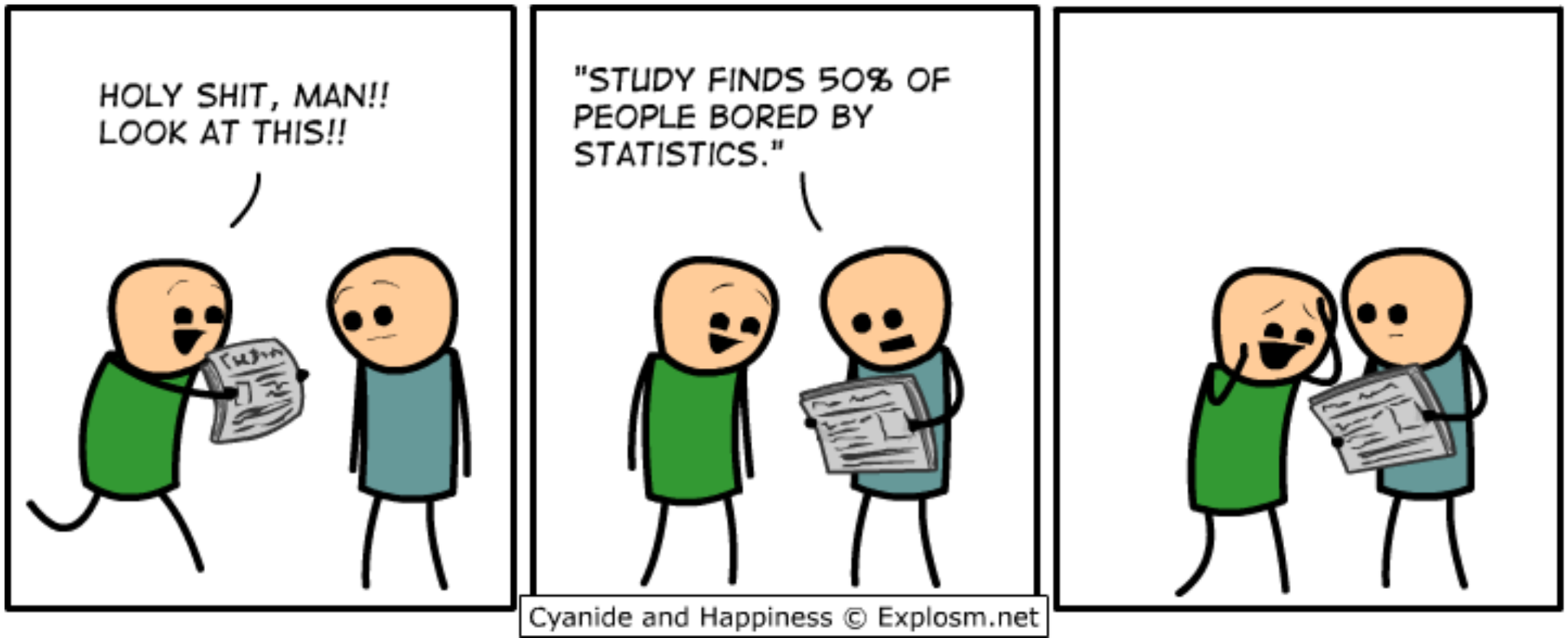
Linear model 4

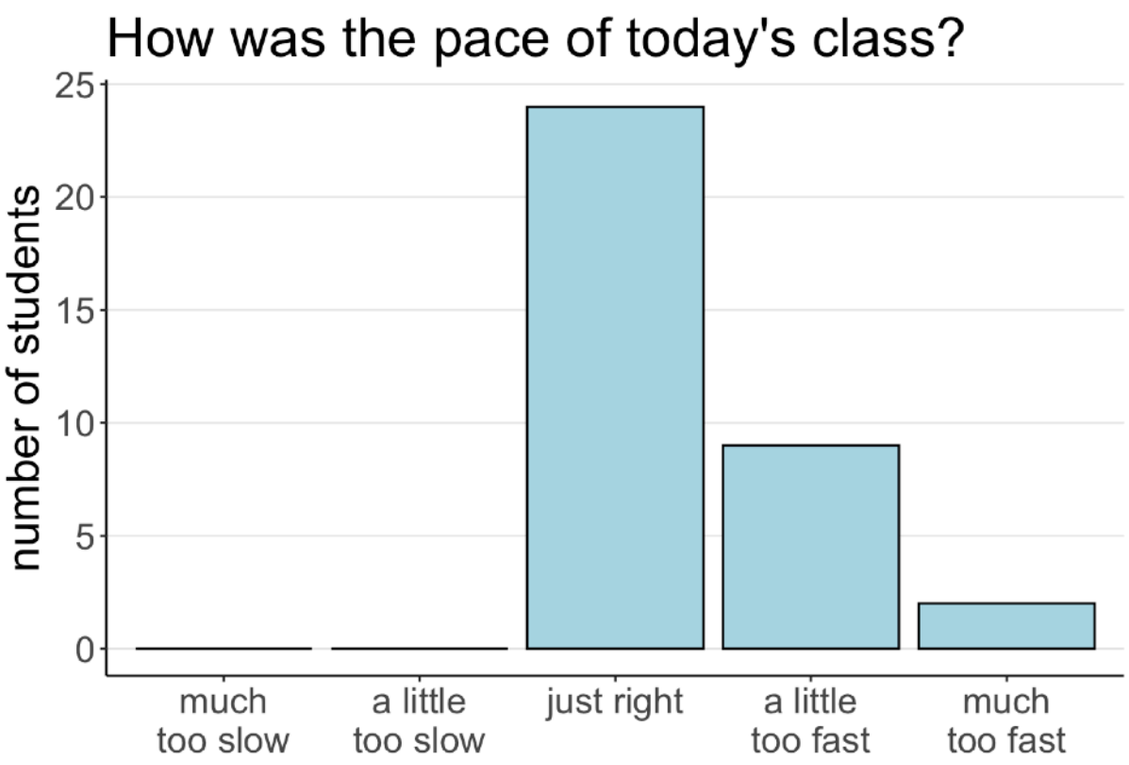


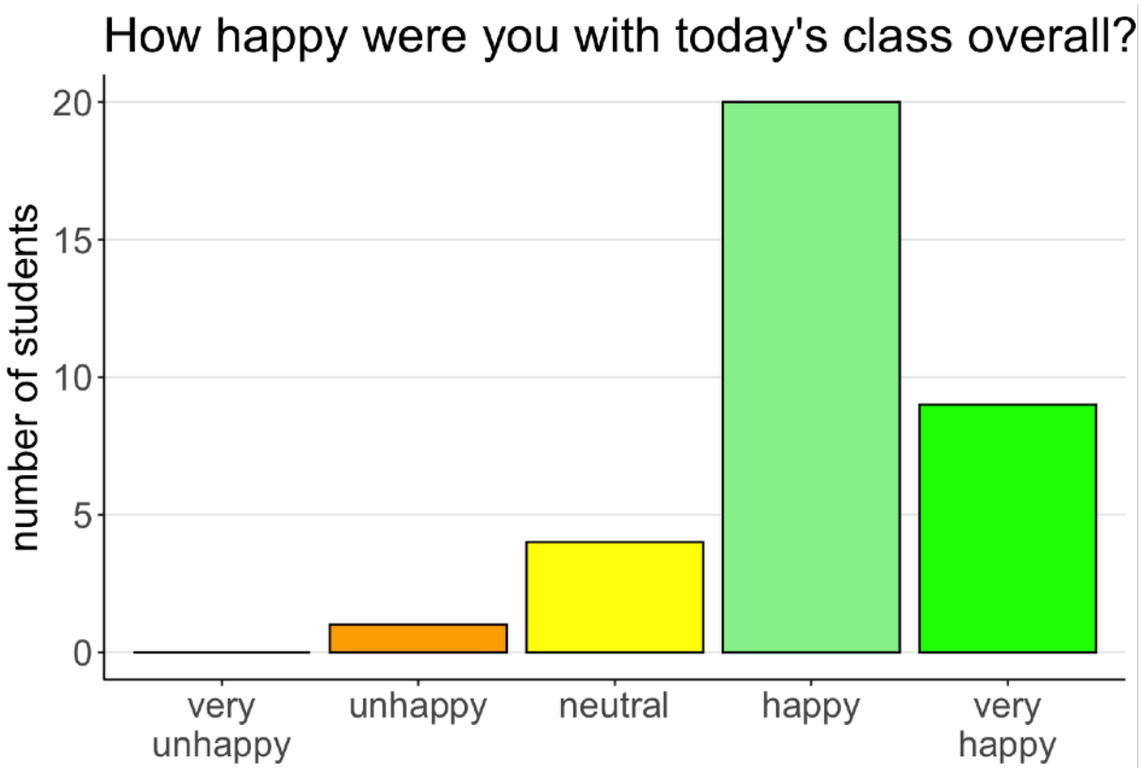
02/05/2020

Logistics

Feedback

Your feedback





still a little too fast

Your feedback

The past few classes have been incredibly helpful for thinking through the computations that underlie regression models and ANOVAs. I've encountered these in past courses, but I feel like I only now really understand what these tests are doing.

Really like going through the same visual representations of model a vs. model c and then code chunks in the presentation! It's very helpful and helps me make sense of it over a few classes. Also like the model comparison approach to stats

Your feedback

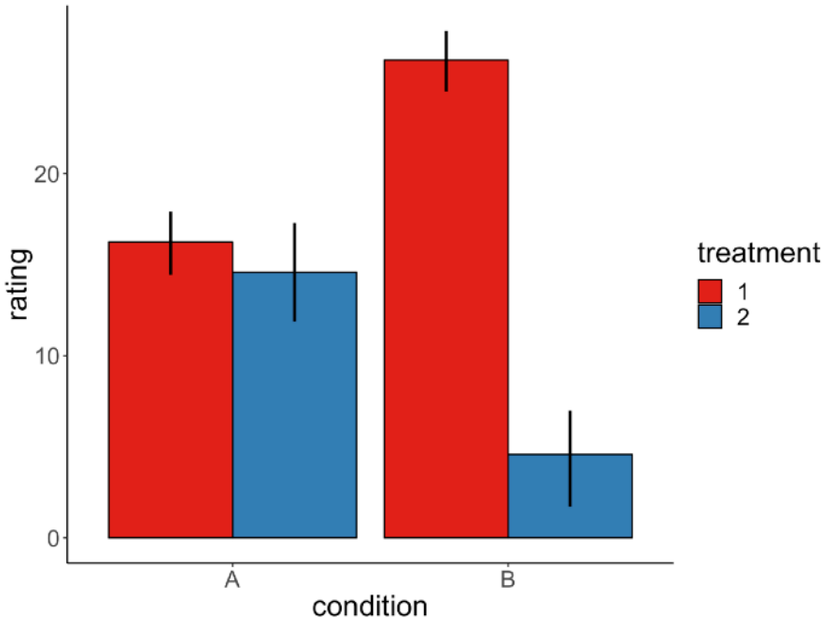
I was shocked that I was so bad at guessing which effects were significant in the interaction plots!

Your feedback

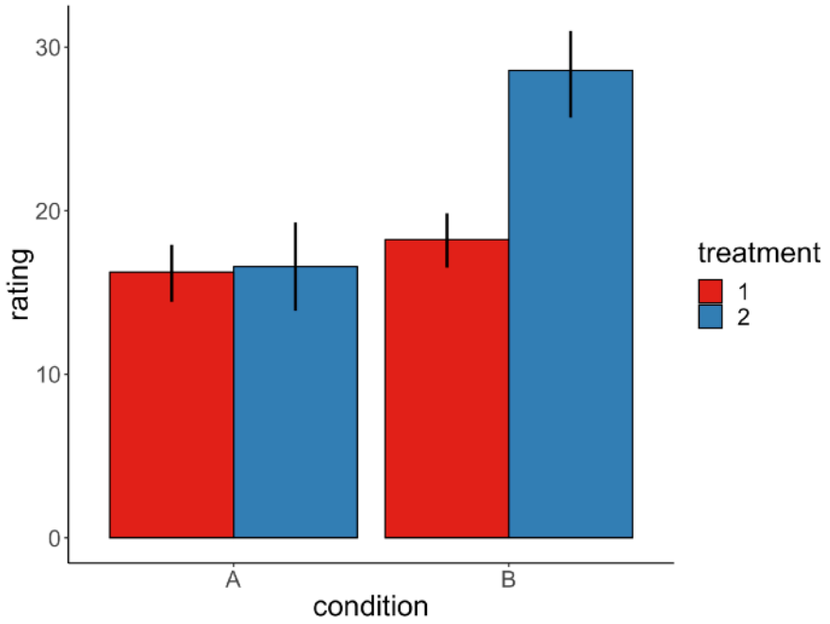
I felt like with some of the content, the key understanding was not put in the slides. Like the game with the bars. The explanation was verbal, and reviewing the slides will be less useful

Solution

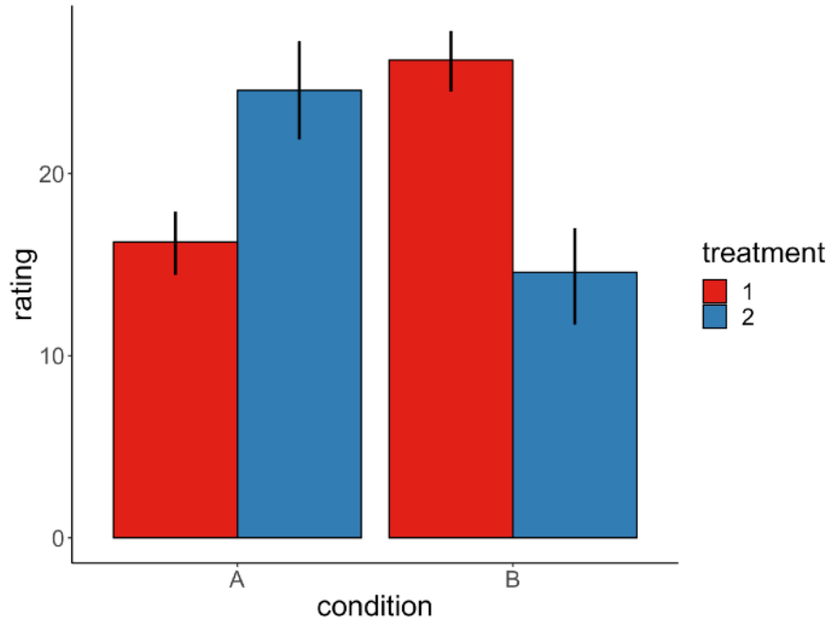
Treatment
Condition x Treatment



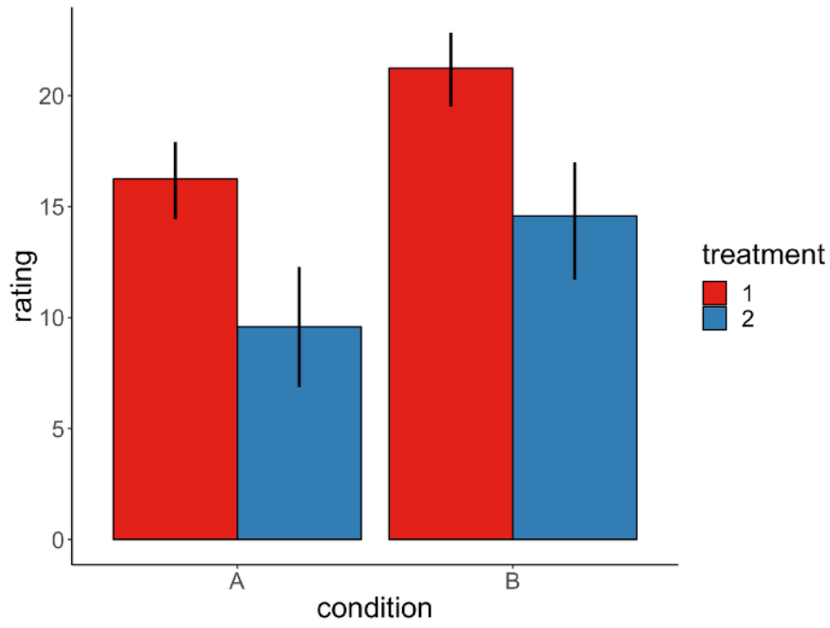
Condition,
Treatment,
Condition x Treatment



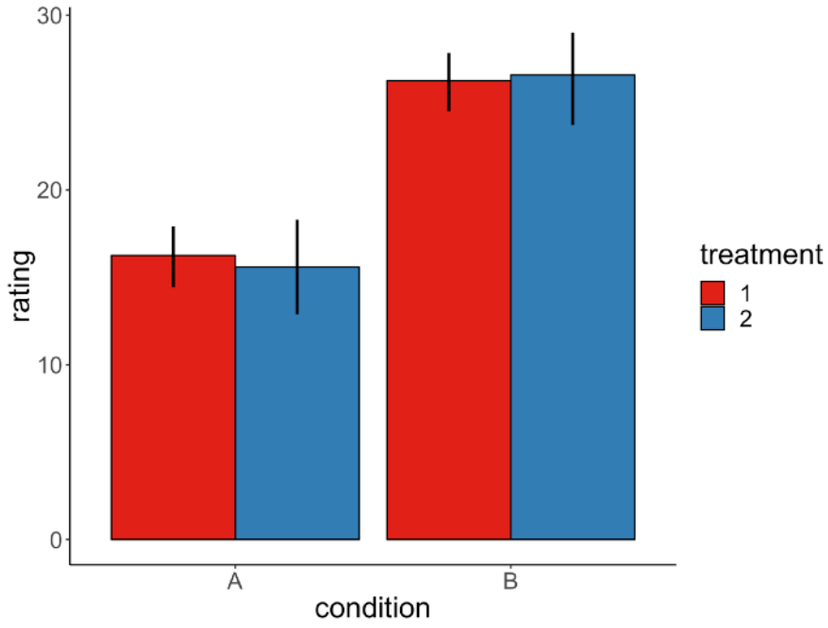
Condition x Treatment



Condition
Treatment

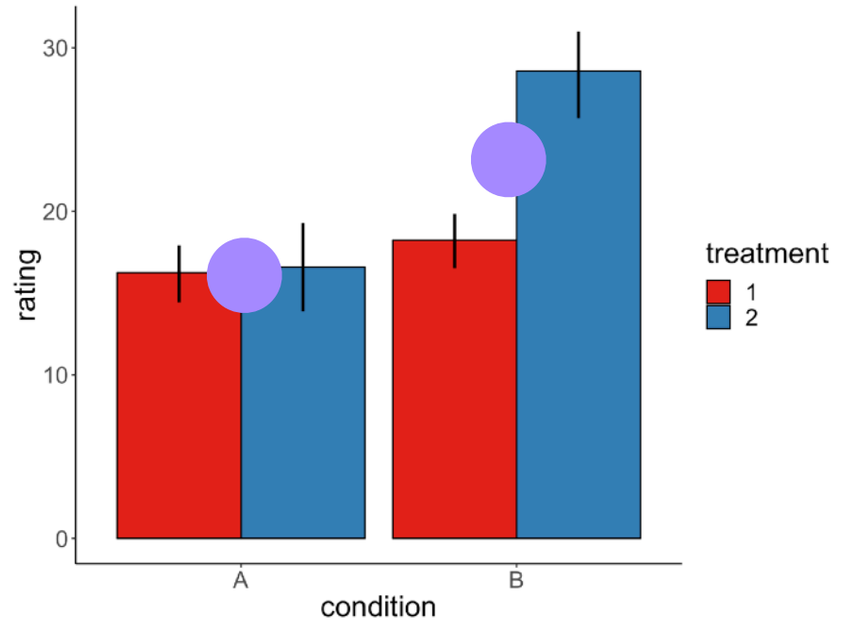


Condition



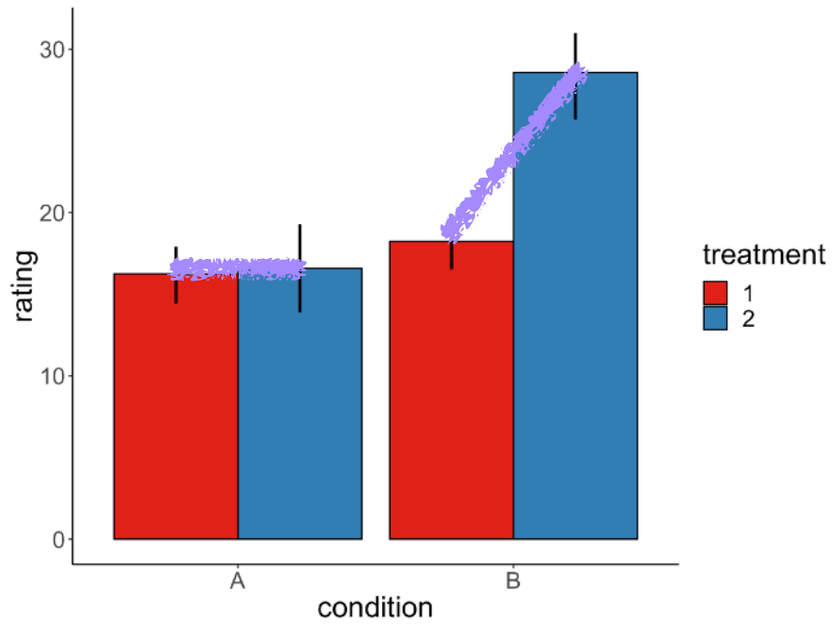
Solution

to detect main effects, try to visualize what the averaged group means would look like



main effect of condition

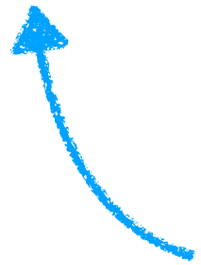
to detect interaction effects, try to visualize whether the slopes are different from each other



interaction effect

Application section

Applications of the linear model



good practice for the midterm!

Midterm

Midterm

Will be released on Friday 7th (after class),
and will be due on **Thursday 13th at 8pm**.

- A (slightly longer) homework assignment.
- You have to work on it on your own.
- It counts a little more. (20% of your final grade)

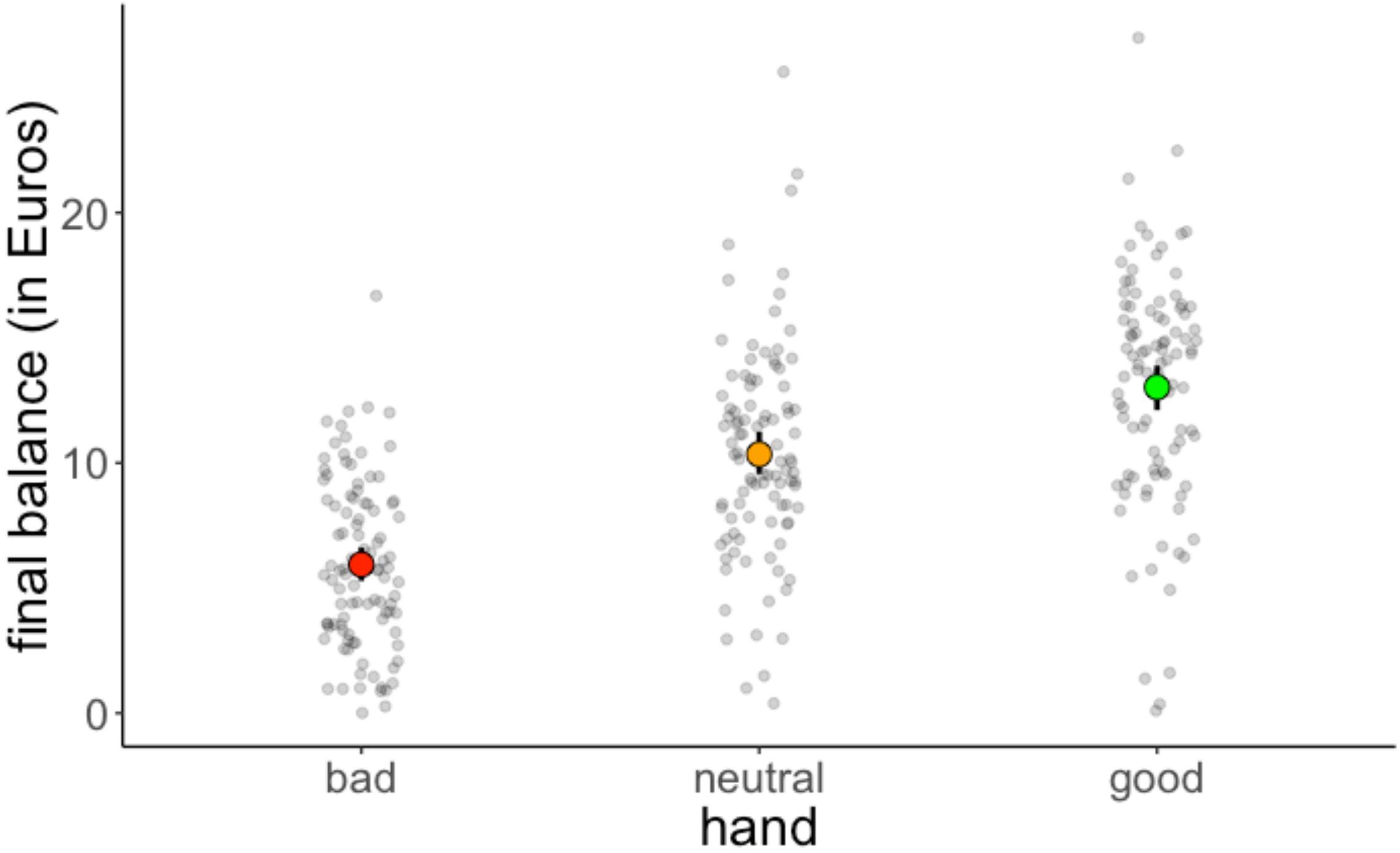
We won't have sections next week.

There will be **no class** on Wednesday 12th next week.

Plan for today

- Contrasts
- Planned comparisons
- Making decisions
- Power analysis
- Effect sizes
- Determining sample size

Do better hands win more money?



Do better hands win more money?

ANOVA

Does card quality affect the final balance?

post-hoc tests

bad vs. **neutral**

neutral vs. **good**

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

Contrasts

```
1 df.poker = df.poker %>%
2   mutate(hand_contrast = factor(hand,
3                                 levels = c("bad", "neutral", "good"),
4                                 labels = c(-1, 0, 1)),
5          hand_contrast = hand_contrast %>% as.character() %>% as.numeric())
```

participant	hand	balance	hand_contrast
1	bad	4.00	-1
2	bad	5.55	-1
3	bad	9.45	-1
51	neutral	11.74	0
52	neutral	10.04	0
53	neutral	9.49	0
101	good	10.86	1
102	good	8.68	1
103	good	14.36	1

Contrasts

```r
df.poker = df.poker %>%
  mutate(hand_contrast = factor(hand,
                                levels = c("bad", "neutral", "good"),
                                labels = c(-1, 0, 1)),
         hand_contrast = hand_contrast %>% as.character() %>% as.numeric())

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

```
Call:
lm(formula = balance ~ hand_contrast, data = df.fit)

Residuals:
    Min      1Q  Median      3Q     Max
-13.214  -2.684  -0.019   2.444  15.858

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    9.7715     0.2381   41.03   <2e-16 ***
hand_contrast  3.5424     0.2917   12.14   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.125 on 298 degrees of freedom
Multiple R-squared:  0.3311,  Adjusted R-squared:  0.3289
F-statistic: 147.5 on 1 and 298 DF,  p-value: < 2.2e-16
```

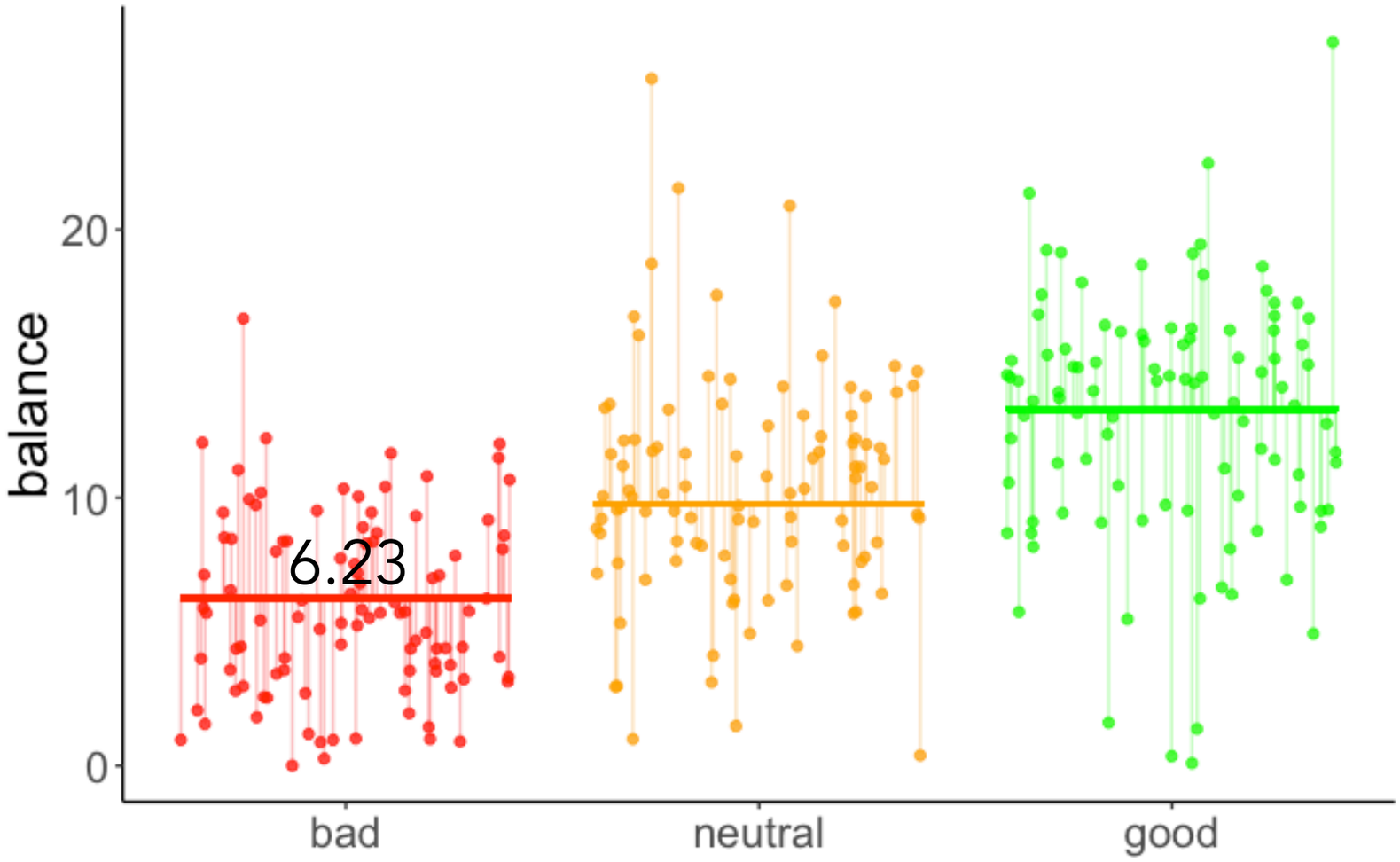
grand mean

significant contrast

Contrasts

name	estimate	std.error	statistic	p.value
intercept	9.77	0.24	41.03	0
contrast	3.54	0.29	12.15	0

$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}$$



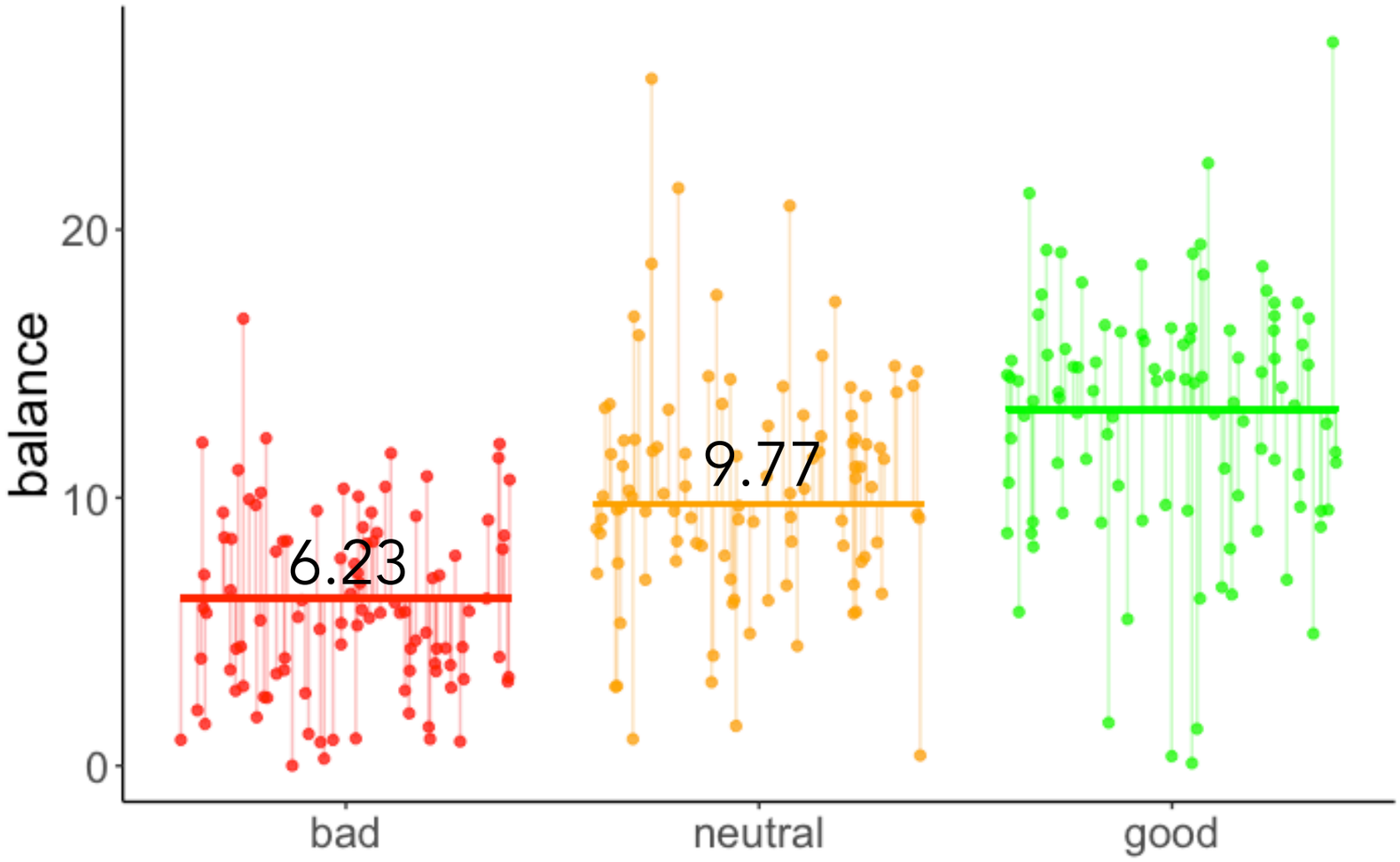
if contrast == -1

$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}_i$$
$$= 9.77 + (-1) \cdot 3.54 = 6.23$$

Contrasts

name	estimate	std.error	statistic	p.value
intercept	9.77	0.24	41.03	0
contrast	3.54	0.29	12.15	0

$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}$$



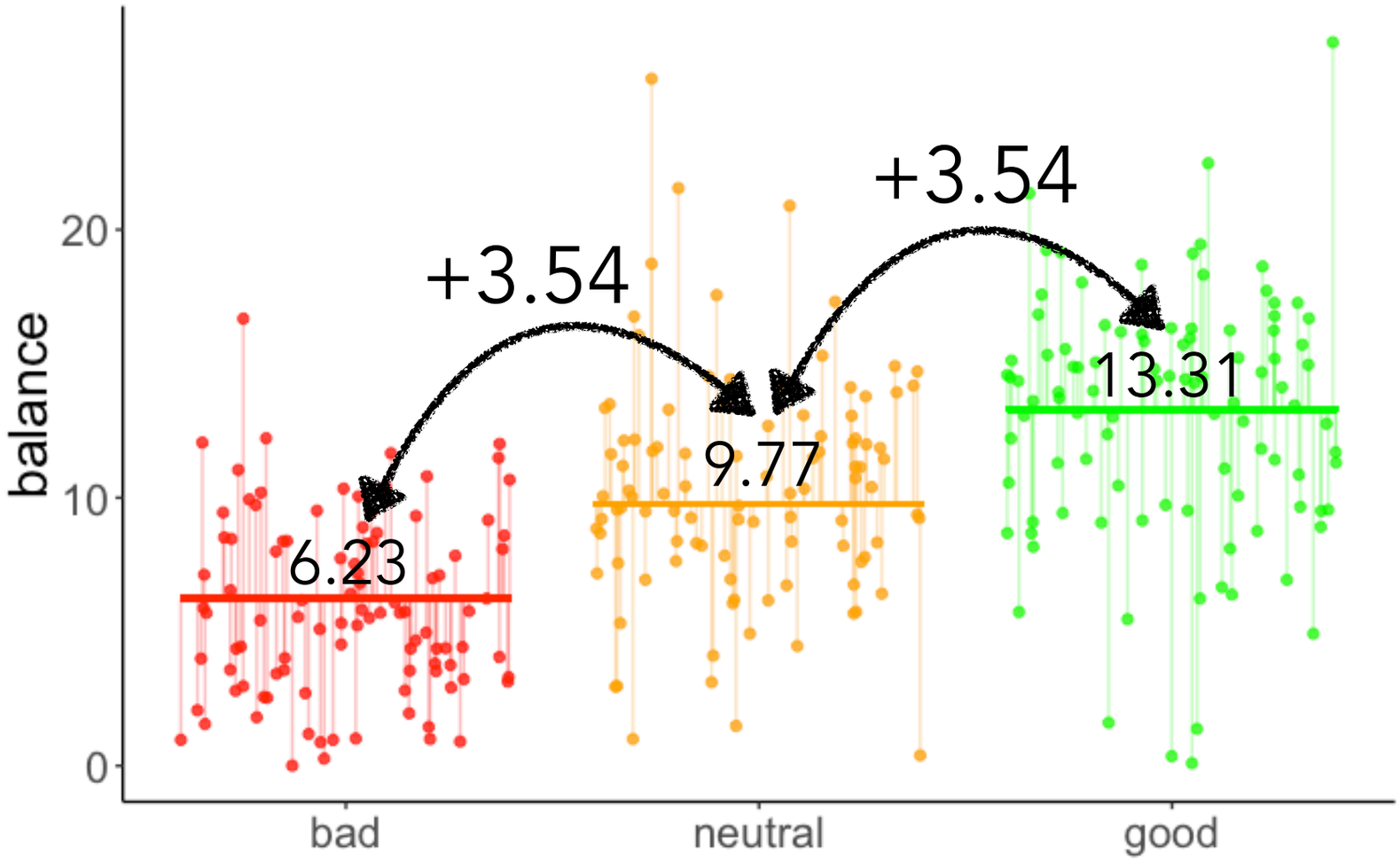
if contrast == 0

$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}_i$$
$$= 9.77 + 0 \cdot 3.54 = 9.77$$

Contrasts

name	estimate	std.error	statistic	p.value
intercept	9.77	0.24	41.03	0
contrast	3.54	0.29	12.15	0

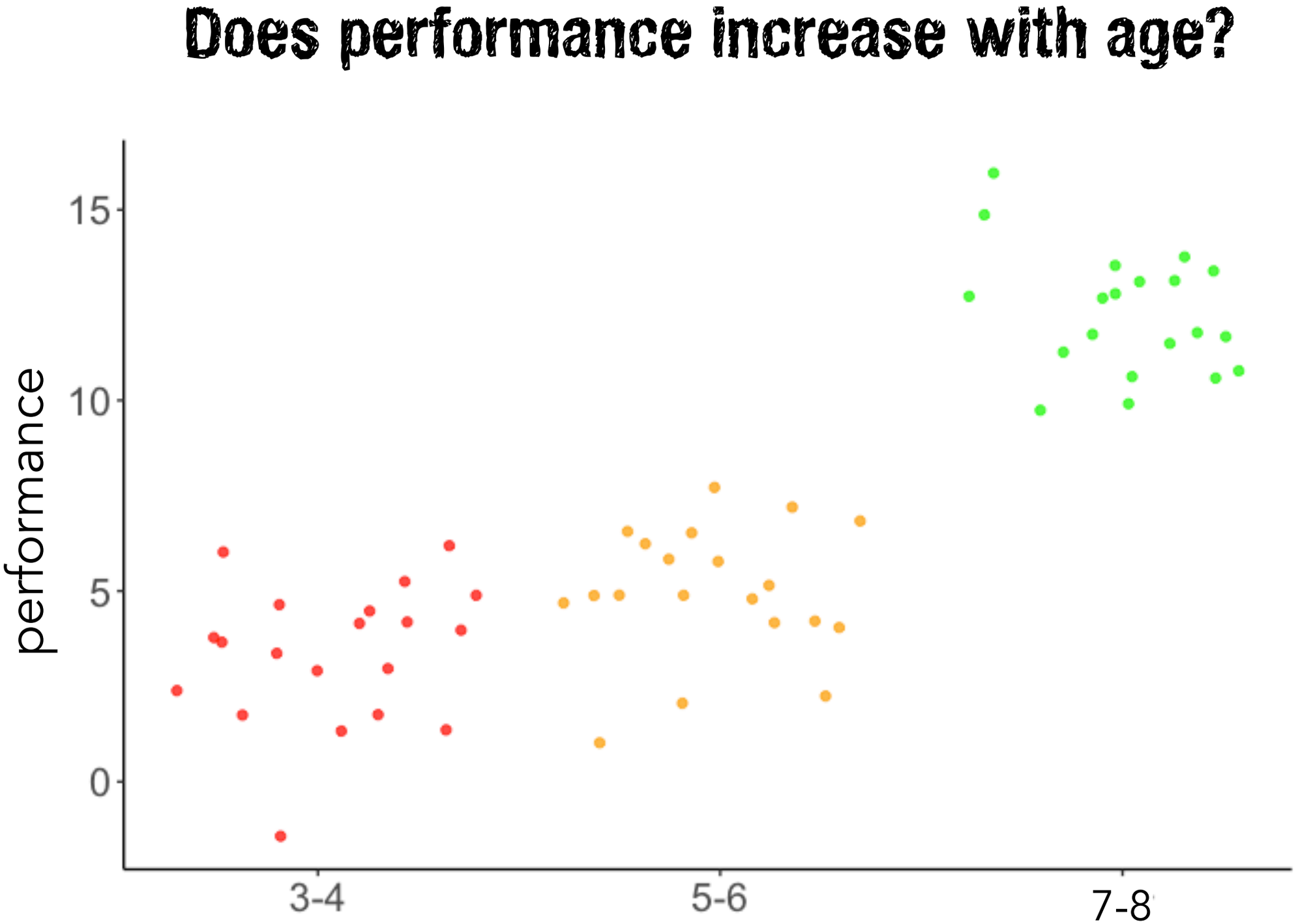
$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}$$



if contrast == 1

$$\widehat{\text{balance}}_i = b_0 + b_1 \cdot \text{hand\_contrast}_i$$
$$= 9.77 + 1 \cdot 3.54 = 13.31$$

Contrasts



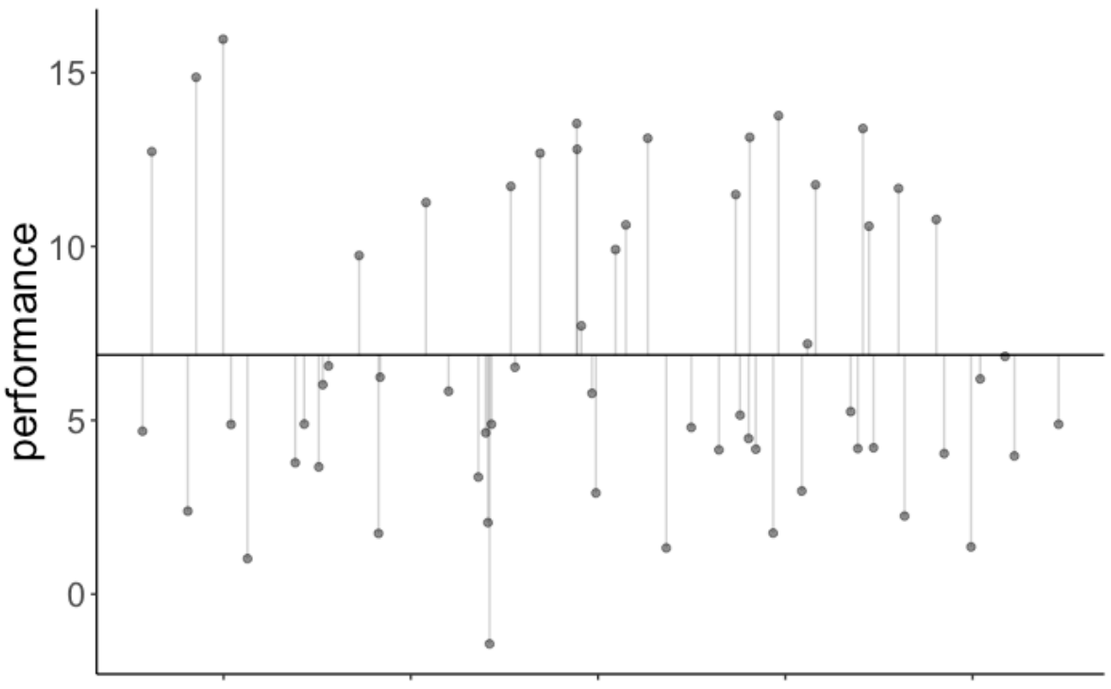
Data from a hypothetical developmental study

Contrasts

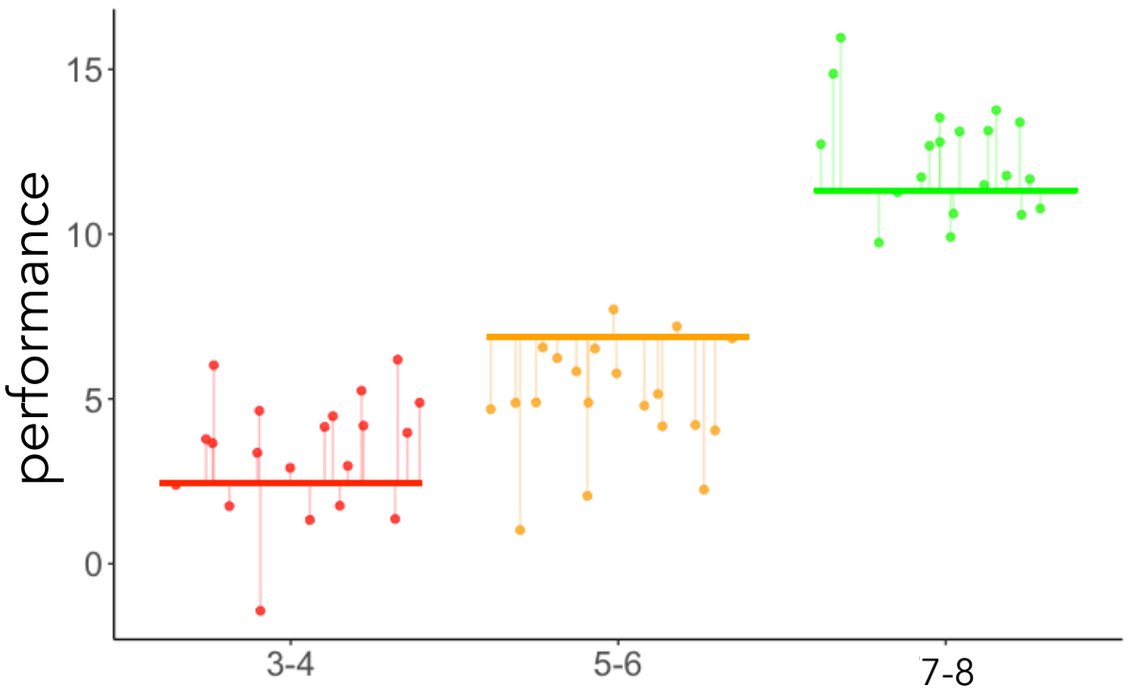
Does performance increase with age?

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

Compact model



Augmented model

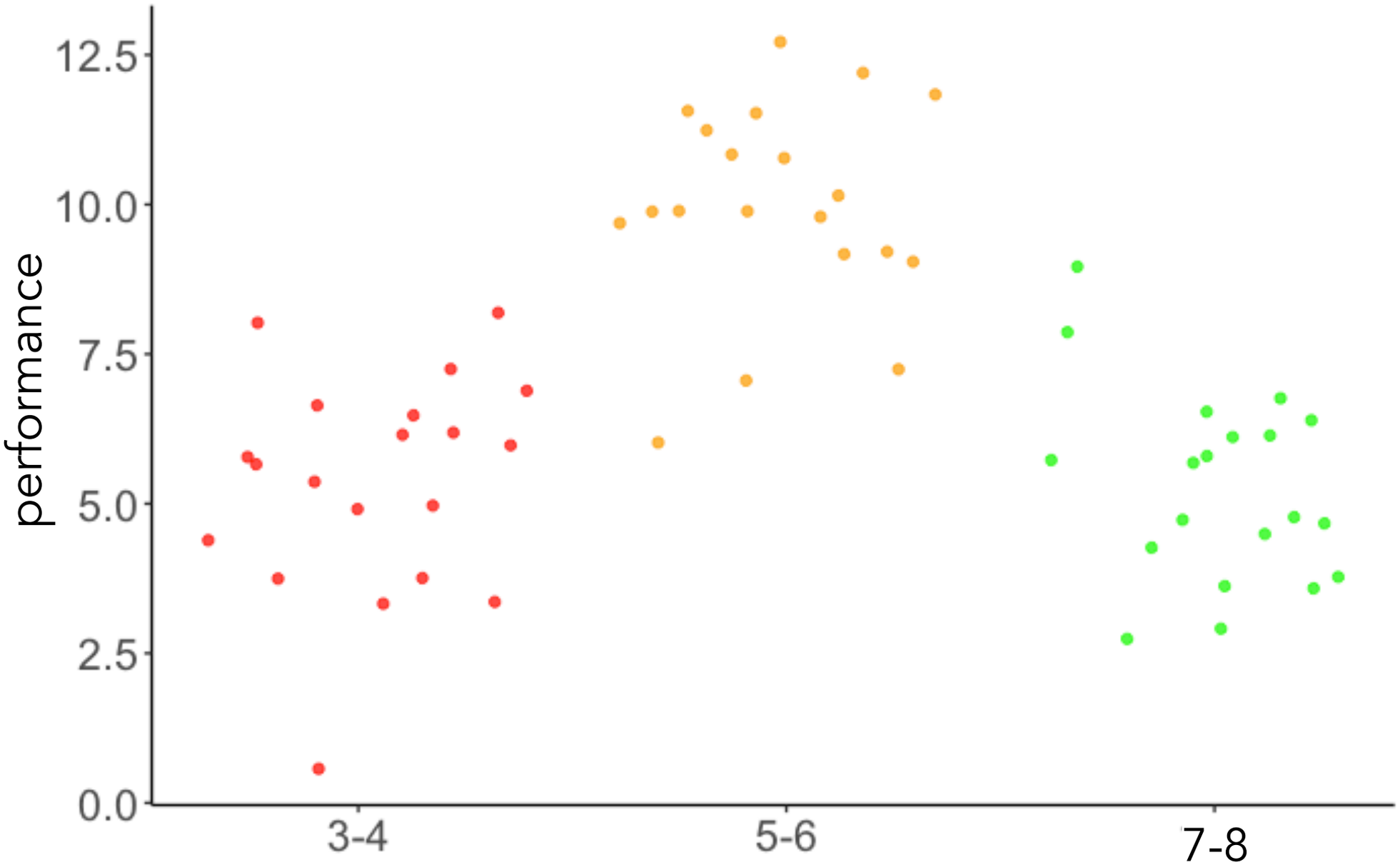


Model comparison

p < .001

Contrasts

Does performance increase with age?



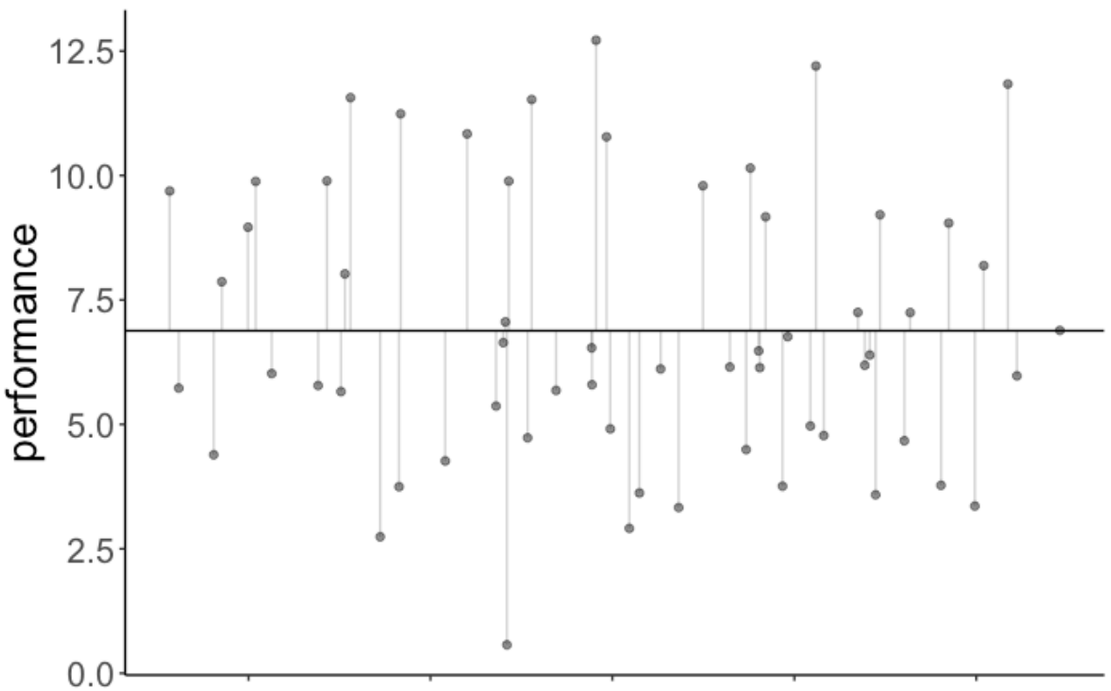
Data from another hypothetical developmental study

Contrasts

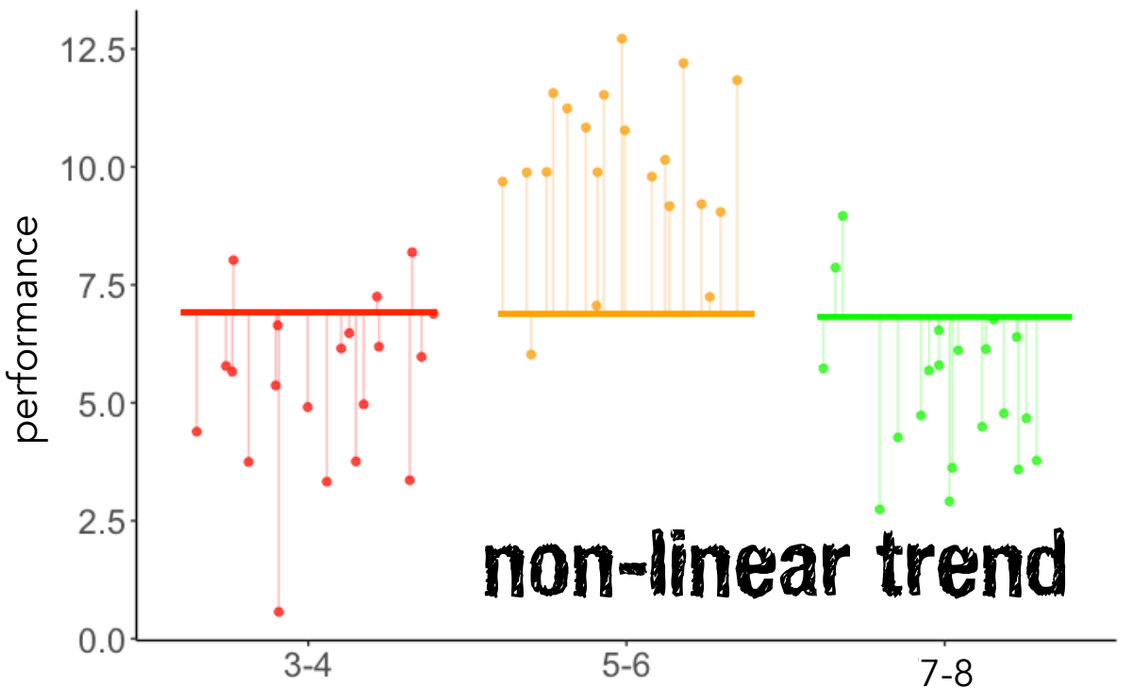
Does performance increase with age?

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

Compact model



Augmented model



Model comparison

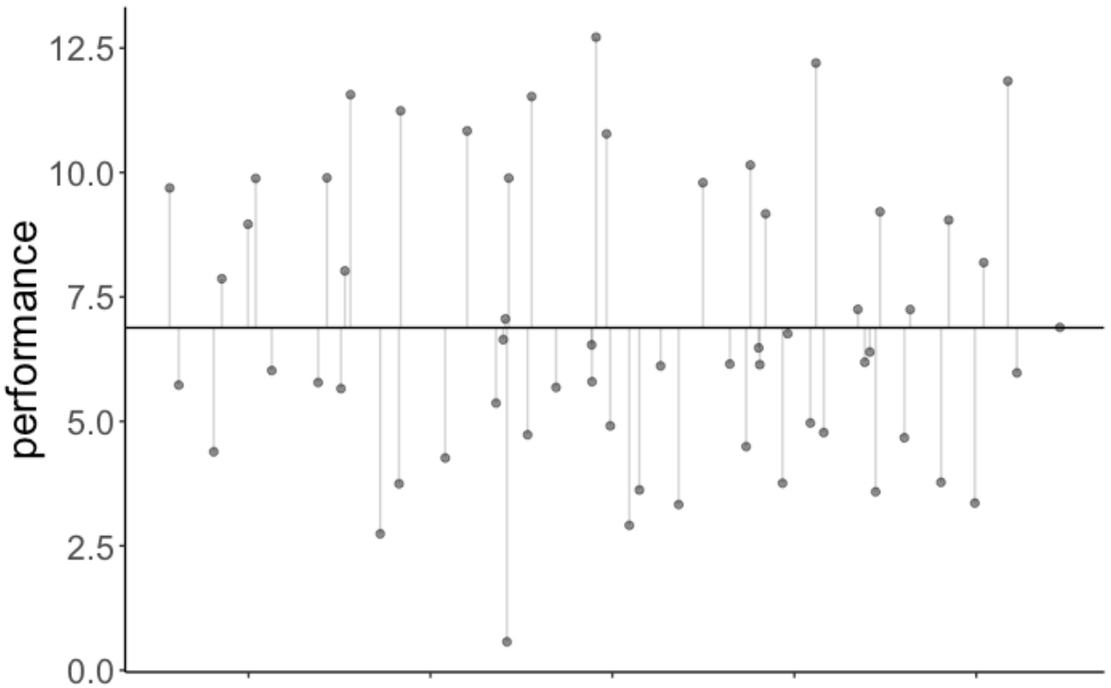
p = .8508

Contrasts

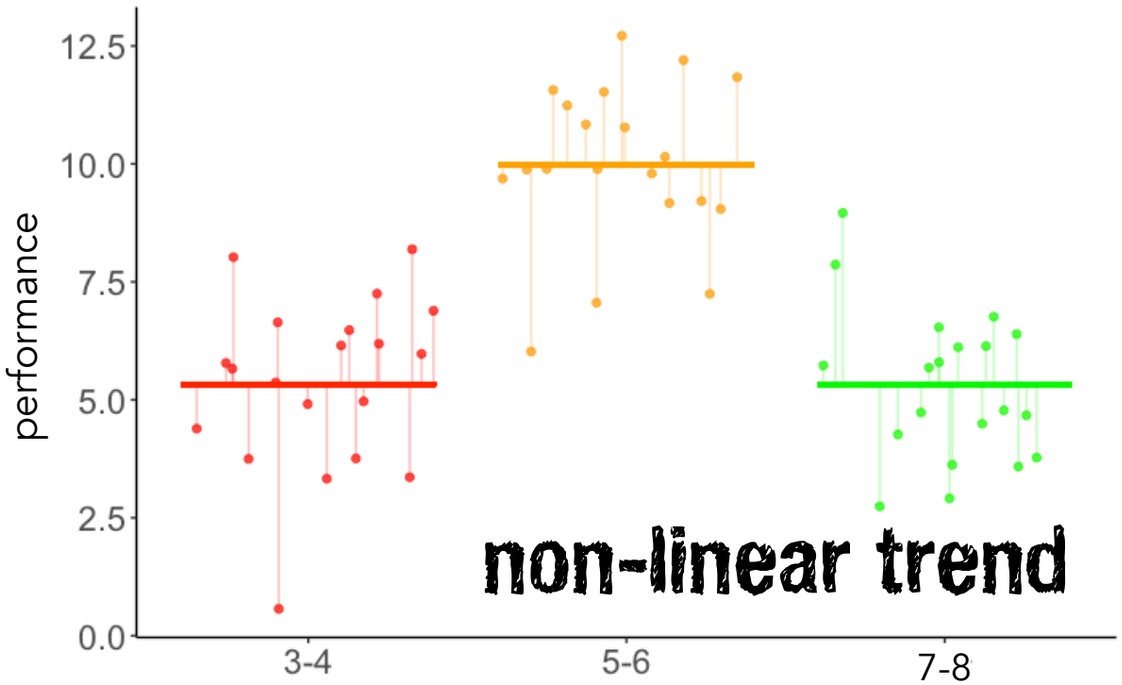
Does performance increase with age?

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

Compact model



Augmented model



Model comparison

p < .001

Contrasts

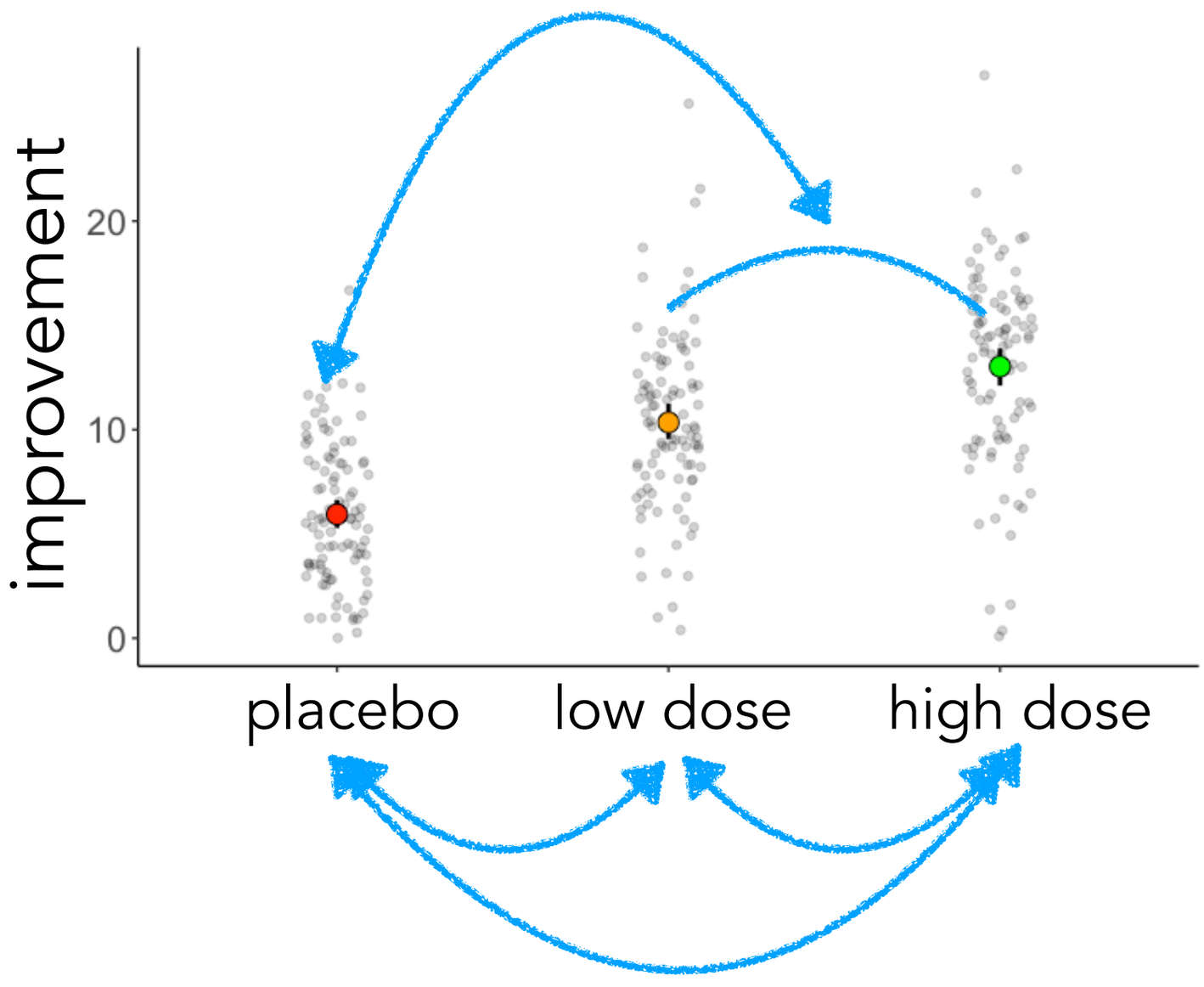
- linear contrasts allow us to ask more specific questions of our data
- rather than asking whether any of the group means are significantly different from each other (ANOVA), we can ask questions such as:
  - Does performance increase with age?
  - Is the overall performance in Condition B and C better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

Planned comparisons

Planned contrasts

must be defined in advance

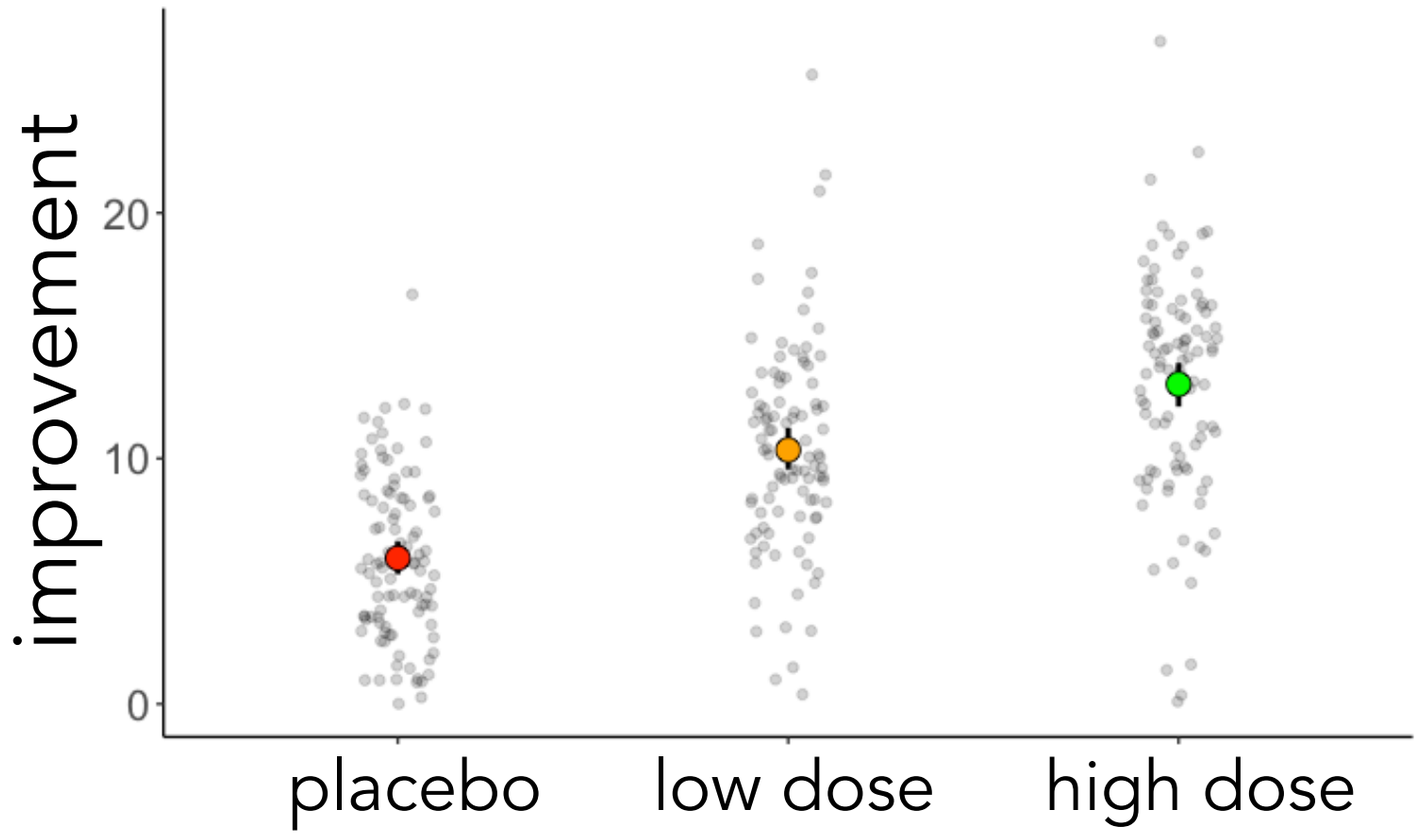
1. Is the treatment different from the placebo?
2. Do the two treatments differ from each other?



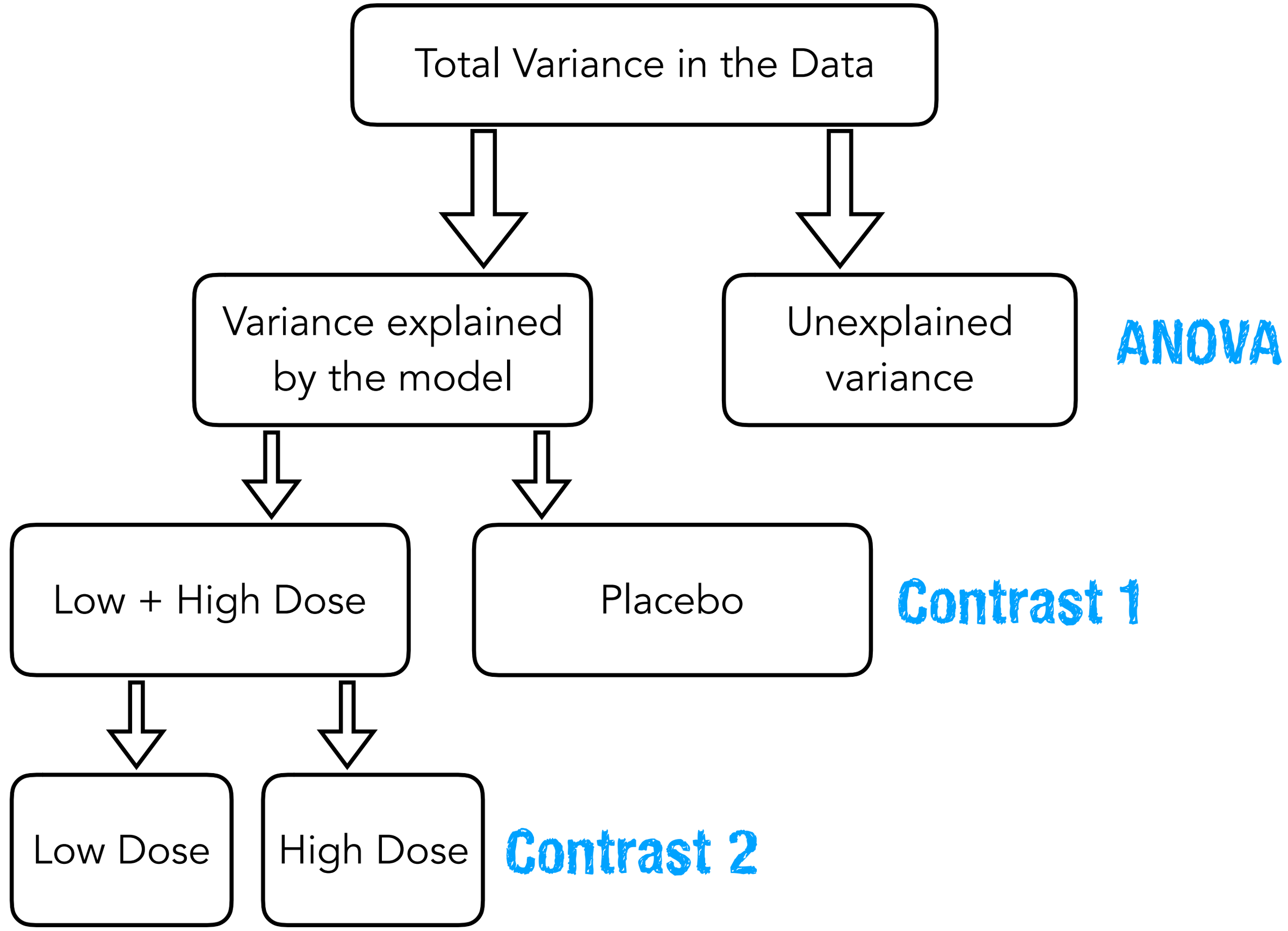
six possible comparisons

if we test all hypotheses, we increase the chance of making a type I error

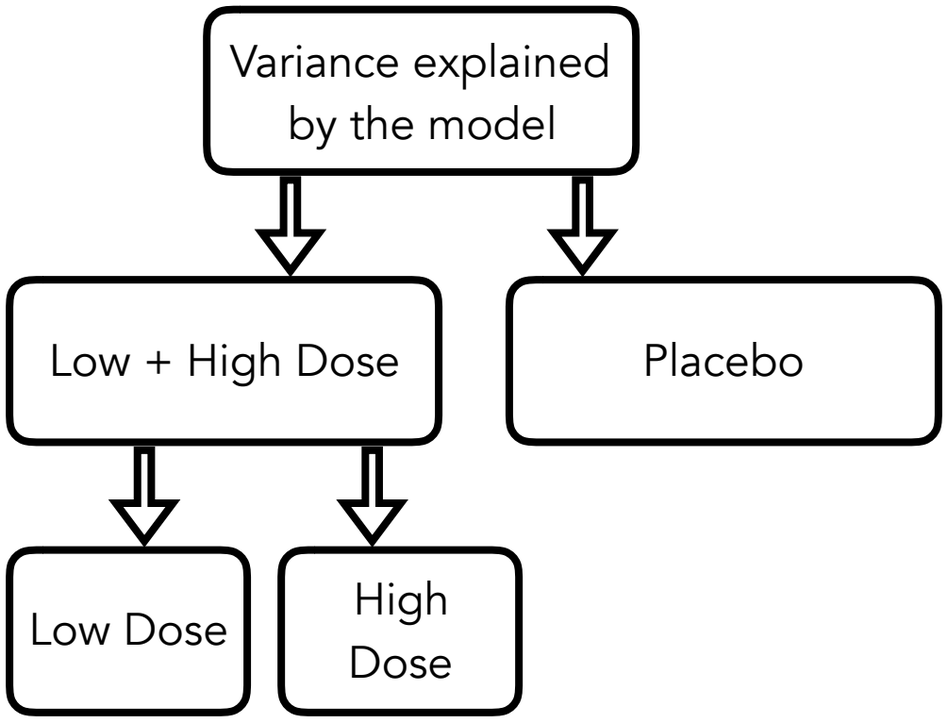
Planned contrasts



Orthogonal contrasts allow us to partition the variance explained by the ANOVA model into a maximum of (#treatment - 1) meaningful and targeted comparisons involving different combinations of means.


Total Variance in the Data
Variance explained by the model
Unexplained variance
ANOVA
Low + High Dose
Placebo
Contrast 1
Low Dose
High Dose
Contrast 2

Planned contrasts



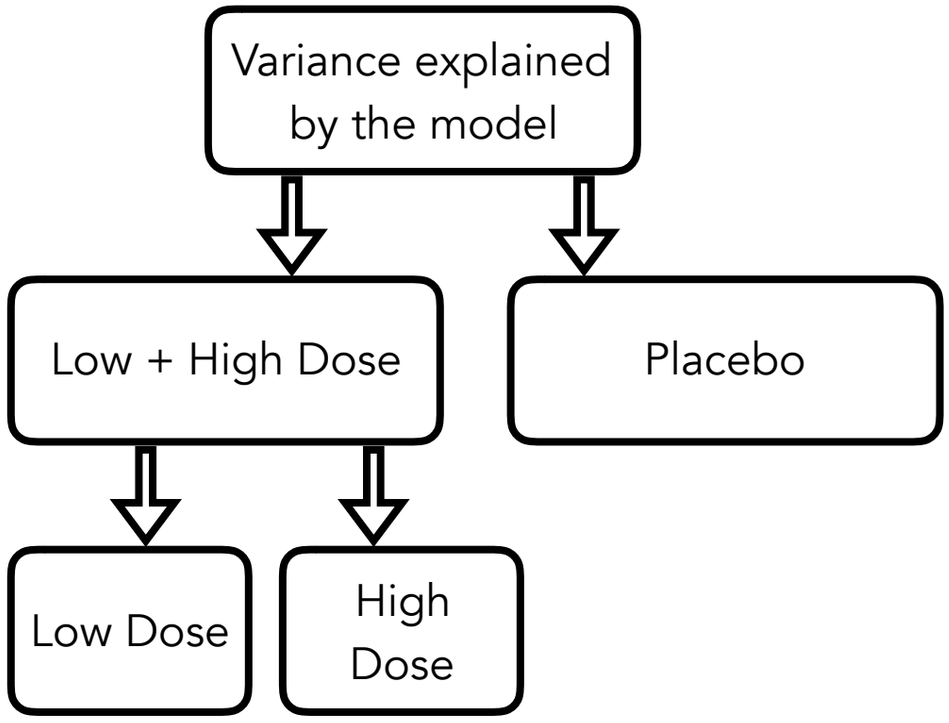
- Each contrast must compare only two "chunks" of variation
- Once a group has been singled out in a contrast, it can't be used in another contrast

Defining orthogonal contrast codes

1. Groups coded with positive weights will be compared against groups with negative weights
2. Assign a 0 to groups who are not involved in a comparison
3. The sum of weights for each comparison should be 0
4. The product of the weights for any pair comparisons should sum to 0

Defining contrast codes

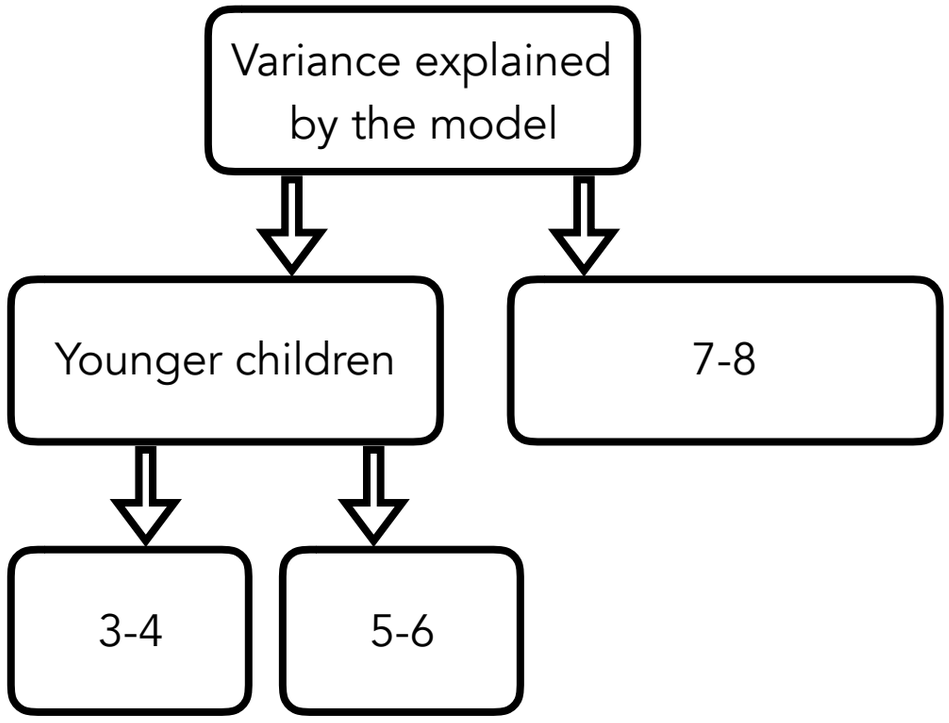
- The sum of weights for each comparison should be 0
- The product of the weights for any pair comparisons should sum to 0



contrast	Low	High	Placebo	sum
Dose vs Placebo	0.5	0.5	-1	0
Low vs. high	-1	1	0	0
Product	-0.5	0.5	0	0

Defining contrast codes

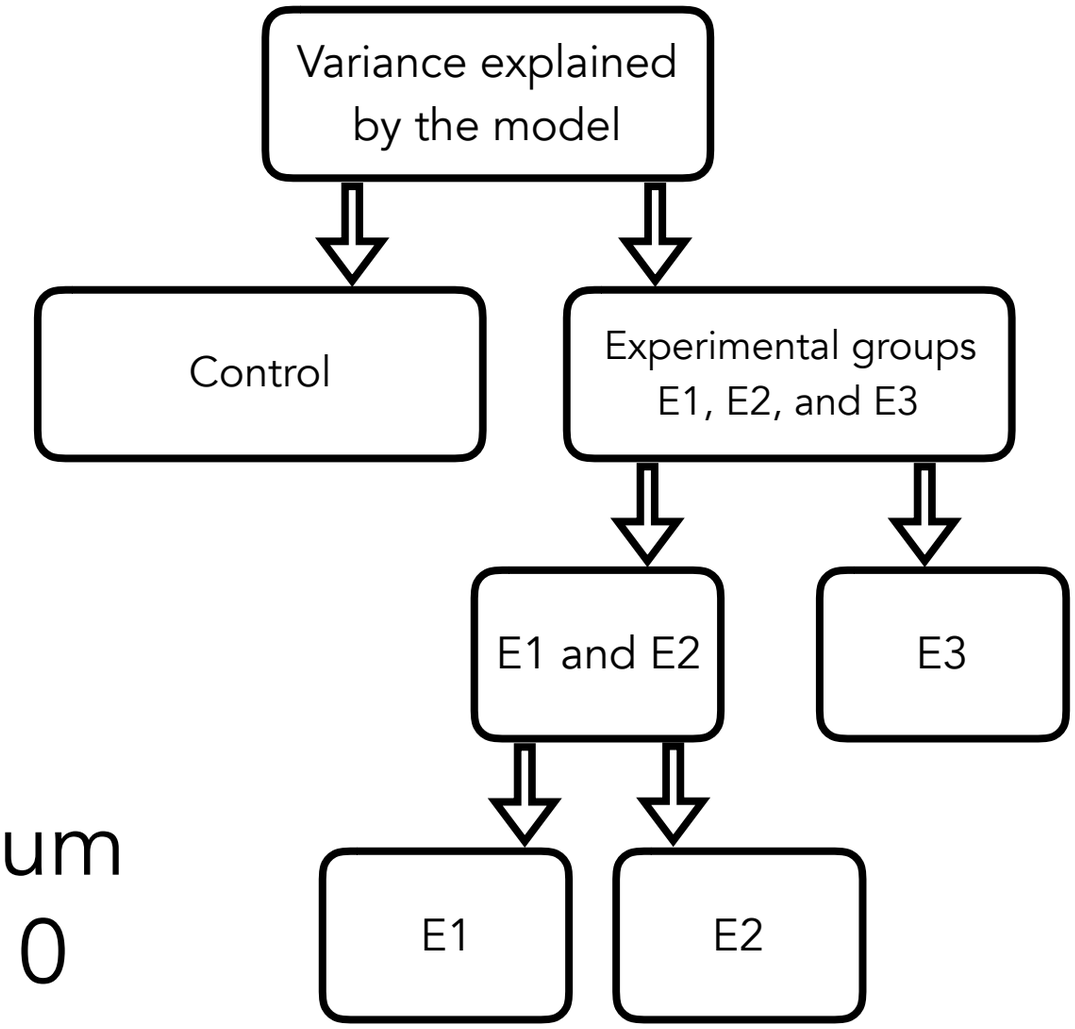
- The sum of weights for each comparison should be 0
- The product of the weights for any pair comparisons should sum to 0



contrast	3-4	5-6	7-8	sum
Young vs. old	-0.5	-0.5	1	0
3-4 vs. 5-6	-1	1	0	0
Product	0.5	-0.5	0	0

Defining contrast codes

- The sum of weights for each comparison should be 0
- The product of the weights for any pair comparisons should sum to 0



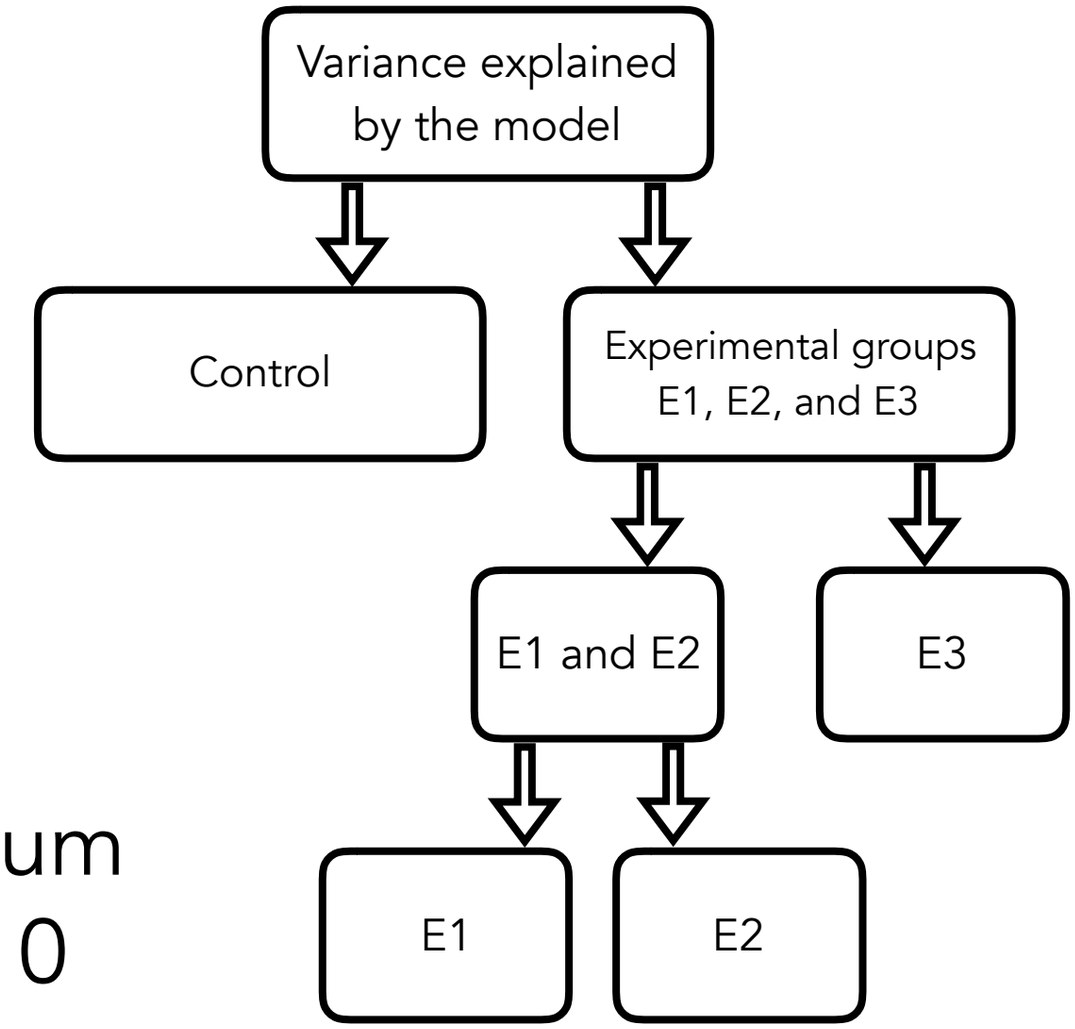
contrast	Control	E1	E2	E3	sum
Control vs. E	-3	1	1	1	0
E1&2 vs. E3	0	-1	-1	2	0
E1 vs. E2	0	-1	1	0	0

Check for products

contrast	Control	E1	E2	E3	sum
Control vs. E	-3	1	1	1	
E1&2 vs. E3	0	-1	-1	2	
product	0	-1	-1	2	0

Defining contrast codes

- The sum of weights for each comparison should be 0
- The product of the weights for any pair comparisons should sum to 0



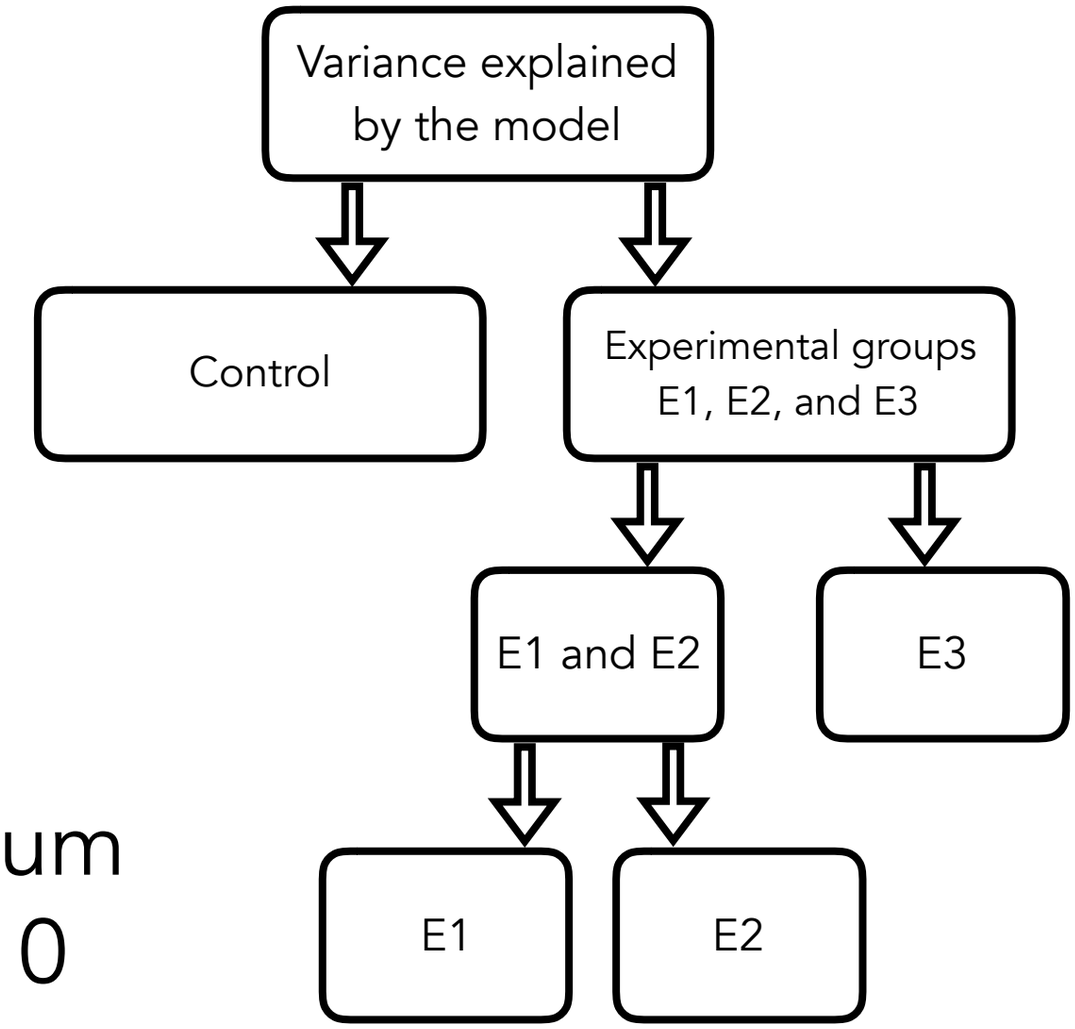
contrast	Control	E1	E2	E3	sum
Control vs. E	-3	1	1	1	0
E1&2 vs. E3	0	-1	-1	2	0
E1 vs. E2	0	-1	1	0	0

Check for products

contrast	Control	E1	E2	E3	sum
Control vs. E	-3	1	1	1	
E1 vs. E2	0	-1	1	0	
product	0	-1	1	0	0

Defining contrast codes

- The sum of weights for each comparison should be 0
- The product of the weights for any pair comparisons should sum to 0



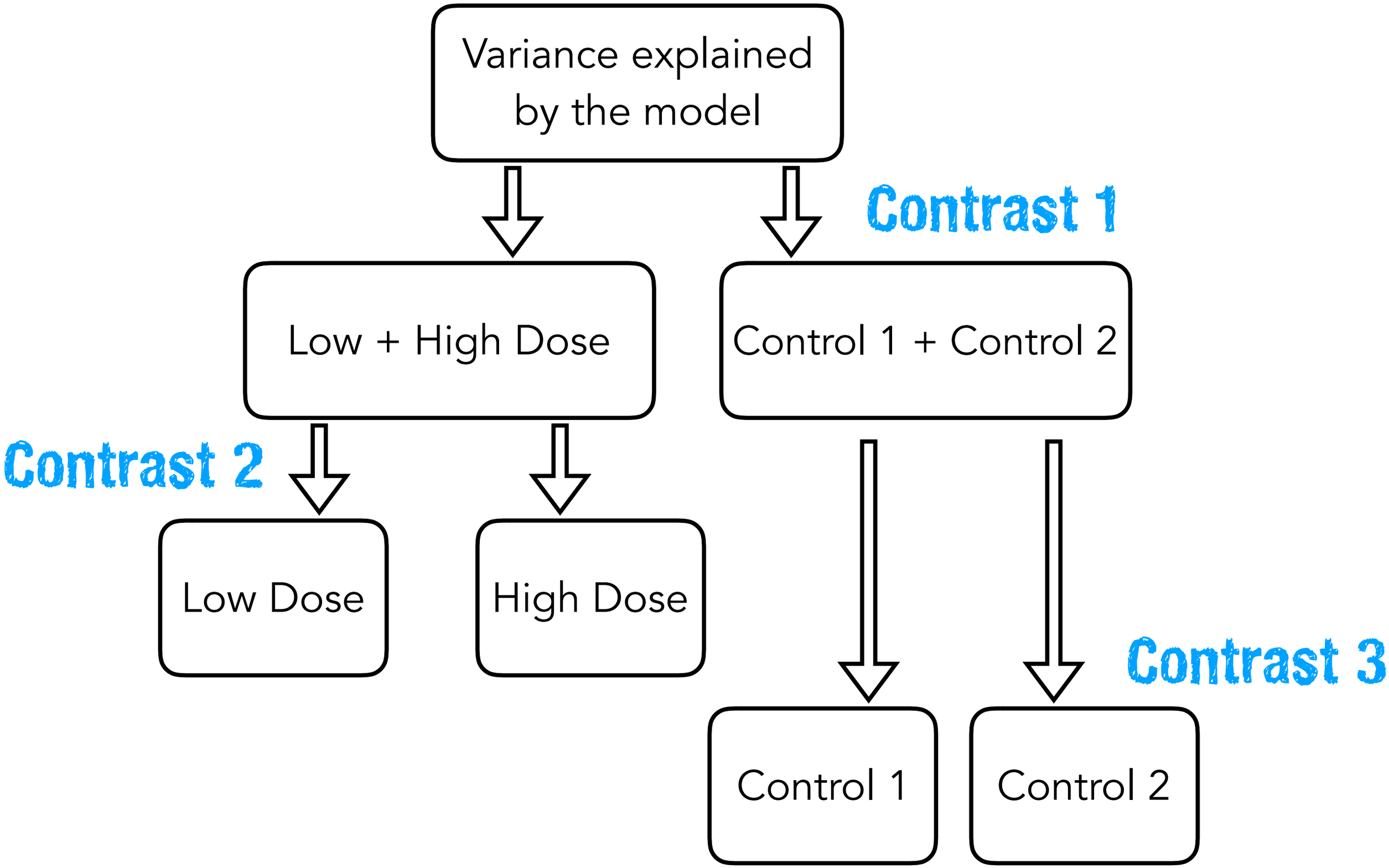
contrast	Control	E1	E2	E3	sum
Control vs. E	-3	1	1	1	0
E1&2 vs. E3	0	-1	-1	2	0
E1 vs. E2	0	-1	1	0	0

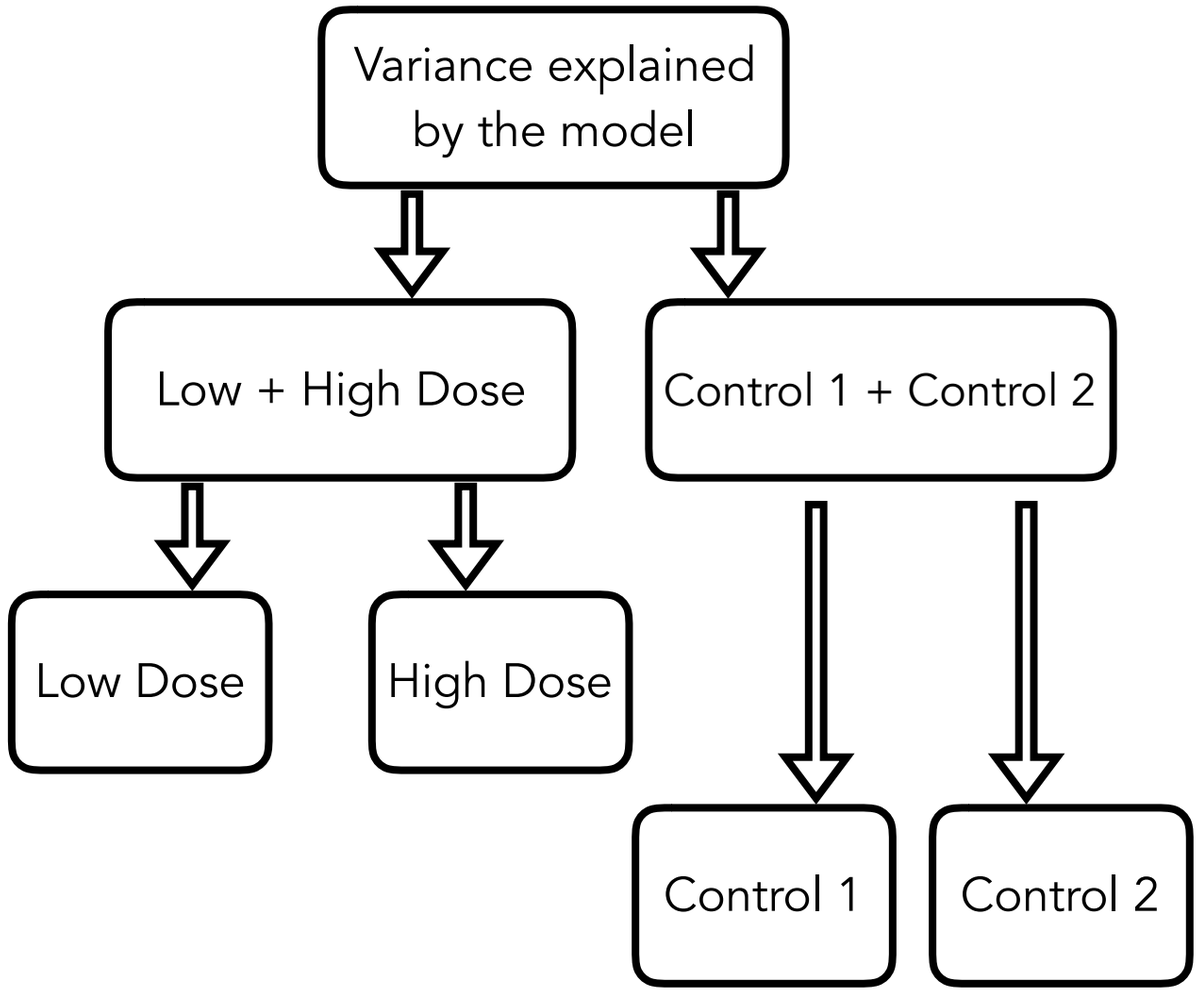
contrasts are orthogonal

Check for products

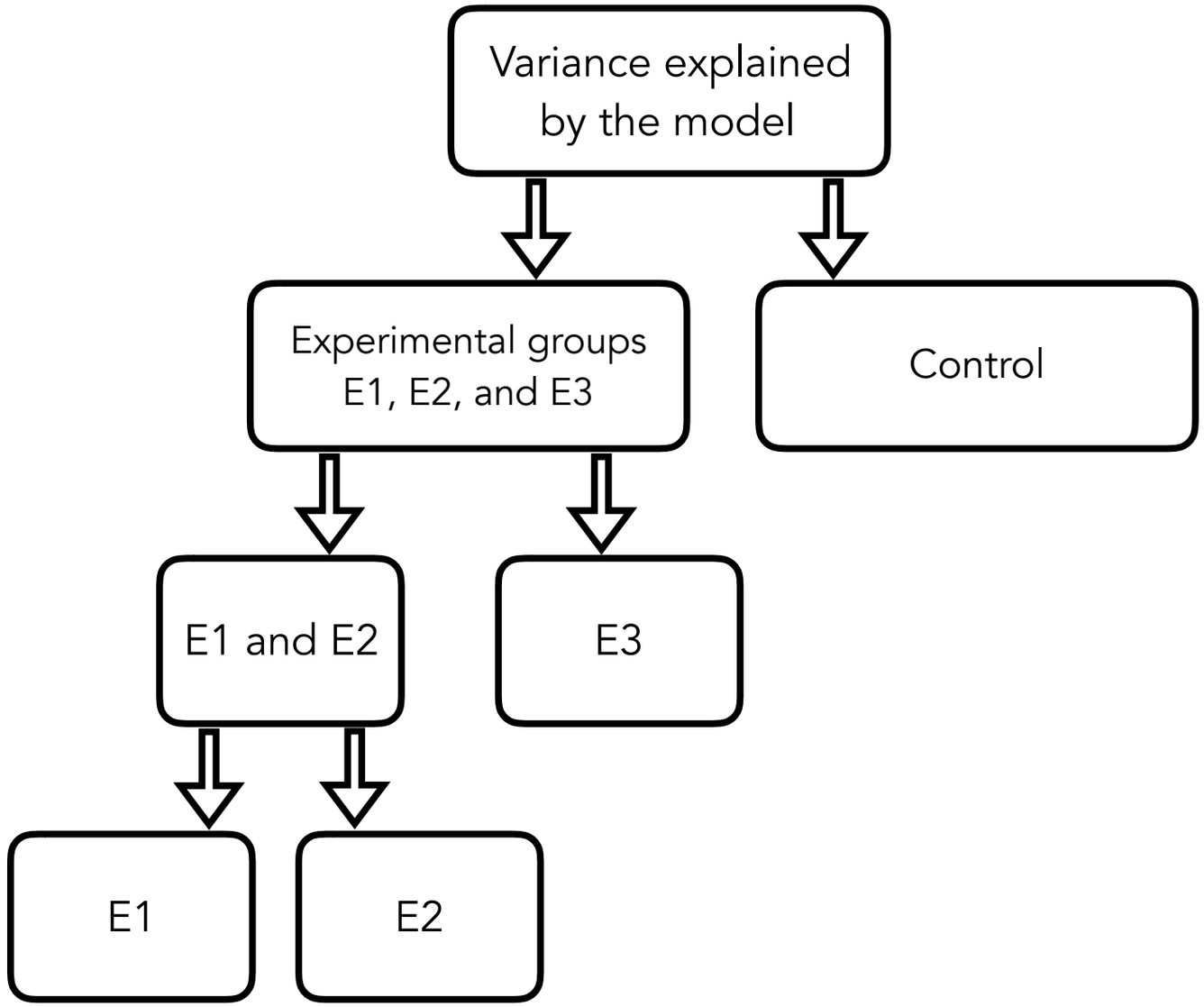
contrast	Control	E1	E2	E3
E1&2 vs. E3	0	-1	-1	2
E1 vs. E2	0	-1	1	0
product	0	1	-1	0

sum 0


Variance explained by the model
Contrast 1
Low + High Dose
Control 1 + Control 2
Contrast 2
Low Dose
High Dose
Contrast 3
Control 1
Control 2



contrast	low	high	C1	C2
low/high vs. C1/C2				
low vs. high				
C1 vs. C2				



contrast	E1	E2	E2	C
E1/E2/E3 vs. C				
E1/E2 vs. E3				
E1 vs. E2				

Planned contrasts in R

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

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

fit linear model

Planned contrasts in R

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

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

Planned contrasts in R

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

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(-1, 1, 0))
```

set up linear contrasts
(make sure they are orthogonal)

Planned contrasts in R

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

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(-1, 1, 0))

# compute estimated marginal means
leastsquare = emmeans(fit, "group")

# run analyses
contrast(leastsquare,
         contrasts,
         adjust = "bonferroni")
```

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
```

Interpreting the coefficients

```
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(-0.5, 0.5, 0))

# compute estimated marginal means
leastsquare = emmeans(fit, "group")

# run analyses
contrast(leastsquare,
         contrasts,
         adjust = "bonferroni")
```

```
[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!

Post hoc tests

```r
library("emmeans")

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

# post hoc tests
leastsquare = emmeans(fit, "group")
pairs(leastsquare,
      adjust = "bonferroni")
```

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
leastsquare = emmeans(fit, c("hand", "skill"))
pairs(leastsquare,
      adjust = "bonferroni")
```

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

Making decisions

Type I Error

Type II Error

H_0: Not pregnant. H_1: Pregnant.

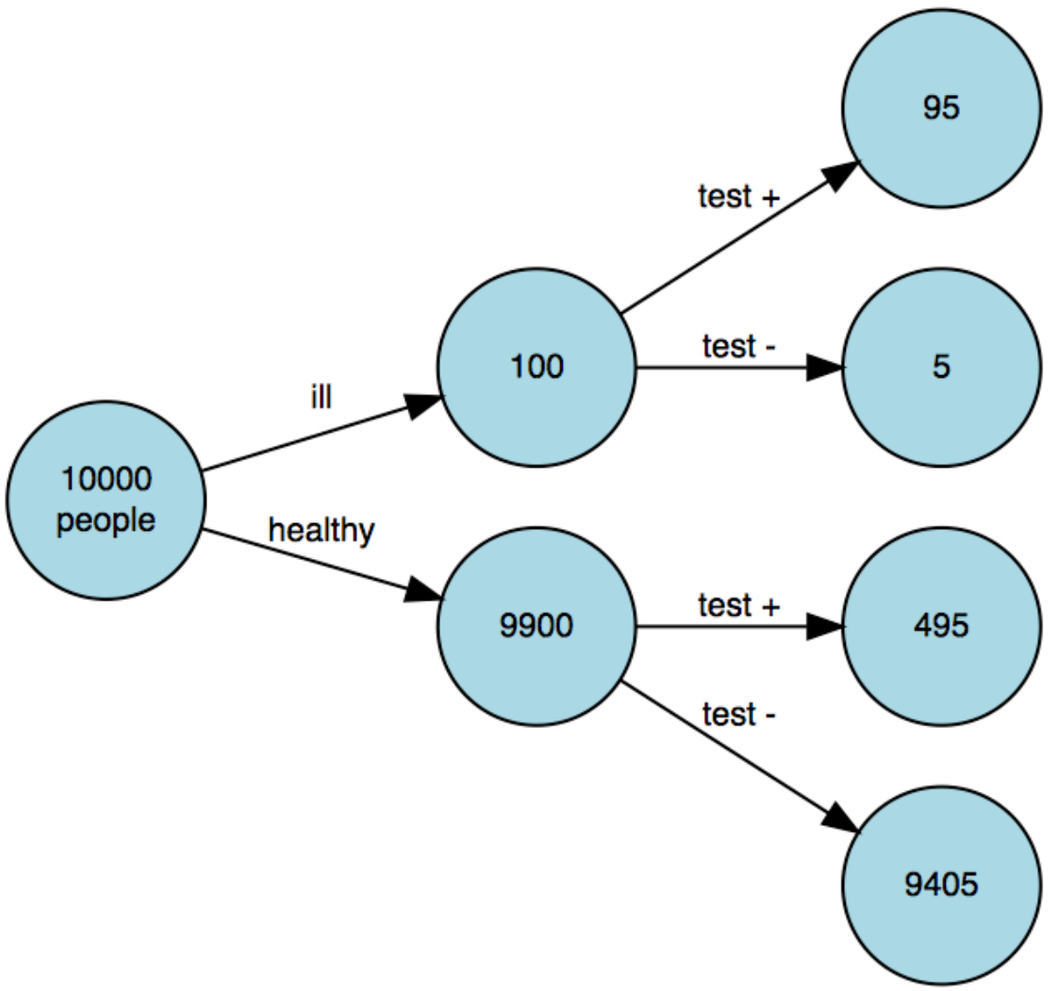
Type I Error: Falsely rejecting the null hypothesis (even though it is true).

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

---

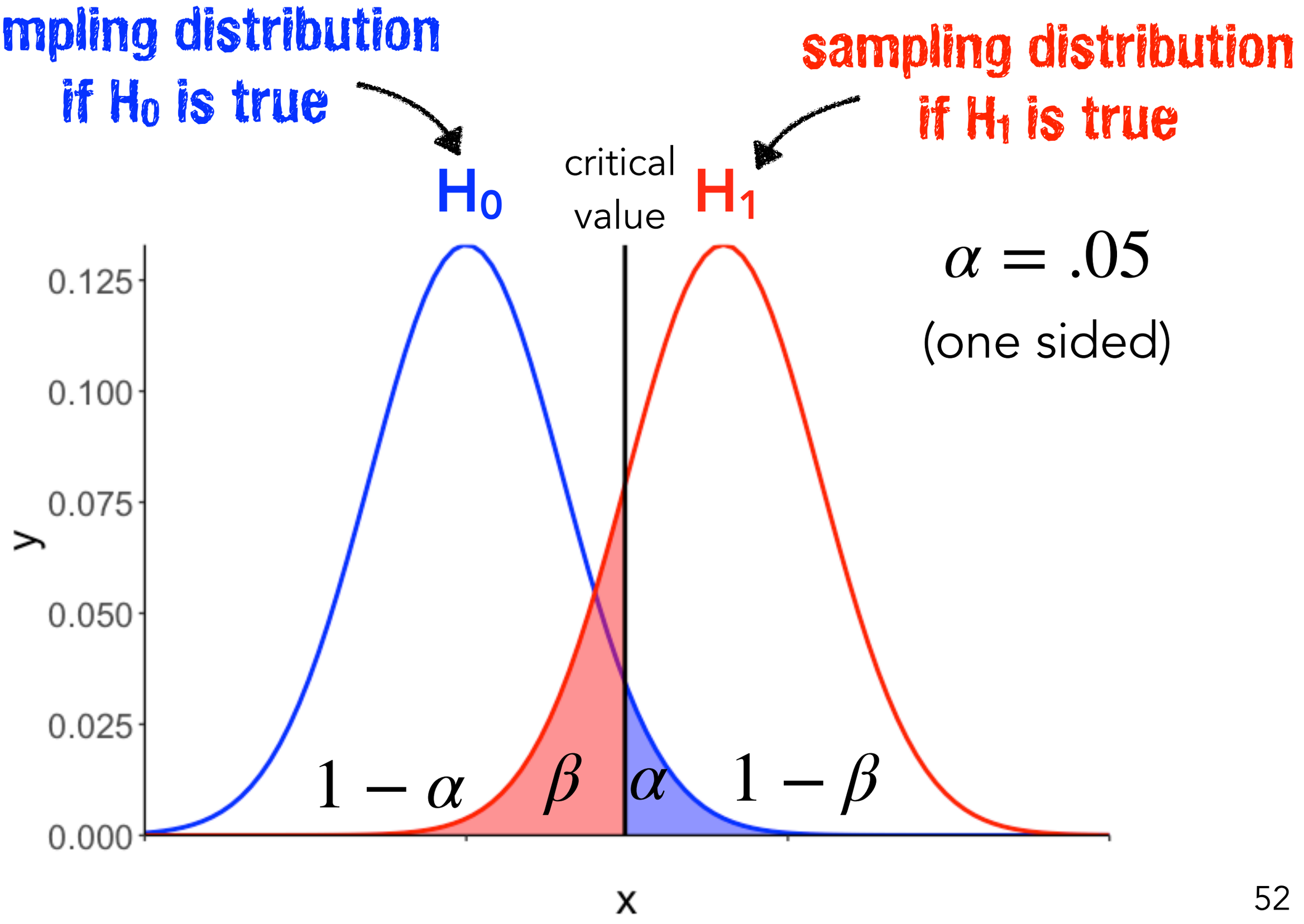
true negative

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

Specificity

$1-\alpha$

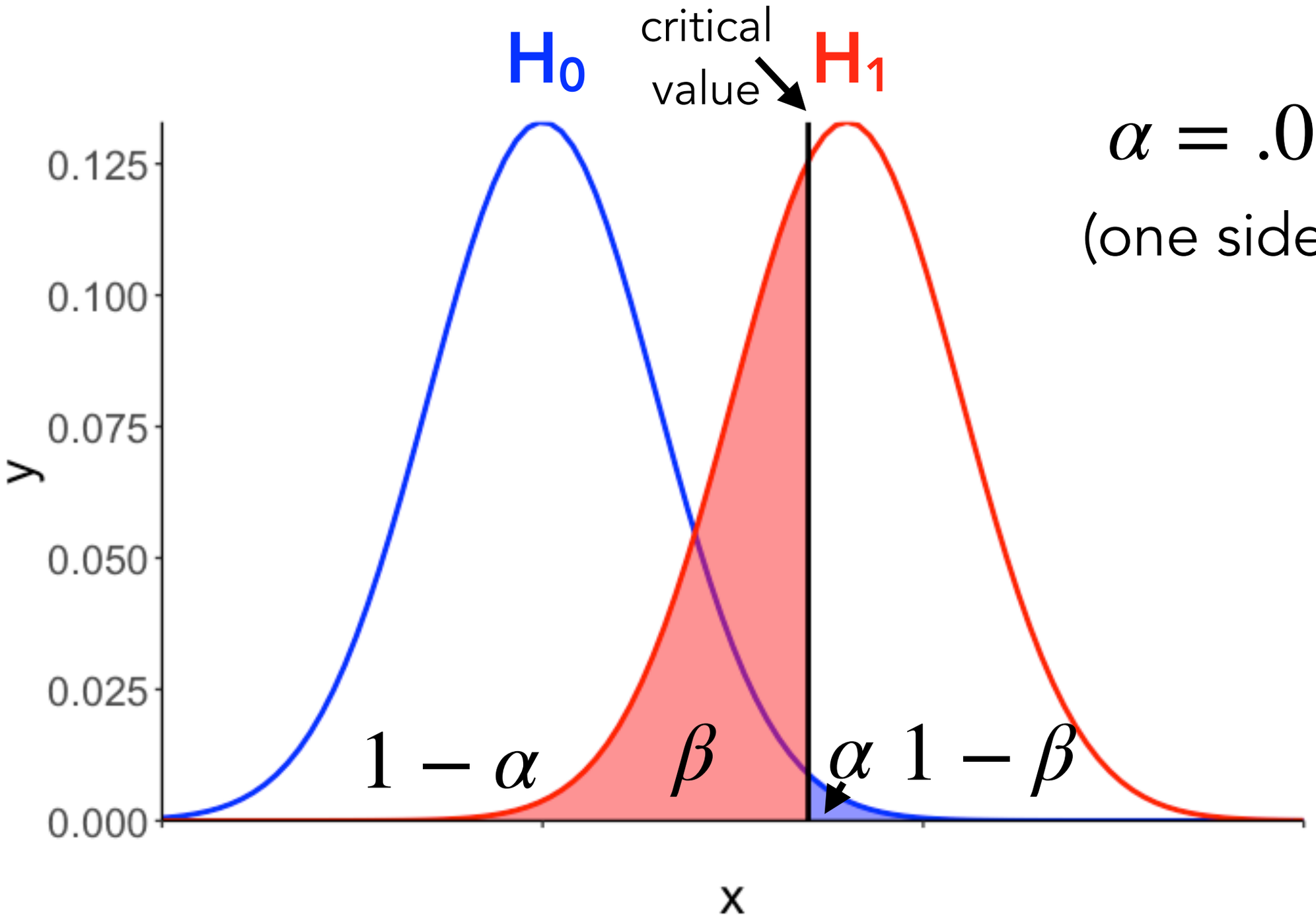
What affects power?



$\alpha = .05$

(one sided)

What affects power?

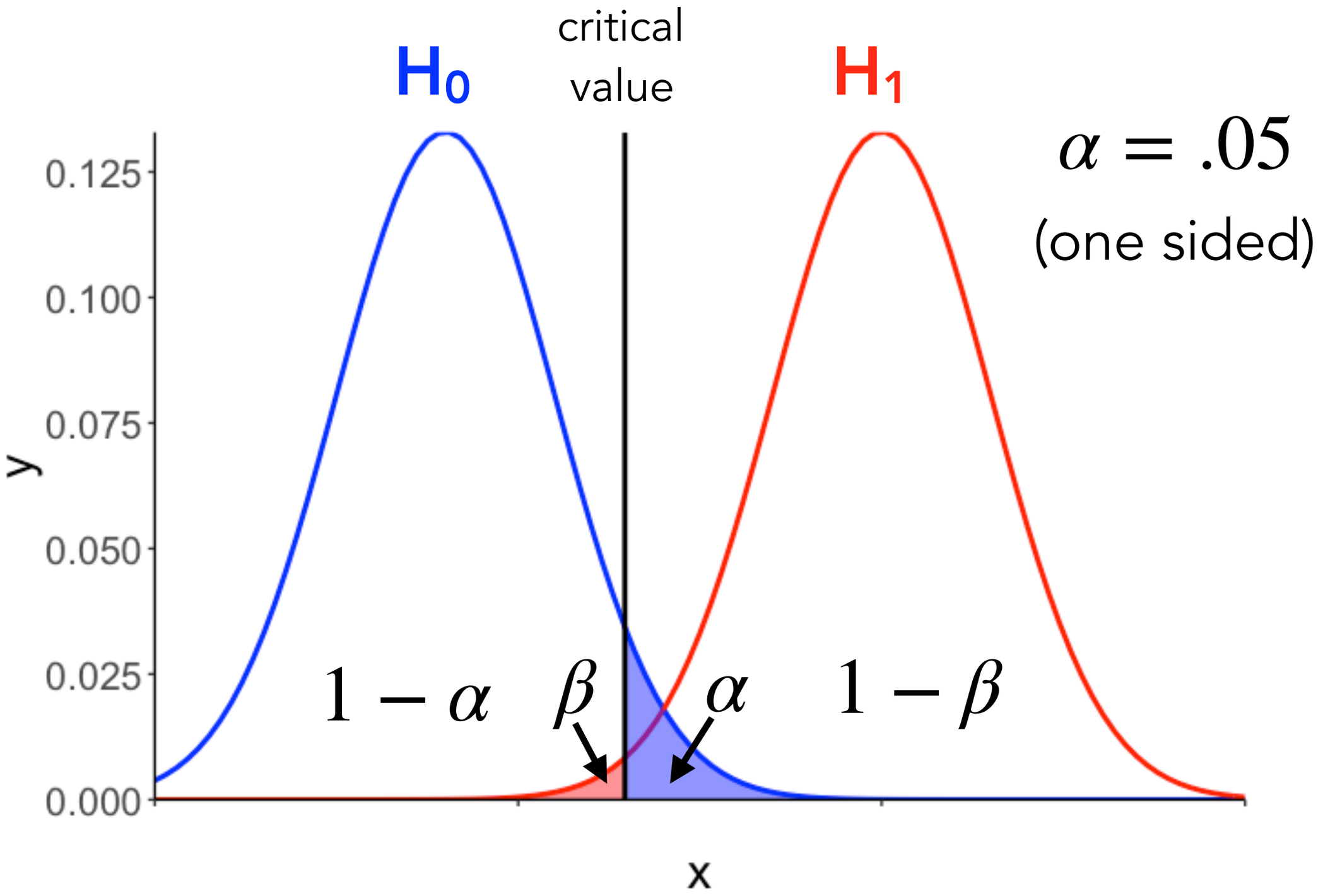


$\alpha = .01$

(one sided)

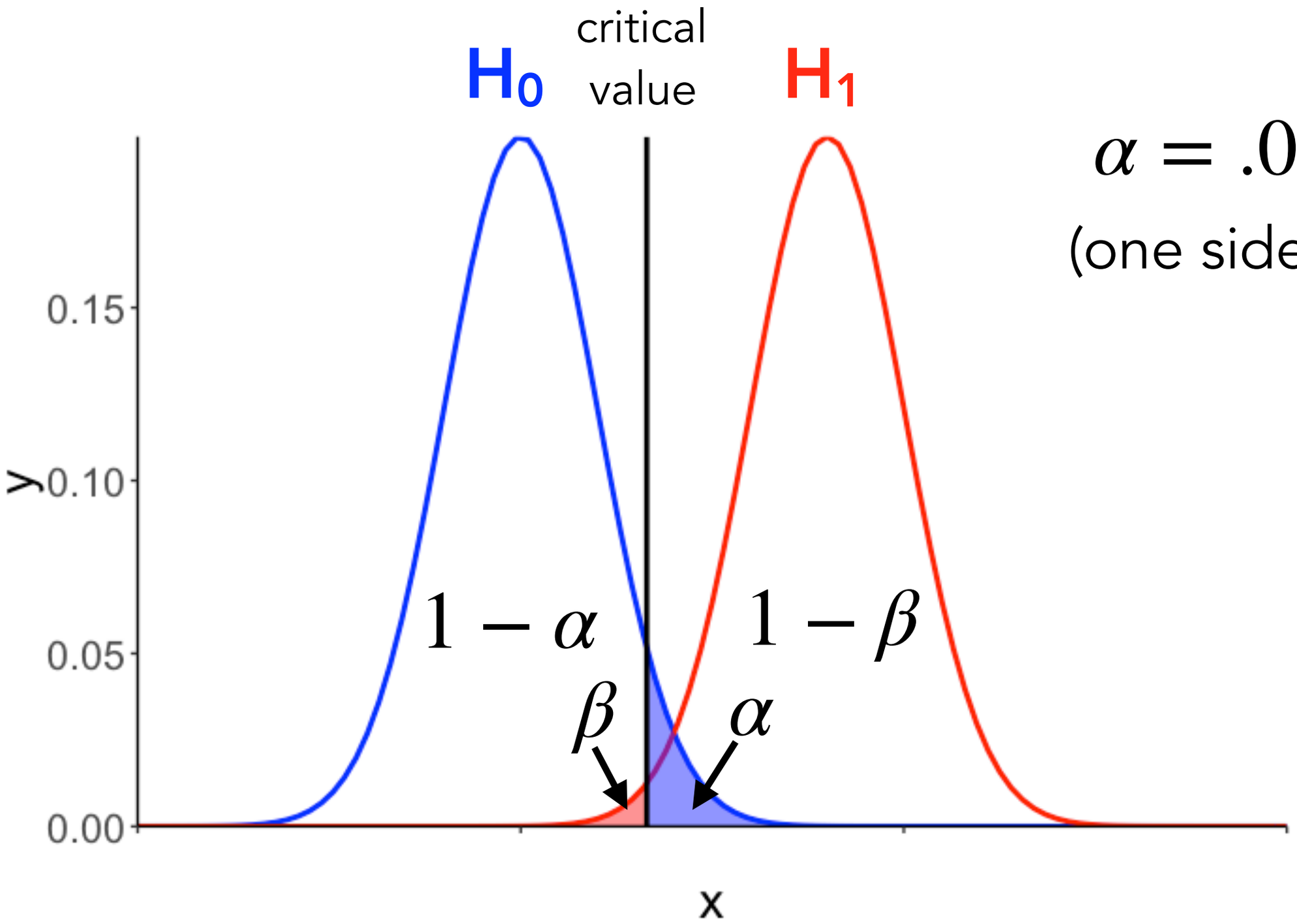
α affects power

What affects power?



distance between means affects power

What affects power?



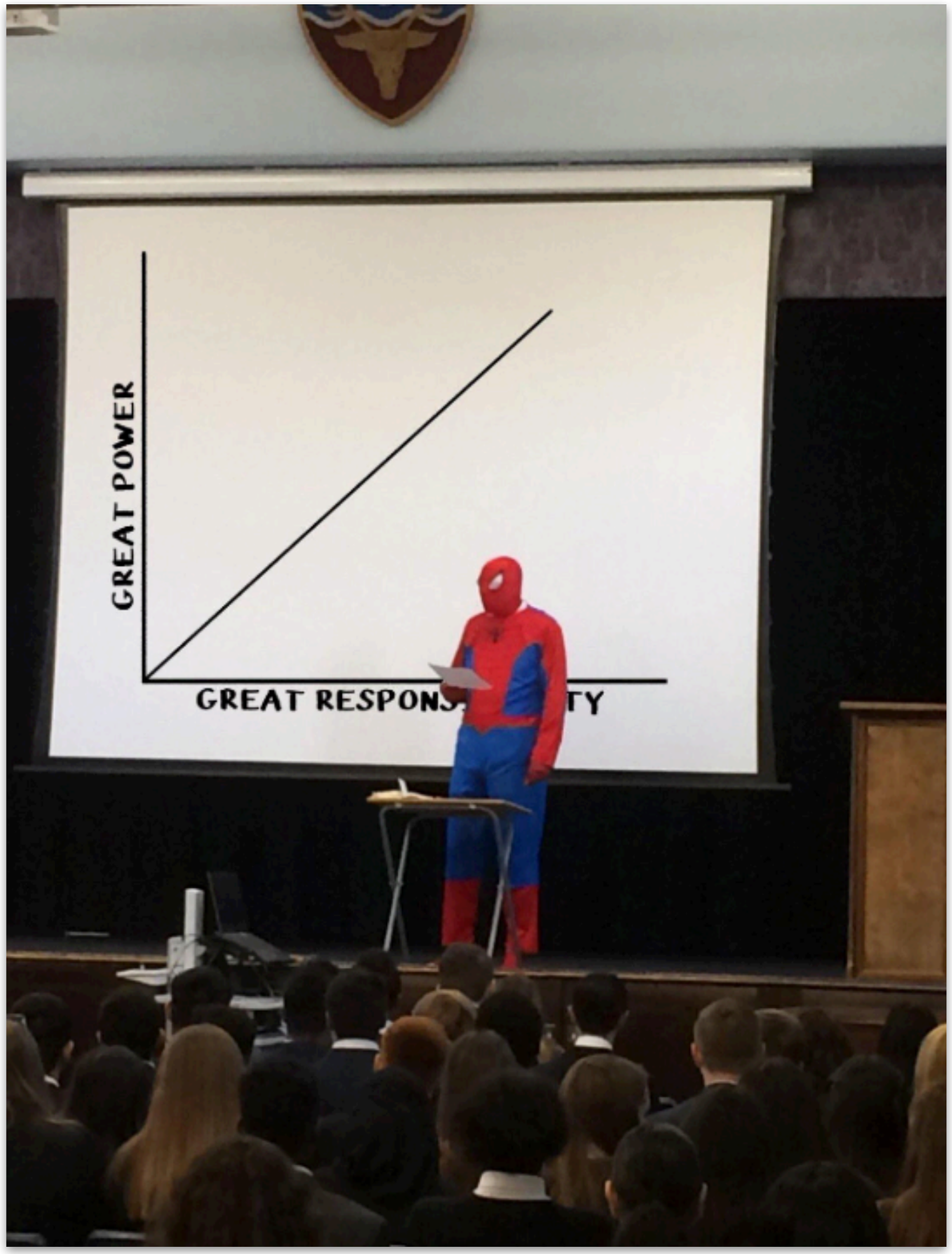
$\alpha = .05$

(one sided)

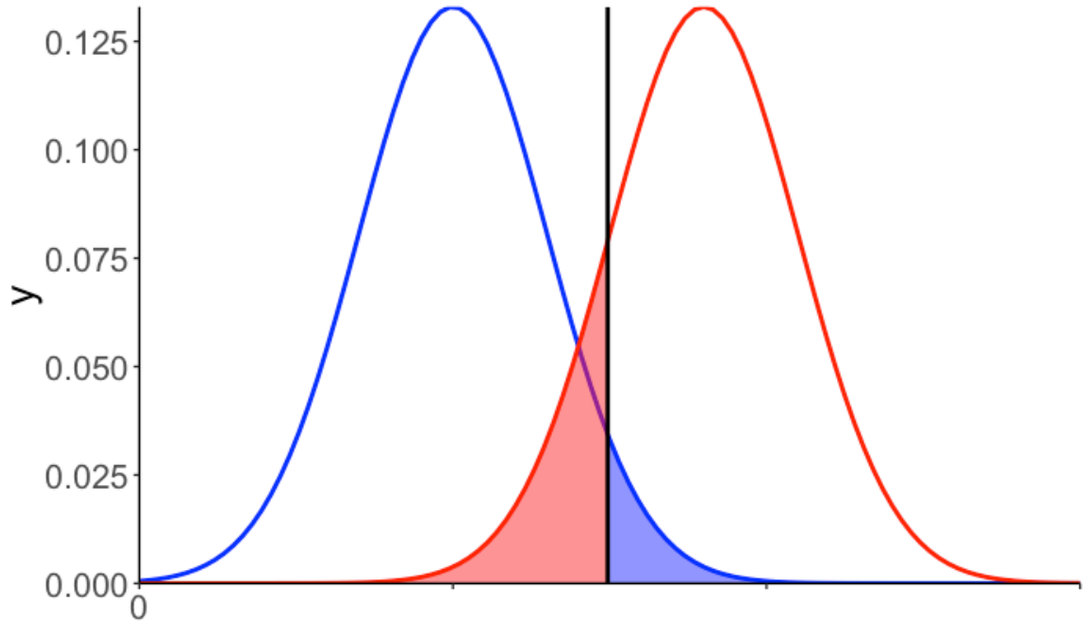
variance affects power

Calculating power

With great power comes ...



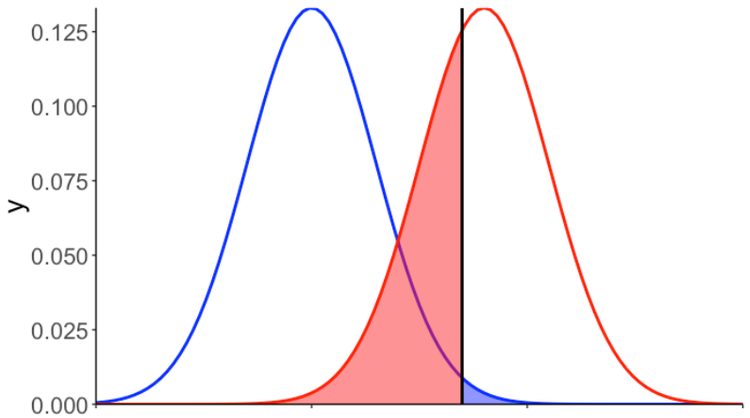
The knobs we can turn to affect power

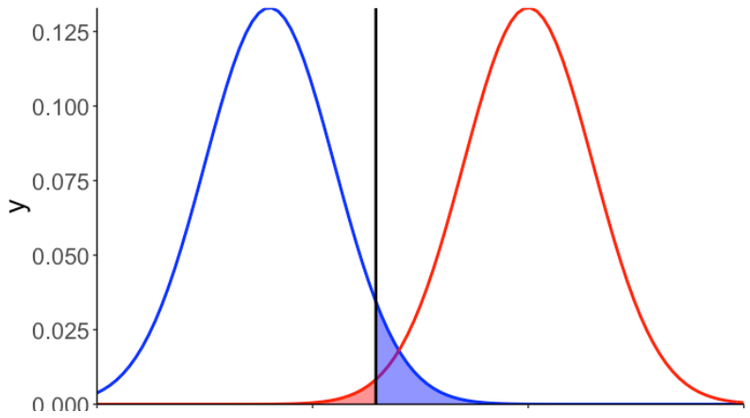


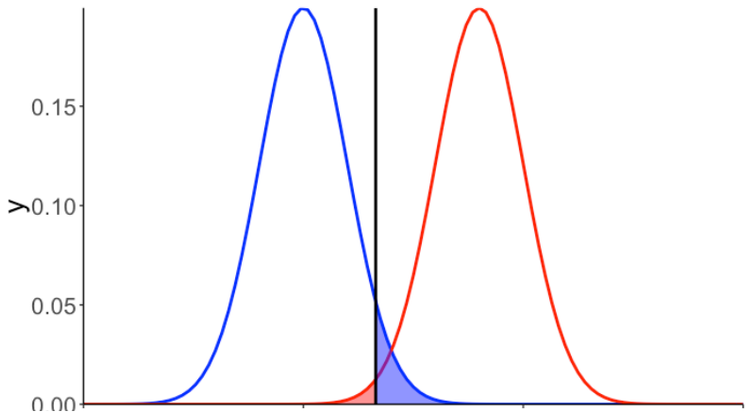
α

effect size

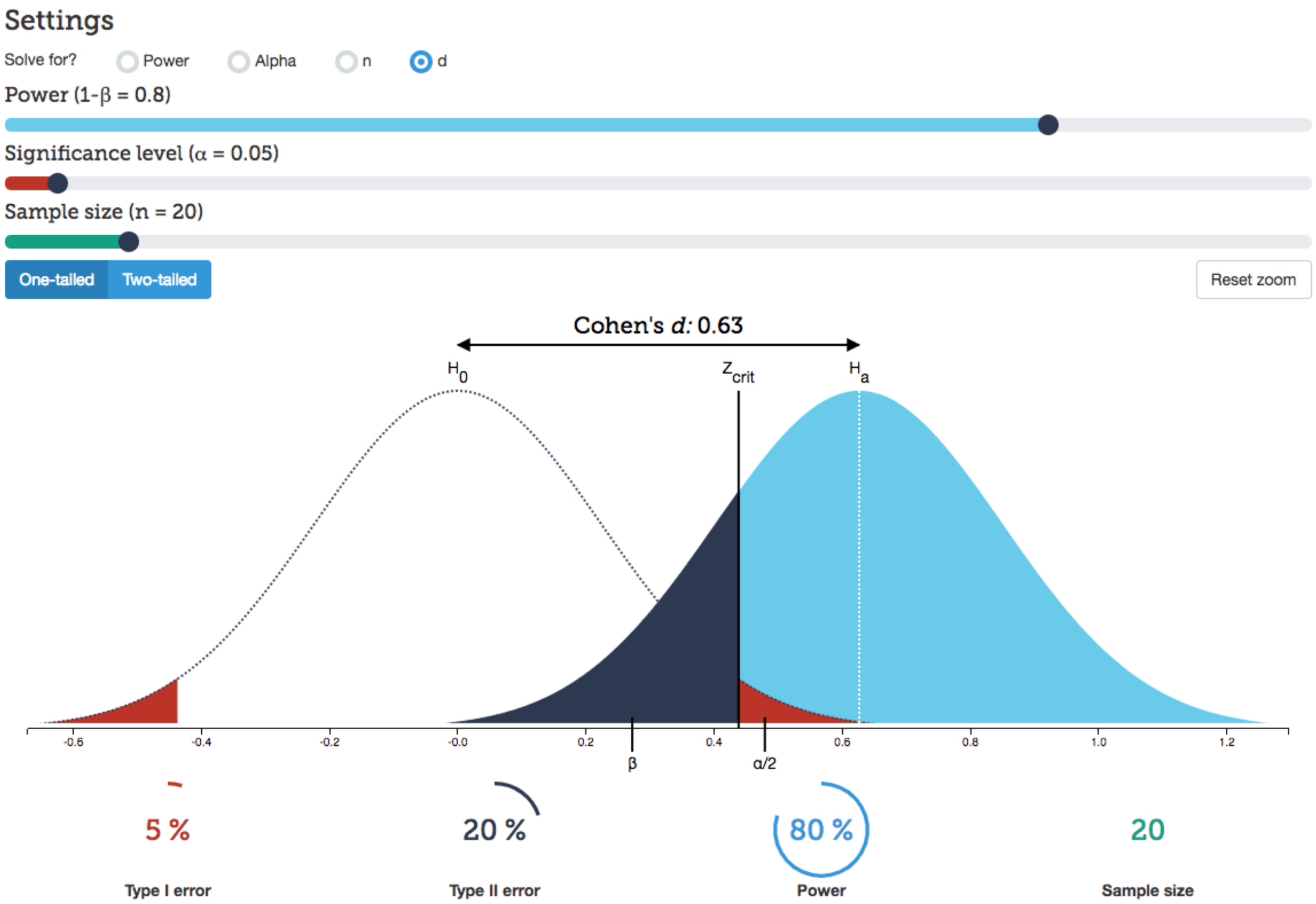
sample size







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)

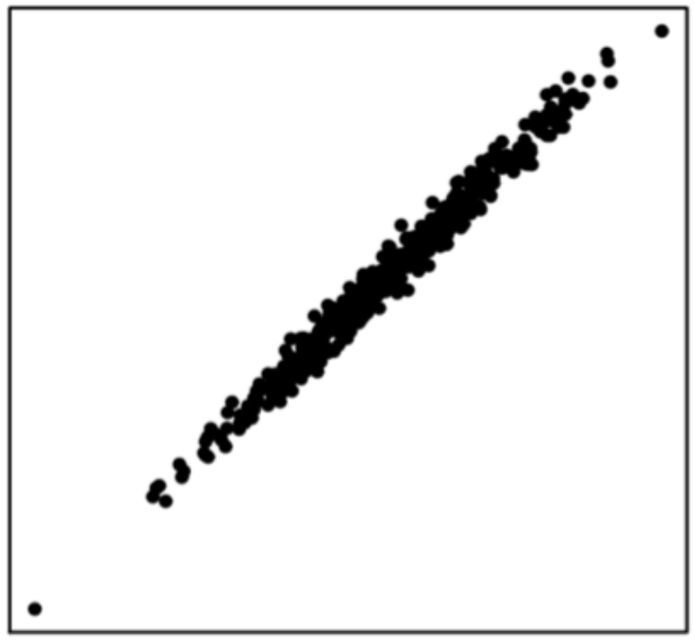
Augmented model SSE(A)

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

SSE = sum of squared errors

Common effect sizes

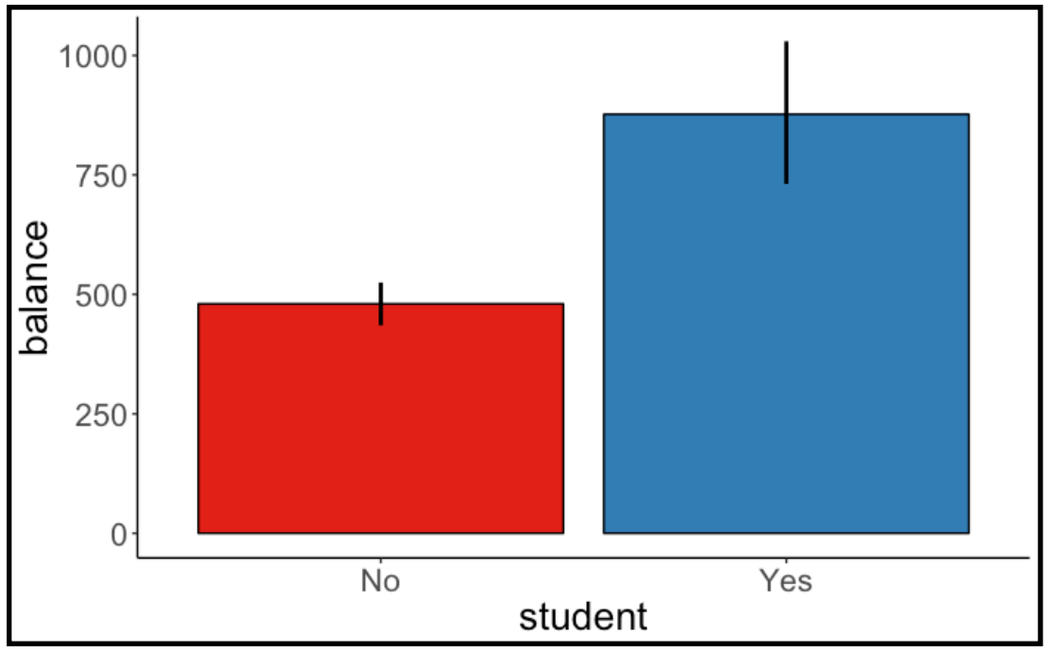
Relationships between variables



r correlation

R^2 variance explained

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

Coefficient of determination R^2

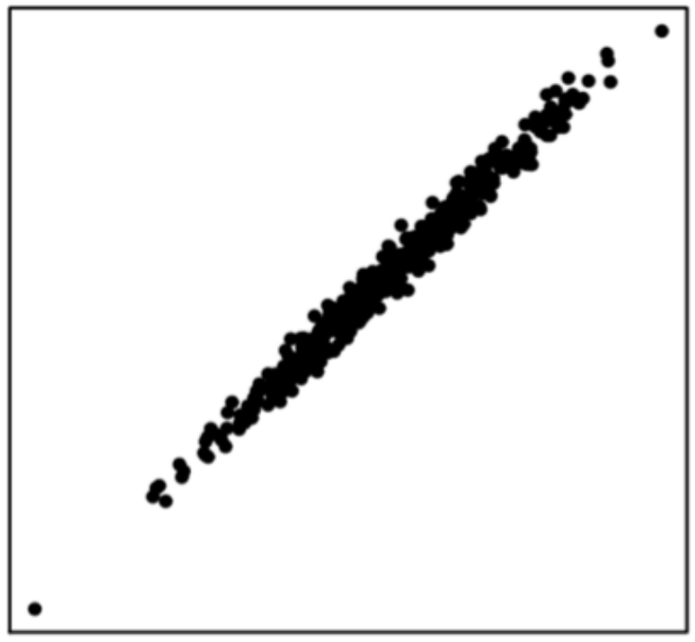
$$R^2 = 1 - \frac{\text{SS}_{\text{residual}}}{\text{SS}_{\text{total}}}$$

equivalent to PRE for the special case in which

$$\text{PRE} = 1 - \frac{\text{SSE}(A)}{\text{SSE}(C)} \qquad \textbf{Model C: } Y_i = \beta_0 + \epsilon_i$$

Common effect sizes

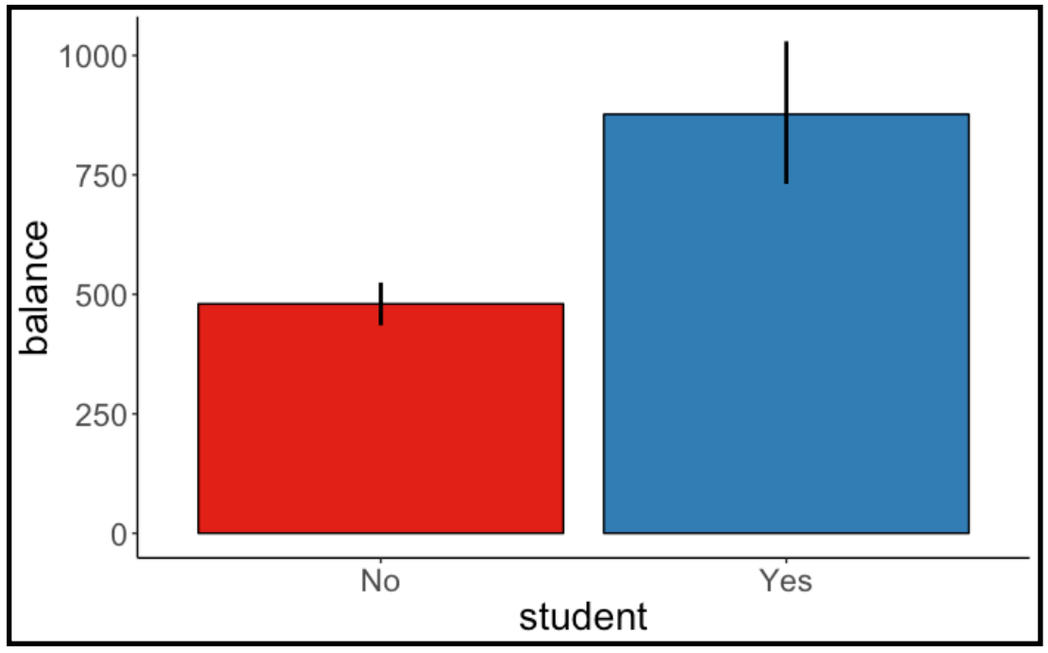
Relationships between variables



r correlation

R^2 variance explained

Differences between groups



Cohen's d

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

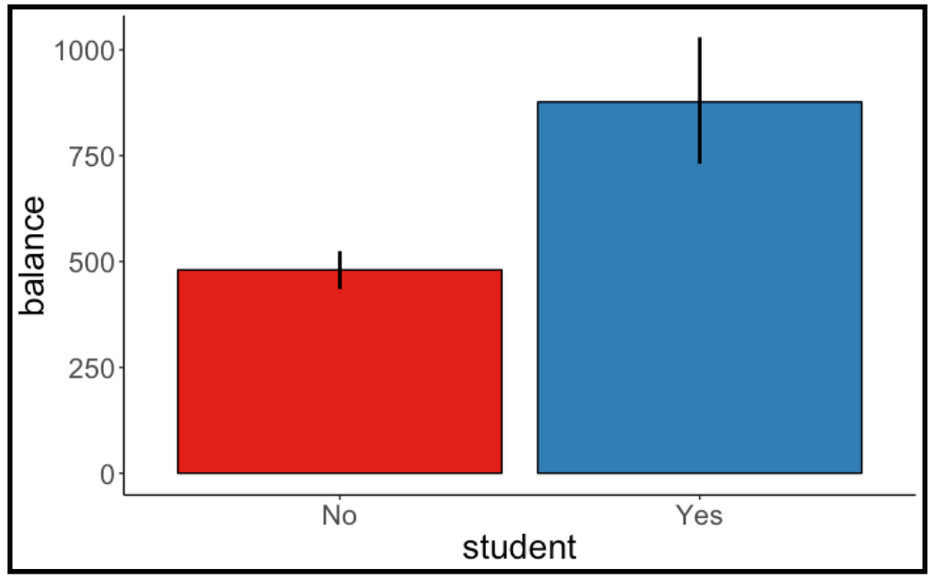
Difference between two means in pooled standard deviation

Cohen's *d* in R

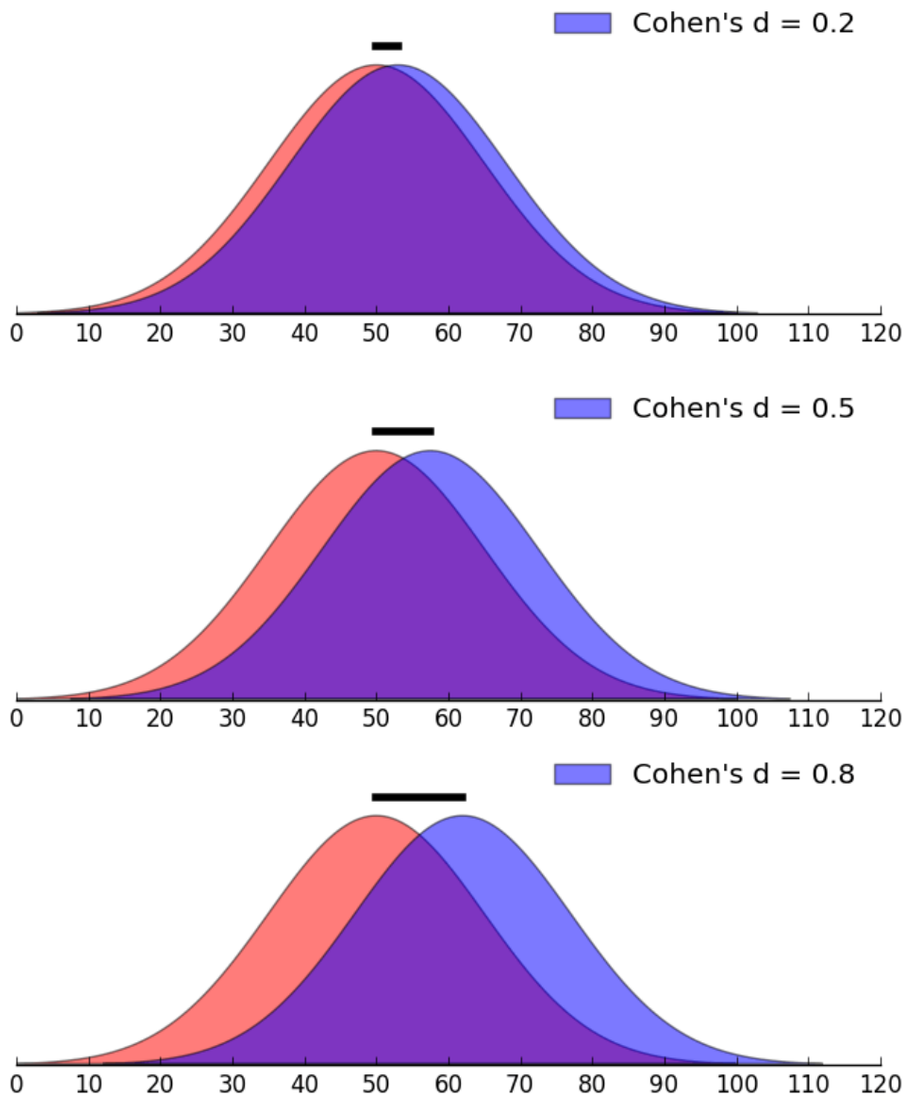
formula

```
1 library("lsr")
2 cohensD(x = balance ~ student,
3         data = df.credit)
```

d = 0.89



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

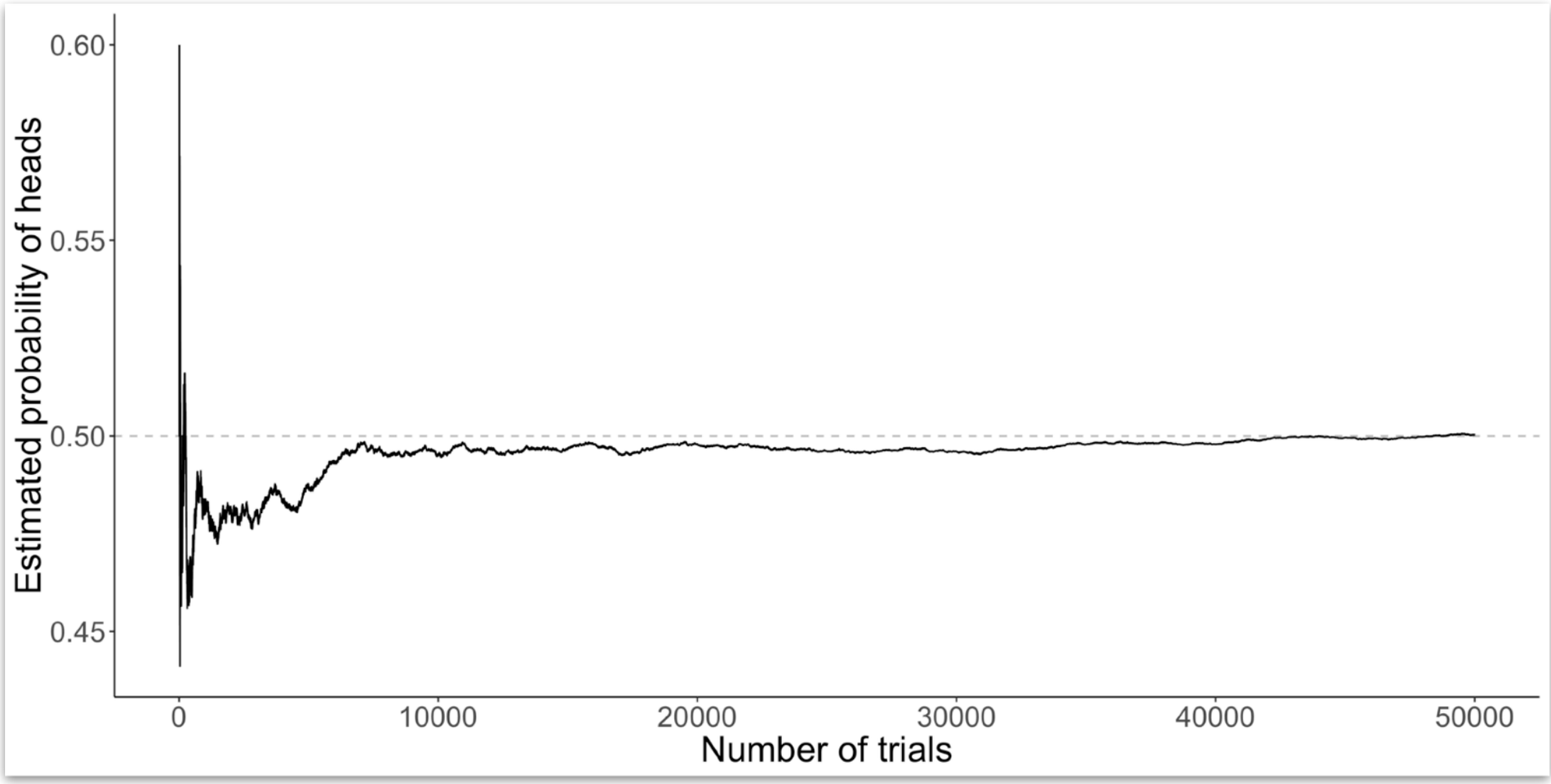


Determining sample size

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

Frequentist interpretation

Probabilities = **long-range frequencies**



law of large numbers = empirical probability will approximate the true probability as the sample size increases

Example

H_0: p(coin = heads) = 0.5

H_1: p(coin = heads) = 0.75

$\alpha = 0.05$

$1 - \beta = 0.8$

How many times should we flip the coin to have a high probability (or *power*), say 0.80, of correctly rejecting H_0 if our coin is indeed loaded to land heads 75% of the time?

```
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")
```

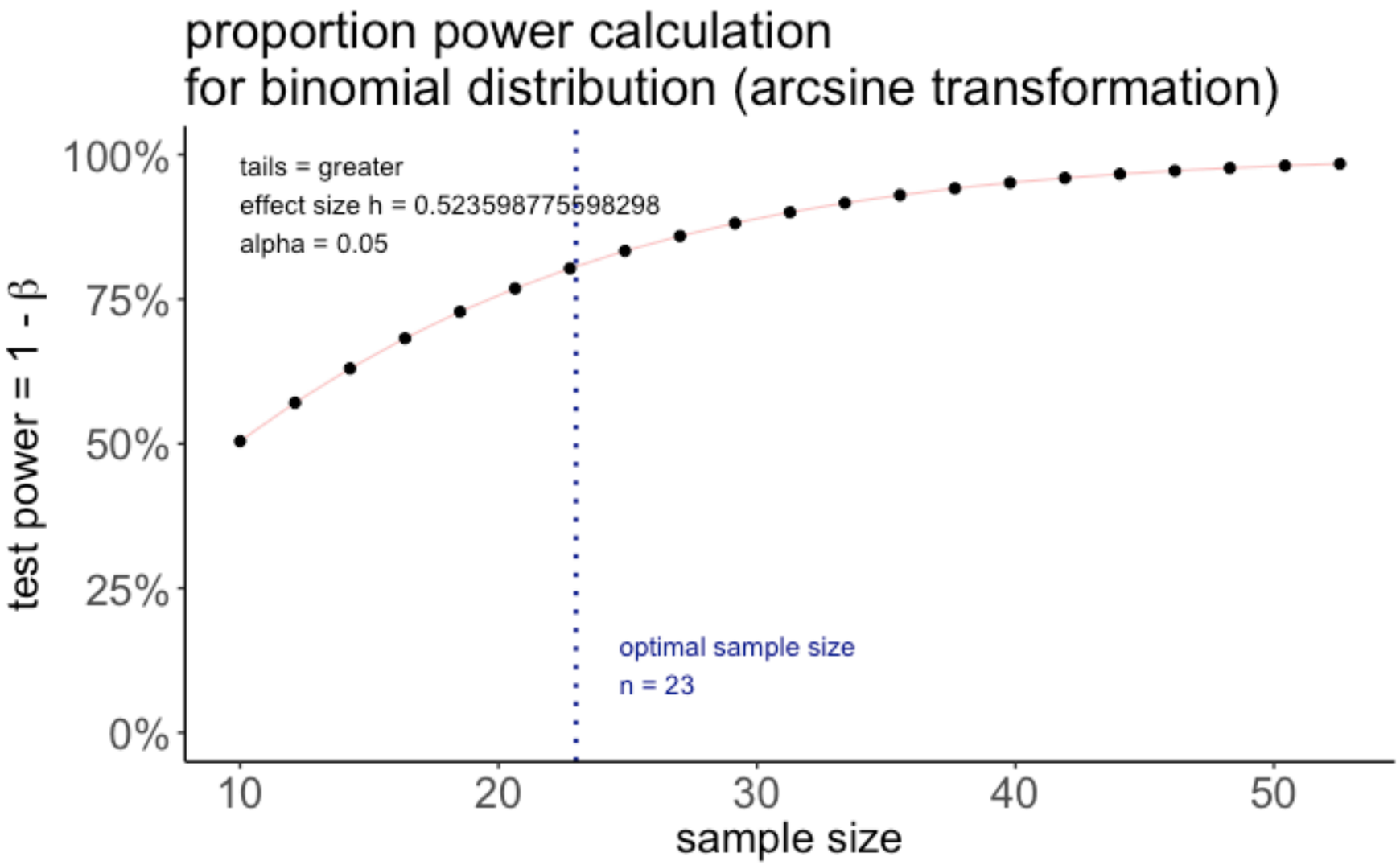
We need to flip it 23 times.

```
proportion power
calculation for binomial
distribution (arcsine
transformation)

h = 0.5235988
n = 22.55126
sig.level = 0.05
power = 0.8
alternative = greater
```

Example

```
1 pwr.p.test(h = ES.h(p1 = 0.75, p2 = 0.50),
2              sig.level = 0.05,
3              power = 0.80,
4              alternative = "greater") %>%
5   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 for a given α
- determine the probability of rejecting the H_0 (given that H_1 is true)

Let's simulate

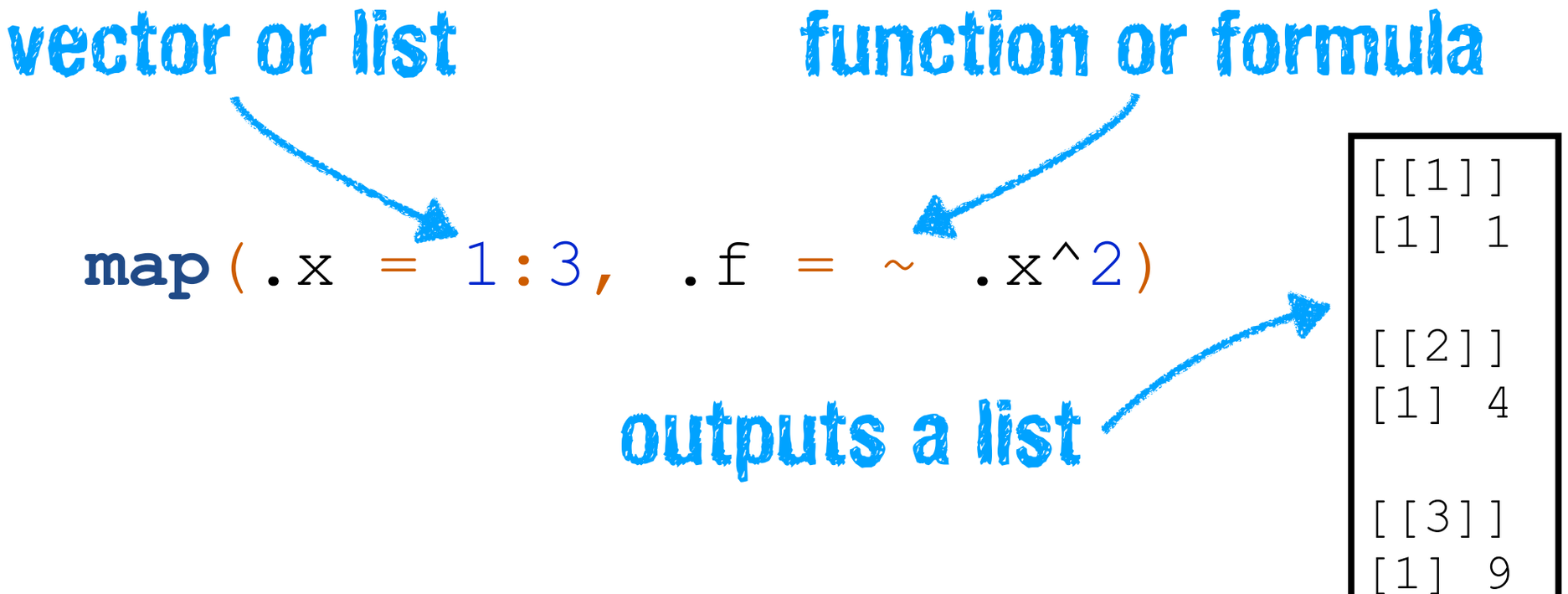
```
library("purrr")
```

automatically loaded with

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

`map()`

map()



- **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**()
- using **map**() allows one to avoid writing loops
- it's extremely powerful in combination with data frames

`map()`

vector or list

function or formula

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

outputs a list

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

[[2]]
[1] 4

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

same same but different

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

anonymous function

`map()`

vector or list

function or formula

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

outputs a list

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

[[2]]
[1] 4

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

different versions of `map()`

differ in what they output

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

outputs a vector

```
[1] 1 4 9
```

map()

same same but different

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

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

map(1:3, ~ .^2)

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

map2()

with multiple arguments

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

```
# with multiple arguments
map2_dbl(.x = 1:3,  .y = 1:3,  .f = ~  .x * .y)
```

Let's simulate ...

```
1 # number of simulations
2 n_simulations = 1000
3
4 # run simulation
5 df.power = crossing(n = seq(10, 50, 1),
6                     simulation = 1:n_simulations,
7                     p = c(0.75, 0.8, 0.85)) %>%
```

set up simulation grid

n	simulation	p
10	1	0.75
10	1	0.80
10	1	0.85
10	2	0.75
10	2	0.80
10	2	0.85
10	3	0.75
10	3	0.80
10	3	0.85
10	4	0.75

Let's simulate ...

```
1 # number of simulations
2 n_simulations = 1000
3
4 # run simulation
5 df.power = crossing(n = seq(10, 50, 1),
6                     simulation = 1:n_simulations,
7                     p = c(0.75, 0.8, 0.85)) %>%
8   mutate(index = 1:n()) %>%
9   mutate(response = rbinom(n = n(), size = n, prob = p)) %>%
```

generate random data

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

Let's simulate ...

```
1  # number of simulations
2  n_simulations = 1000
3
4  # run simulation
5  df.power = crossing(n = seq(10, 50, 1),
6                      simulation = 1:n_simulations,
7                      p = c(0.75, 0.8, 0.85)) %>%
8    mutate(index = 1:n()) %>%
9    mutate(response = rbinom(n = n(), size = n, prob = p)) %>%
10   group_by(index, simulation, p) %>%
11   nest() %>%
```

put data in a list column

	index	simulation	p	data
1	1	1	0.75	list(n = 10, response = 7)
2	2	1	0.80	list(n = 10, response = 8)
3	3	1	0.85	list(n = 10, response = 9)
4	4	2	0.75	list(n = 10, response = 9)
5	5	2	0.80	list(n = 10, response = 9)
6	6	2	0.85	list(n = 10, response = 10)
7	7	3	0.75	list(n = 10, response = 9)
8	8	3	0.80	list(n = 10, response = 5)
9	9	3	0.85	list(n = 10, response = 6)
10	10	4	0.75	list(n = 10, response = 7)

Let's simulate ...

	index	simulation	p	data	fit
1	1	1	0.75	list(n = 10, response = 7)	list(statistic = c(`number of successes` = 7), parame...
2	2	1	0.80	list(n = 10, response = 9)	list(statistic = c(`number of successes` = 9), parame...
3	3	1	0.85	list(n = 10, response = 9)	list(statistic = c(`number of successes` = 9), parame...
4	4	2	0.75	list(n = 10, response = 8)	list(statistic = c(`number of successes` = 8), parame...
5	5	2	0.80	list(n = 10, response = 8)	list(statistic = c(`number of successes` = 8), parame...
6	6	2	0.85	list(n = 10, response = 10)	list(statistic = c(`number of successes` = 10), param...
7	7	3	0.75	list(n = 10, response = 7)	list(statistic = c(`number of successes` = 7), parame...
8	8	3	0.80	list(n = 10, response = 8)	list(statistic = c(`number of successes` = 8), parame...
9	9	3	0.85	list(n = 10, response = 9)	list(statistic = c(`number of successes` = 9), parame...
10	10	4	0.75	list(n = 10, response = 6)	list(statistic = c(`number of successes` = 6), parame...

```
12    mutate(fit = map(data,
13                     ~ binom.test(x = .$response,
14                                  n = .$n,
15                                  p = 0.5,
16                                  alternative = "two.sided")),
```

run binomial test for each cell in the data column

map(list, function) applies function to each element of the list

Let's simulate ...

	index	simulation	p	p.value	n	response
1	1	1	0.75	1.093750e-01	10	8
2	2	1	0.80	1.953125e-03	10	10
3	3	1	0.85	3.437500e-01	10	7
4	4	2	0.75	2.148438e-02	10	9
5	5	2	0.80	2.148438e-02	10	9
6	6	2	0.85	3.437500e-01	10	7
7	7	3	0.75	3.437500e-01	10	7
8	8	3	0.80	1.093750e-01	10	8
9	9	3	0.85	2.148438e-02	10	9
10	10	4	0.75	2.148438e-02	10	9

```
13    mutate(fit = map(data,
14                     ~ binom.test(x = .$response,
15                                  n = .$n,
16                                  p = 0.5,
17                                  alternative = "two.sided")),
18           p.value = map_dbl(fit, ~ .$p.value)) %>%
19    unnest(data) %>%
20    select(-fit)
```

extract p-values

Let's simulate ...

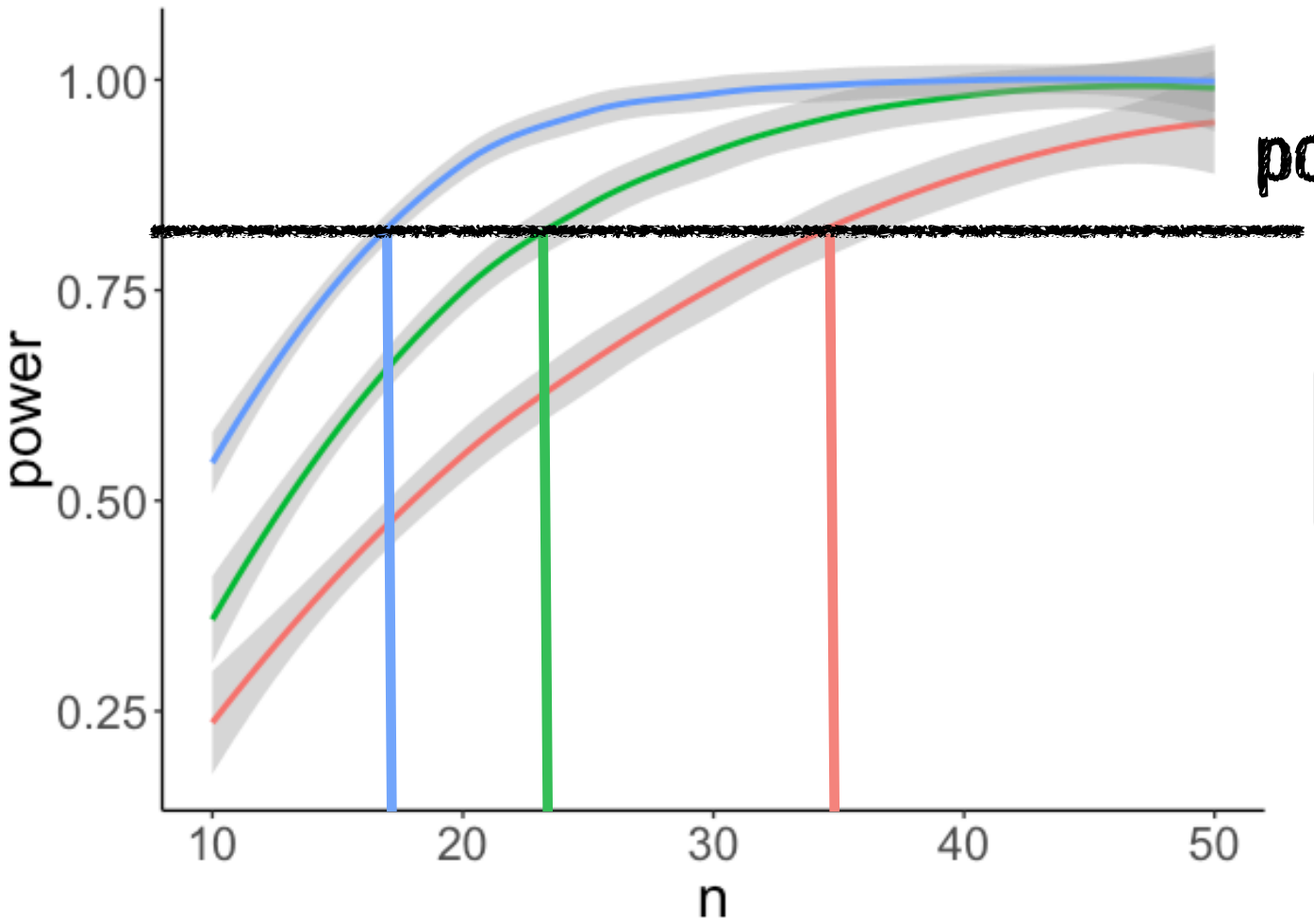
```
1 # data frame for plot
2 df.plot = df.power %>%
3   group_by(n, p) %>%
4   summarize(power = sum(p.value < 0.05) / n())
```

probability of rejecting H_0 when H_1 is true

Power

$$1 - \beta$$

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



p = probability of coin coming up heads

n	p	power
10	0.75	0.24
10	0.8	0.38
10	0.85	0.58
11	0.75	0.24
11	0.8	0.40
11	0.85	0.50
12	0.75	0.45
12	0.8	0.55
12	0.85	0.82
13	0.75	0.26

Let's simulate ...

```r
# analytic solution
pwr.p.test(h = ES.h(0.5, 0.75),
           power = 0.8,
           alternative = "two.sided")
```

```
proportion power calculation for binomial distribution (arcsine
transformation)

h = 0.5235988
n = 28.62923
sig.level = 0.05
power = 0.8
alternative = two.sided
```

```r
# based on simulations
df.plot %>%
  filter(p == 0.75, near(power, 0.8, tol = 0.02))
```

n	p	power
30	0.75	0.81

cool function that filters on close values

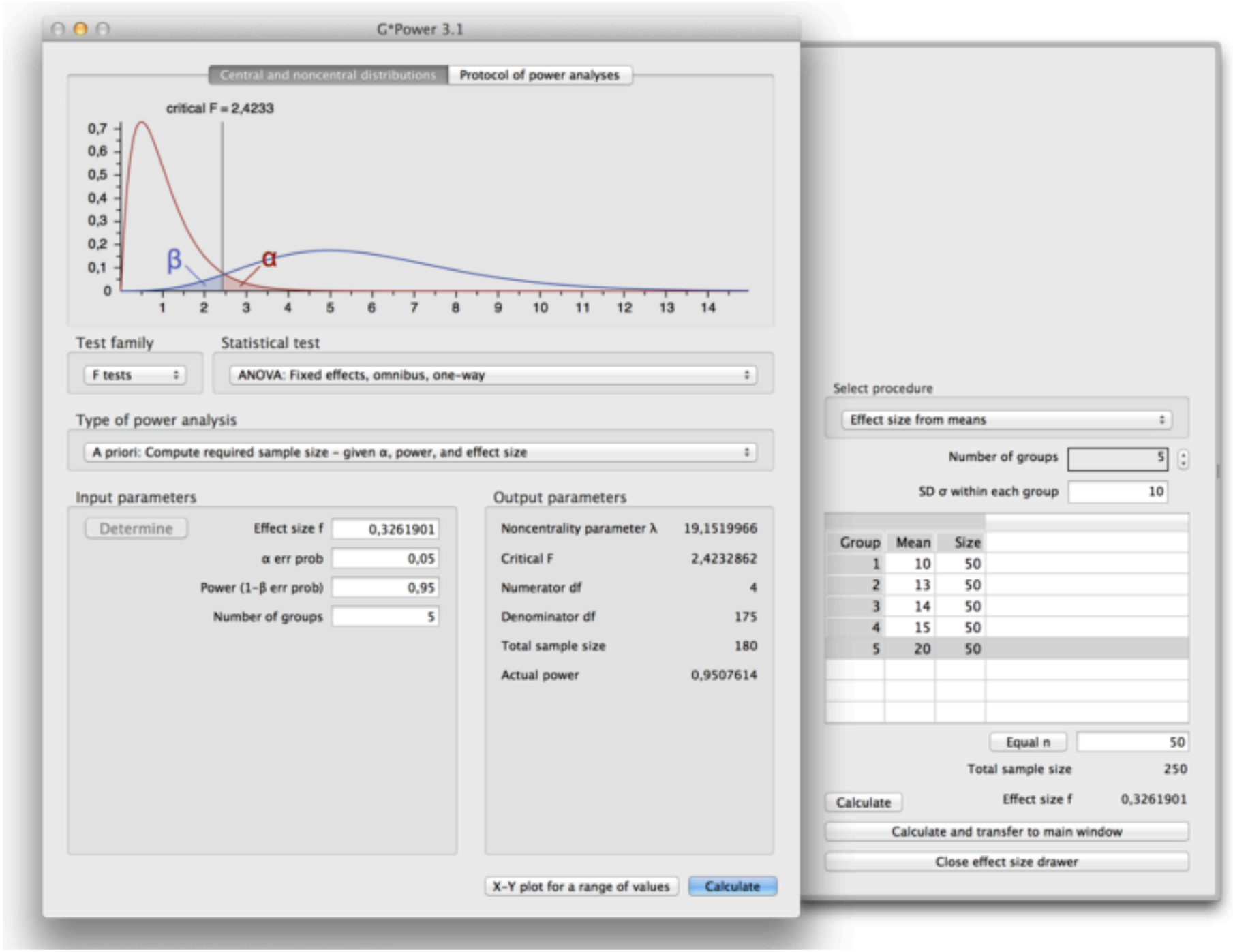
Let's simulate ...

- 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)

G*Power 3.1: Alternative software for power calculations



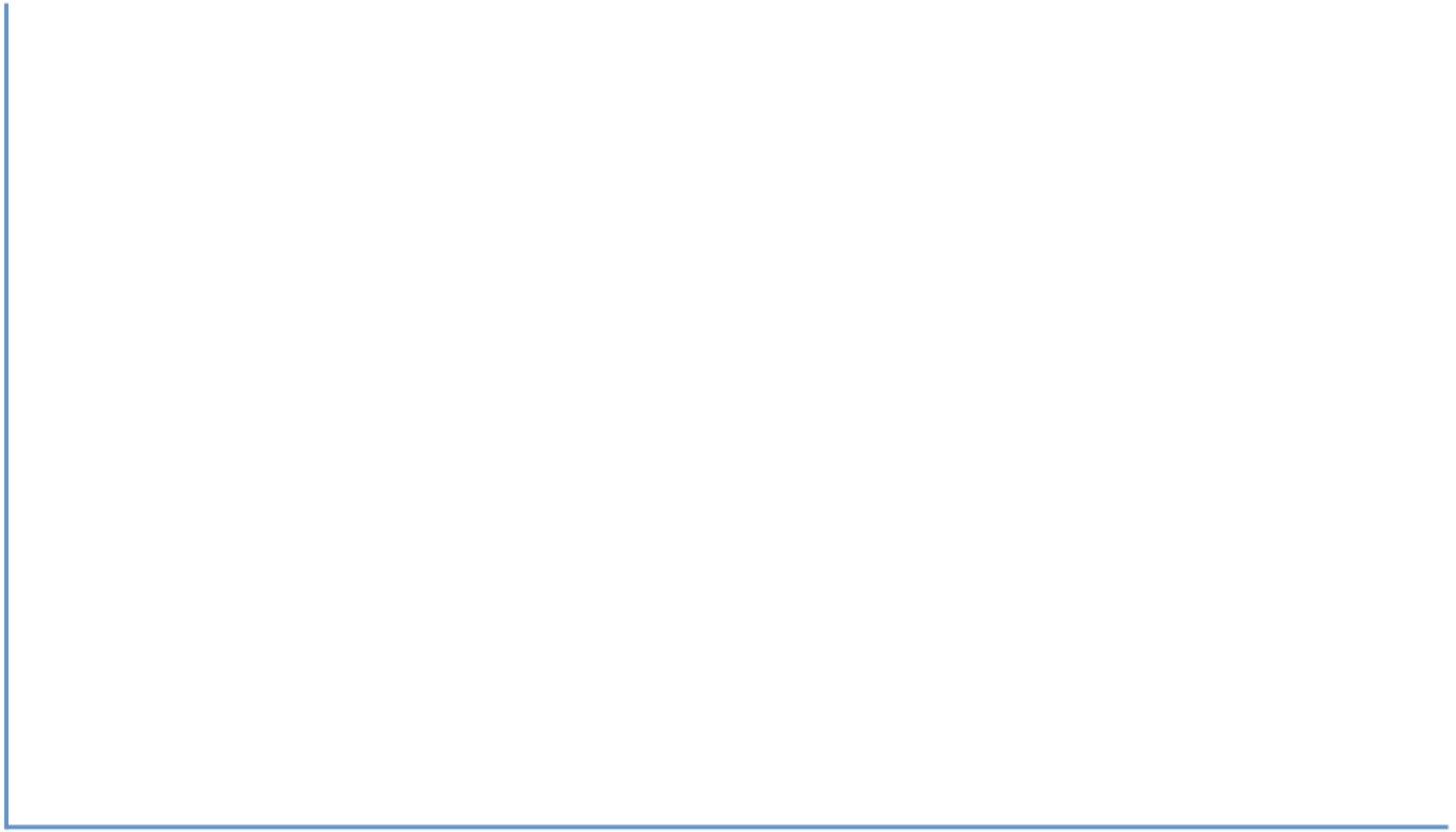
http://www.gpower.hhu.de/

Summary

- Contrasts
- Planned comparisons
- Making decisions
- Power analysis
- Effect sizes
- Determining sample size

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!