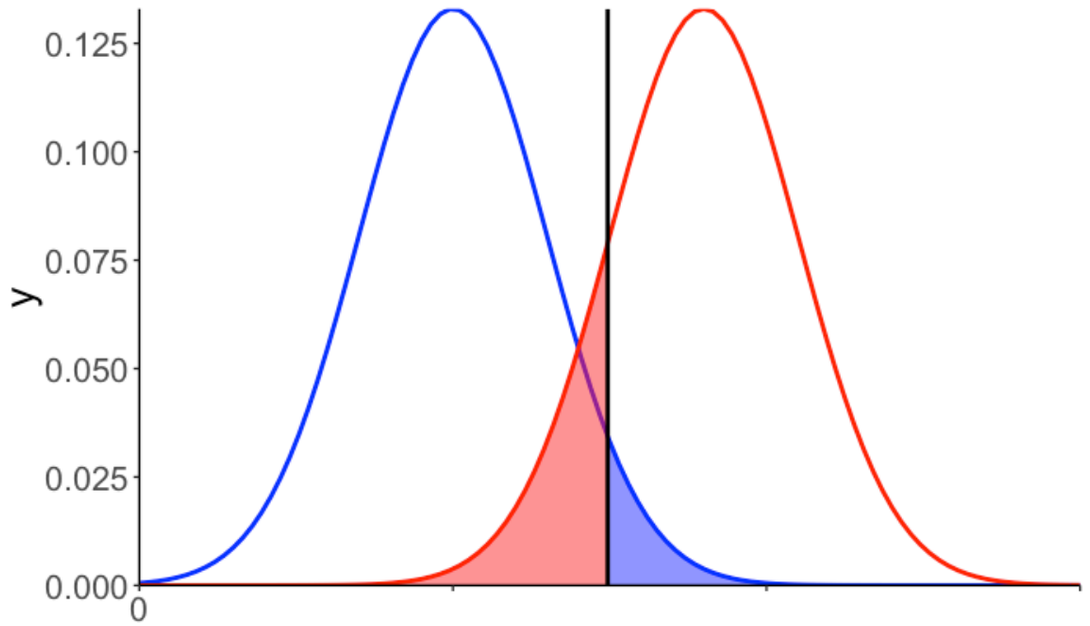
variance affects power

Calculating power

With great power comes ...



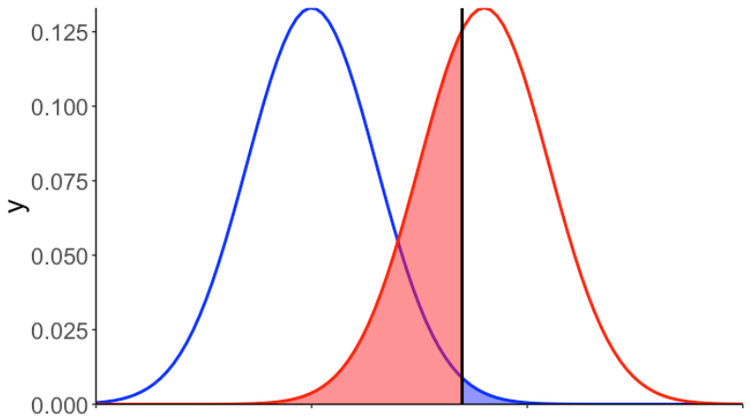
The knobs we can turn to affect power

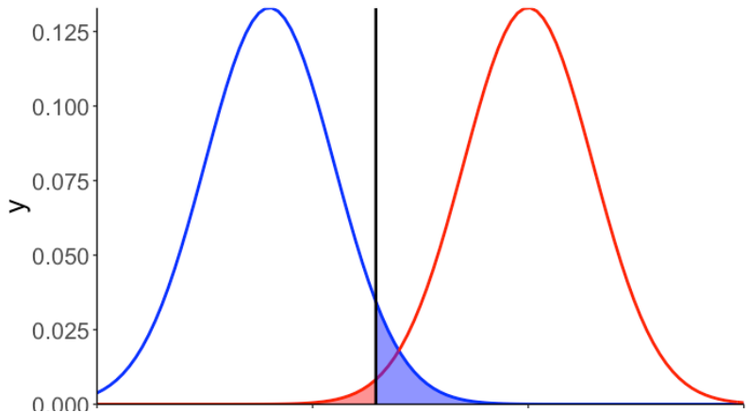


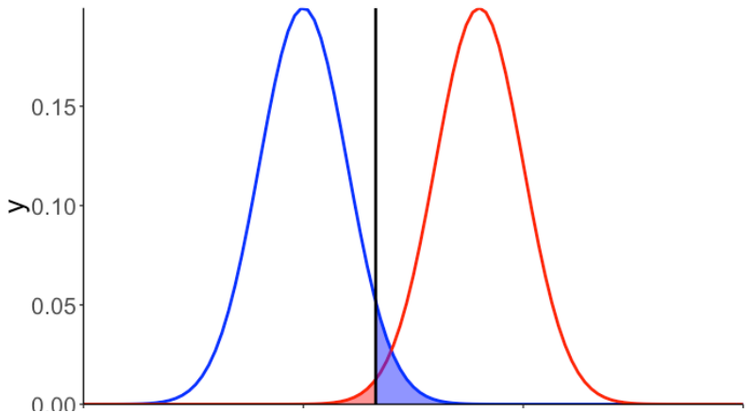
α

effect size

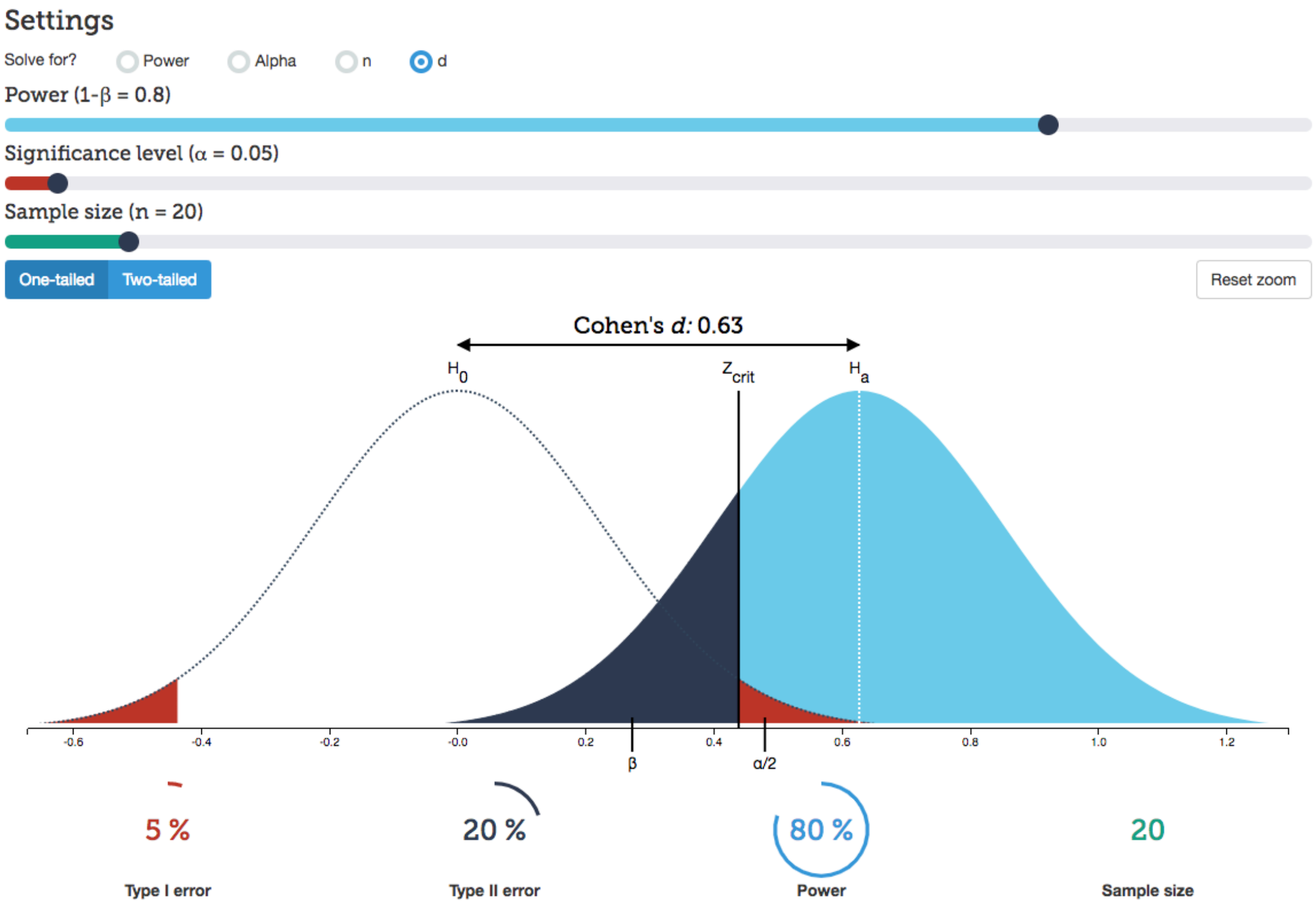
sample size







Visualization demo



https://rpsychologist.com/d3/NHST/

The **power** of a binary hypothesis test is the probability that the test rejects the null hypothesis (H_0) when a **specific** alternative hypothesis (H_1) is true.

H_0: Students and non-students have the same balance.

Model C

$$Y_i = \beta_0 + \epsilon_i$$

$$\beta_1 = 0$$

H_1: Students and non-students have different balances.

Model A

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

$$\beta_1 \neq 0$$

We cannot calculate power in this case.
We need a specific alternative hypothesis!

The **power** of a binary hypothesis test is the probability that the test rejects the null hypothesis (H_0) when a **specific** alternative hypothesis (H_1) is true.

H_0: Students and non-students have the same balance.

Model C

$$Y_i = \beta_0 + \epsilon_i$$

$$\beta_1 = 0$$

H_1: Students and non-students have different balances.

Model A

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i$$

$$\beta_1 = 300$$

We can calculate power in this case (since we have a specific alternative hypothesis)!

Effect sizes

Effect sizes

- a p-value tells us whether we can reject the H_0
- effect sizes is a measure of the strength of the actual effect

Why can't we just use p-values as a measure of the effect size?

$$F = \frac{\text{PRE}/(\text{PA} - \text{PC})}{(1 - \text{PRE})/(n - \text{PA})}$$

PRE = proportional reduction in error
PA = # parameters in the augmented model
PC = # parameters in the compact model
n = sample size

any PRE will become significant if *n* gets large enough

statistical vs. practical significance

Effect sizes

PRE = **p**roportional **r**eduction in **e**rror

Compact model SSE(C)

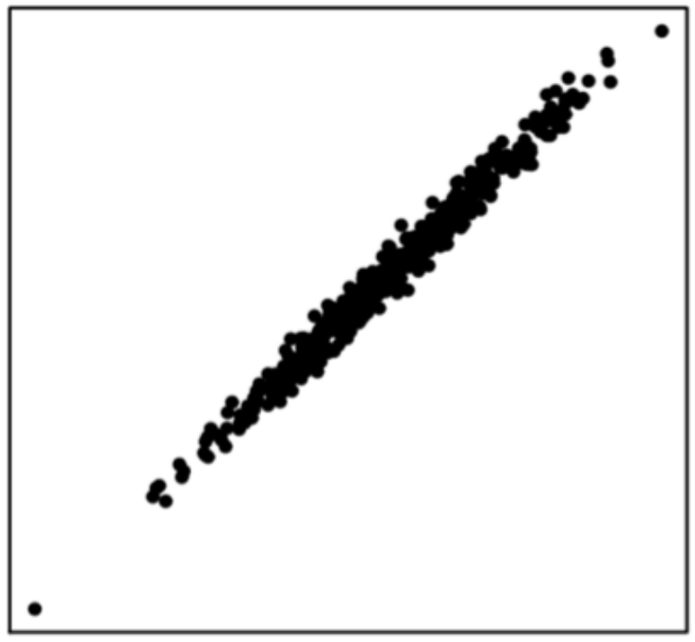
Augmented model SSE(A)

$$\text{PRE} = 1 - \frac{\text{SSE}(A)}{\text{SSE}(C)}$$

SSE = sum of squared errors

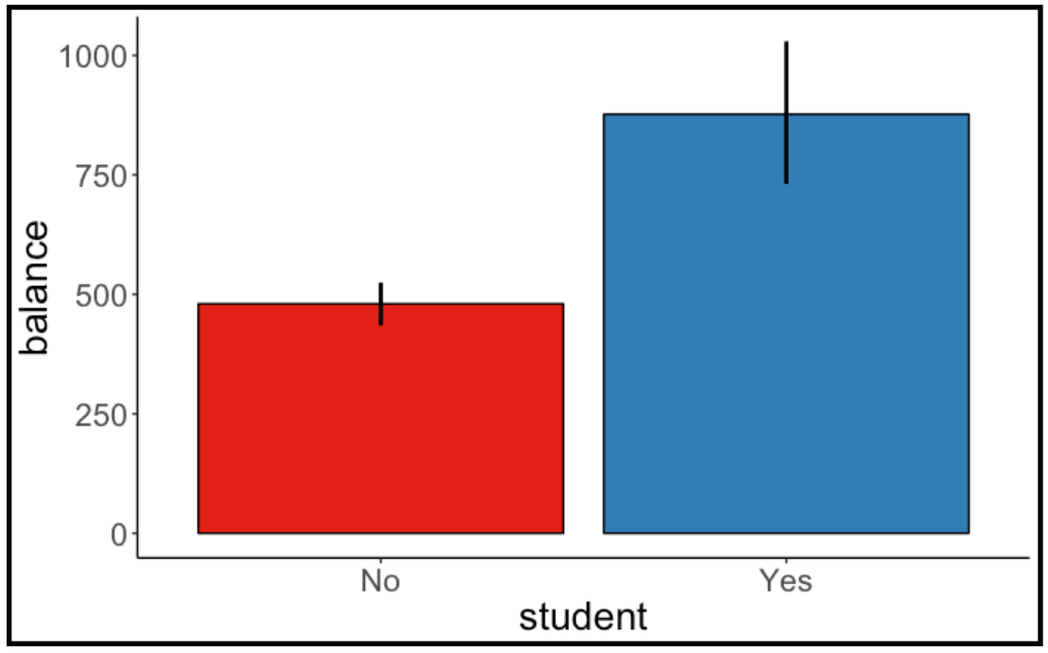
Common effect sizes

Relationships between variables



r correlation

Differences between groups



Cohen's d

Correlation

Pearson correlation

$$r(X, Y) = \frac{\sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \overline{X})^2} \cdot \sqrt{\sum_{i=1}^{n} (Y_i - \overline{Y})^2}}$$

Cohen's guidelines for the social sciences

Effect size	*r*
Small	0.1
Medium	0.3
Large	0.5

depends very much on the domain

Cohen's *d*

- standardized difference between two means

absolute difference between means

$$d = \frac{|\bar{y}_1 - \bar{y}_2|}{s_p}$$

pooled standard variation

$$s_p = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

Effect size	***d***
Very small	0.01
Small	0.20
Medium	0.50
Large	0.80
Very large	1.20
Huge	2.0

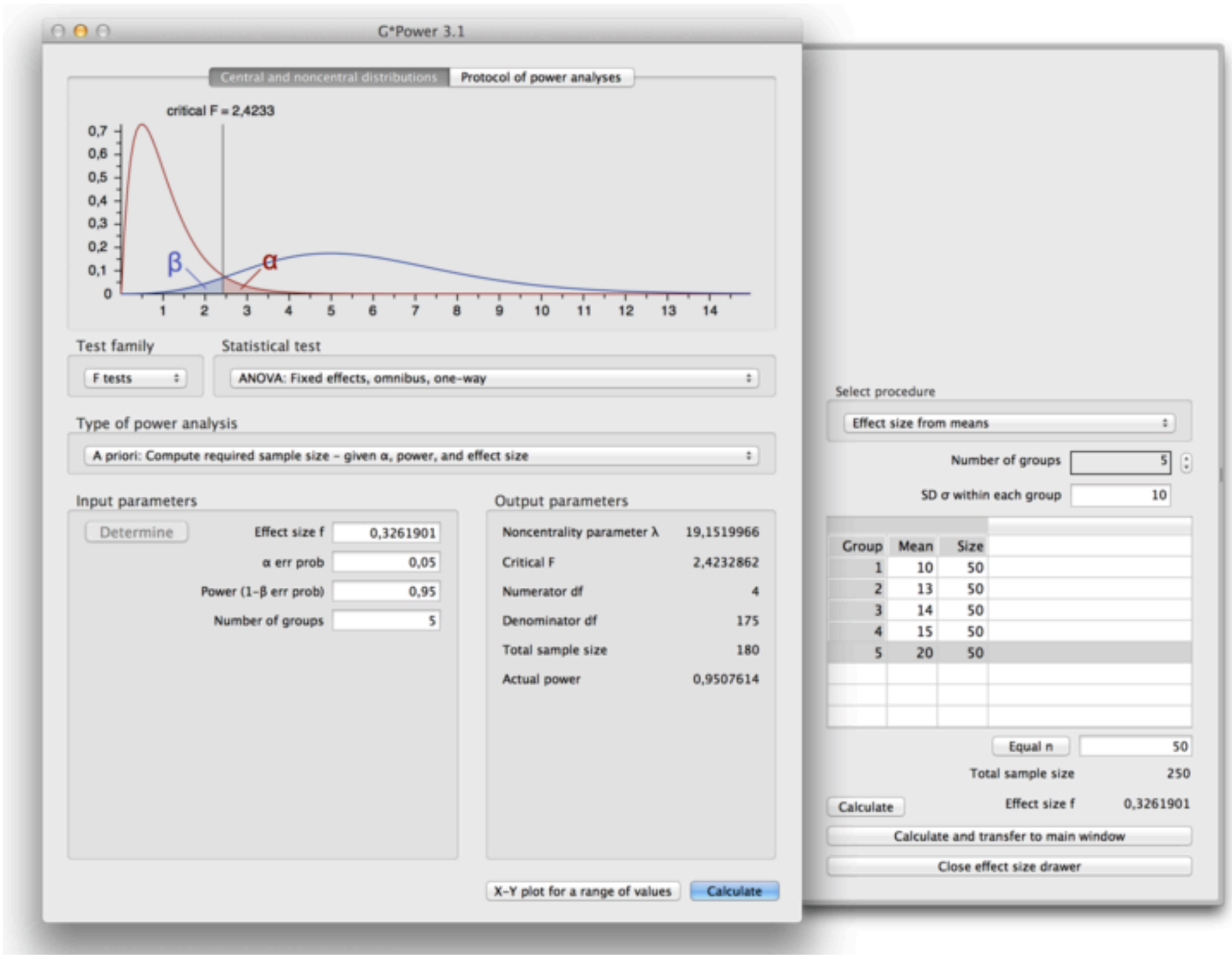
Difference between two means in pooled standard deviation

Determining sample size

How many participants do I need to run to have a good chance of detecting a true effect?

G*Power 3.1: Alternative software for power calculations

Option 1

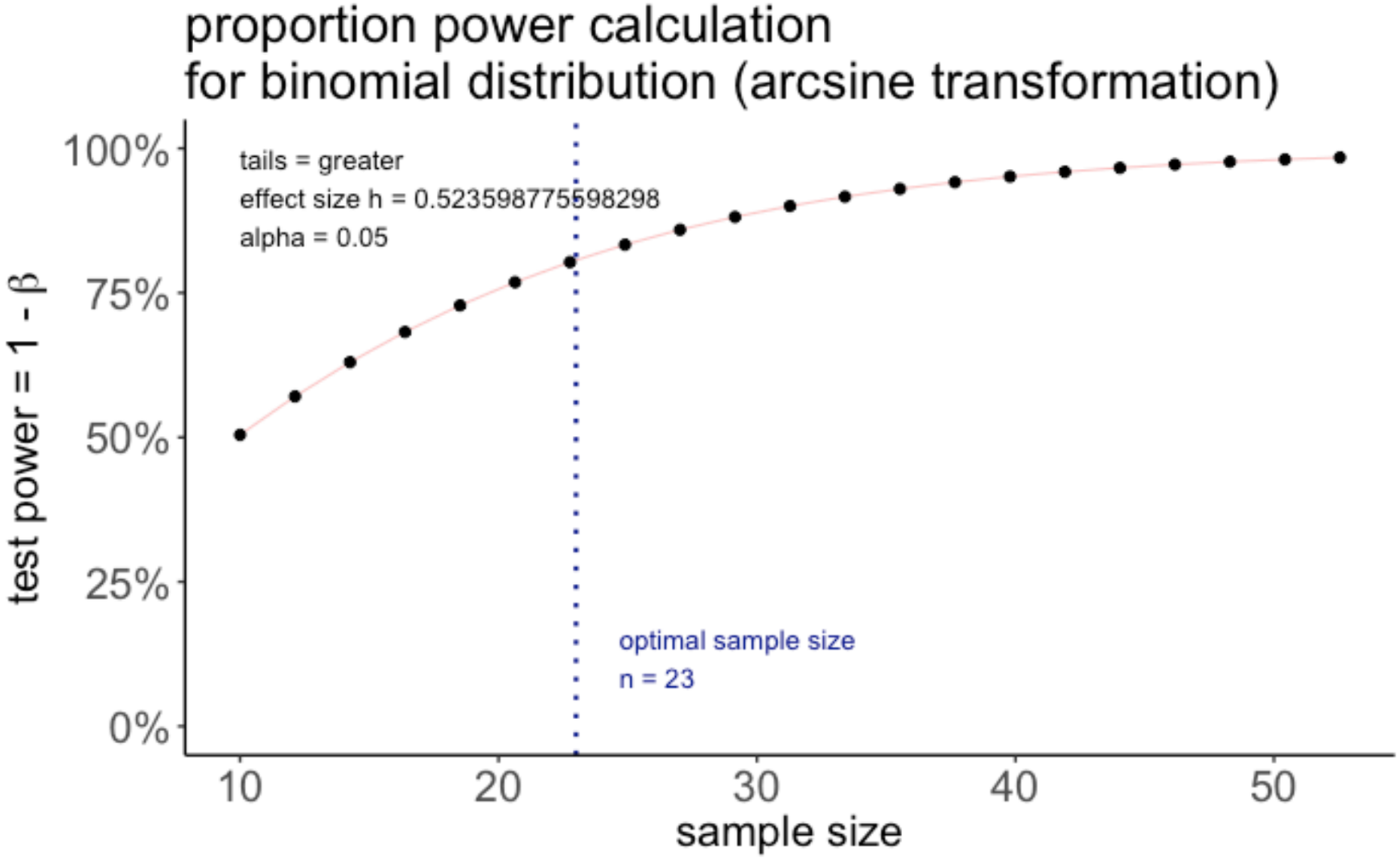


http://www.gpower.hhu.de/

"pwr" package in R

Option 2

```
1  library("pwr")
2  pwr.p.test(h = ES.h(p1 = 0.75, p2 = 0.50),
3             sig.level = 0.05,
4             power = 0.80,
5             alternative = "greater") %>%
6   plot()
```



Power simulation recipe

- assume:
  - α, *n*, effect size
- simulate a large number of data sets of size *n* with the specified effect size
- for each data set, run a statistical test to calculate the p-value
- determine the probability of rejecting the H_0 (given that H_1 is true)

Power analysis via simulation

Simulating a power analysis

Power simulation recipe

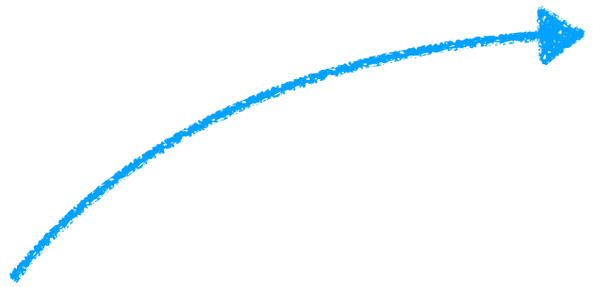
- assume:
  - α, *n*, effect size
- simulate a large number of data sets of size *n* with the specified effect size
- for each data set, run a statistical test to calculate the p-value
- determine the probability of rejecting the H_0 (given that H_1 is true)

Let's simulate ...

```r
# make reproducible
set.seed(1)

# number of simulations
n_simulations = 5

# run simulation
expand_grid(n = seq(10, 40, 2),
            simulation = 1:n_simulations,
            p = 0.75) %>%
  mutate(index = 1:n(),
         .before = n) %>%
  group_by(index, n, p, simulation) %>%
  mutate(response = rbinom(n = 1,
                           size = n,
                           prob = p),
         p.value = binom.test(x = response,
                              n = n,
                              p = 0.5,
                              alternative = "two.sided")$p.value) %>%
  group_by(n, p) %>%
  summarize(power = sum(p.value < 0.05) / n())
```

Let's simulate ...

```
# make reproducible
set.seed(1)

# number of simulations
n_simulations = 5

# run simulation
expand_grid(n = seq(10, 40, 2),
            simulation = 1:n_simulations,
            p = 0.75) %>%
  mutate(index = 1:n(),
         .before = n) %>%
```



index	n	simulation	p
1	10	1	0.75
2	10	2	0.75
3	10	3	0.75
4	10	4	0.75
5	10	5	0.75
6	12	1	0.75
7	12	2	0.75
8	12	3	0.75
9	12	4	0.75
10	12	5	0.75

Let's simulate ...

```
 1 # make reproducible
 2 set.seed(1)
 3
 4 # number of simulations
 5 n_simulations = 5
 6
 7 # run simulation
 8 expand_grid(n = seq(10, 40, 2),
 9               simulation = 1:n_simulations,
10               p = 0.75) %>%
11   mutate(index = 1:n(),
12          .before = n) %>%
13   group_by(index, n, p, simulation) %>%
14   mutate(response = rbinom(n = 1,
15                            size = n,
16                            prob = p),
```

index	n	simulation	p	response
1	10	1	0.75	8
2	10	2	0.75	7
3	10	3	0.75	8
4	10	4	0.75	8
5	10	5	0.75	7
6	12	1	0.75	10
7	12	2	0.75	8
8	12	3	0.75	11
9	12	4	0.75	10
10	12	5	0.75	11

Let's simulate ...

index	n	simulation	p	response	p.value
1	10	1	0.75	8	0.11
2	10	2	0.75	7	0.34
3	10	3	0.75	8	0.11
4	10	4	0.75	8	0.11
5	10	5	0.75	7	0.34
6	12	1	0.75	10	0.04
7	12	2	0.75	8	0.39

```r
# make reproducible
set.seed(1)

# number of simulations
n_simulations = 5

# run simulation
expand_grid(n = seq(10, 40, 2),
            simulation = 1:n_simulations,
            p = 0.75) %>%
  mutate(index = 1:n(),
         .before = n) %>%
  group_by(index, n, p, simulation) %>%
  mutate(response = rbinom(n = 1,
                           size = n,
                           prob = p),
         p.value = binom.test(x = response,
                              n = n,
                              p = 0.5,
                              alternative = "two.sided")$p.value) %>%
```

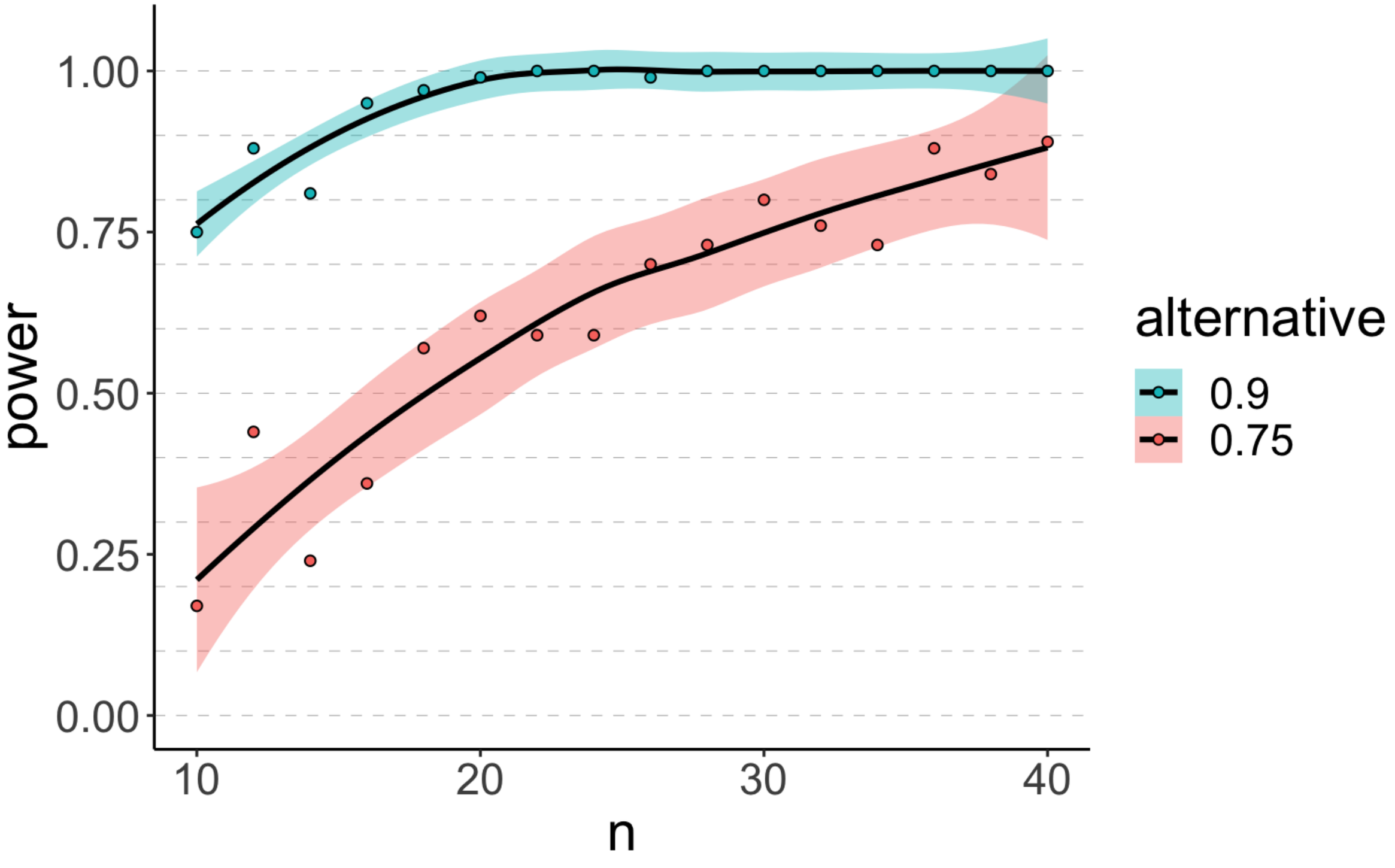
Let's simulate ...

```r
# make reproducible
set.seed(1)

# number of simulations
n_simulations = 5

# run simulation
expand_grid(n = seq(10, 40, 2),
            simulation = 1:n_simulations,
            p = 0.75) %>%
  mutate(index = 1:n(),
         .before = n) %>%
  group_by(index, n, p, simulation) %>%
  mutate(response = rbinom(n = 1,
                           size = n,
                           prob = p),
         p.value = binom.test(x = response,
                              n = n,
                              p = 0.5,
                              alternative = "two.sided")$p.value) %>%
  group_by(n, p) %>%
  summarize(power = sum(p.value < 0.05) / n())
```

n	p	power
10	0.75	0.2
12	0.75	0.2
14	0.75	0.4
16	0.75	0.2
18	0.75	0.6
20	0.75	0.8
22	0.75	0.6
24	0.75	0.4
26	0.75	0.6
28	0.75	0.8
30	0.75	0.8

Let's simulate ...

in this example, I looked at the power for two different alternative hypotheses



Let's simulate ...

- here, I've used a simple example (Binomial test)
- but: we can use the same recipe for any statistical test that we are planning on running

Power simulation recipe

- assume:
  - α, n, effect size
- simulate a large number of data sets of size n with the specified effect size
- for each data set, run a statistical test to calculate the p-value for a given α
- determine the probability of rejecting the H_0 (given that H_1 is true)

Let's simulate

```
library("purrr")
```

automatically loaded with

```
library("tidyverse")
```

`map()`

`map()`

vector or list

function

```
map(.x = 1:3, .f = ~ .^2)
```

outputs a list

```
[[1]]
[1] 1

[[2]]
[1] 4

[[3]]
[1] 9
```

- **`map`**`(list, function)` applies a function to each element of the list
- it's a unified version of the many different **`apply`**`()` functions in base R
- you already know a cousin of **`map`**`()`: **`replicate`**`()`
- use **`map`**`()`, don't write `for(){}` loops!
- it's extremely powerful in combination with data frames

map()

same same but different

```r
map(.x = 1:3, .f = ~ .x^2)

map(1:3, ~ .x^2)

map(1:3, ~ .^2)

map(.x = 1:3, .f = function(.x) .x^2)

# using a function
square = function(x){x^2}
map(1:3, square)
```

INTERACTIVE COURSE

Foundations of Functional Programming with purrr

Start Course For Free | Bookmark

4 hours | 13 Videos | 44 Exercises | 7,769 Participants | 3,750 XP

Course Description

Lists can be difficult to both understand and manipulate, but they can pack a ton of information and are very powerful. In this course, you will learn to easily extract, summarize, and manipulate lists and how to export the data to your desired object, be it another list, a vector, or even something else! Throughout the course, you will work with the purrr package and a variety of datasets from the repurrrsive package, including data from Star Wars and Wes Anderson films and data collected about GitHub users and GitHub repos. Following this course, your list skills will be purrrfect!

1 Simplifying Iteration and Lists With purrr FREE

0%

Iteration is a powerful way to make the computer do the work for you. It can also be an area of coding where it is easy to make lots of typos and simple mistakes. The purrr package helps simplify iteration so you can focus on the next step, instead of finding typos.

The power of iteration	50 xp
Introduction to iteration	100 xp
Iteration with purrr	100 xp
More iteration with for loops	100 xp
More iteration with purrr	100 xp
Subsetting lists; it's not that hard!	50 xp
Subsetting lists	100 xp
Subsetting list elements	100 xp

This course is part of these tracks:

Intermediate Tidyverse Toolbox

DataCamp Content Creator

Course Instructor

This instructor prefers to remain anonymous.

See More

COLLABORATOR(S)

Chester Ismay

Becca Robins

https://www.datacamp.com/courses/foundations-of-functional-programming-with-purrr

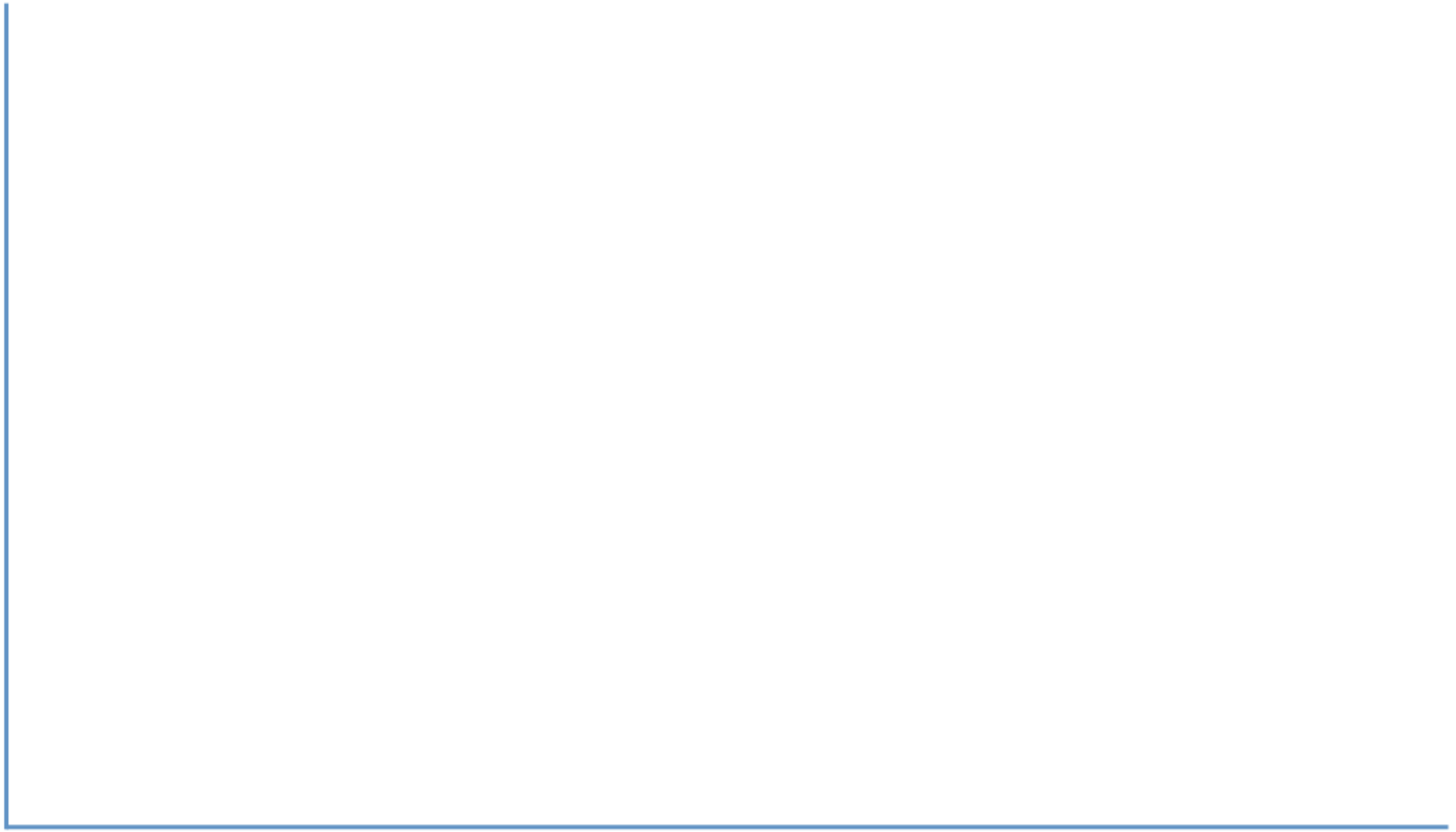
Plan for today

- Quick recap
- Unbalanced designs
- Linear contrasts
  - Testing specific hypotheses with linear contrasts
  - `emmeans` for handling linear contrasts in R
- Power analysis
  - Making decisions
  - Calculating power
  - Effect sizes
  - Determining sample size
- Power analysis via simulation
- Learn about more advanced simulation techniques in R
  - `map()`
  - list columns: `nest()`, `unnest()`

will share a 25 minute video lecture

Feedback

How was the pace of today's class?



How happy were you with today's class overall?

What did you like about today's class? What could be improved next time?

Thank you!