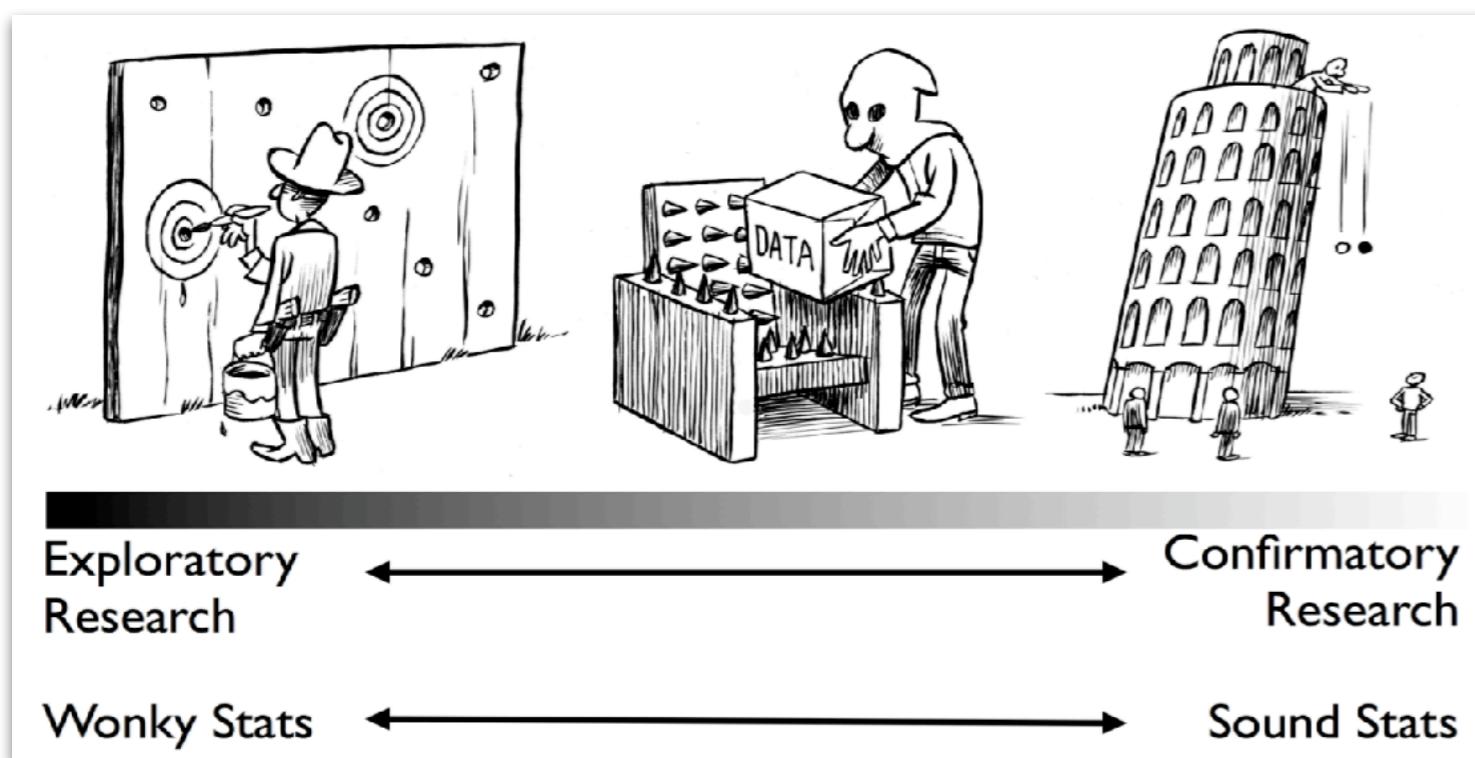


# Power analysis



Chat

What is the strangest gift you have ever received?

To: Everyone ▾ More ▾

Type message here...

COLLABORATIVE PLAYLIST

**psych252**

<https://tinyurl.com/psych252spotify22>

PLAY ...

We're listening to "Fleur tropicale" by "Francis Bebey" submitted by Tobi

02/04/2022

# **Logistics**

# Midterm

will be available a little bit after class today

Psych 252 Midterm

My name goes here

## Introduction

This is a take-home exam. The exam is open notes and open book (in short, you can use any source of information you like as long as you work on the exam by yourself). The maximum score is 120 points. Please adhere to the honor code.

The late policy submission policy is:

- We will subtract 2% from your points for each hour that the midterm is submitted late but before midnight. For example, 2% will be subtracted if you submit between 8pm and 9pm, or 8% if you submit between 11pm and midnight.
- 20% will be subtracted if you submit after midnight on Thursday but before 8pm on Friday, February 19th.
- No points will be granted if you submit later than 8pm on Friday, February 19th.

For questions that require written responses, please make sure to show any relevant tables, summaries (e.g. from `lm()` or `anova()`), or visualizations. Some of the code chunks have existing code that you can use to build your code around.

When asked to report results, please do so like you would in a scientific article (see examples from class).

- Please leave the `\clearpage` commands where they are. This makes sure that each question is printed on a separate page in the pdf.
- Some code chunks are set to `eval=F`, make sure to set these to `eval=T` before knitting the final version.
- We note for each question how many points you can get. You can get up to 120 points in total.
- Good coding style matters! We will add or subtract up to 5 points depending on style.

If you have any questions about the midterm, please post them on Piazza addressed to the instructors only. We will answer your question and may choose to share both your question and our answer with the rest of the group.

Best of luck with the midterm!

## Honor Code

The Honor Code is the University's statement on academic integrity written by students in 1921. It articulates University expectations of students and faculty in establishing and maintaining the highest standards in academic work:

1. The Honor Code is an undertaking of the students, individually and collectively:
  - a. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
  - b. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

## Part 1: Wash your hands!

A student investigated how effective different methods are for eliminating bacteria. She tested four different methods: (1) washing her hands with water only, (2) with regular soap, (3) with antibacterial soap (ABS), or (4) using an antibacterial spray (AS). She suspected that the number of bacteria on her hands might vary considerably from day to day. To account for this, she generated random numbers to determine on what day she would use which treatment. After each treatment, she placed her right hand on a sterile media plate to measure bacteria growth. She incubated each plate for 2 days after which she counted the bacteria colonies. She replicated this procedure 9 times for each of the four treatments.

Note: For statistical analysis purposes, we make the assumption that the individual measurements are independent from each other.

## Part 2: Life satisfaction

In this exercise, we are interested in seeing what affects life satisfaction. We have a (fake) data set with the following variables:

Table 1: Variables in the satisfaction data set.

variable	description
id	participant id
age	age in years
kids	number of kids
jobsatis	job satisfaction (1 = not at all, 7 = very much)
marsatis	marital satisfaction (1 = not at all, 7 = very much)
lifsatis	life satisfaction (1 = not at all, 7 = very much)

## Part 3: You've got the power!

In this exercise, we'll take a look at determining what group sample size we would need in order to achieve adequate statistical power to test our research hypothesis of interest. We will be using the data from "data/power.csv" and you can see its visualization in "figure/df\_power.png".

Tip: We will provide saved checkpoints for each problem. Feel free to look ahead to compare your results with ours.

# Next week

no class on Wednesday (more time to work on midterm)

no sections

no office hours (at least not for questions about the midterm)

due **Friday 11th before 8pm** (one extra day)

# Plan for today

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

will share a 25  
minute video lecture

# Quick recap

# Quick recap: lm() output

## 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   33.6981   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)

```

10

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	56208745	1	18131167	108.99 < 2.2e-16 ***
---								
Signif. codes:	0	'****'	0.001	'**'	0.01	'*'	0.05	.
	.'	0.1	' '	1				

deterministic mapping  
between t and F

$$t^2 = F$$

$$10.44^2 = 108.99$$

anova () gives me Fs  
but lm() gives me ts

```

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   33.1953   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: 407.5 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

```

## lm() output



what does this mean?

not the overall  
effect of  
income!!

instead the  
predicted effect  
of income for  
non-students!

```

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

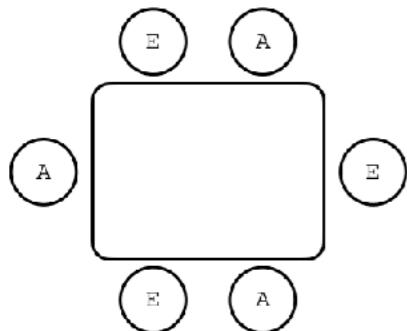
```

# Quick recap: Categorical predictors

## What's the role of skill vs. chance in poker?

### Abstract

Adopting a quasi-experimental approach, the present study examined the extent to which the influence of poker playing skill was more important than card distribution. Three average players and three experts sat down at a six-player table and played 60 computer-based hands of the poker variant "Texas Hold'em" for money. In each hand, one of the average players and one expert received (a) better-than-average cards (winner's box), (b) average cards (neutral box) and (c) worse-than-average cards (loser's box). The standardized manipulation of the card distribution controlled the factor of chance to determine differences in performance between the average and expert groups. Overall, 150 individuals participated in a "fixed-limit" game variant, and 150 individuals participated in a "no-limit" game variant.



- During the game, one expert player and one average player received
- the winning hand 15 times and the losing hand 5 times (winner's box condition)
  - the winning hand 10 times and the losing hand 10 times (neutral box condition)
  - the winning hand 5 times and the losing hand 15 times (loser's box condition)

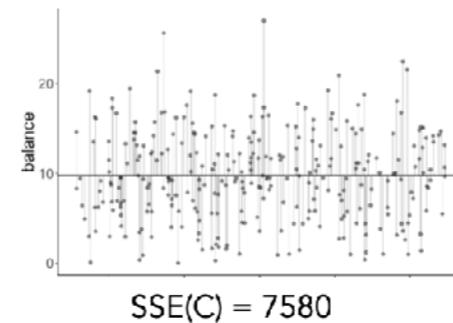
16

$H_0$ : Card quality does not affect the final balance.

### Model C

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

### Model prediction



### Fitted model

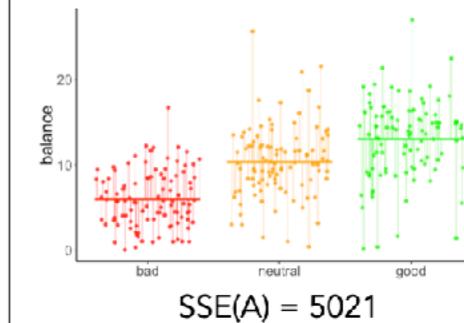
$$\widehat{\text{balance}}_i = 9.77$$

$H_1$ : Card quality affects the final balance.

### Model A

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

### Model prediction



### Fitted model

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

21

## Does card quality affect the final balance?

$$\text{SSE}(C) = 7580$$

$$\text{PRE} = 1 - \frac{\text{SSE}(A)}{\text{SSE}(C)} \quad \text{worth it?}$$

$$\text{SSE}(A) = 5021$$

$$= 1 - \frac{5021}{7580} \approx 0.34$$

```

1 # fit the models
2 fit_c = lm(formula = balance ~ 1, data = df.poker)
3 fit_a = lm(formula = balance ~ hand, data = df.poker)
4
5 # compare via F-test
6 anova(fit_c, fit_a)

```

### Analysis of Variance Table

Model 1: balance ~ 1

Model 2: balance ~ 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

## Interpreting the results

`lm(formula = balance ~ 1 + hand, data = df.poker)`

Call:

`lm(formula = balance ~ hand, data = df.poker)`

Residuals:

Min	1Q	Median	3Q	Max
-12.9264	-2.5902	-0.0115	2.6573	15.2834

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	5.9415	0.4111	14.451	< 2e-16 ***
handneutral	4.4051	0.5815	7.576	4.55e-13 ***
handgood	7.0849	0.5815	12.185	< 2e-16 ***

Signif. codes: 0 '\*\*\*\*' 0.001 '\*\*\*' 0.01 '\*\*' 0.05 '\*' 0.1 '.' 1

Residual standard error: 4.111 on 297 degrees of freedom

Multiple R-squared: 0.3377, Adjusted R-squared: 0.3332

F-statistic: 75.7 on 2 and 297 DF, p-value: < 2.2e-16

22

# Quick recap: Categorical predictors

## One-way ANOVA

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

```
Analysis of Variance Table

Response: balance
          Df Sum Sq Mean Sq F value    Pr(>F)
hand        2 2559.4 1279.7 75.703 < 2.2e-16 ***
Residuals 297 5020.6   16.9
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What do these mean?

```
1 # fit the models
2 fit_c = lm(formula = balance ~ 1, data = df.poker)
3 fit_d = lm(formula = balance ~ hand, data = df.poker)
4
5 # compare via F-test
6 anova(fit_c, fit_d)
```

```
Analysis of Variance Table

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

## Asking more specific questions

**Is there a difference in the final balance between neutral hands and good hands?**

```
1 df.poker %>%
2   filter(hand %in% c("neutral", "good")) %>%
3   lm(formula = balance ~ hand,
4       data = .) %>%
5   summary()
```

```
Call:
lm(formula = balance ~ hand, data = .)

Residuals:
    Min      1Q  Median      3Q     Max 
-12.9264 -2.7141  0.2585  2.7184 15.2834 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 10.3466    0.4448  23.26 < 2e-16 ***
handgood    2.6798    0.6291   4.26 3.16e-05 ***  
---
Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.448 on 198 degrees of freedom
Multiple R-squared:  0.08396, Adjusted R-squared:  0.07933 
F-statistic: 18.15 on 1 and 198 DF, p-value: 3.158e-05
```

## One-way ANOVA

### Variance decomposition



variance_total	variance_model	variance_residual
7580	2559	5021

## Analysis of variance

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

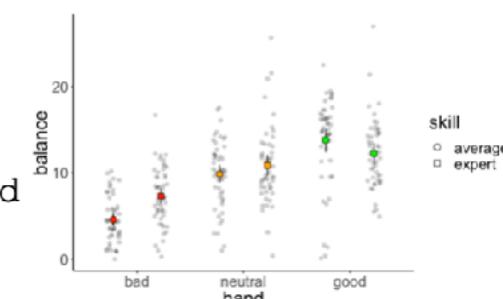
```
Analysis of Variance Table

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



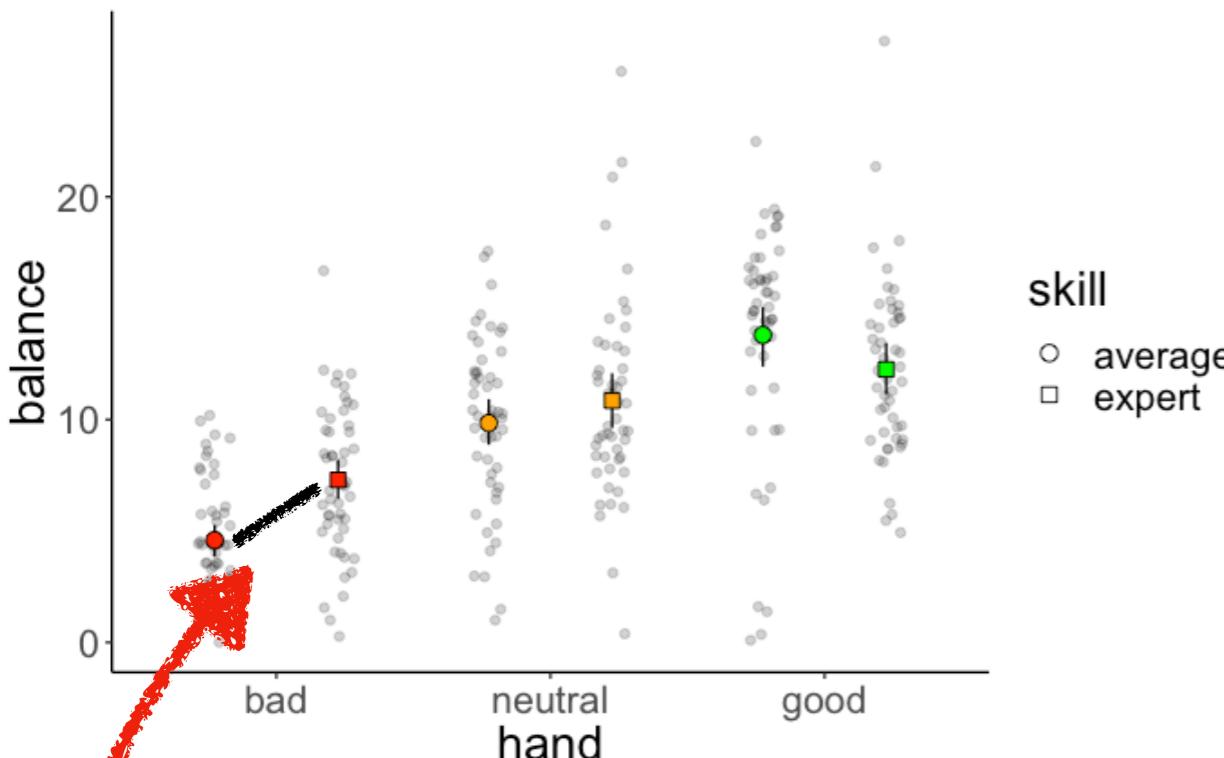
# Parameter interpretation

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

```
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:  '***' 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
```

there was a significant effect of 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()

```

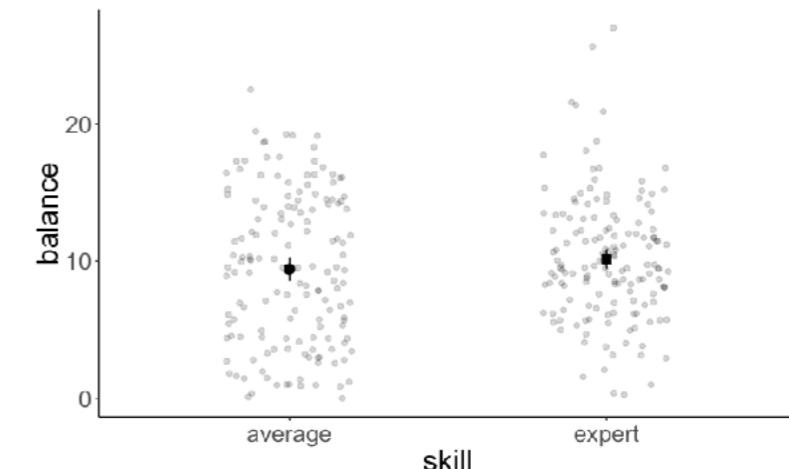
Analysis of Variance Table

	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

# Effects in an ANOVA

- **main effect:** effect of one independent variable on the dependent variable
- **interaction effect:** when the effect of one independent variable depends on the level of another
- **simple effect:** comparison between two specific cell means

# Interpreting parameters

lm() gives simple effects

lm(formula = balance ~ hand \* skill,  
data = df.poker)

```
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
```

anova() gives main effects,  
and interactions

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

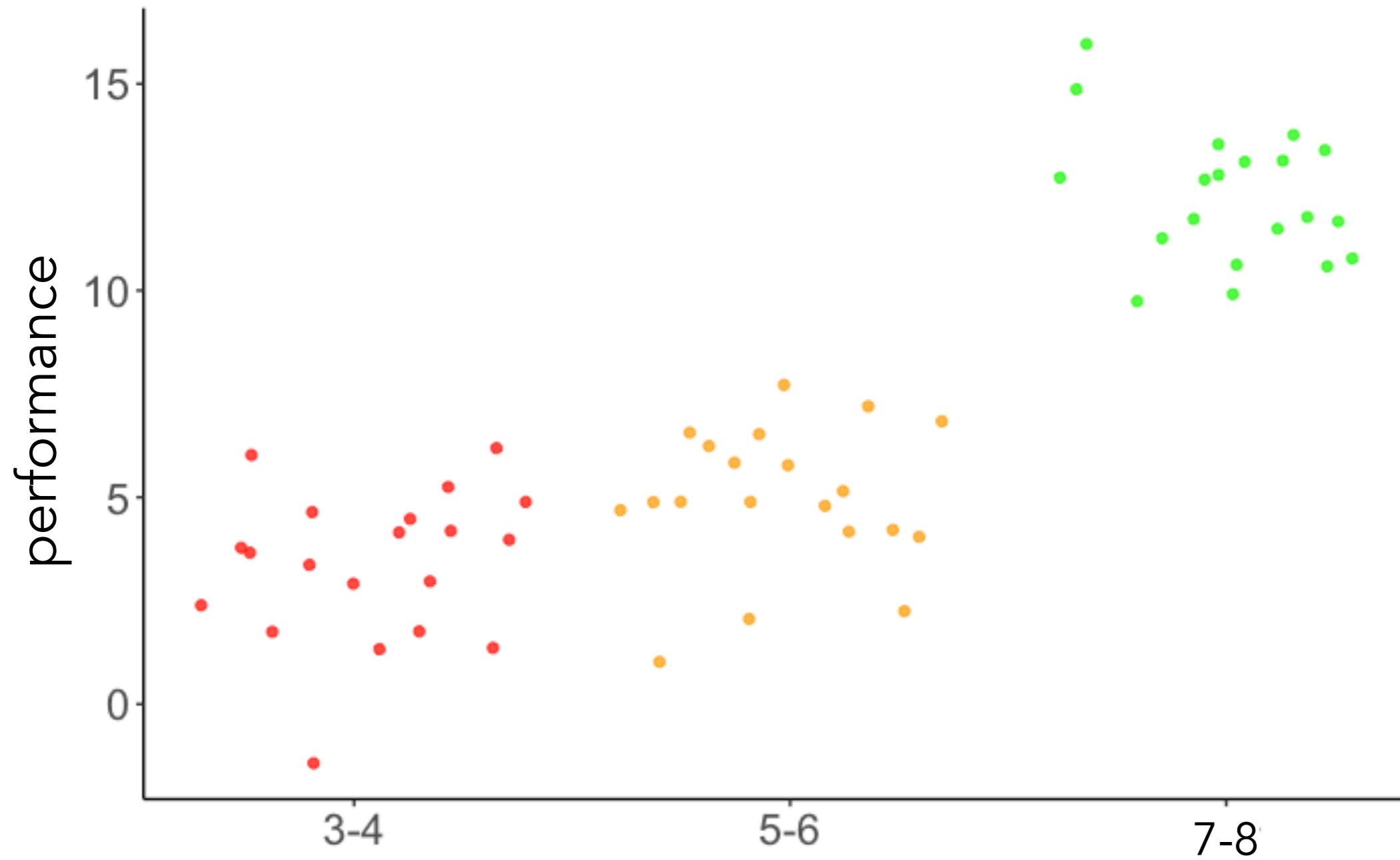
```
Analysis of Variance Table  
  
Response: balance  
              Df Sum Sq Mean Sq F value Pr(>F)  
hand          2 2559.4 1279.70 79.1692 < 2.2e-16 ***  
skill         1   39.3   39.35  2.4344 0.1197776  
hand:skill   2   229.0  114.49  7.0830 0.0009901 ***  
Residuals  294 4752.3   16.16  
---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

# **Linear contrasts**

# **Testing (more) specific hypotheses with linear contrasts**

# Contrasts

**Does performance increase with age?**



**Data from a hypothetical developmental study**

# Do better hands win more money?



ANOVA

## Does card quality affect the final balance?

post-

3-4 vs. 5-6

its

5-6 vs. 7-8



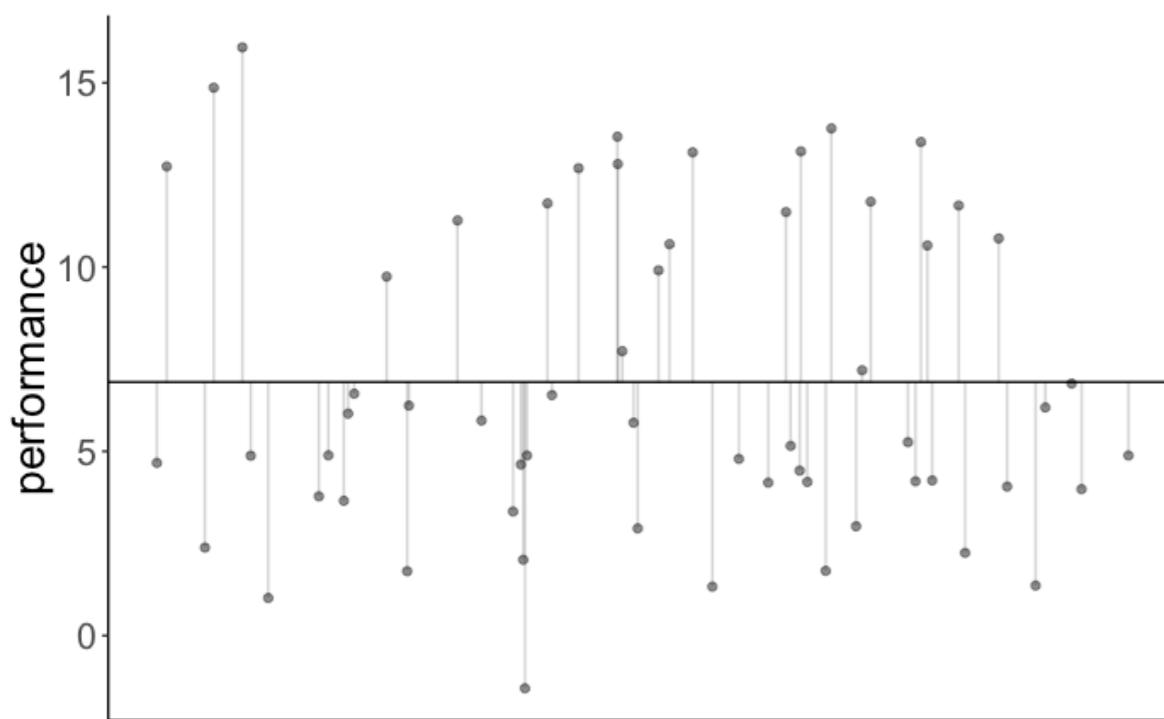
Is there are more direct way of asking this question with a statistical model?

# Contrasts

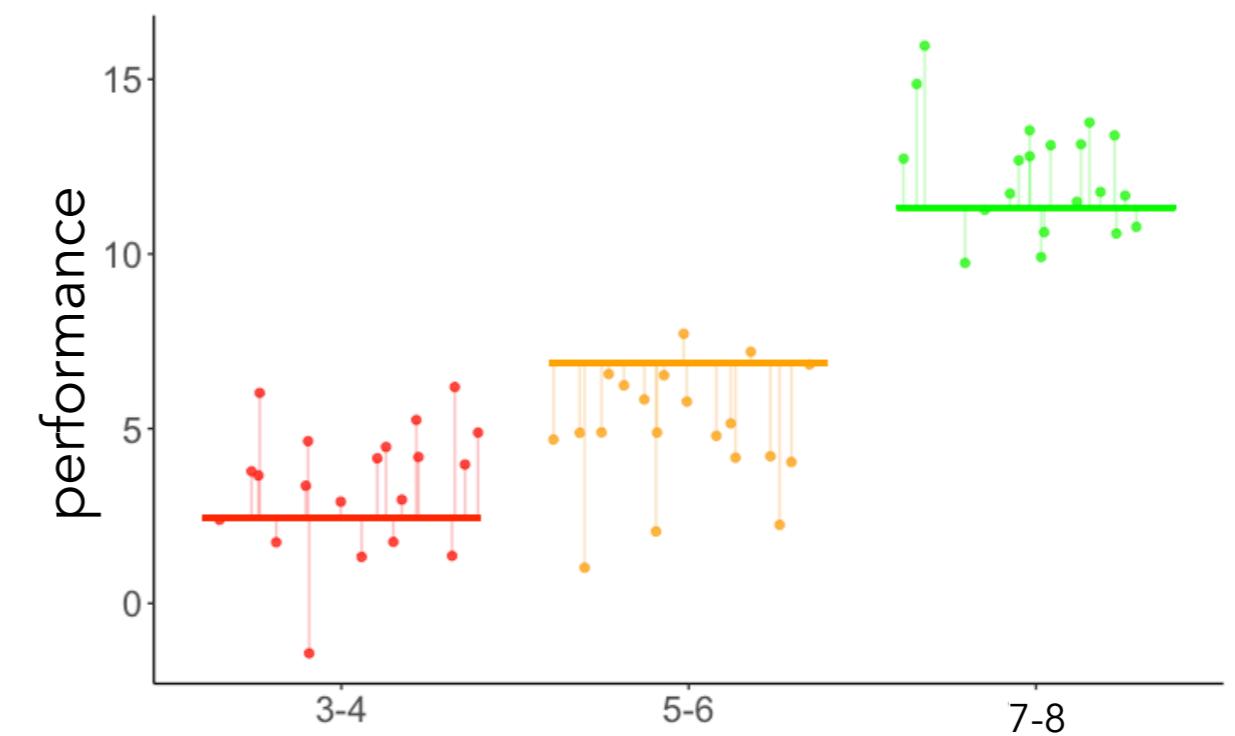
**Does performance increase with age?**

contrasts = c(-1, 0, 1)

**Compact model**



**Augmented model**



**Model comparison**

$p < .001$

# **emmeans for handling linear contrasts in R**

# Linear contrasts

## ~~How to use contrasts in R~~

In short: don't bother.<sup>1</sup>

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.

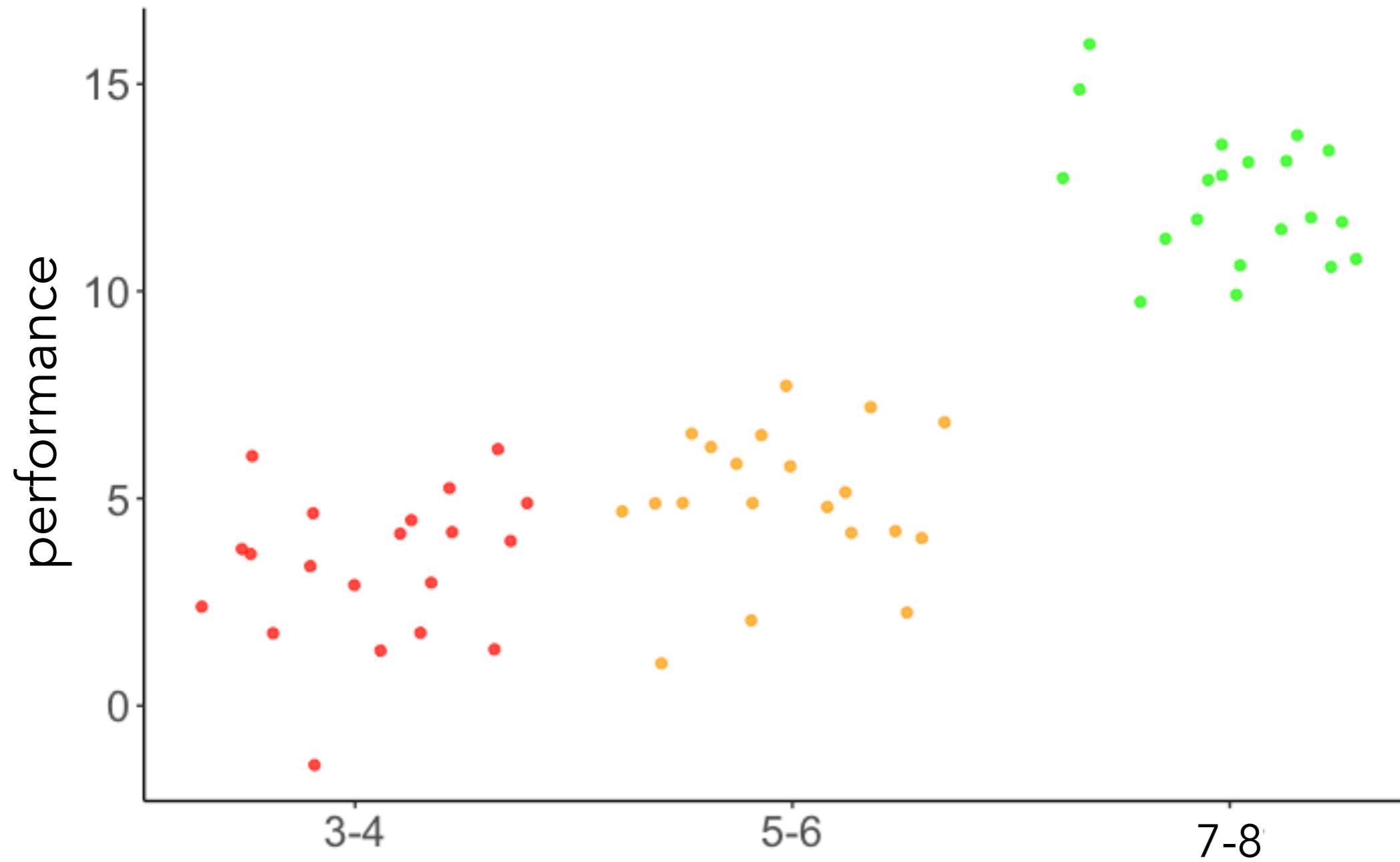
Luckily for me, someone came along and fixed the situation: [emmeans](#). emmeans frames contrasts as a question you pose to a model: you can ask for all pairwise comparisons and get back that. `lm` and `summary` treat the same problem as fitting abstract coefficients, and you are left to answer your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

**Does performance increase with age?**



**Data from a hypothetical developmental study**

# Linear contrasts in R

```
1 library("emmeans") # for calculating contrasts  
2  
3 # fit the linear model  
4 fit = lm(formula = performance ~ group,  
5           data = df.development)
```

fit linear model

# Linear contrasts in R

```
1 library("emmeans") # for calculating contrasts  
2  
3 # fit the linear model  
4 fit = lm(formula = performance ~ group,  
5           data = df.development)  
6  
7 # check factor levels  
8 levels(df.development$group) [1] "3-4" "5-6" "7-8"
```

check factor levels before  
defining contrasts

# Linear contrasts in R

```
1 library("emmeans") # for calculating contrasts
2
3 # fit the linear model
4 fit = lm(formula = performance ~ group,
5           data = df.development)
6
7 # check factor levels
8 levels(df.development$group) [1] "3-4" "5-6" "7-8"
9
10 # define the contrasts of interest
11 contrasts = list(young_vs_old = c(-0.5, -0.5, 1),
12                   three_vs_five = c(-0.5, 0.5, 0)))
```

set up linear contrasts

# Linear contrasts in R

```
1 library("emmeans") # for calculating contrasts
2
3 # fit the linear model
4 fit = lm(formula = performance ~ group,
5           data = df.development)
6
7 # check factor levels
8 levels(df.development$group) [1] "3-4" "5-6" "7-8"
9
10 # define the contrasts of interest
11 contrasts = list(young_vs_old = c(-0.5, -0.5, 1),
12                   three_vs_five = c(-0.5, 0.5, 0))
13
14 # compute significance test on contrasts
15 fit %>%
16   emmeans("group",
17           contr = contrasts,
18           adjust = "bonferroni") %>%
19   pluck("contrasts")
```

compute the results

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

```
1 library("emmeans") # for calculating contrasts
2
3 # fit the linear model
4 fit = lm(formula = performance ~ group,
5           data = df.development)
6
7 # check factor levels
8 levels(df.development$group)
9
10 # define the contrasts of interest
11 contrasts = list(young_vs_old = c(-1, -1, 2),
12                   three_vs_five = c(-1, 1, 0))
13
14 # compute significance test on contrasts
15 fit %>%
16   emmeans("group",
17           contr = contrasts,
18           adjust = "bonferroni") %>%
19   pluck("contrasts")
```

hypothesis tests  
are the same!

[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

# Defining contrasts

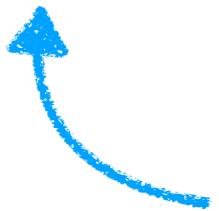
- groups that we don't want to include in the comparison get a 0
- groups that we want to compare with one another should sum to 0
- this also means that all the contrasts together should sum to 0

## Example:

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

# Post hoc tests

```
1 fit = lm(formula = performance ~ group,  
2           data = df.development)  
3  
4 # pairwise differences between all the groups  
5 fit %>%  
6   emmeans(pairwise ~ group) %>%  
7   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

```
1 # fit the model  
2 fit = lm(formula = balance ~ hand + skill,  
3           data = df.poker)  
4  
5 # post hoc tests  
6 fit %>%  
7   emmeans(pairwise ~ hand + skill,  
8             adjust = "bonferroni") %>%  
9   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

that's a lot of tests!

... not

P value adjustment: bonferroni method for 15 tests

all pairwise tests between groups

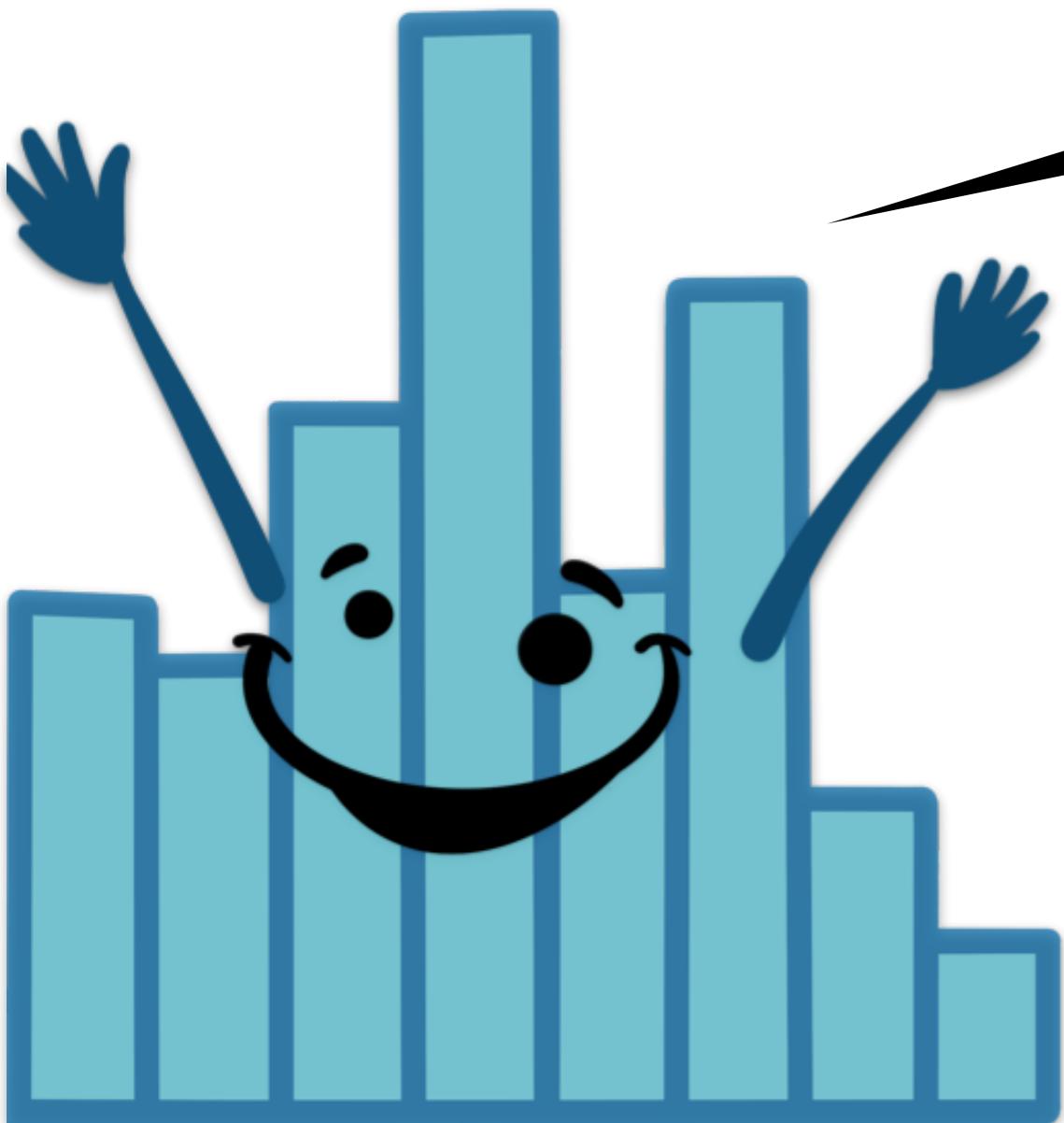
# Contrasts

- 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?

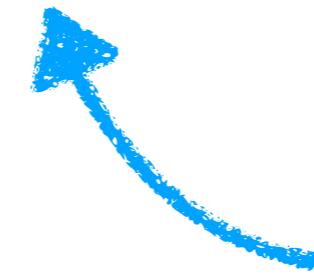
We're listening to "Wake up" by "Sudan Archives"  
submitted by Tobi

02:00

stretch break!



# 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()
```

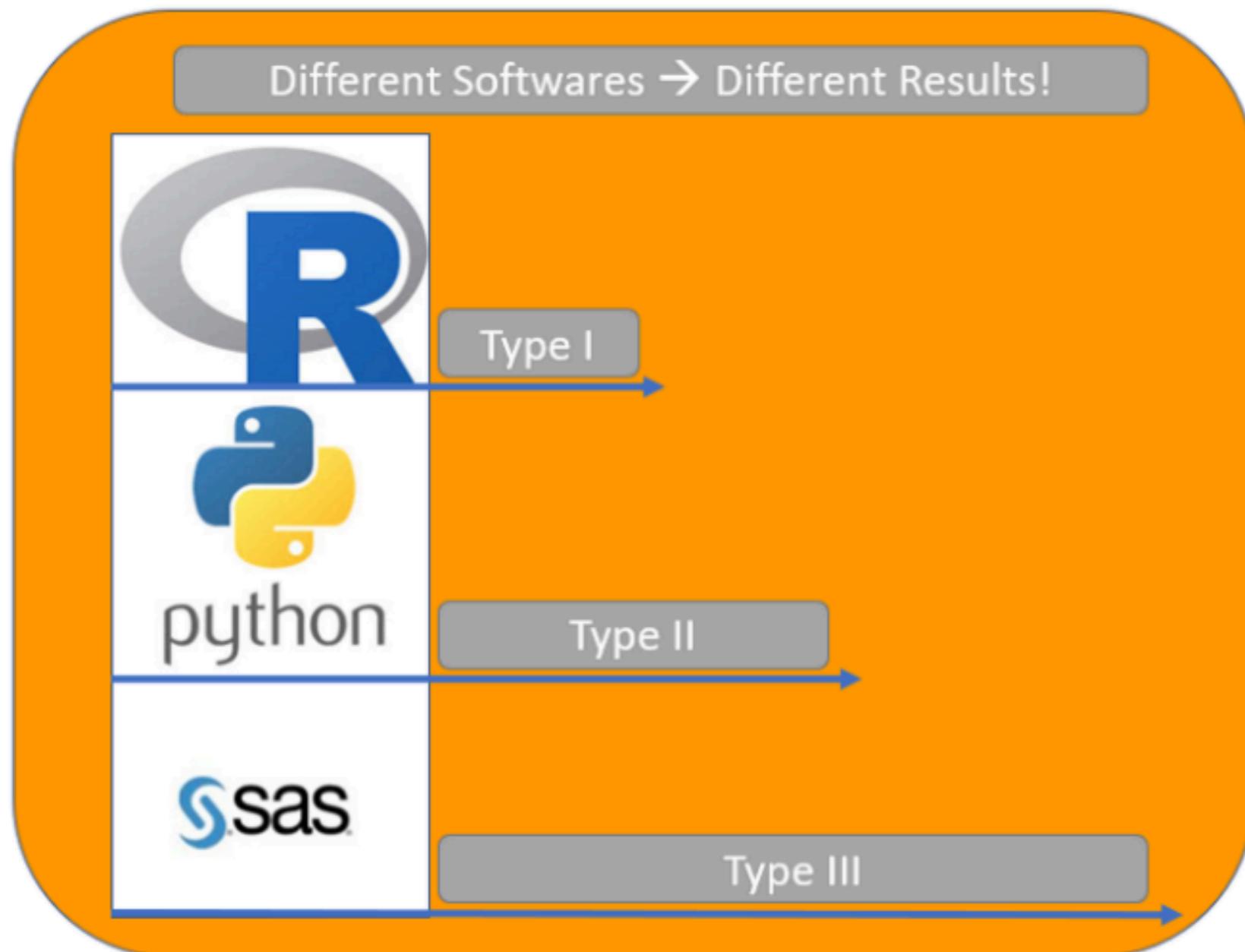
Analysis of Variance Table						
Response: balance						
	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
hand	2	2419.8	1209.92	69.8845	<2e-16	***
skill	1	39.6	39.59	2.2867	0.1316	
Residuals	286	4951.5	17.31			
---						
Signif. codes:	0	'***'	0.001	'**'	0.01	'*'
	0.05	'. '	0.1	' '	1	

# 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!

# 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

# 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"

```
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			

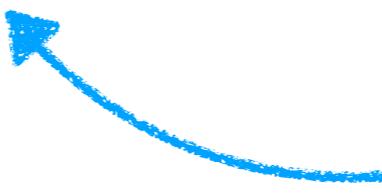
---

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

# 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"

```
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!

# **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.

Power

$$1 - \beta$$

Sensitivity

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

$$\beta$$

Type II error

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

$$\alpha$$

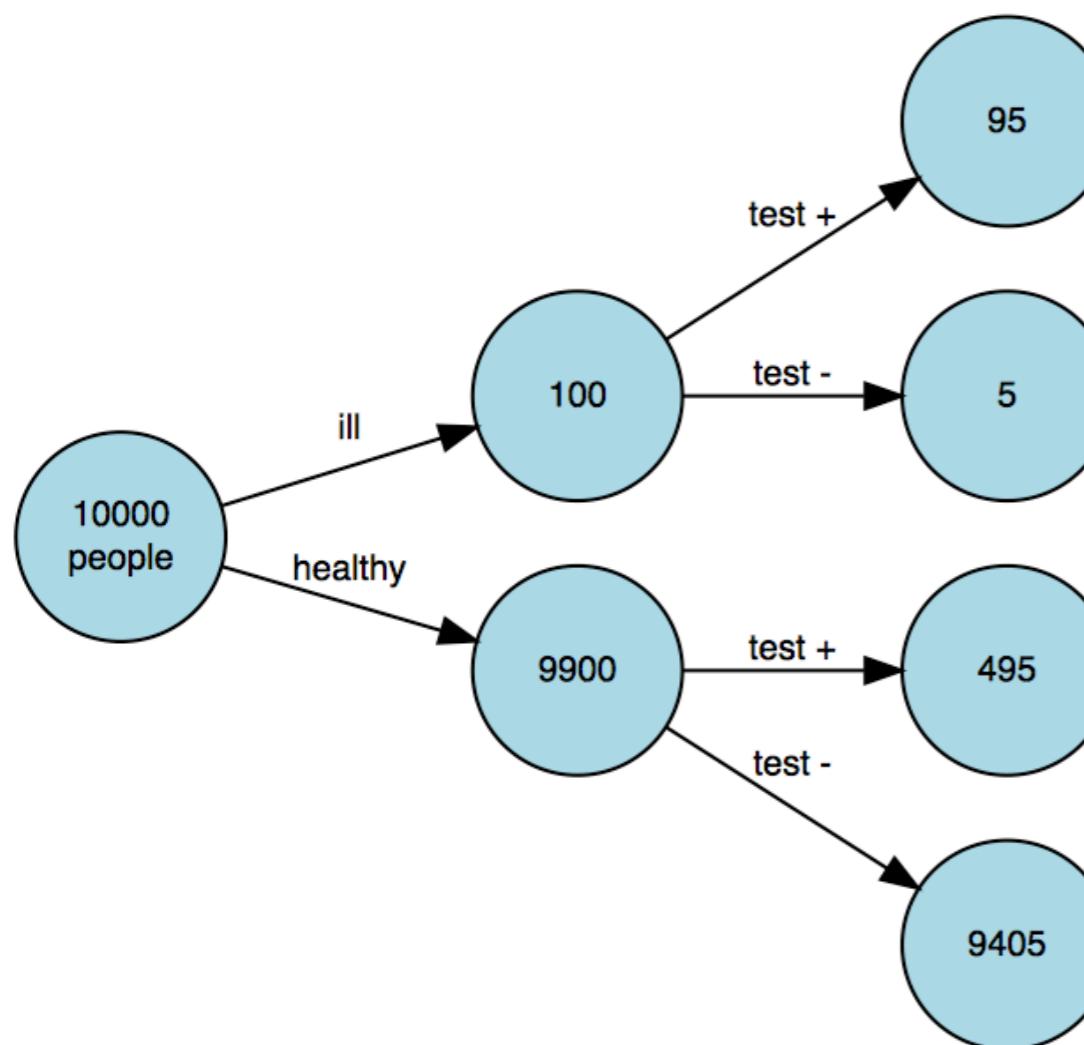
Type I error

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

$$1 - \alpha$$

Specificity

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



true positive

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

false negative

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

false positive

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

true negative

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

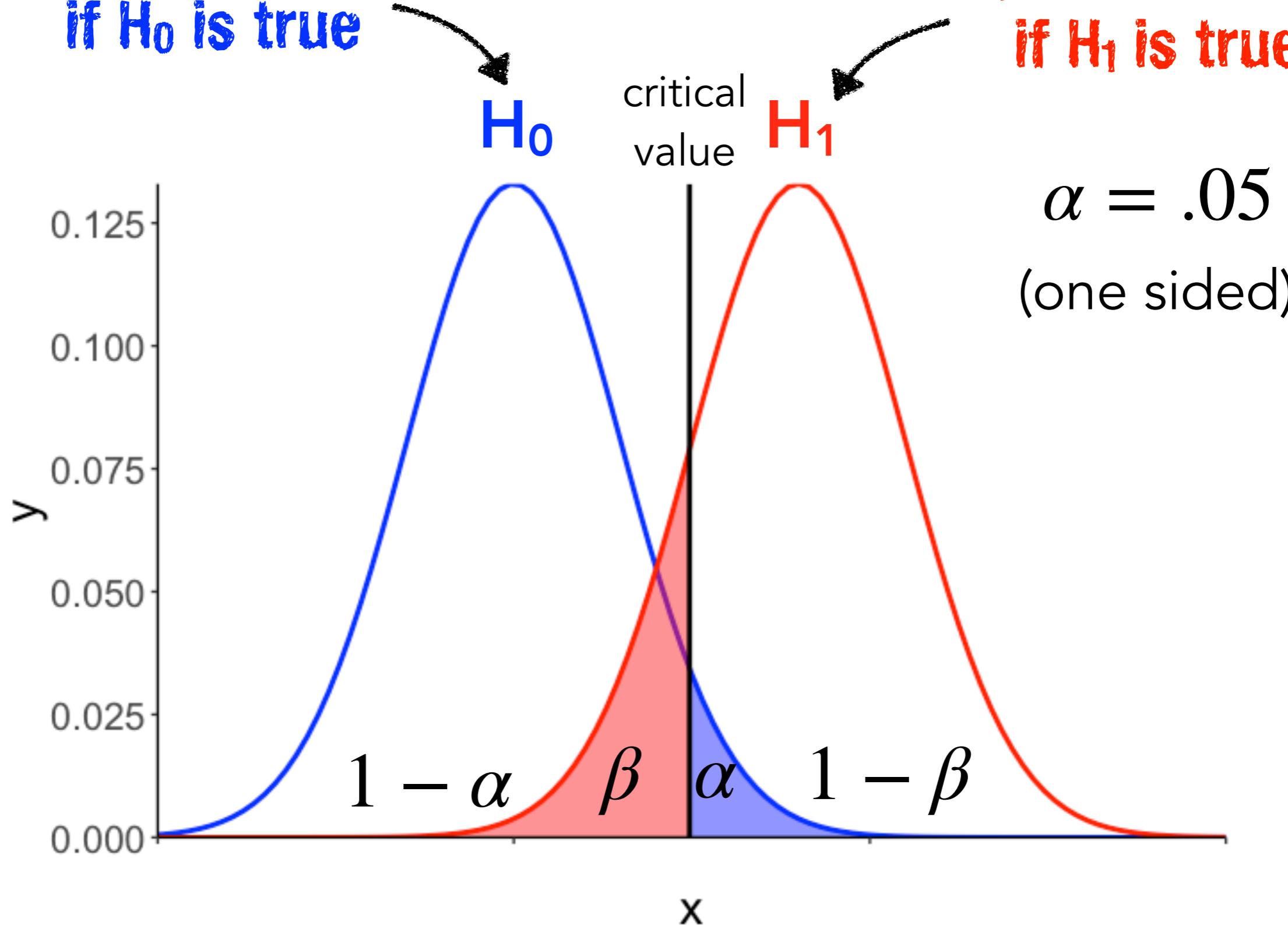
# What affects power?

sampling distribution

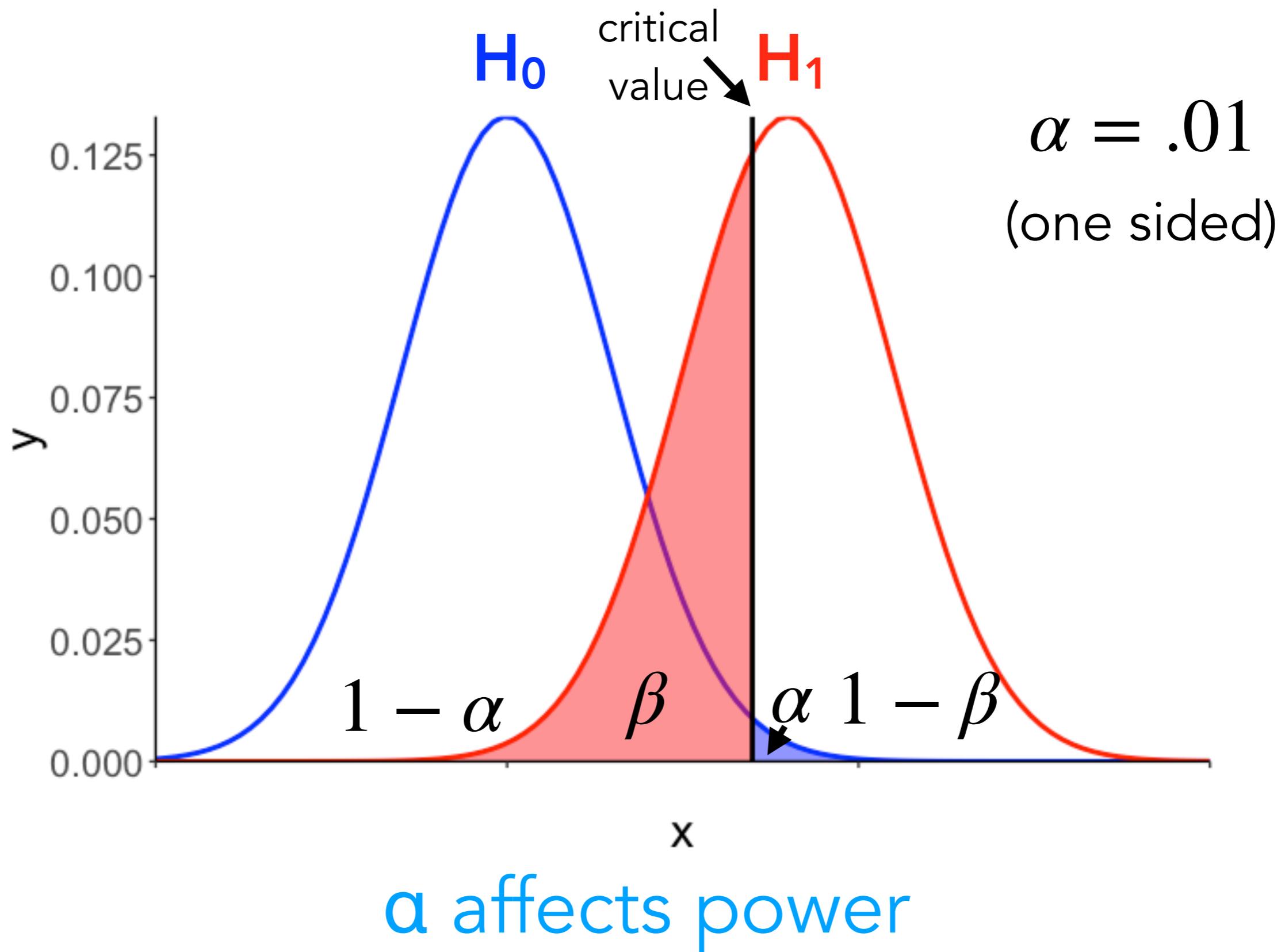
if  $H_0$  is true

sampling distribution

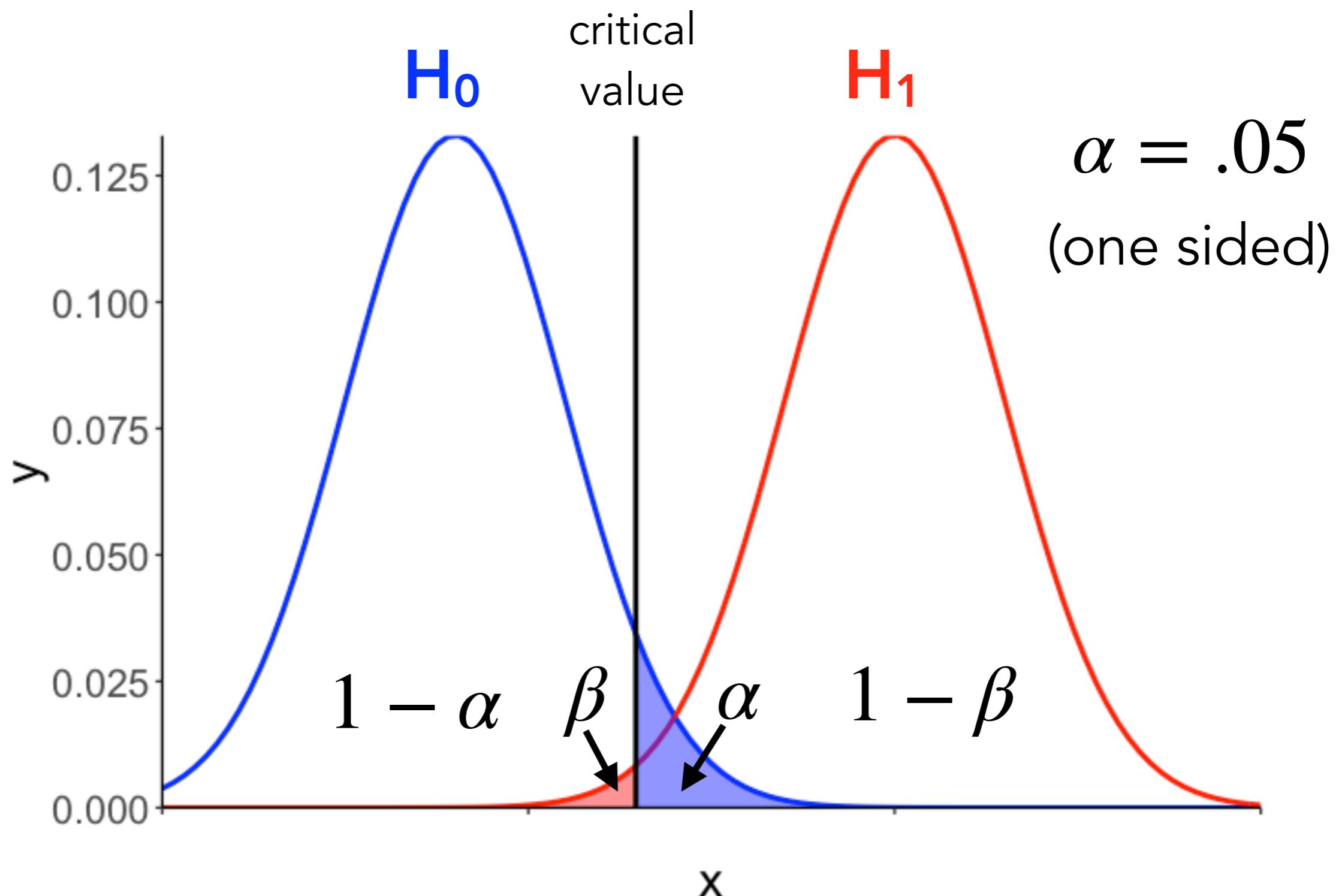
if  $H_1$  is true



# What affects power?

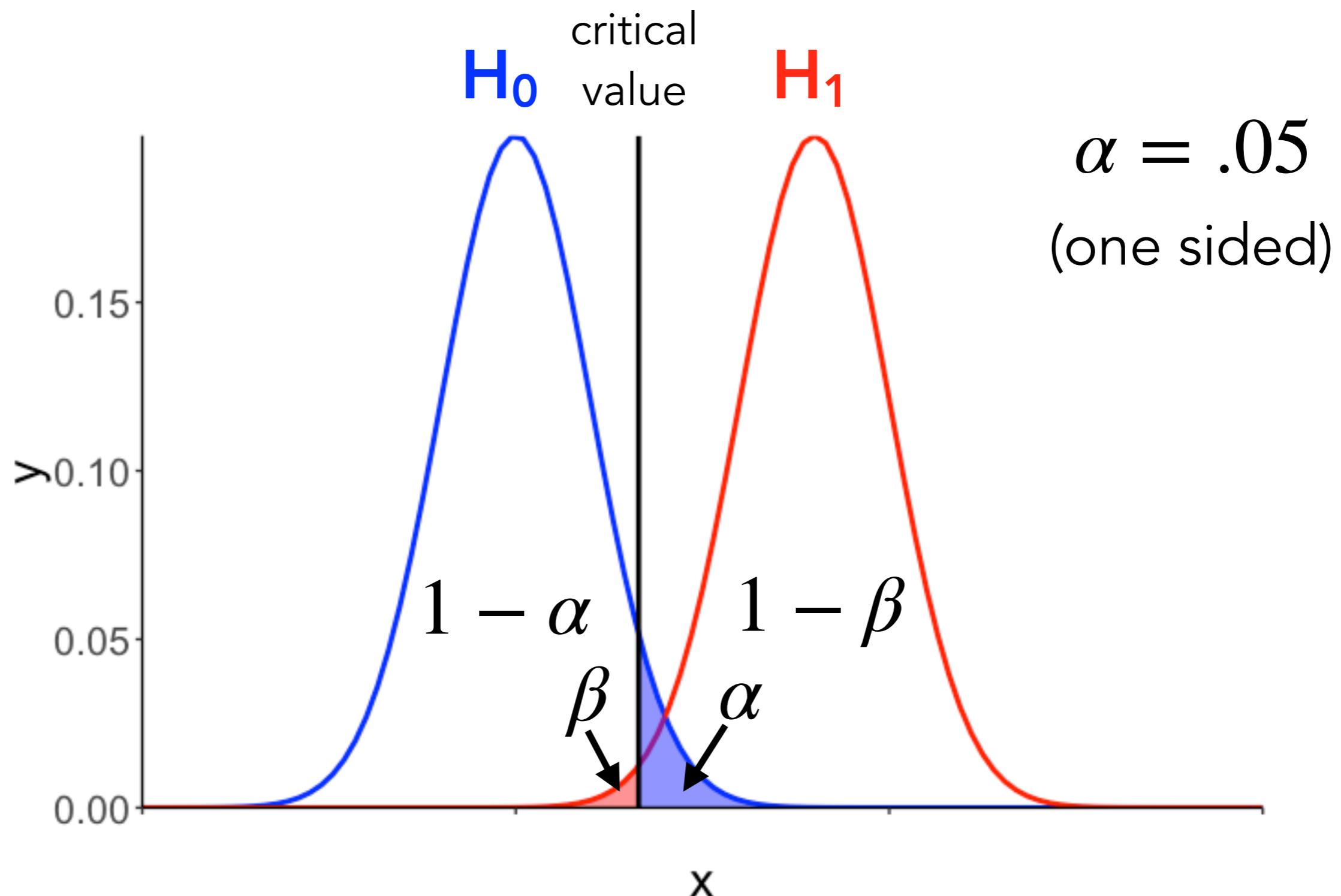


# What affects power?



distance between means affects power

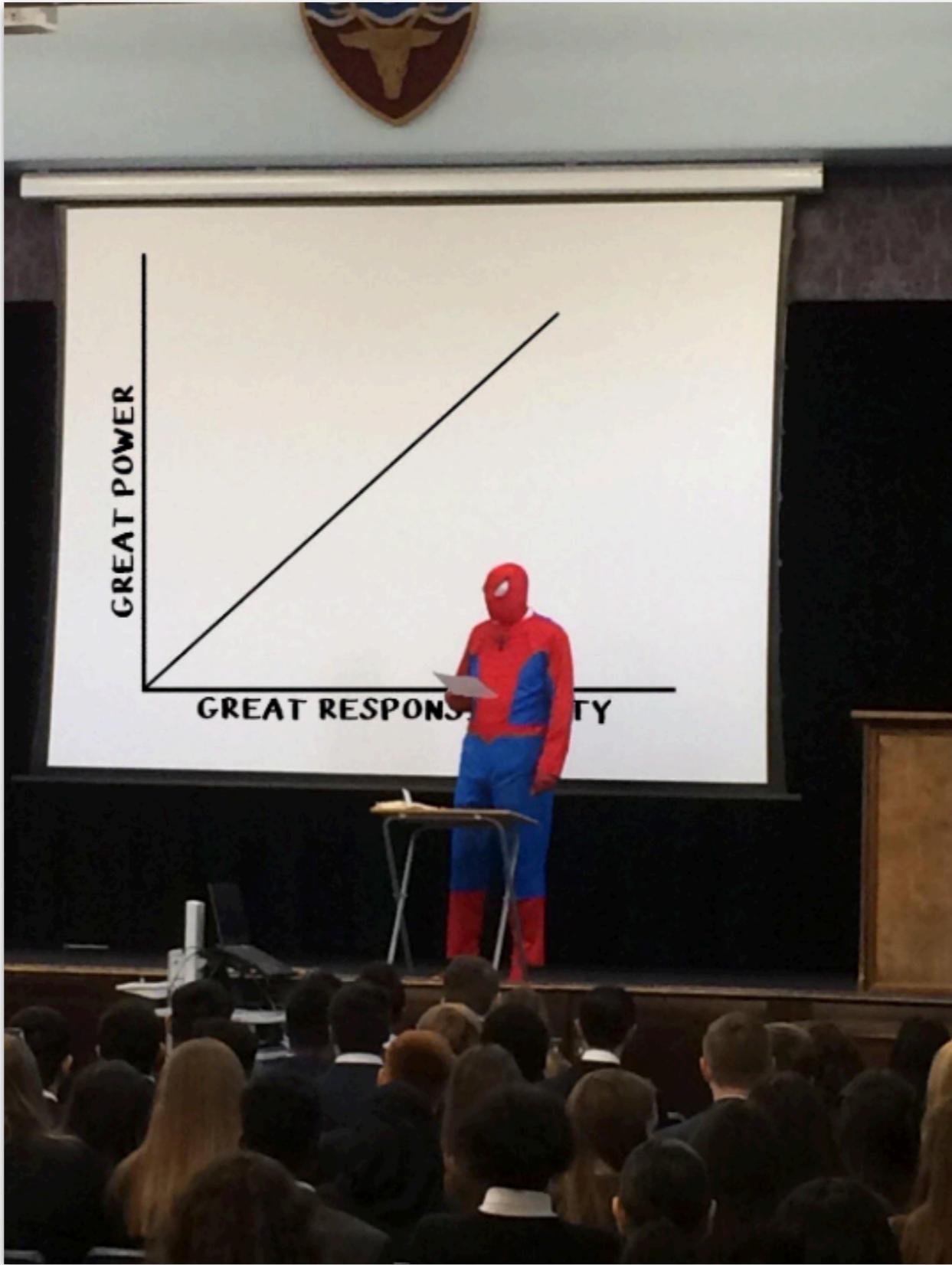
# What affects power?



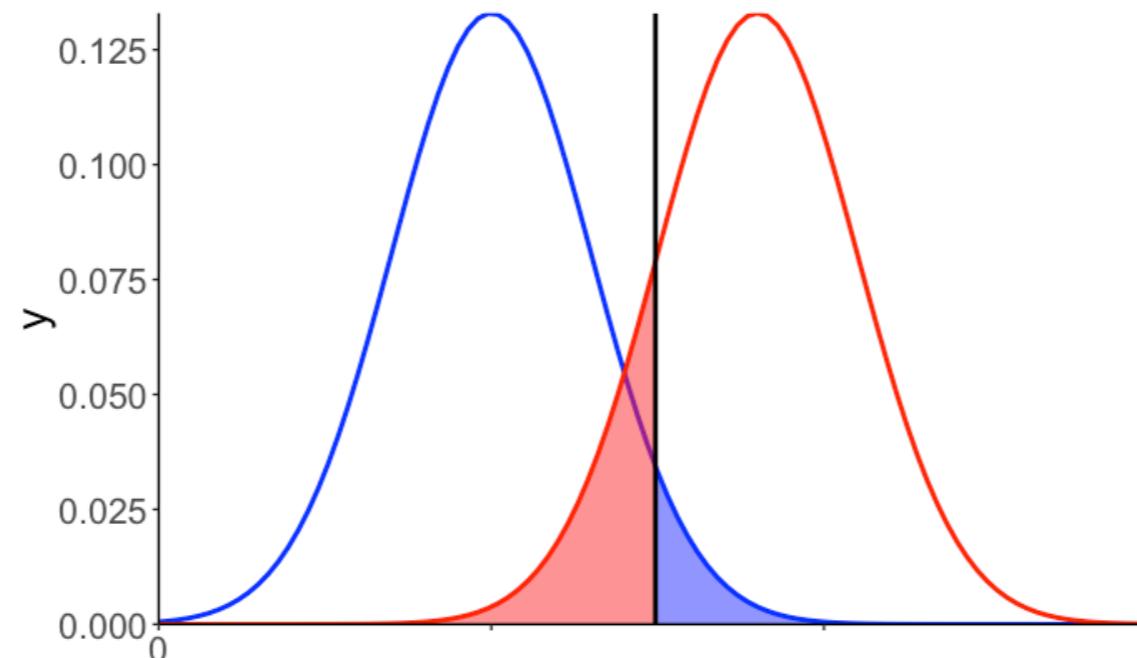
variance affects power

# **Calculating power**

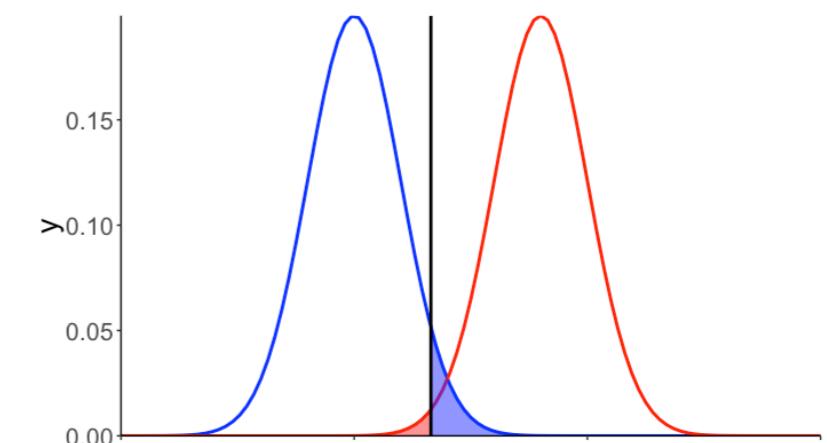
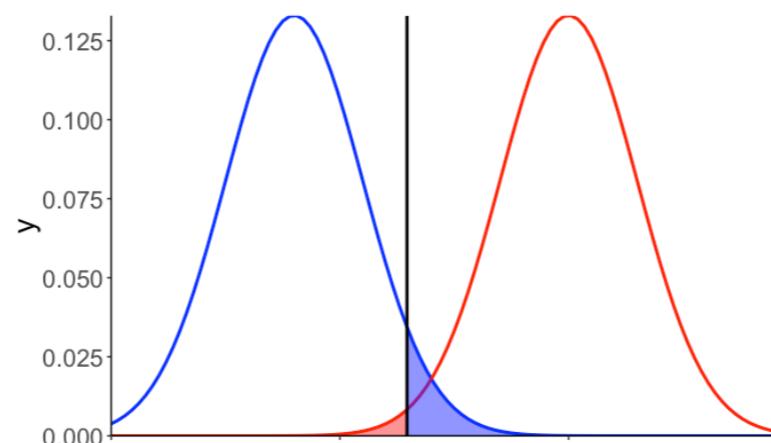
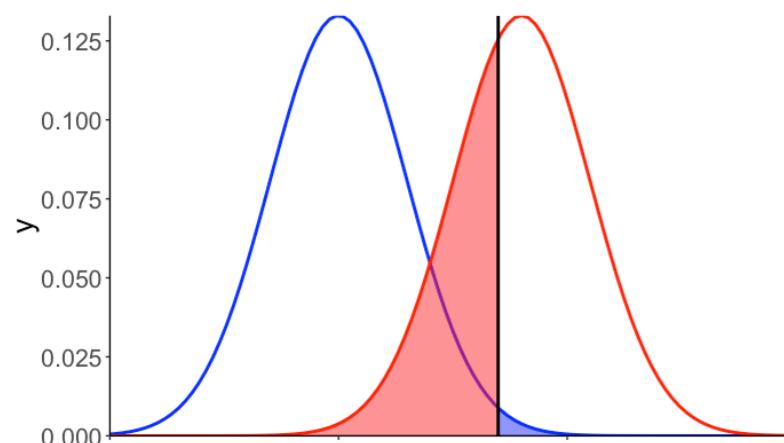
# With great power comes ...



# The knobs we can turn to affect power



$\alpha$       effect size      sample size



# Visualization demo

## Settings

Solve for?  Power  Alpha  n  d

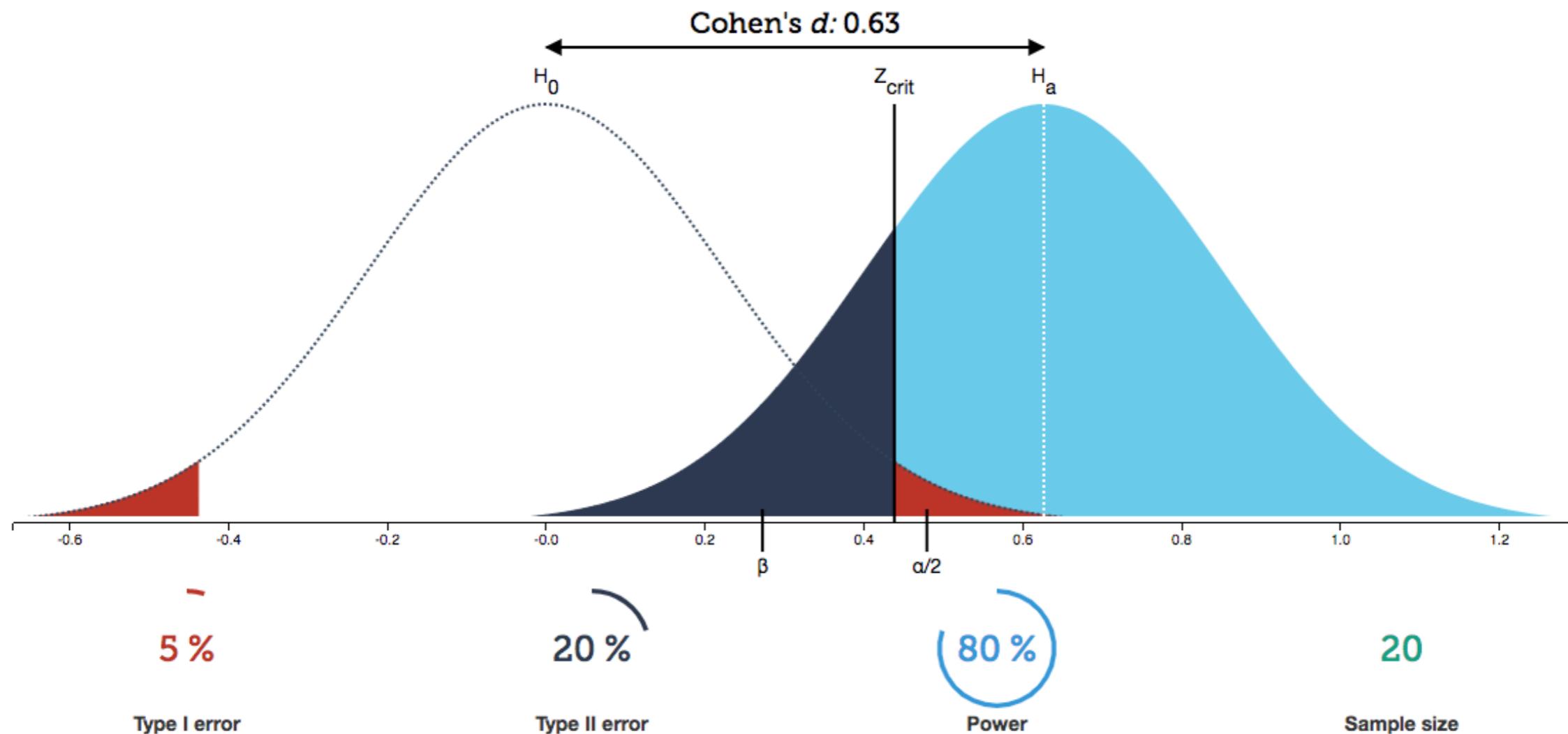
Power ( $1-\beta = 0.8$ )

Significance level ( $\alpha = 0.05$ )

Sample size ( $n = 20$ )

One-tailed  Two-tailed

Reset zoom



<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** = proportional reduction in error

**Compact model**

SSE(C)

**Augmented model**

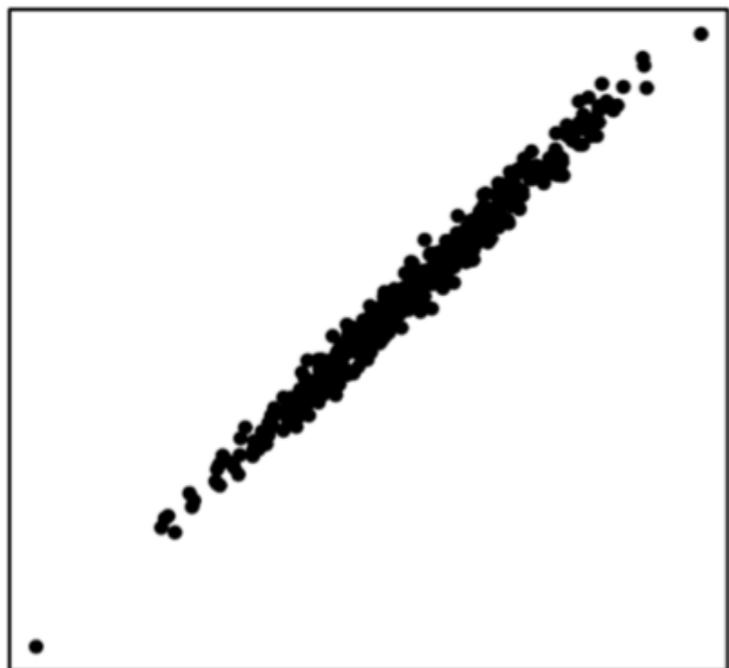
SSE(A)

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

SSE = sum of squared errors

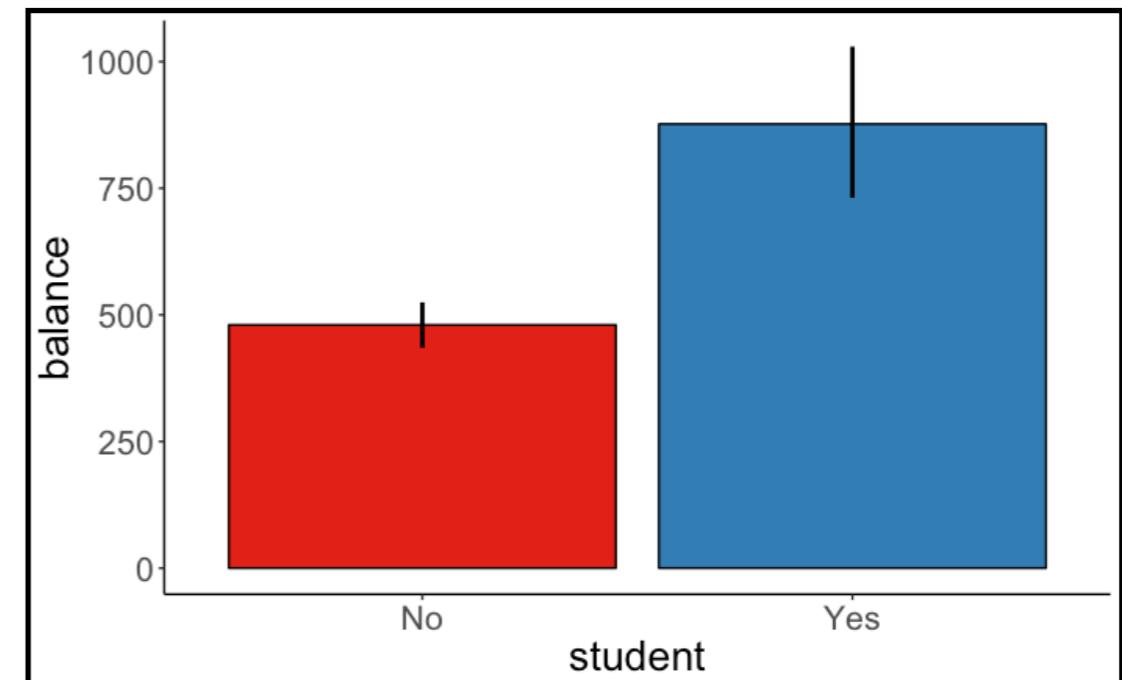
# Common effect sizes

Relationships between variables



$r$  correlation

Differences between groups



Cohen's  $d$

# Correlation

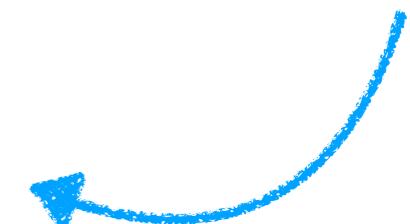
## Pearson correlation

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

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

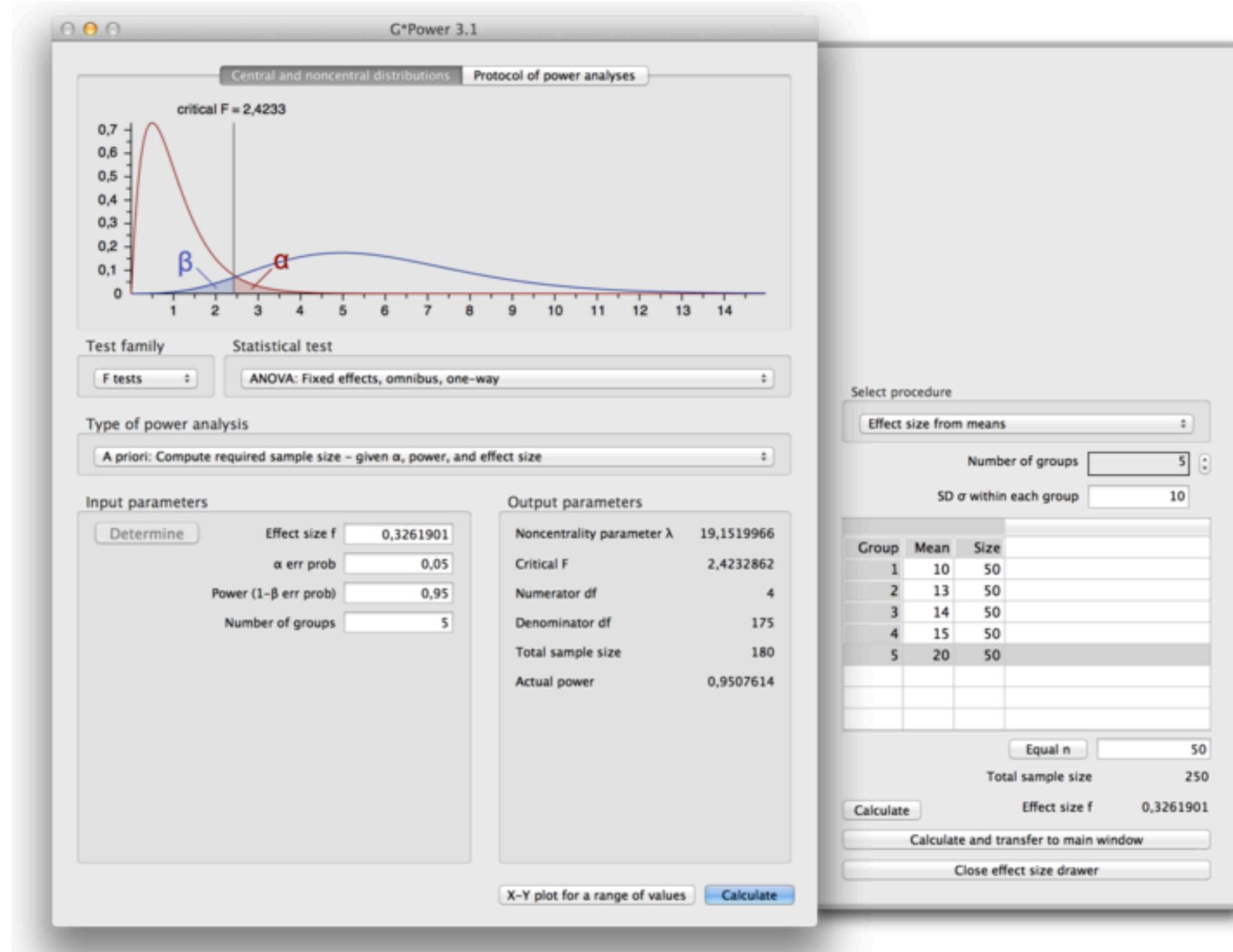
Difference between two means in pooled standard deviation

# 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

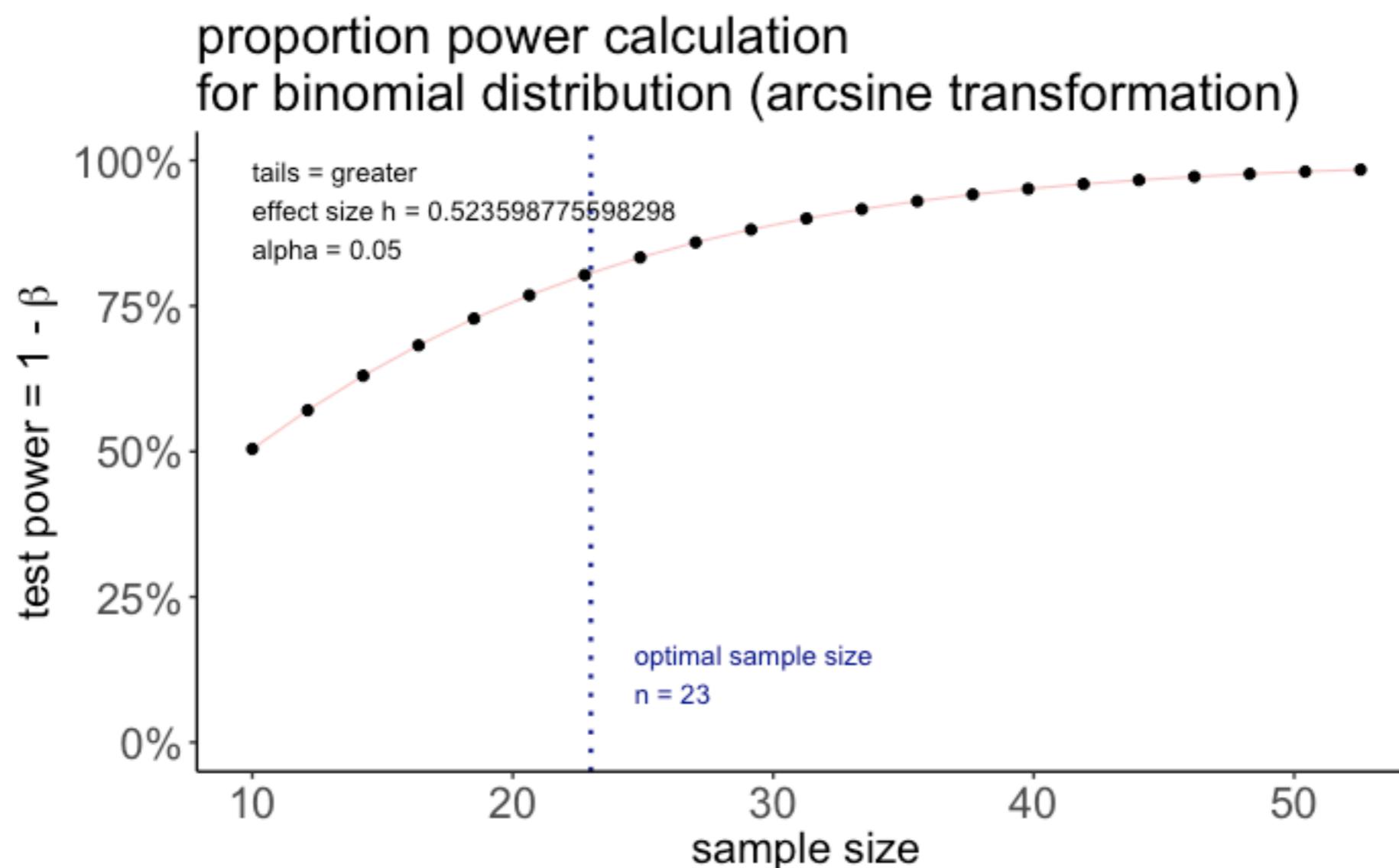
Option 1



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

# "pwr" package in R

```
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:
  - $\alpha$ ,  $n$ , 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

# Let's simulate

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

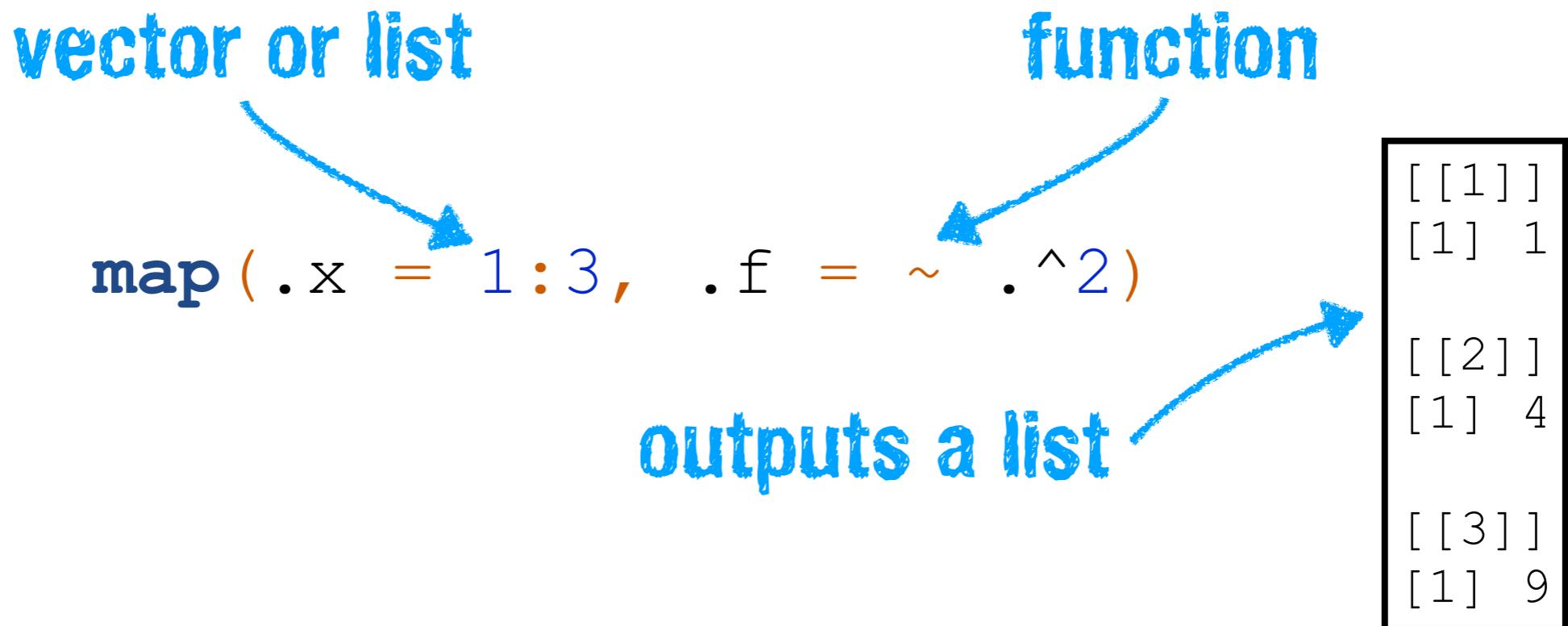


automatically loaded with  
**library**("tidyverse")



**map ( )**

# map()



- `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

```
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)
```

# Power analysis example

# Power simulation recipe

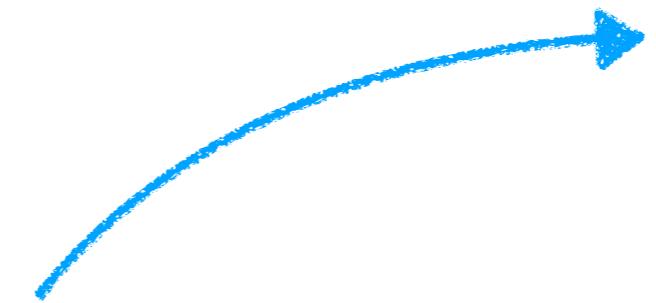
- assume:
  - $\alpha$ ,  $n$ , effect size
- simulate a large number of data sets of size  $n$  with the specified effect size
- for each data set, run a statistical test to calculate the p-value
- determine the probability of rejecting the  $H_0$  (given that  $H_1$  is true)

# Let's simulate ...

```
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, simulation) %>%
14 mutate(response = rbinom(n = n(),
15                         size = n,
16                         prob = p)) %>%
17 group_by(index, simulation, p) %>%
18 nest() %>%
19 mutate(fit = map(.x = data,
20                     .f = ~ binom.test(x = .$response,
21                               n = .$n,
22                               p = 0.5,
23                               alternative = "two.sided")) ) %>%
24 mutate(coef = map(.x = fit,
25                      .f = ~ tidy(.))) %>%
26 select(simulation, p, index, coef) %>%
27 unnest(cols = coef) %>%
28 rename(n = parameter) %>%
29 group_by(n, p) %>%
30 summarize(power = sum(p.value < 0.05) / n()) %>%
31 ungroup()
```

# 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) %>%
```

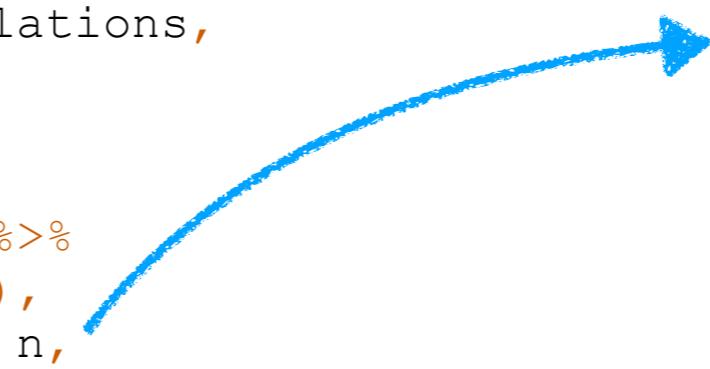


index	n	simulation	p
1	10	1	0.75
2	10	2	0.75
3	10	3	0.75
4	10	4	0.75
5	10	5	0.75
6	12	1	0.75
7	12	2	0.75
8	12	3	0.75
9	12	4	0.75
10	12	5	0.75

# 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, simulation) %>%
14 mutate(response = rbinom(n = n(),
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 ...

```
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, simulation) %>%
14 mutate(response = rbinom(n = n(),
15                         size = n,
16                         prob = p)) %>%
17 group_by(index, simulation, p) %>%
18 nest() %>%
```

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

# Let's simulate ...

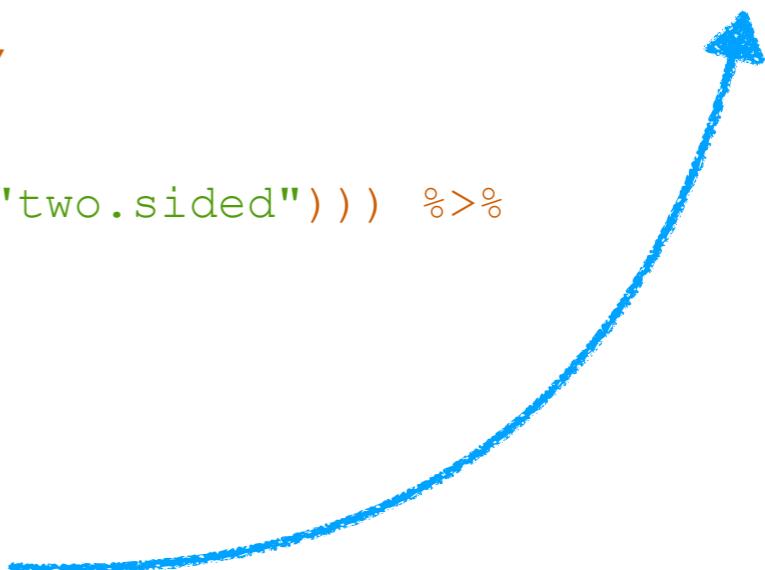
```
1 # make reproducible
2 set.seed(1)
3
4 # number of simulations
5 n_simulations = 5
6
7 # run simulation
8 expand_grid(n = seq(10, 40, 2),
9               simulation = 1:n_simulations,
10              p = 0.75) %>%
11  mutate(index = 1:n(),
12        .before = n) %>%
13  group_by(index, n, simulation) %>%
14  mutate(response = rbinom(n = n(),
15                           size = n,
16                           prob = p)) %>%
17  group_by(index, simulation, p) %>%
18  nest() %>%
19  mutate(fit = map(.x = data,
20                  .f = ~ binom.test(x = .$response,
21                                    n = .$n,
22                                    p = 0.5,
23                                    alternative = "two.sided")) ) %>%
24  mutate(coef = map(.x = fit,
25                    .f = ~ tidy(.))) %>%
26  select(simulation, p, index, coef) %>%
27  unnest(cols = coef) %>%
```

simulation	p	index	estimate	statistic	p.value	n	conf.low	conf.high	method	alternative
1	0.75	1	1.00	10	0.00	10	0.69	1.00	Exact binomial test	two.sided
2	0.75	2	0.70	7	0.34	10	0.35	0.93	Exact binomial test	two.sided
3	0.75	3	0.60	6	0.75	10	0.26	0.88	Exact binomial test	two.sided
4	0.75	4	0.70	7	0.34	10	0.35	0.93	Exact binomial test	two.sided
5	0.75	5	0.80	8	0.11	10	0.44	0.97	Exact binomial test	two.sided

# 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, simulation) %>%
14 mutate(response = rbinom(n = n(),
15                         size = n,
16                         prob = p)) %>%
17 group_by(index, simulation, p) %>%
18 nest() %>%
19 mutate(fit = map(.x = data,
20                     .f = ~ binom.test(x = .\$response,
21                               n = .\$n,
22                               p = 0.5,
23                               alternative = "two.sided") )) %>%
24 mutate(coef = map(.x = fit,
25                      .f = ~ tidy(.))) %>%
26 select(simulation, p, index, coef) %>%
27 unnest(cols = coef) %>%
28 rename(n = parameter) %>%
29 group_by(n, p) %>%
30 summarize(power = sum(p.value < 0.05) / n()) %>%
31 ungroup()
```

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 ...

- 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:
  - $\alpha$ ,  $n$ , 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  $\alpha$
- determine the probability of rejecting the  $H_0$  (given that  $H_1$  is true)

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!

This course is part of these tracks:

Intermediate Tidyverse Toolbox

### 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 |



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

will share a 25  
minute video lecture

# **Feedback**

# How was the pace of today's class?

much      a little      just      a little      much  
too      too      right      too      too  
slow      slow

# 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!