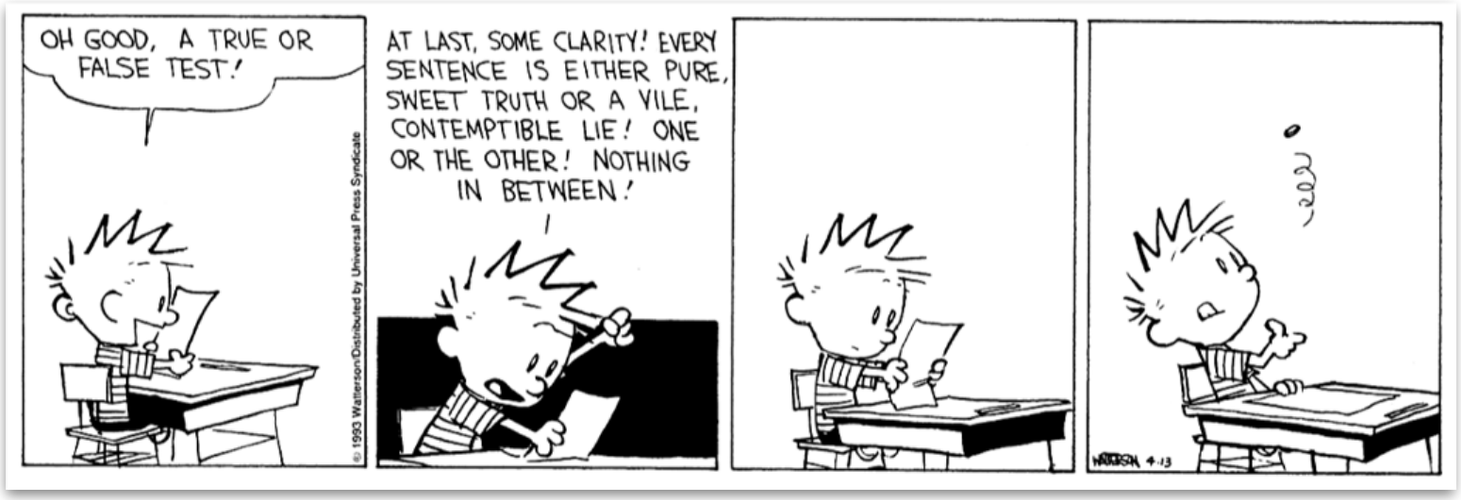
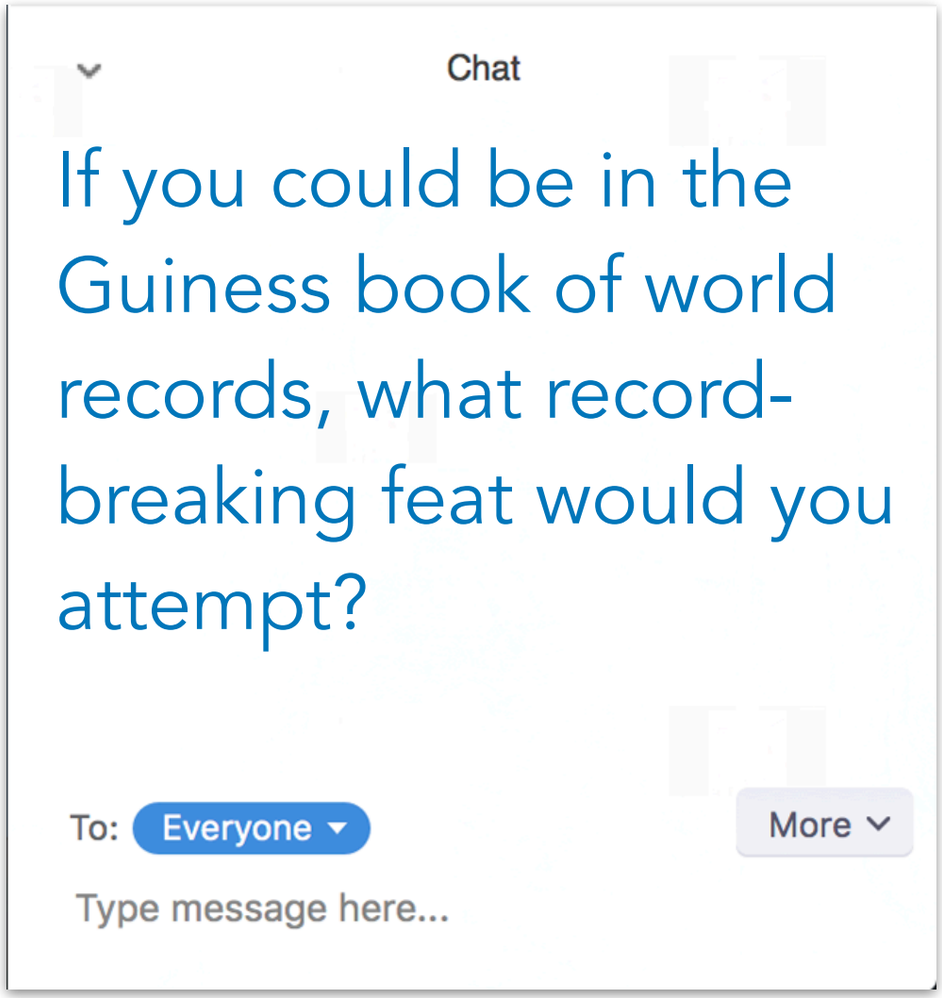
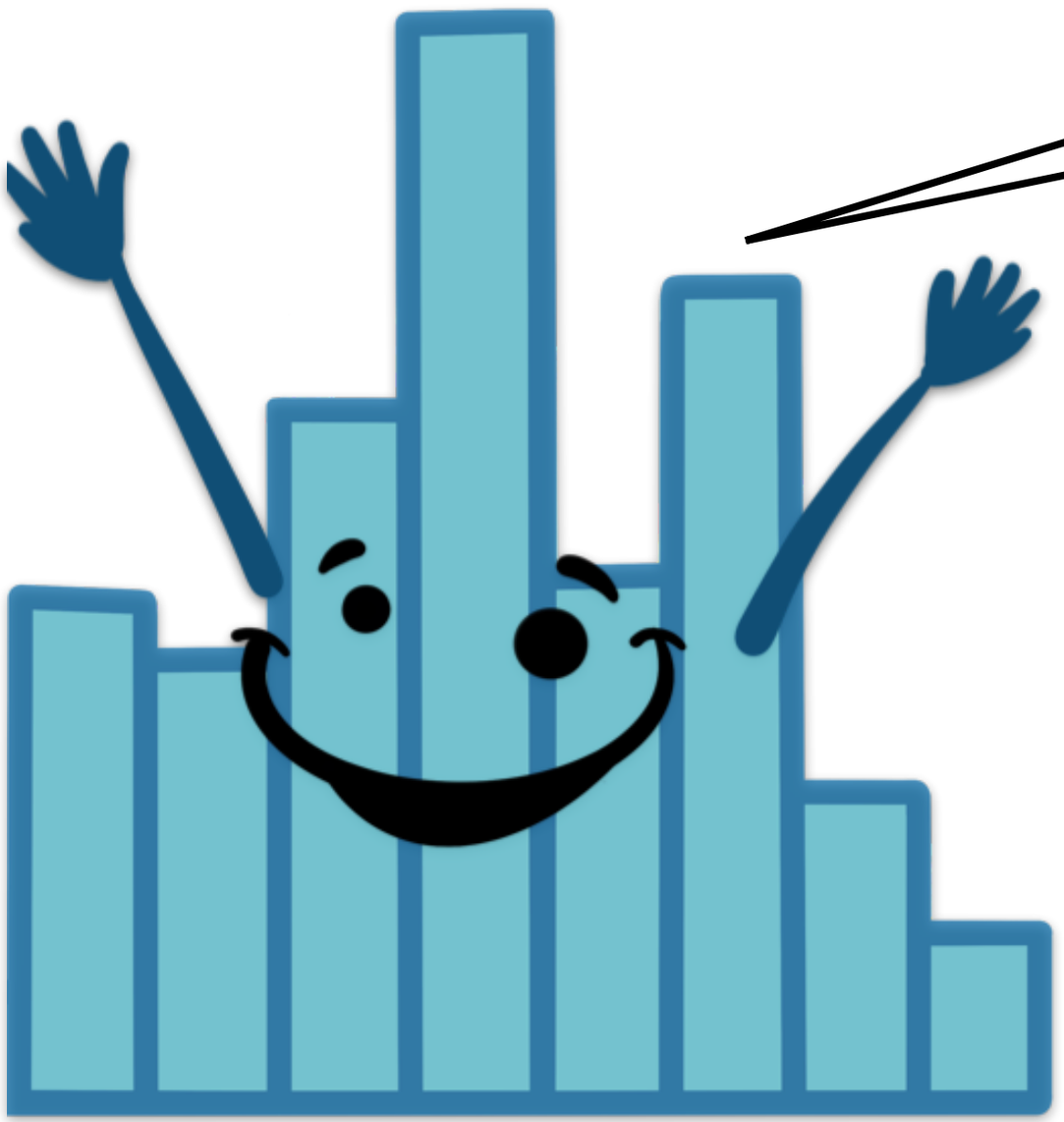
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

Model comparison





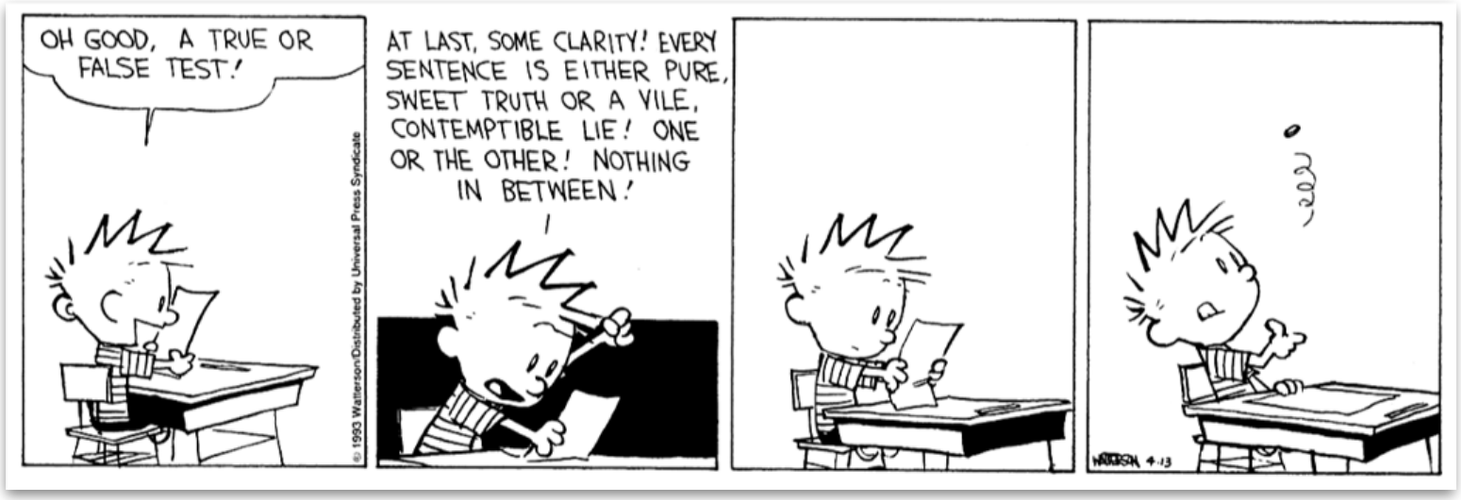
02/22/2021

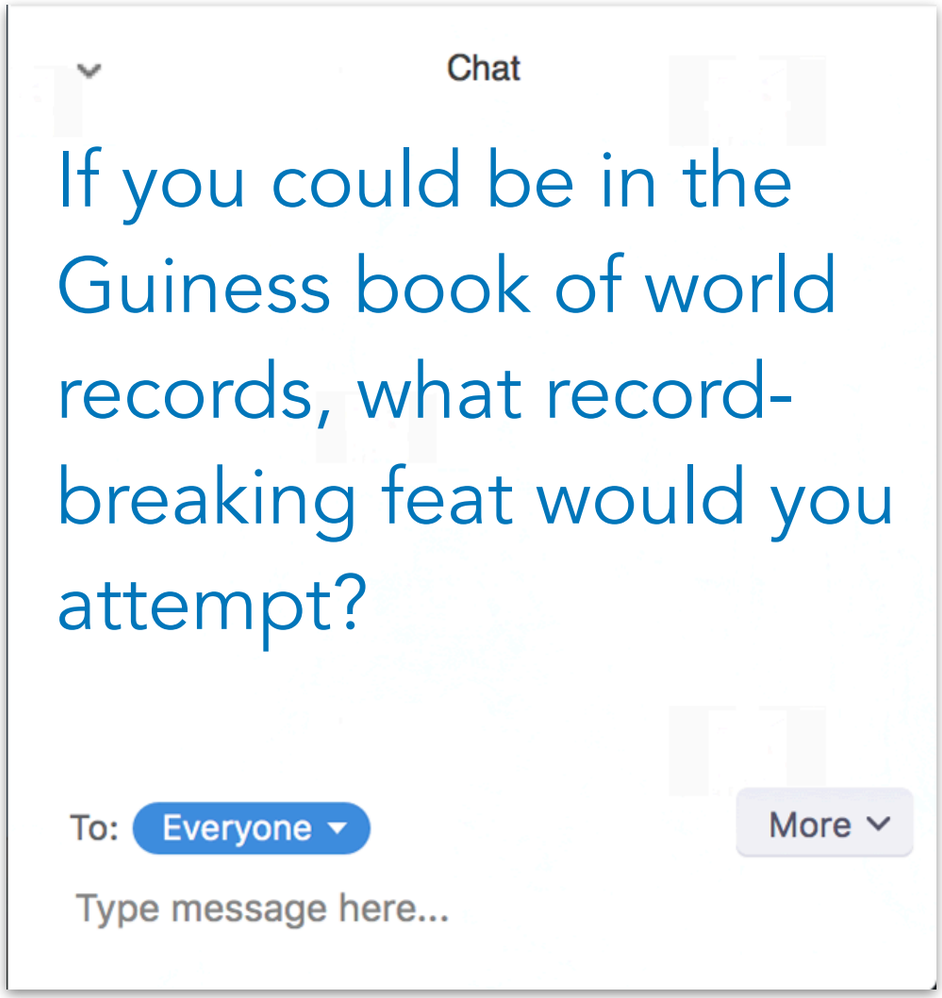


Remember to record the lecture!

PSYCH 252: Statistical Methods for Behavioral and Social Sciences

Model comparison





02/22/2021

Plan for today

- Model comparison
  - Cross-validation
  - AIC and BIC
- Linear mixed effects models

Model comparison

The general procedure

1. Define H_0 as Model C (compact) and H_1 as Model A (augmented)
2. Fit model parameters to the data
3. Calculate the proportional reduction of error (PRE) in our sample
4. Decide whether the augmented model is **worth it** by comparing the observed PRE in our sample to the sampling distribution of PRE (assuming that H_0 is true)

Any problems with our approach?

sometimes it doesn't work ...

Model C

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

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{student}_i + \epsilon_i$$

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{student}_i + \epsilon_i$$

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{student}_i + \epsilon_i$$

Model A

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{student}_i + \beta_2 \cdot \text{age}_i + \epsilon_i$$ ✓

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{student}_i + \beta_2 \cdot \text{age}_i + \epsilon_i$$ ✓

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{age}_i + \epsilon_i$$ ✗

$$\text{balance}_i = \beta_0 + \beta_1 \cdot \text{age}_i + \beta_2 \cdot \text{degree}_i + \epsilon_i$$ ✗

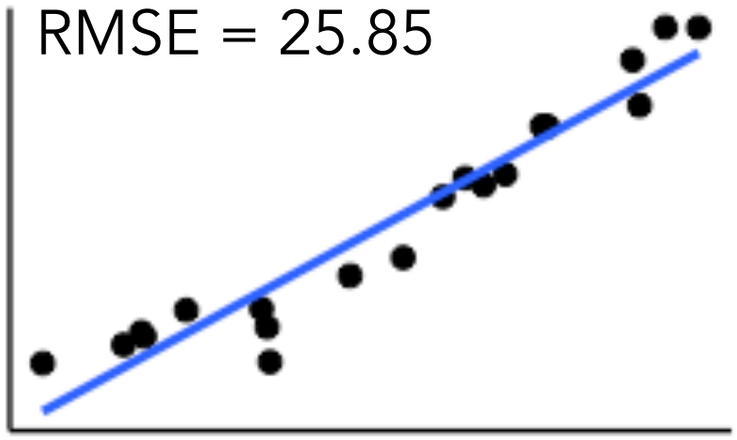
Tools for model comparison

- **`anova()`:** compare a compact model with an augmented model via the F-test
  - problem: only works for nested models (where the augmented model contains all the predictors of the compact model and more)
- **What if we want to compare models that aren't nested?**
  - Cross-validation
  - AIC and BIC
  - Bayesian data analysis (we'll get there soon)

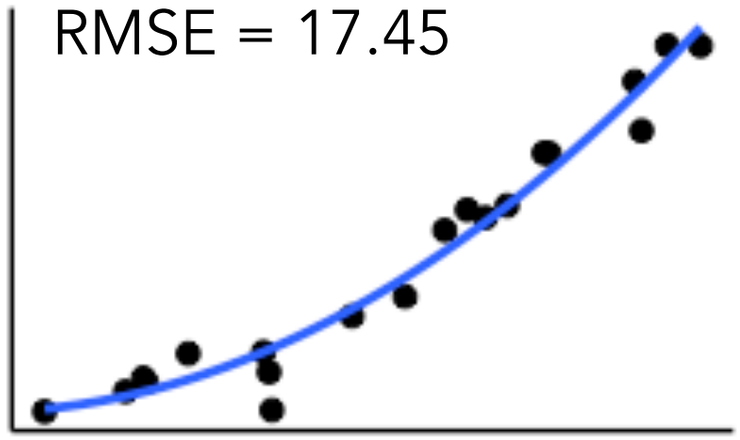
Cross-validation

Which model describes the data best?

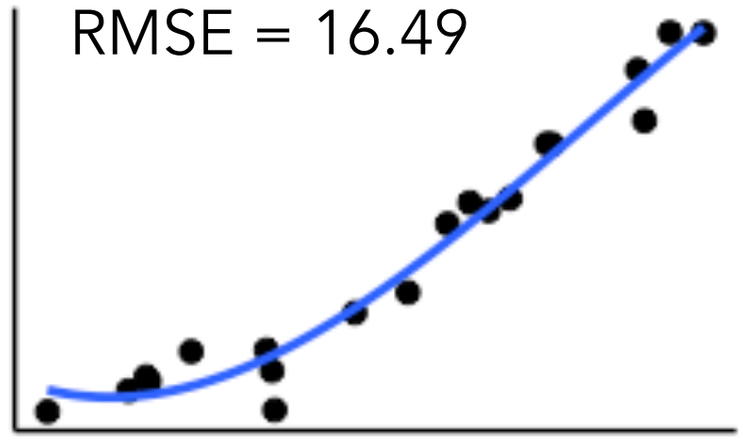
RMSE = 25.85

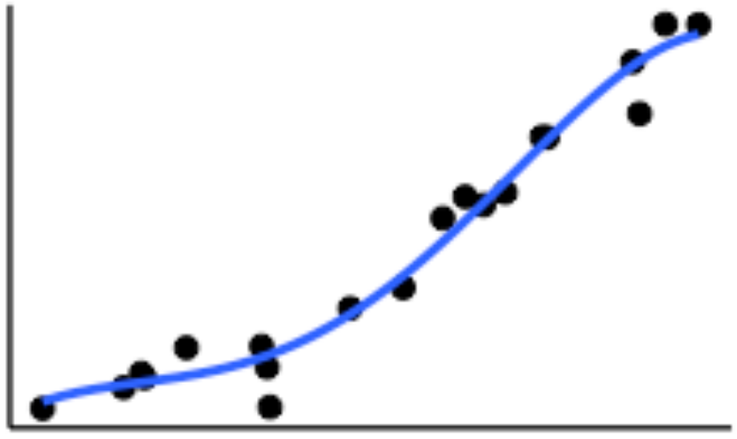


RMSE = 17.45

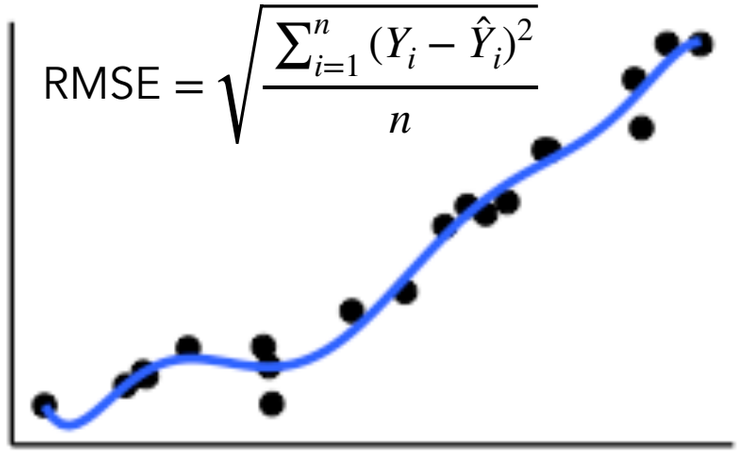


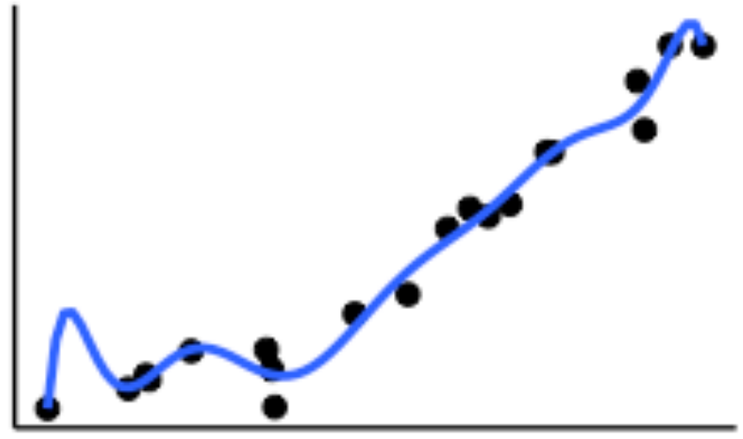
RMSE = 16.49

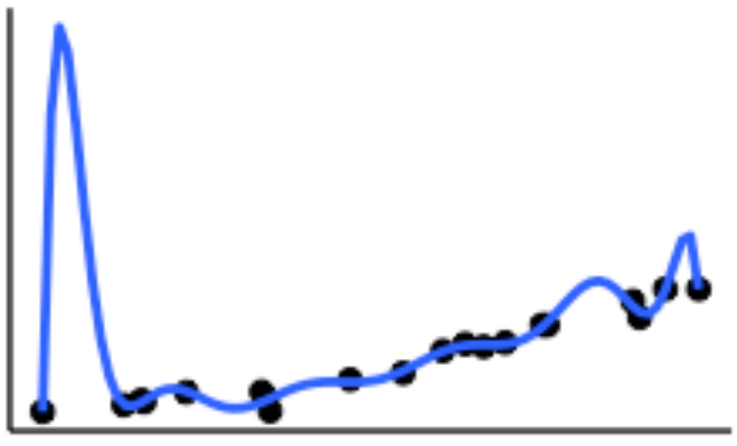


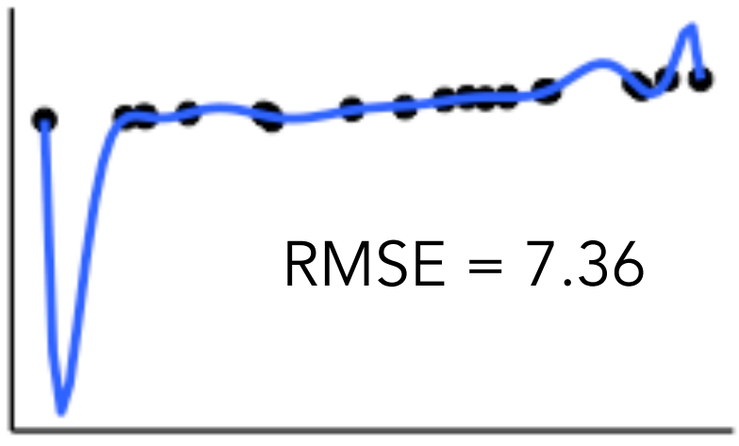


$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{n}}$$

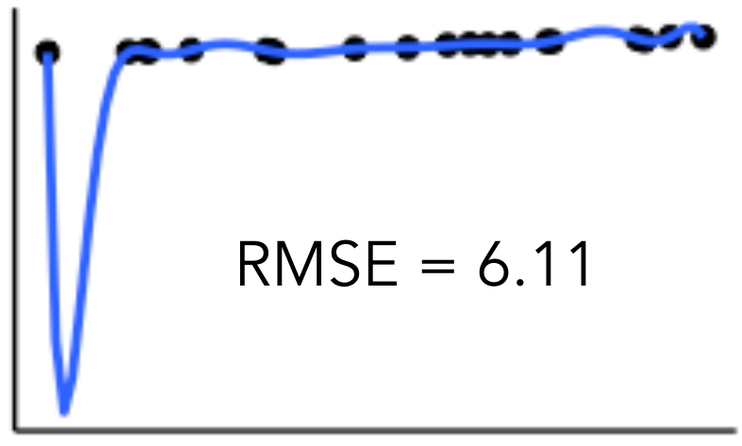








RMSE = 7.36

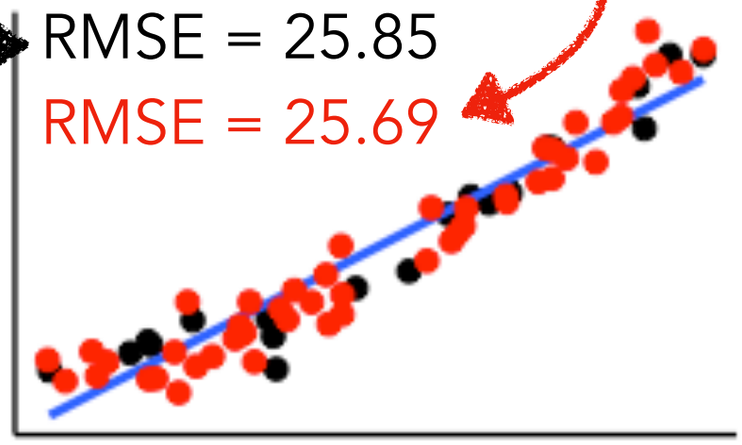


RMSE = 6.11

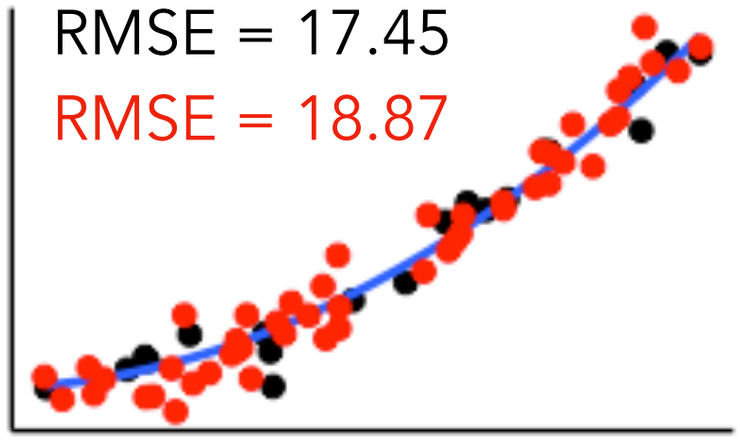
Which model describes the data best?

training error validation error

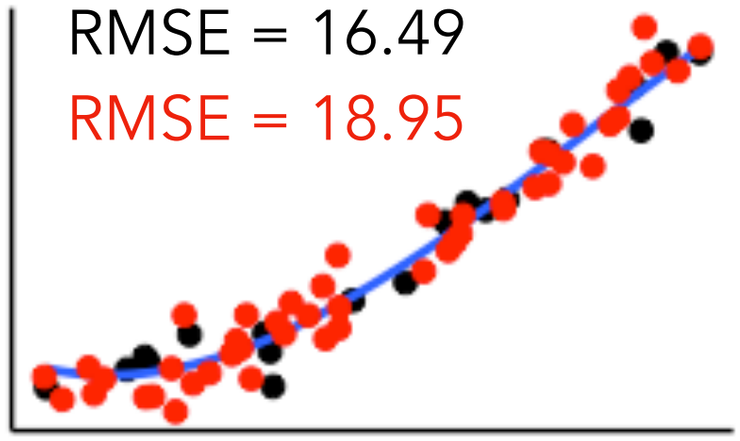
RMSE = 25.85
RMSE = 25.69

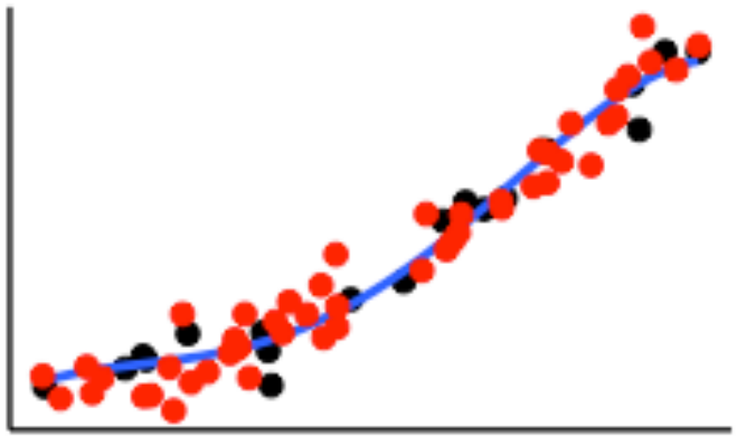


RMSE = 17.45
RMSE = 18.87

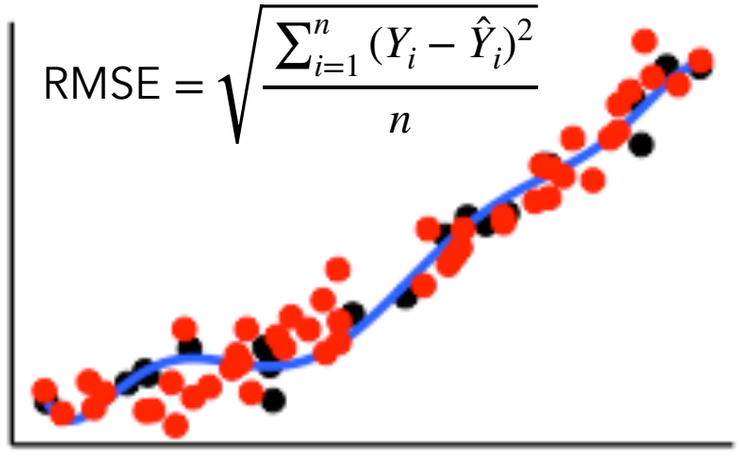


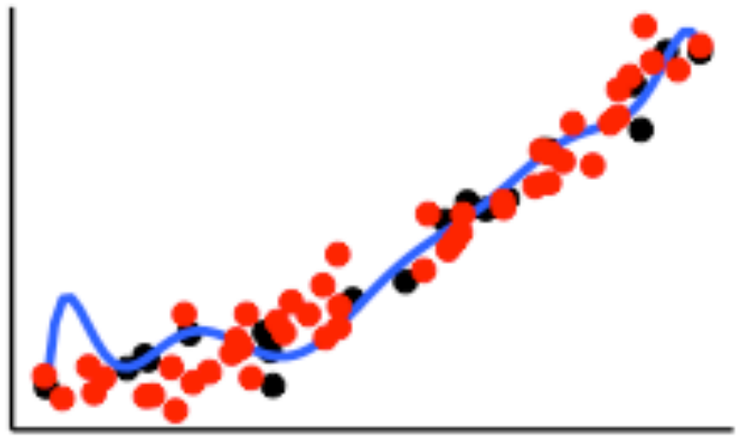
RMSE = 16.49
RMSE = 18.95

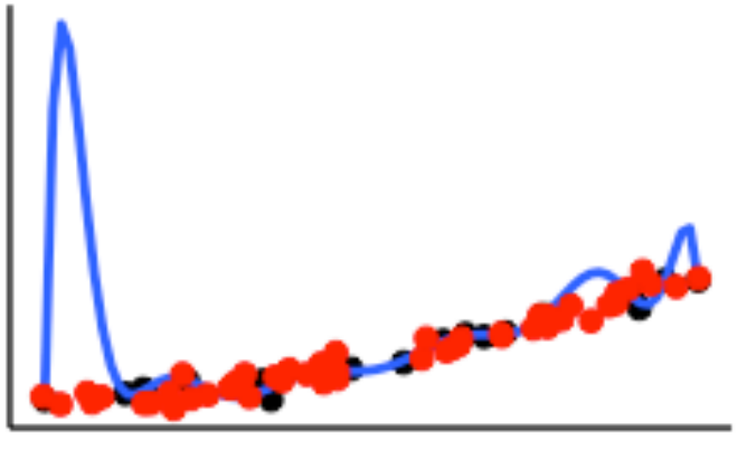


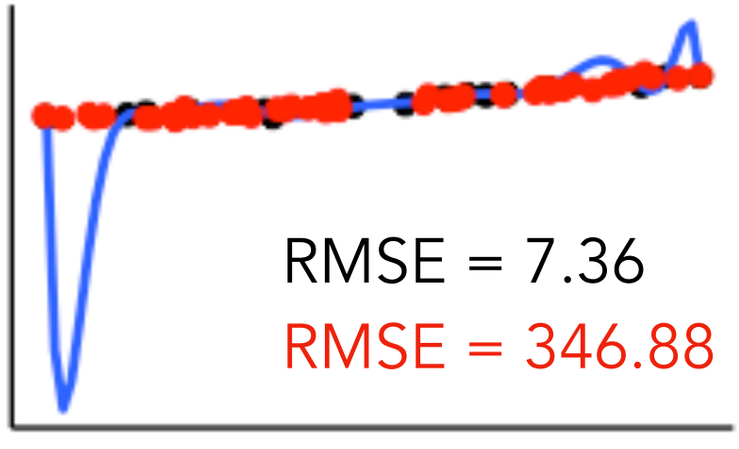


$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}}$$

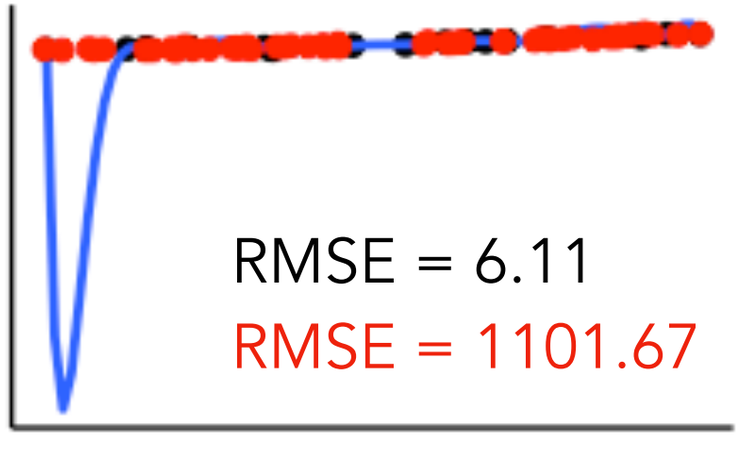








RMSE = 7.36
RMSE = 346.88



RMSE = 6.11
RMSE = 1101.67

Underfitting vs. Overfitting

ONE WAY TO EXPLAIN UNDERFITTING

Underfitting vs. Overfitting

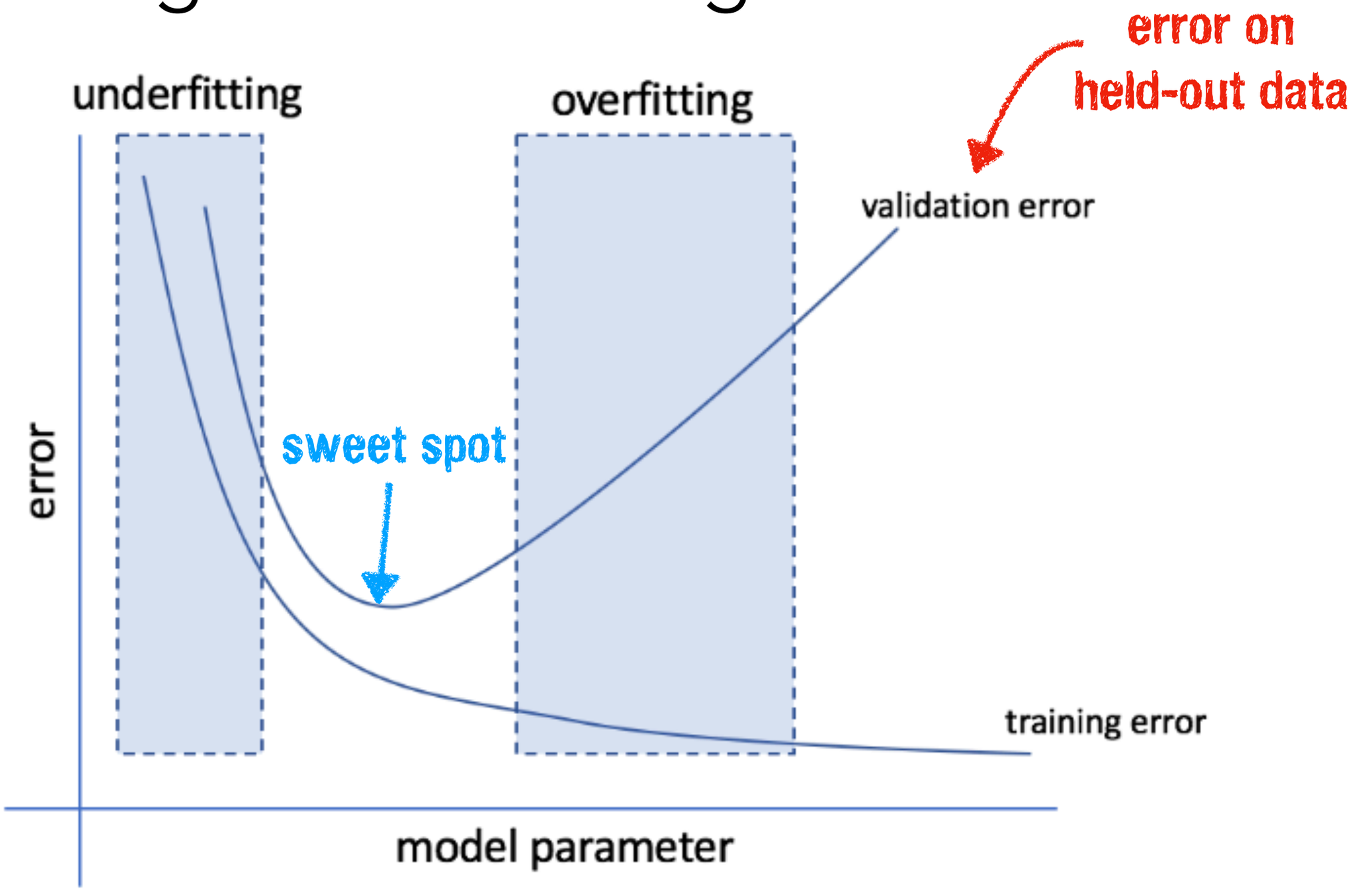
- **a good model should**:
  - explain the actual data well
  - predict future data well
- **bias-variance tradeoff**:
  - **bias** = error from erroneous assumptions in the model, high bias can cause a model to miss the relevant relations between predictors and outcome

underfitting

- **variance** = error from sensitivity to small fluctuations in the data, high variance can cause a model to fit the random noise in the data

overfitting

Underfitting vs. Overfitting



the goal is to find the **sweet spot** between underfitting and overfitting

Leave-one-out crossvalidation

LOO

Leave One Out

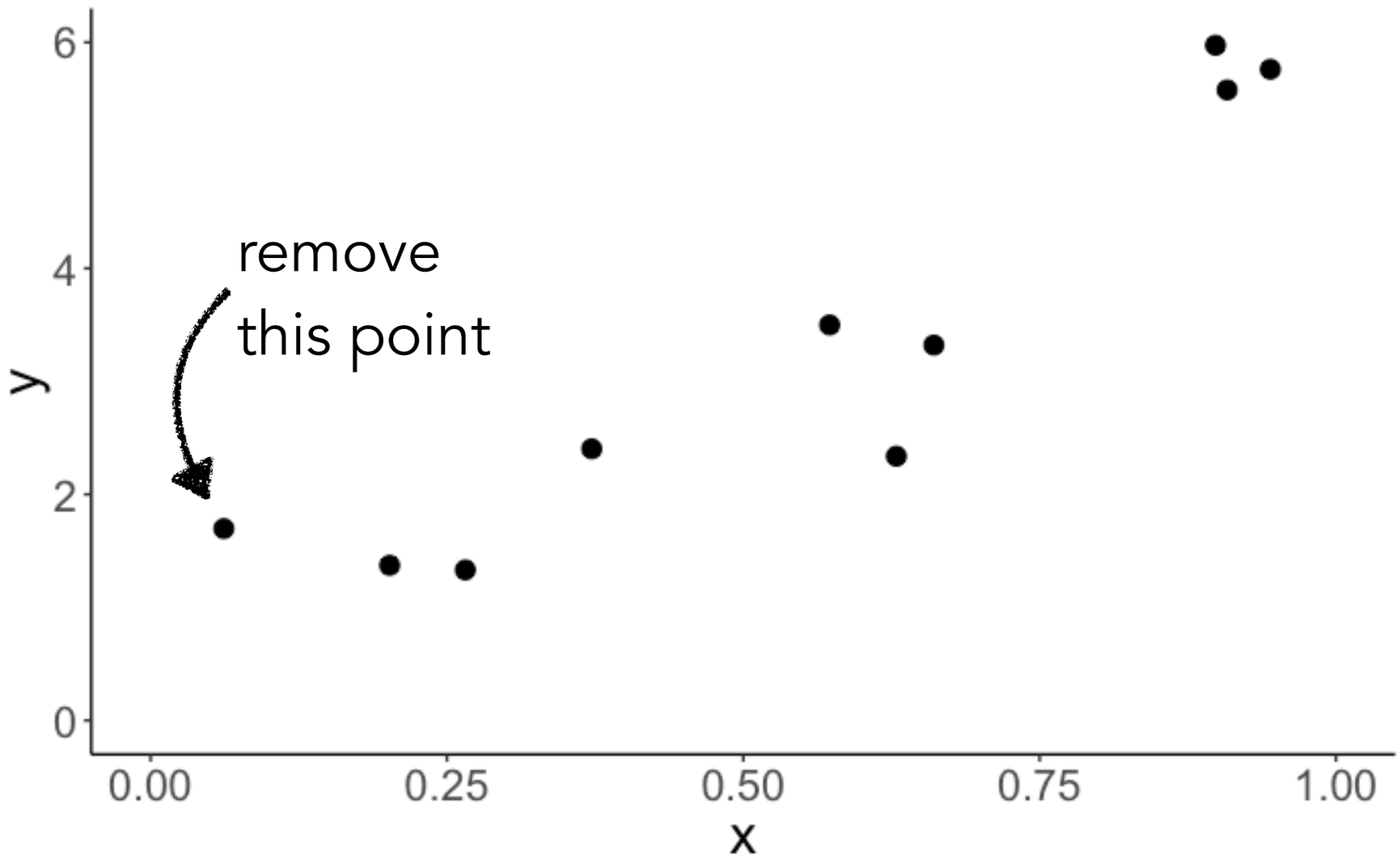
non-inspirational quote

Leave one out cross-validation

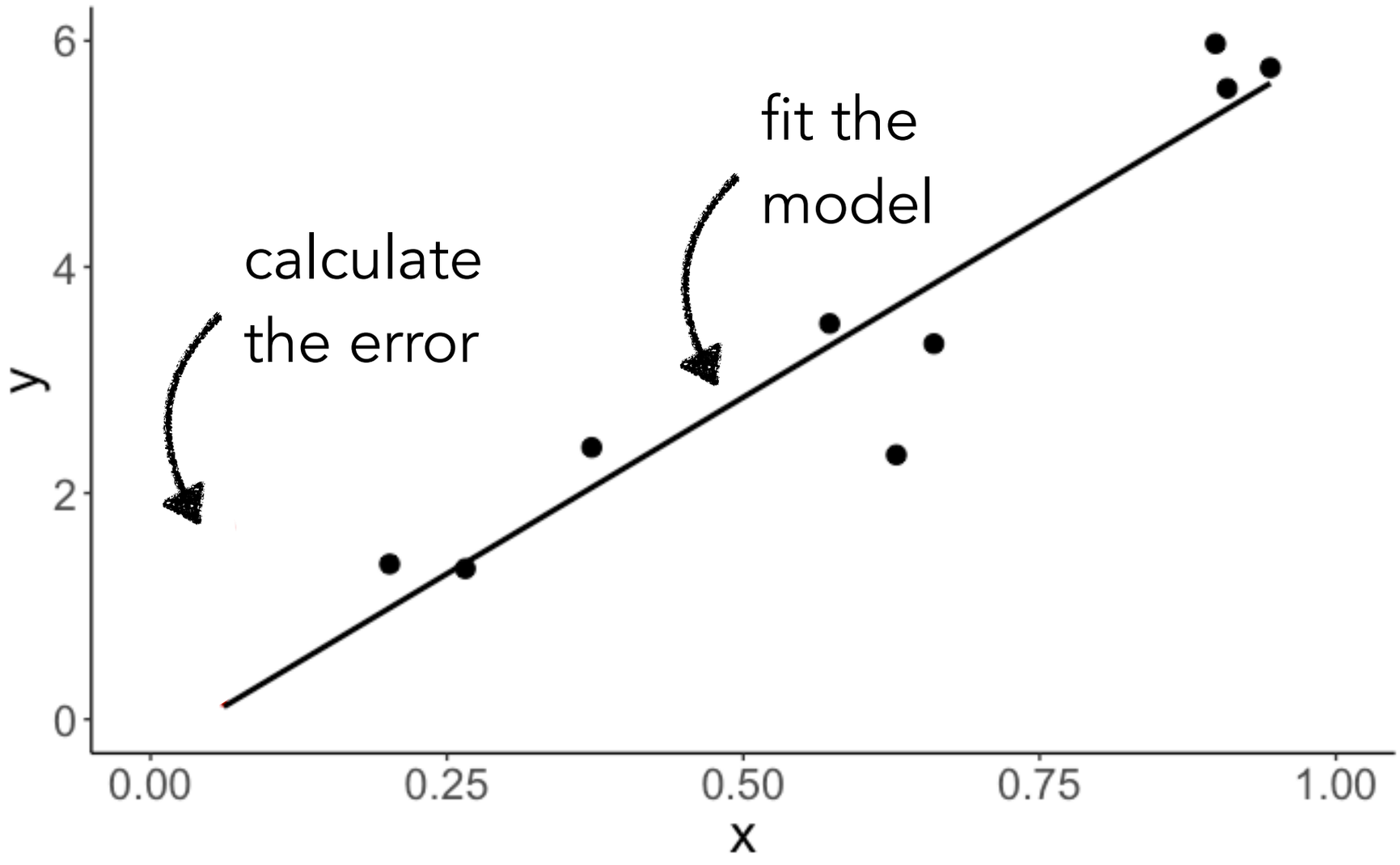
- train the model on all the data points except for one
- calculate the prediction error for the held-out data point

repeat for all data points

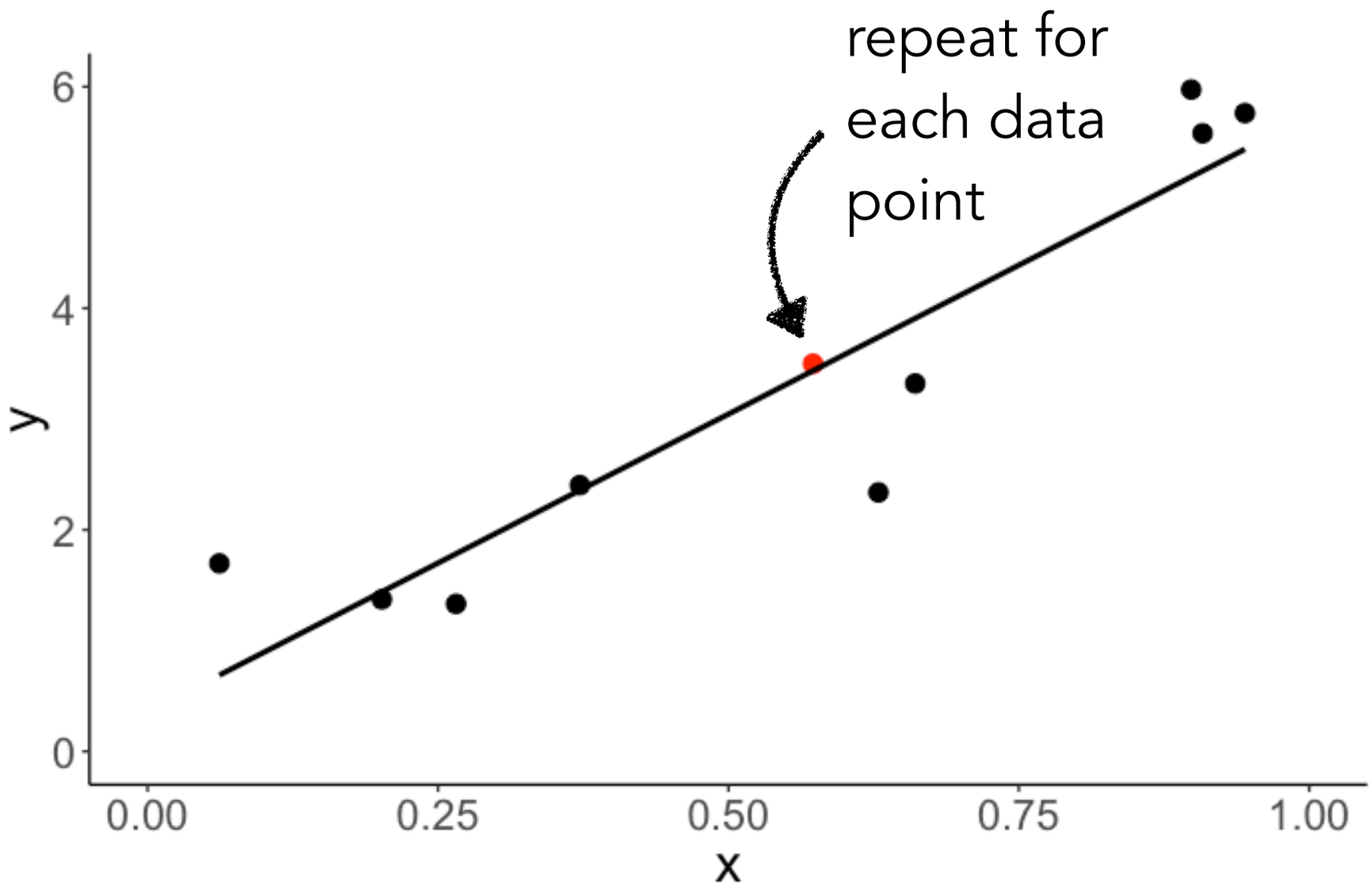
Leave one out cross-validation



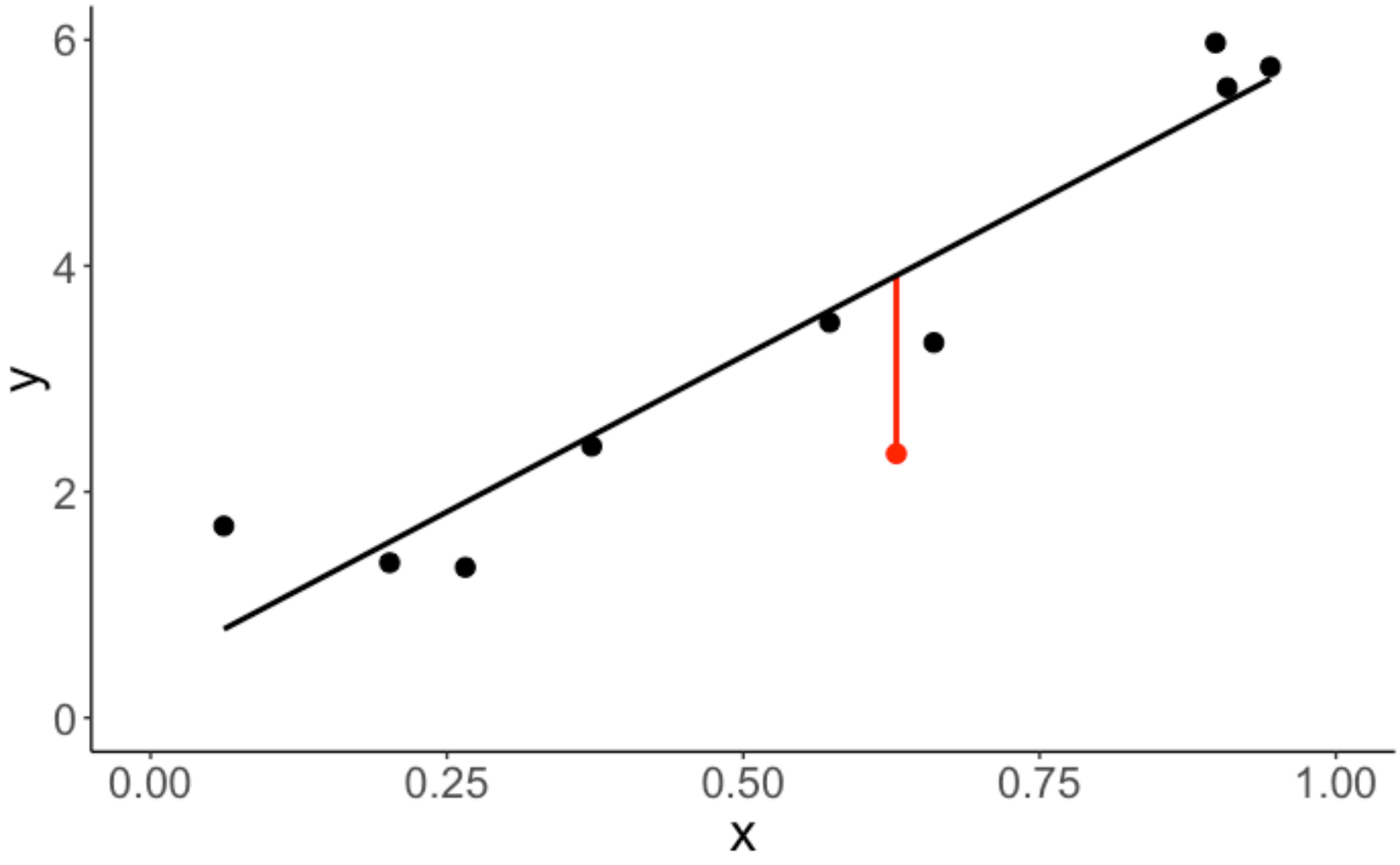
Leave one out cross-validation



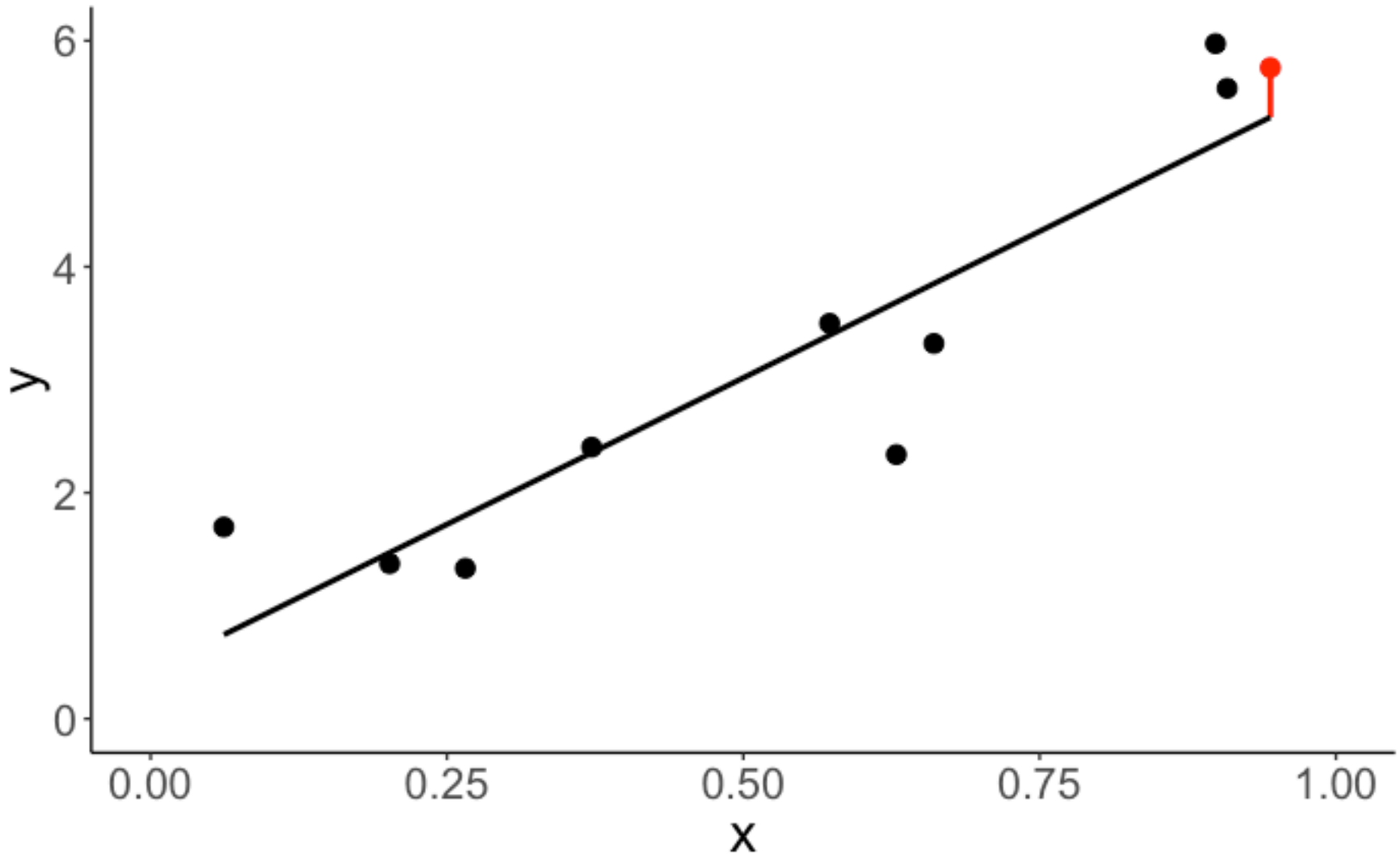
Leave one out cross-validation



Leave one out cross-validation

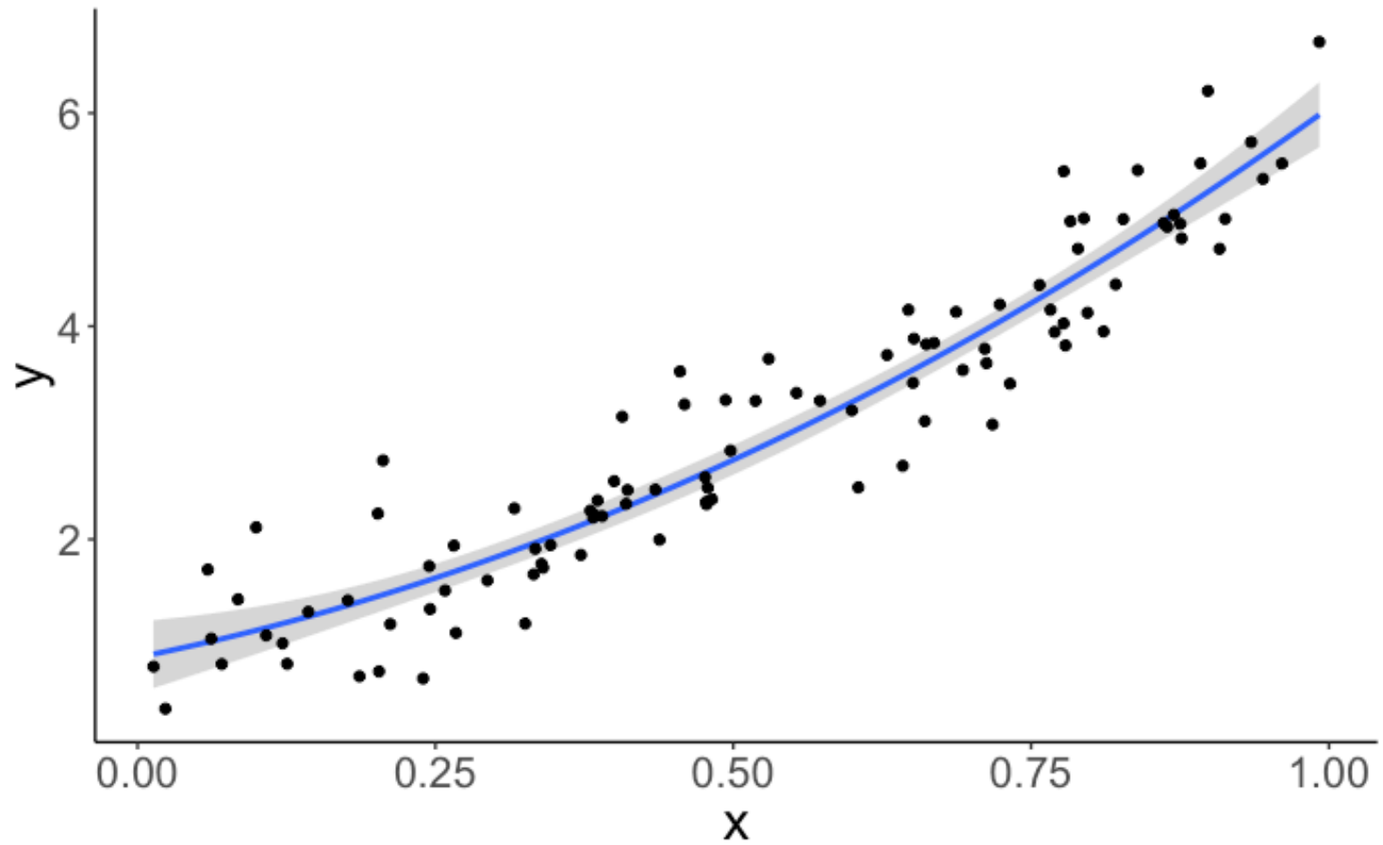


Leave one out cross-validation



Leave-one out crossvalidation

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

# parameters
sample_size = 100
b0 = 1
b1 = 2
b2 = 3
sd = 0.5
```



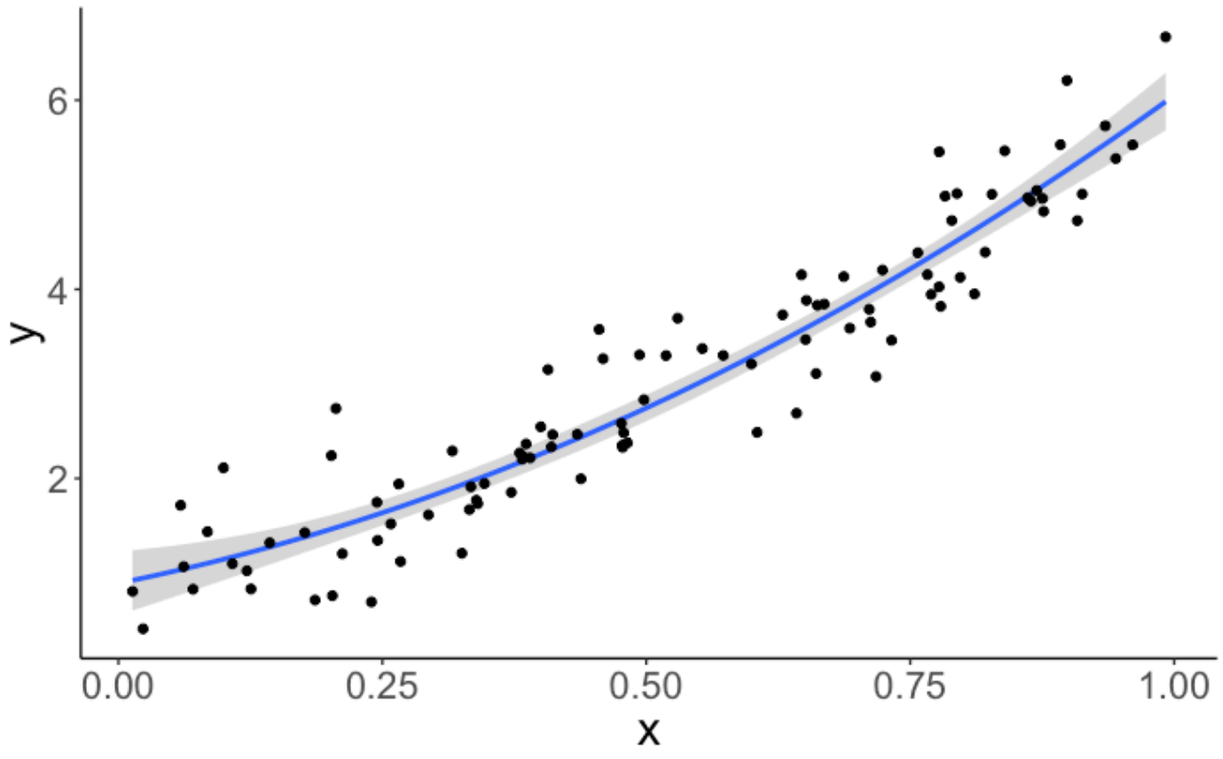

```r

# sample
df.data = tibble(
  participant = 1:sample_size,
  x = runif(sample_size, min = 0, max = 1),
  y = b0 + b1*x + b2*x^2 + rnorm(sample_size, sd = sd)
)
```

Leave-one out crossvalidation

ground truth

$$y_i = 1 + 2 \cdot x_i + 3 \cdot x_i^2 + e$$

$$e \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = 0.5)$$



candidate models

simple $\hat{y}_i = b_0 + b_1 \cdot x_i$

correct $\hat{y}_i = b_0 + b_1 \cdot x_i + b_2 \cdot x_i^2$

complex $\hat{y}_i = b_0 + b_1 \cdot x_i + b_2 \cdot x_i^2 + b_3 \cdot x_i^3$

we could do an F-test here since the models are nested ...

Leave-one out crossvalidation

```
1  library("modelr")
2
3  df.cross = df.data %>%
4    crossv_loo() %>% # function which generates training and test data sets
5    mutate(model_simple = map(train, ~ lm(y ~ 1 + x, data = .)),
6           model_correct = map(train, ~ lm(y ~ 1 + x + I(x^2), data = .)),
7           model_complex = map(train, ~ lm(y ~ 1 + x + I(x^2) + I(x^3), data = .))) %>%
8    pivot_longer(cols = contains("model"),
9                 names_to = "index",
10                values_to = "model") %>%
11   mutate(rmse = map2_dbl(.x = model, .y = test, .f = ~ rmse(.x, .y)))
```

nice package for basic cross-validation

index	mean_rmse
simple	0.65
correct	0.42
complex	0.41

complex model has the lowest error on the training data

Leave-one out crossvalidation

```
1 library("modelr")
2
3 df.cross = df.data %>%
4   crossv_loo() %>%
```

splits the data set into training and test

	train	test	.id
1	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	1
2	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	2
3	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	3
4	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	4
5	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	5
6	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	6
7	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	7
8	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	8
9	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	9
10	list(data = list(participant = 1:10, x = c(0.265508663…	list(data = list(participant = 1:10, x = c(0.265508663…	10

each entry is simply a pointer to the data set plus some indices `.idx` for the filtered data points

Leave-one out crossvalidation

```
1 library("modelr")
2
3 df.cross = df.data %>%
4   crossv_loo() %>%
5   mutate(model_simple = map(train, ~ lm(y ~ 1 + x, data = .)),
6          model_correct = map(train, ~ lm(y ~ 1 + x + I(x^2), data = .)),
7          model_complex = map(train, ~ lm(y ~ 1 + x + I(x^2) + I(x^3), data = .))) %>%
```

fit three different models to the training data set

Leave-one out crossvalidation

```
library("modelr")

df.cross = df.data %>%
  crossv_loo() %>%
  mutate(model_simple = map(train, ~ lm(y ~ 1 + x, data = .)),
         model_correct = map(train, ~ lm(y ~ 1 + x + I(x^2), data = .)),
         model_complex = map(train, ~ lm(y ~ 1 + x + I(x^2) + I(x^3), data = .))) %>%
  pivot_longer(cols = contains("model"),
               names_to = "index",
               values_to = "model") %>%
  mutate(rmse = map2_dbl(.x = model, .y = test, .f = ~ rmse(.x, .y)))
```

calculate the root mean squared error for each model on the test data set

```
df.cross %>%
  group_by(index) %>%
  summarize(mean_rmse = mean(rmse))
```

index	mean_rmse
simple	0.65
correct	0.48
complex	0.70

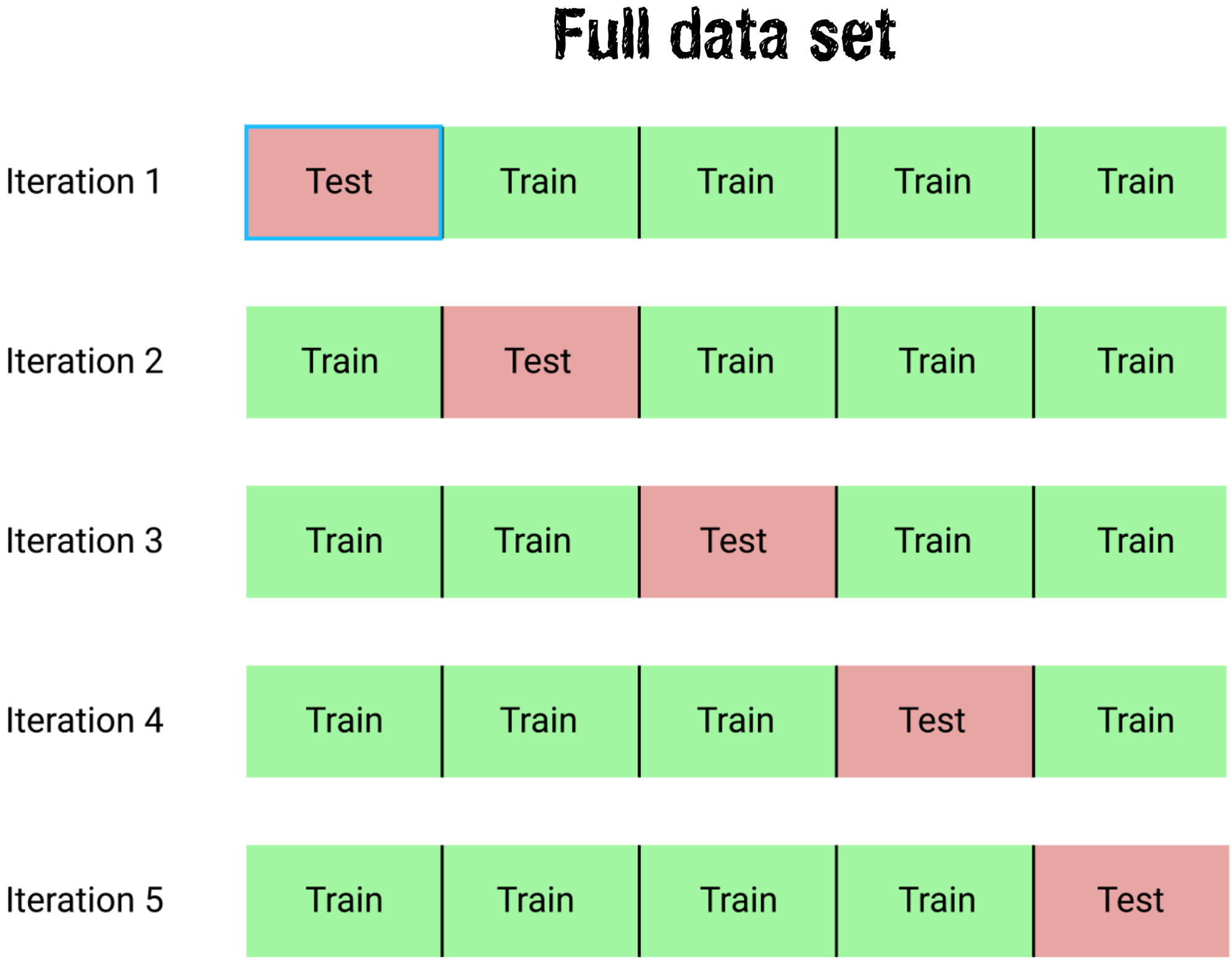
the correct model has the lowest prediction error

Leave-one out crossvalidation

Any potential problems with LOO?

k-fold cross validation

k-fold crossvalidation



k-fold crossvalidation

k-fold crossvalidation

```
df.cross = df.data %>%
  crossv_kfold(k = 10) %>%
  mutate(model_simple = map(train, ~ lm(y ~ 1 + x, data = .)),
         model_correct = map(train, ~ lm(y ~ 1 + x + I(x^2), data = .)),
         model_complex = map(train, ~ lm(y ~ 1 + x + I(x^2) + I(x^3), data = .))) %>%
  pivot_longer(cols = contains("model"),
               names_to = "index",
               values_to = "model") %>%
  mutate(rsquare = map2_dbl(model, test, rsquare))
```

using R^2 as a measure

why didn't we use R^2 for LOO?

this wouldn't work for LOO since we only have one data point in the test data...

index	median_rsquare
simple	0.839
correct	0.865
complex	0.860

the correct model accounts for the most variance in the test data

k-fold vs. leave-one-out crossvalidation

- **LOO**:
  - trained on **more** data
    - more variance
    - less bias
- **k-fold**:
  - trained on **less** data
    - less variance
    - more bias

Monte Carlo crossvalidation

Monte Carlo crossvalidation

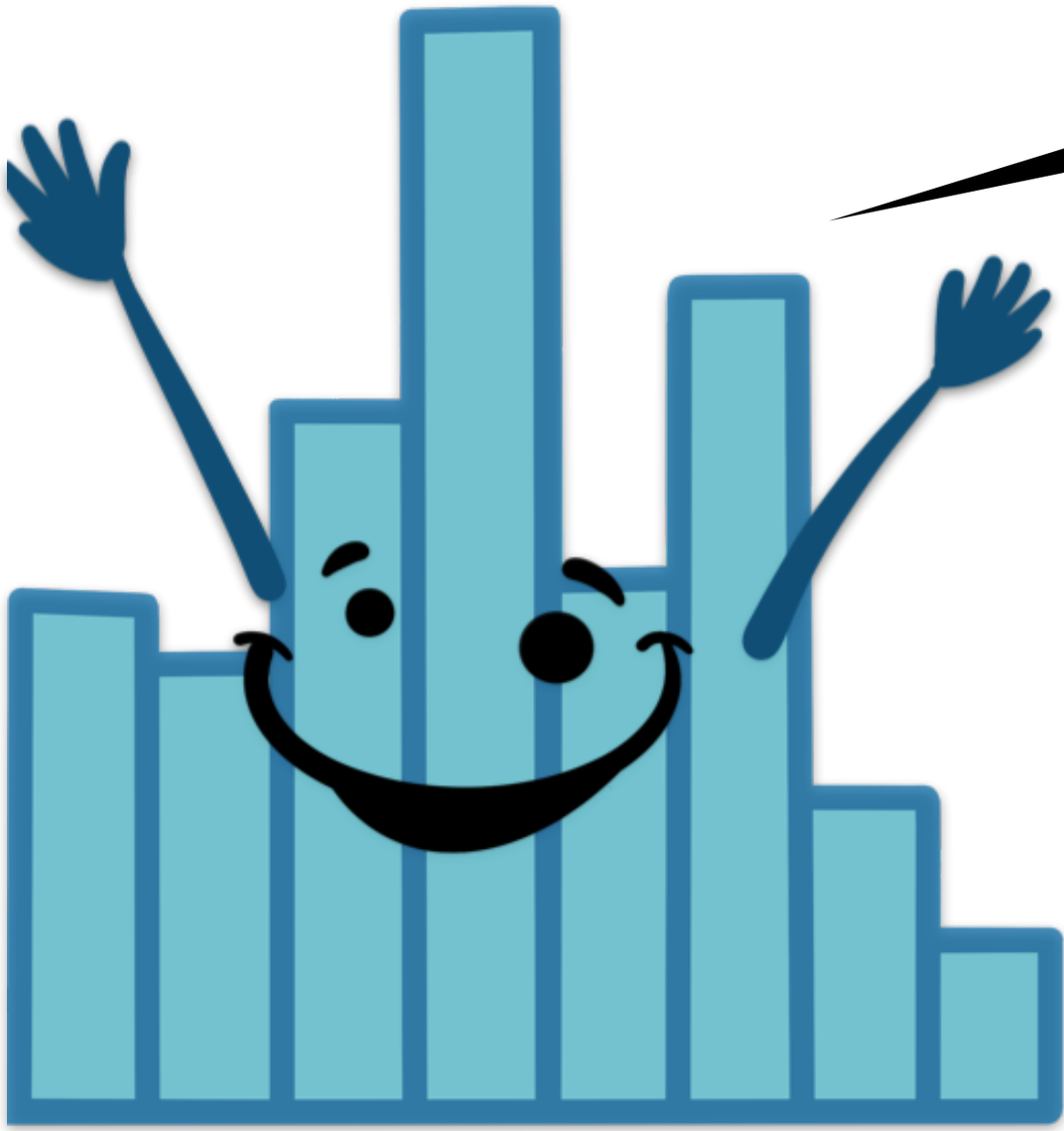
random splits into training and test data

```
crossv_mc(n = 50, test = 0.5)
```

number of training-test splits

proportion of test data in each split

- splits can also be done in a **stratified** way (check out the `tidymodels` package)
- for example, generate training data that has the same percentage of cases from each group
- fit data from some participants, test on data from other participants, ...



01:00

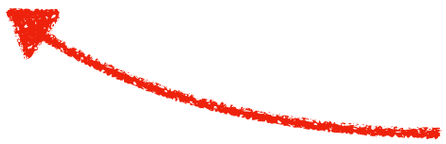
stretch break!

time

AIC and BIC

AIC and BIC

- AIC = Akaike Information Criterion
- BIC = Bayesian Information Criterion



not that much Bayesian about it ...

$$\mathrm{AIC} = 2k - 2\ln(L)$$

$$\mathrm{BIC} = \ln(n)k - 2\ln(\hat{L})$$

$\hat{L}$ = maximized value of the likelihood function of the model

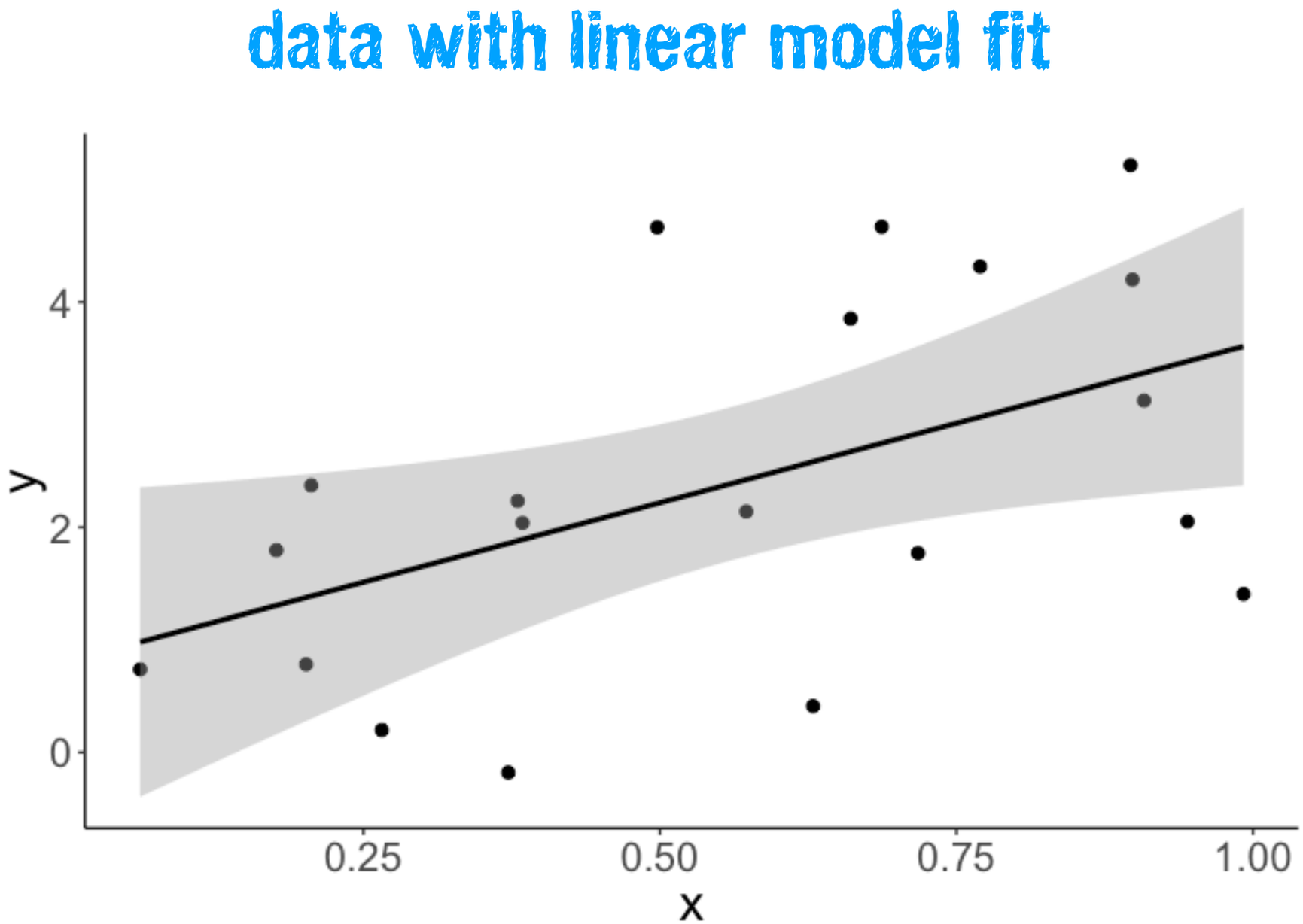
k = number of parameters in the model

n = number of observations

AIC and BIC

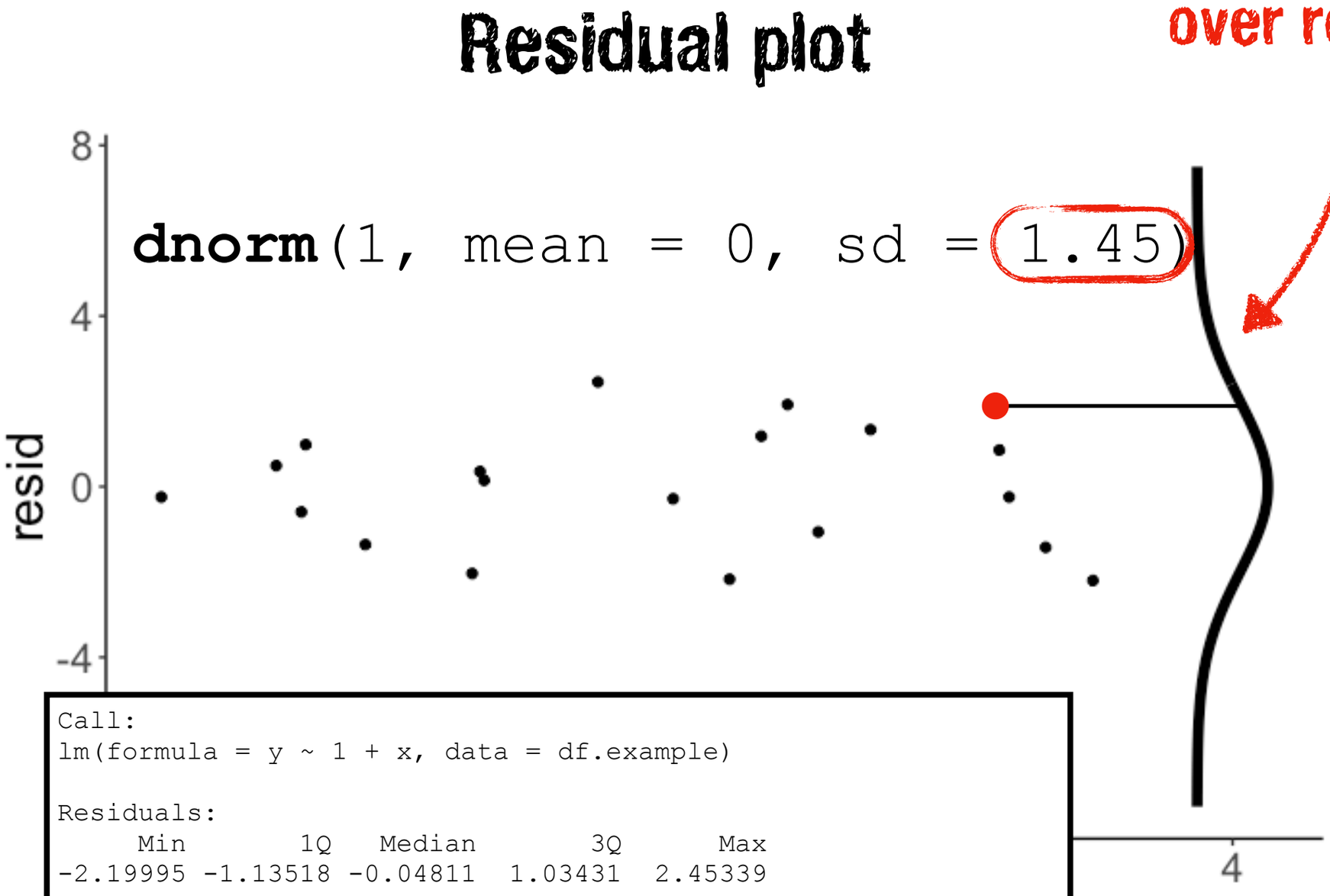
- How do we get the likelihood of our model?
  - in a linear regression, minimizing least squares is equivalent to maximizing the likelihood of the data given the model
- Assumptions of the linear model:
  - residuals are normally distributed with:
    - `mean = 0` and `sd = sigma`
  - calculate overall likelihood by computing the likelihood of each residual, and then multiplying

AIC and BIC



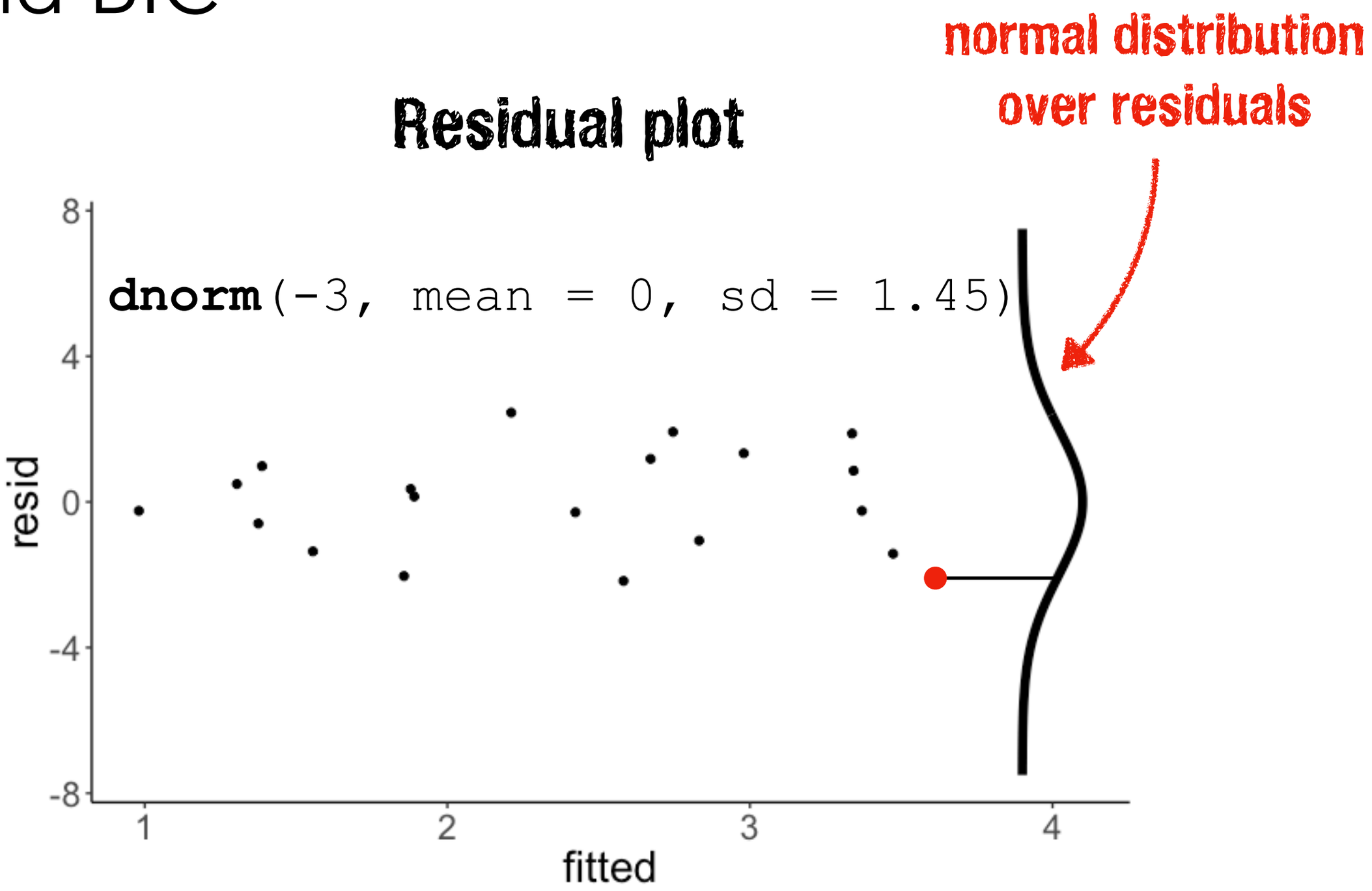
AIC and BIC

normal distribution over residuals




```
Call:
lm(formula = y ~ 1 + x, data = df.example)

Residuals:
     Min       1Q   Median       3Q      Max
-2.19995 -1.13518 -0.04811  1.03431  2.45339

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8062     0.7157   1.126   0.2748
x             2.8232     1.1374   2.482   0.0232 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.449 on 18 degrees of freedom
Multiple R-squared:  0.255,Adjusted R-squared:  0.2136
F-statistic: 6.161 on 1 and 18 DF,  p-value: 0.02315
```

AIC and BIC



since the data points are independent, we can calculate the overall likelihood by multiplying the likelihood of each observation

AIC and BIC

```
dnorm(1.88, mean = 0, sd = 1.45) = 0.12
```

```
# generate some data
df.like = tibble(
  x = runif(20, min = 0, max = 1),
  y = 1 + 3 * x + rnorm(20, sd = 2)
)

# fit the model
fit = lm(formula = y ~ x,
         data = df.like)

# model summary
fit %>%
  glance()
```

x	y	fitted	resid	likelihood
0.90	5.22	3.34	1.88	0.12
0.27	0.20	1.56	-1.36	0.18
0.37	-0.18	1.86	-2.04	0.10
0.57	2.14	2.42	-0.28	0.27
0.91	3.13	3.37	-0.24	0.27
0.20	0.78	1.38	-0.59	0.25
0.90	4.20	3.34	0.86	0.23
0.94	2.05	3.47	-1.42	0.17
0.66	3.85	2.67	1.18	0.20
0.63	0.41	2.58	-2.17	0.09

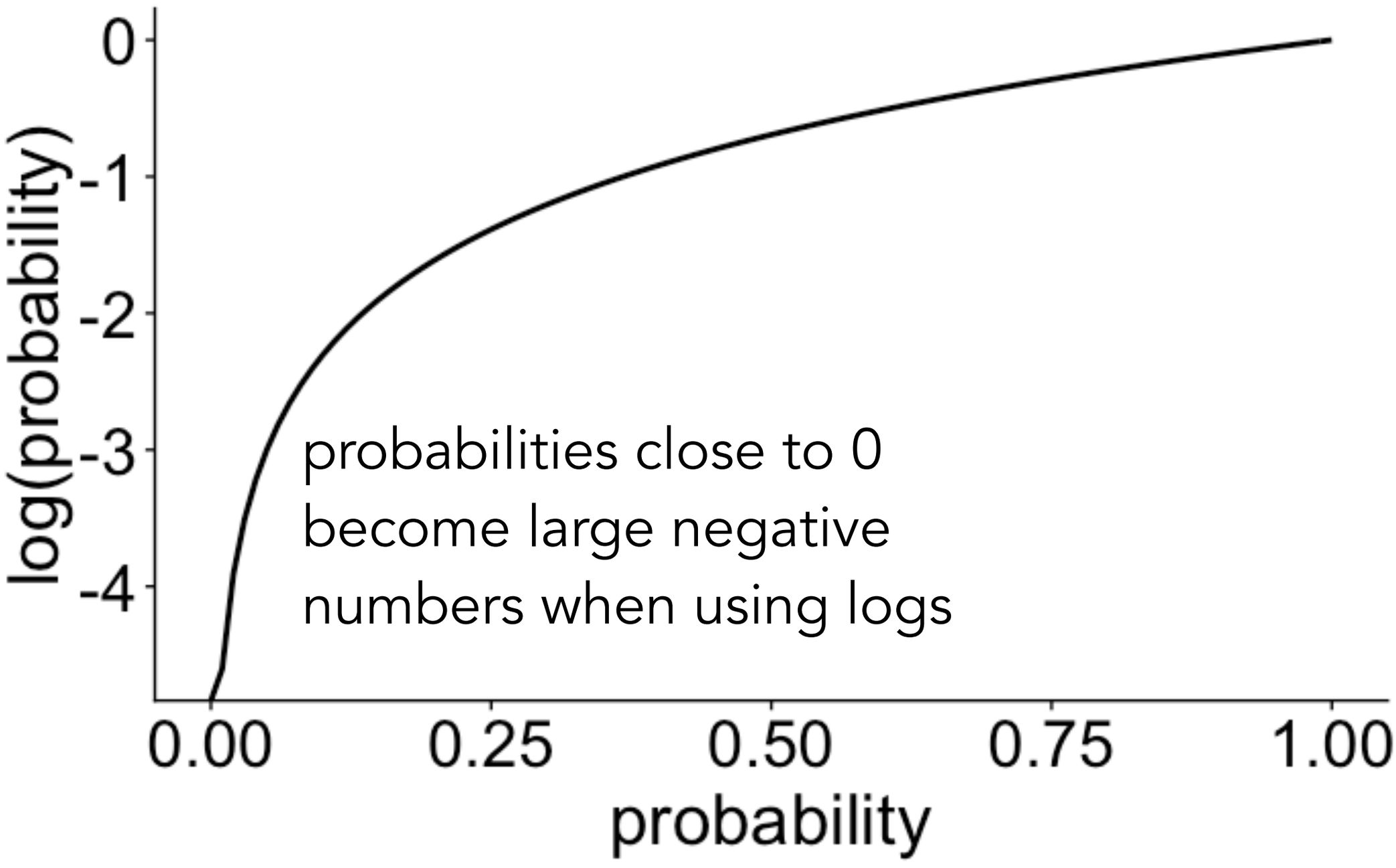
$$\sum_{i=1}^{n} \ln(\text{likelihood})$$

inferred standard deviation of the error

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.25	0.21	1.45	6.16	0.02	2	-34.74	75.47	78.46	37.77	18

$$e \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = 1.45)$$

`log()` is your friend!



Clue guide to probability

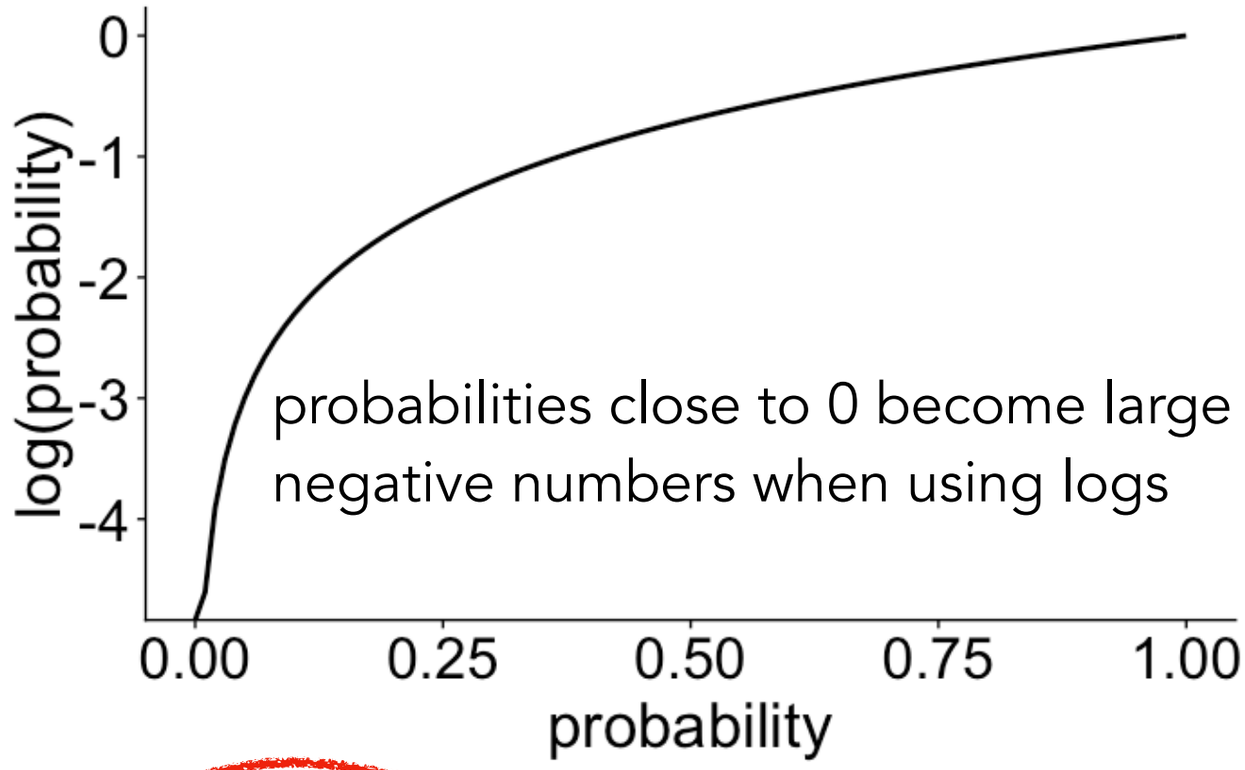
- *joint probability*:
- if A and B are independent then
- **Definition**: $p(A, B) = p(A) \cdot p(B)$

- $p(\text{Prof Plum, candle stick}) =$
  $p(\text{Prof Plum}) \cdot p(\text{candle stick}) =$
  $\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$

Who?

What?

`log()` is your friend!



multiplying probabilities

$$0.01 \cdot 0.01 \cdot 0.01 \cdot 0.01 = 0.00000001$$

number becomes extremely small quickly

take log()

```
log(0.01)= -4.60517
```

number becomes large but that's ok

summing logs

$$(-4.60517) + (-4.60517) + (-4.60517) + (-4.60517) = -18.42068$$

transform back into probability

```
exp(-18.42068)= 0.00000001
```

often not necessary since we just use logLikelihood

AIC and BIC

dnorm(1.88, mean = 0, sd = 1.45) = 0.12

```
# generate some data
df.like = tibble(
   x = runif(20, min = 0, max = 1),
   y = 1 + 3 * x + rnorm(20, sd = 2)
)

# fit the model
fit = lm(formula = y ~ x,
         data = df.like)

# model summary
fit %>%
   glance()
```

x	y	fitted	resid	likelihood
0.90	5.22	3.34	1.88	0.12
0.27	0.20	1.56	-1.36	0.18
0.37	-0.18	1.86	-2.04	0.10
0.57	2.14	2.42	-0.28	0.27
0.91	3.13	3.37	-0.24	0.27
0.20	0.78	1.38	-0.59	0.25
0.90	4.20	3.34	0.86	0.23
0.94	2.05	3.47	-1.42	0.17
0.66	3.85	2.67	1.18	0.20
0.63	0.41	2.58	-2.17	0.09

inferred standard deviation of the error

$$\sum_{i=1}^{n} \ln(\text{likelihood})$$

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.25	0.21	1.45	6.16	0.02	2	-34.74	75.47	78.46	37.77	18

$$e \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = 1.45)$$

AIC and BIC

$$\text{AIC} = 2k - 2\ln(\hat{L})$$

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L})$$

$\ln(\hat{L})$ = maximized value of the likelihood function of the model **-34.74**

k = number of parameters in the model **3**

n = number of observations **20**

the `sd` of the normal distribution modeling the residuals counts as a parameter

```
lm(formula = y ~ 1 + x, data = df.example)
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.25	0.21	1.45	6.16	0.02	2	-34.74	75.47	78.46	37.77	18

AIC and BIC

$$\text{AIC} = 2k - 2\ln(\hat{L}) = 2 \cdot 3 - 2 \cdot (-34.74) = 75.47$$

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L}) = \ln(20) \cdot 3 - 2 \cdot (-34.74) = 78.46$$

$\ln(\hat{L})$ = maximized value of the likelihood function of the model **-34.74**

k = number of parameters in the model **3**

n = number of observations **20**

the `sd` of the normal distribution modeling the residuals counts as a parameter

```
lm(formula = y ~ 1 + x, data = df.example)
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.25	0.21	1.45	6.16	0.02	2	-34.74	75.47	78.46	37.77	18

AIC and BIC

$$\text{AIC} = 2k - 2\ln(\hat{L})$$

$$\text{BIC} = \ln(n)k - 2\ln(\hat{L})$$

ΔBIC	Evidence against higher BIC
0 to 2	Not worth more than a bare mention
2 to 6	Positive
6 to 10	Strong
>10	Very Strong

- for both AIC and BIC, *lower* is better!
- neither provide a test of a model in the sense of testing a null hypothesis
  - AIC or BIC tell us nothing about the absolute quality of a model, only the quality relative to other models
- BIC generally penalizes free parameters more strongly than AIC (though it depends on the size of *n*)

What shall I use when?

- Use it all!
- ideally, the different measures provide converging evidence

Table 2
Summary of the model results. Values for *r* and RMSE indicate means (with 5% and 95% quantiles in parentheses) based on 100 split-half cross-validation runs. BIC scores are based on running the models on the full data set.

Model	*r*	RMSE	BIC
Difference & pivotality	.86 (.66, .95)	10.56 (6.17, 17.21)	158.59
Difference	.70 (.30, .90)	26.92 (16.4, 40.6)	209.74
Pivotality	.63 (.41, .77)	14.23 (11.39, 17.54)	199.53
Optimality	.66 (.42, .84)	14.55 (10.54, 17.91)	199.47

Note: BIC = Bayesian Information Criterion (lower values indicate better model performance).

What shall I use when?

- Use it all!
- ideally, the different measures provide converging evidence

Table 2
Summary of the model results. Values for *r* and RMSE indicate means (with 5% and 95% quantiles in parentheses) based on 100 split-half cross-validation runs. BIC scores are based on running the models on the full data set.

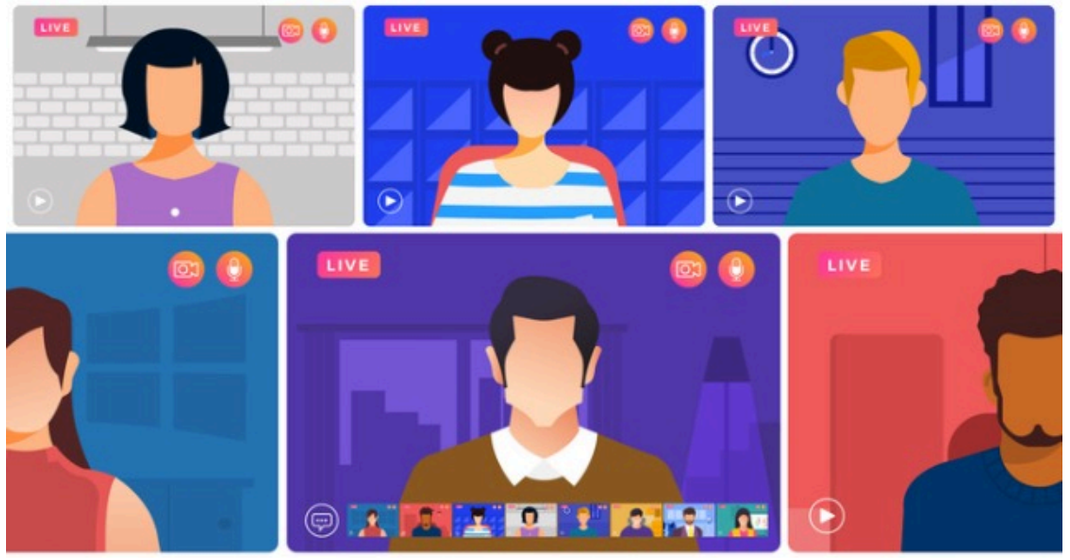
Model	*r*	RMSE	BIC
Difference & pivotality	.86 (.66, .95)	10.56 (6.17, 17.21)	158.59
Difference	.70 (.30, .90)	26.92 (16.4, 40.6)	209.74
Pivotality	.63 (.41, .77)	14.23 (11.39, 17.54)	199.53
Optimality	.66 (.42, .84)	14.55 (10.54, 17.91)	199.47

Note: BIC = Bayesian Information Criterion (lower values indicate better model performance).

Plan for today

- Model comparison
  - Cross-validation
  - AIC and BIC
- Linear mixed effects models

Breakout rooms



Tasks:

1. **What are linear mixed effects models?**
2. **When should we use them?**

Size: ~3 people

Time: 4 minutes

Report: Share your group's thinking in the main room.

Linear mixed effects models

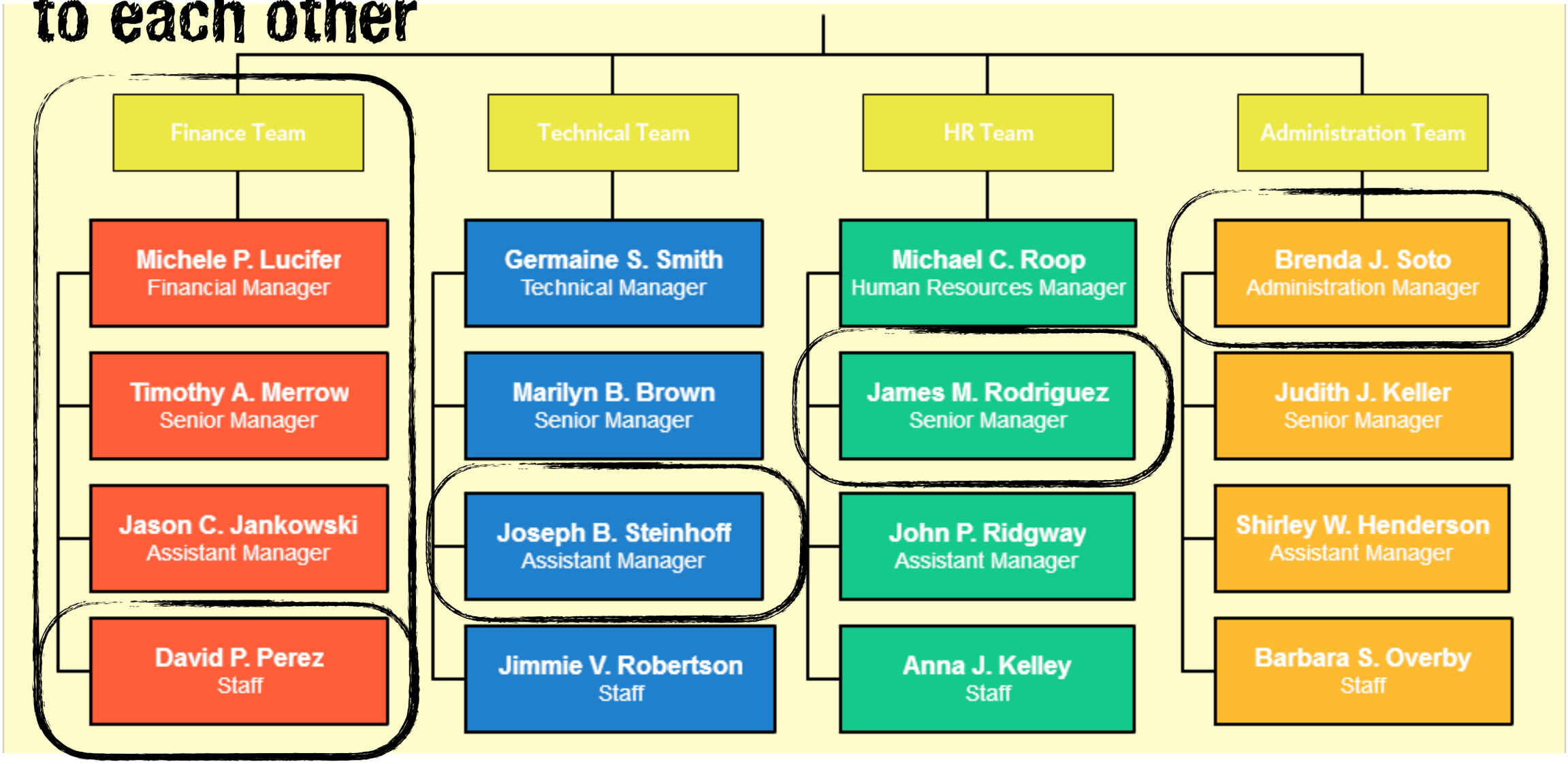
Dependence

- so far, all the models that we've discussed (linear model with different kinds of predictors and contrasts) make the assumption that the residuals are **iid** (independent, and identically distributed)

- often this assumption is violated
  - **psychology experiments**: many observations from the same participants
  - **survey data**: different populations between different states in the US
  - **time series**: distribution at $t + 1$ depends on t

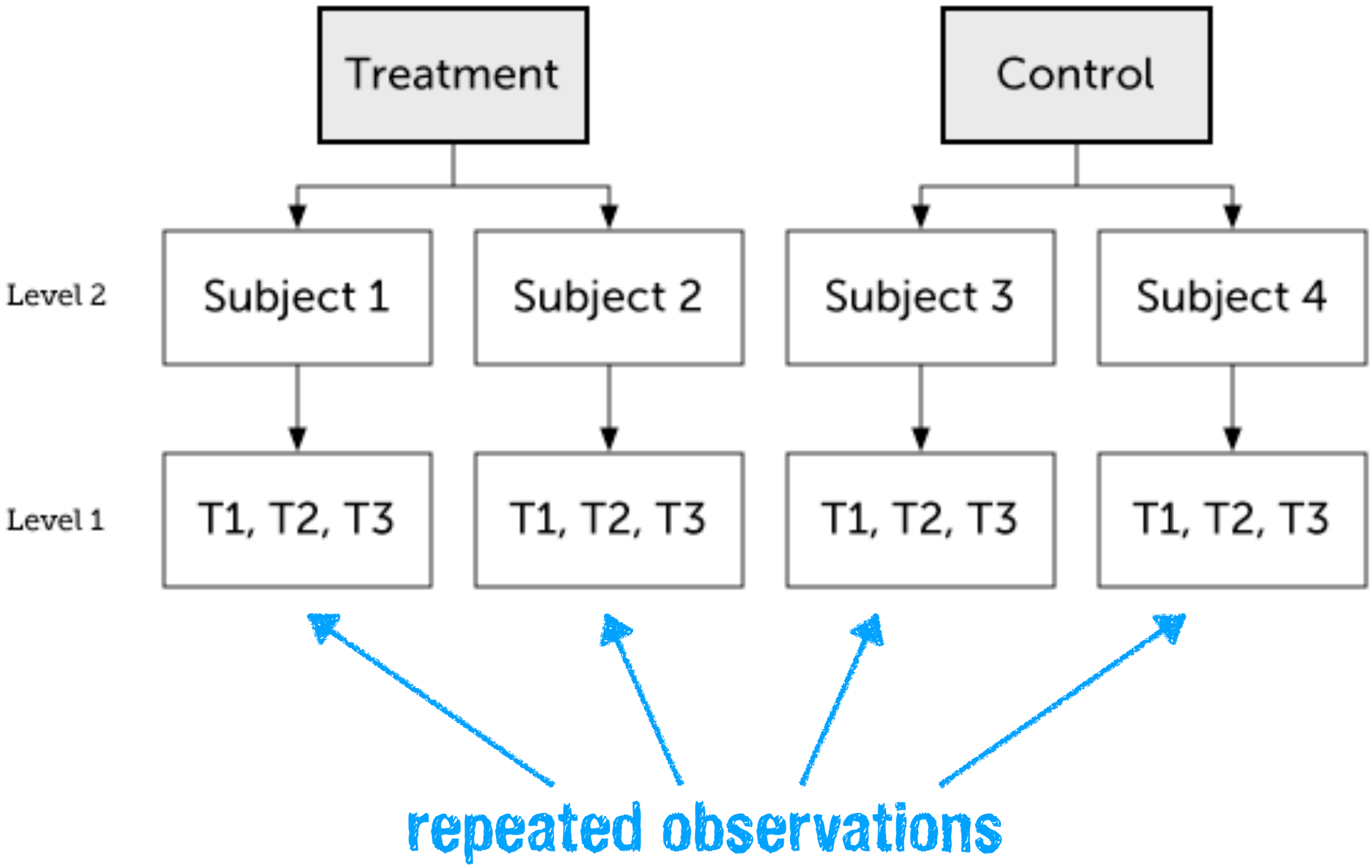
Main use cases: **Hierarchical models**

more similar to each other



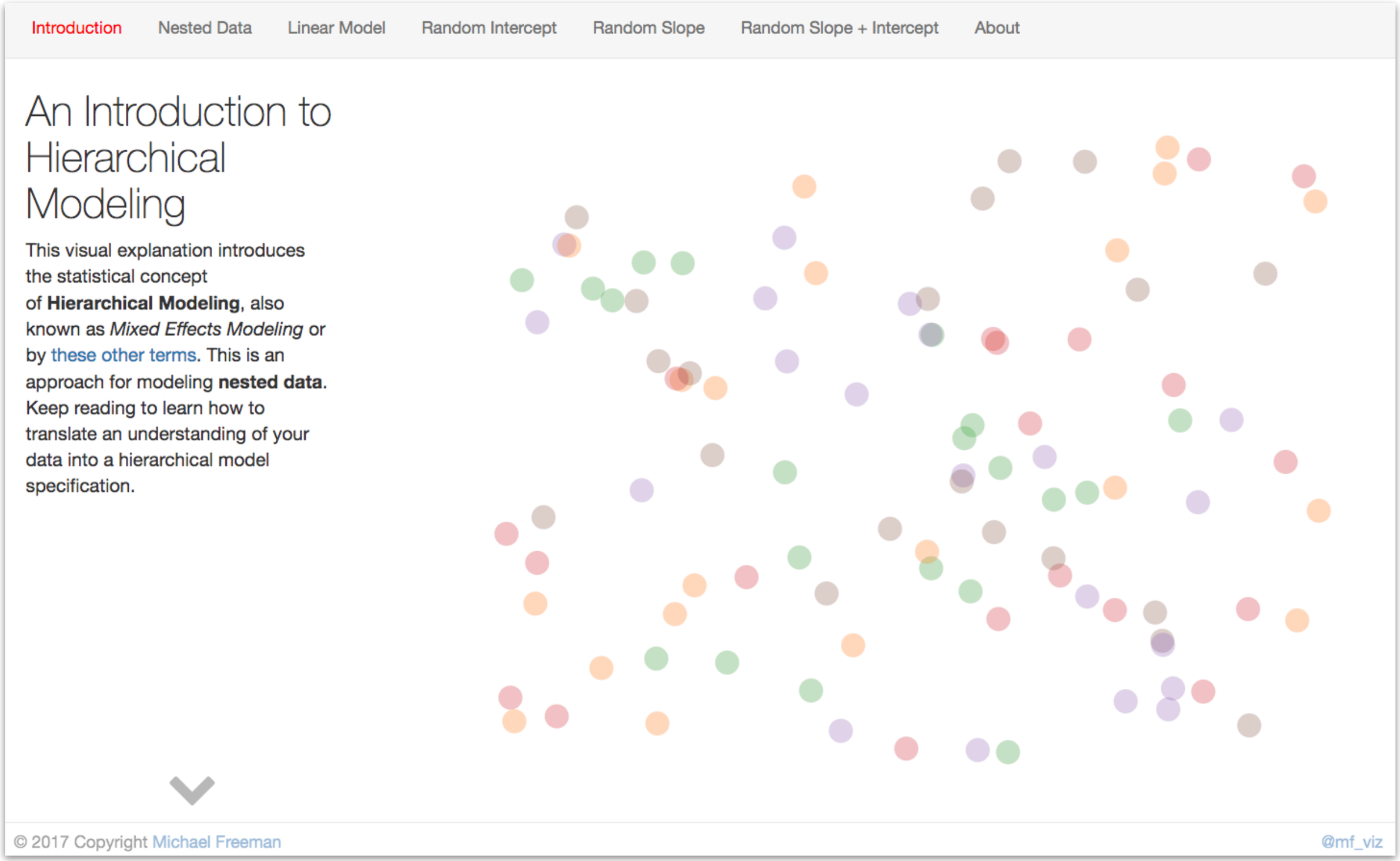
less similar to each other

Main use cases: **Longitudinal models**



general points about mixed effects models

- **fixed effects**:
  - often: factors that we manipulate experimentally
  - parameters are estimated --> we are interested in characterizing the relationship between this variable and the outcome
- **random effects**:
  - variation we want to control for
  - often: differences between participants (or items) in our experiment
  - sampling viewpoint: we explicitly model the variation in participants (or items)

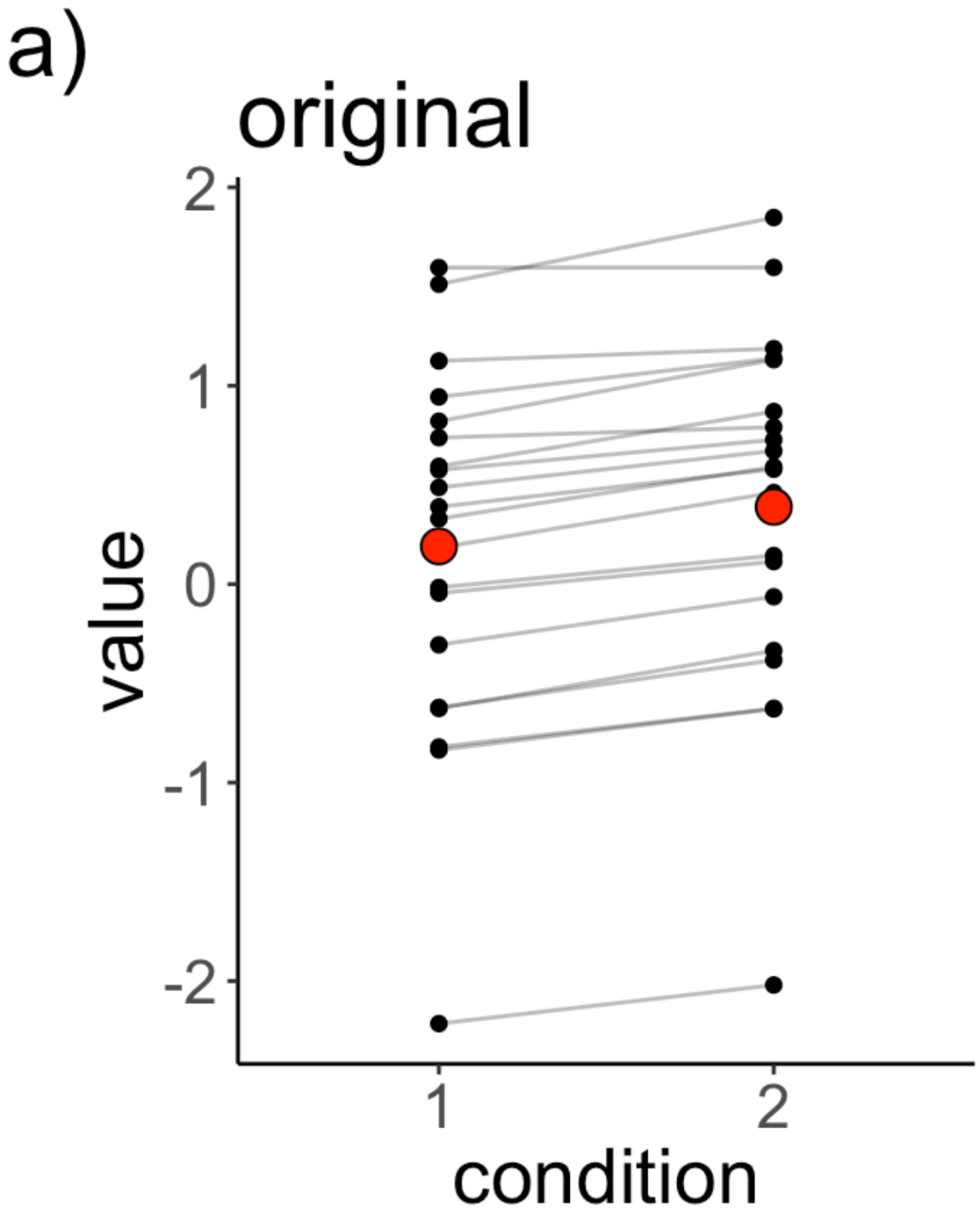


http://mfviz.com/hierarchical-models/

Dependence

Does it really matter?

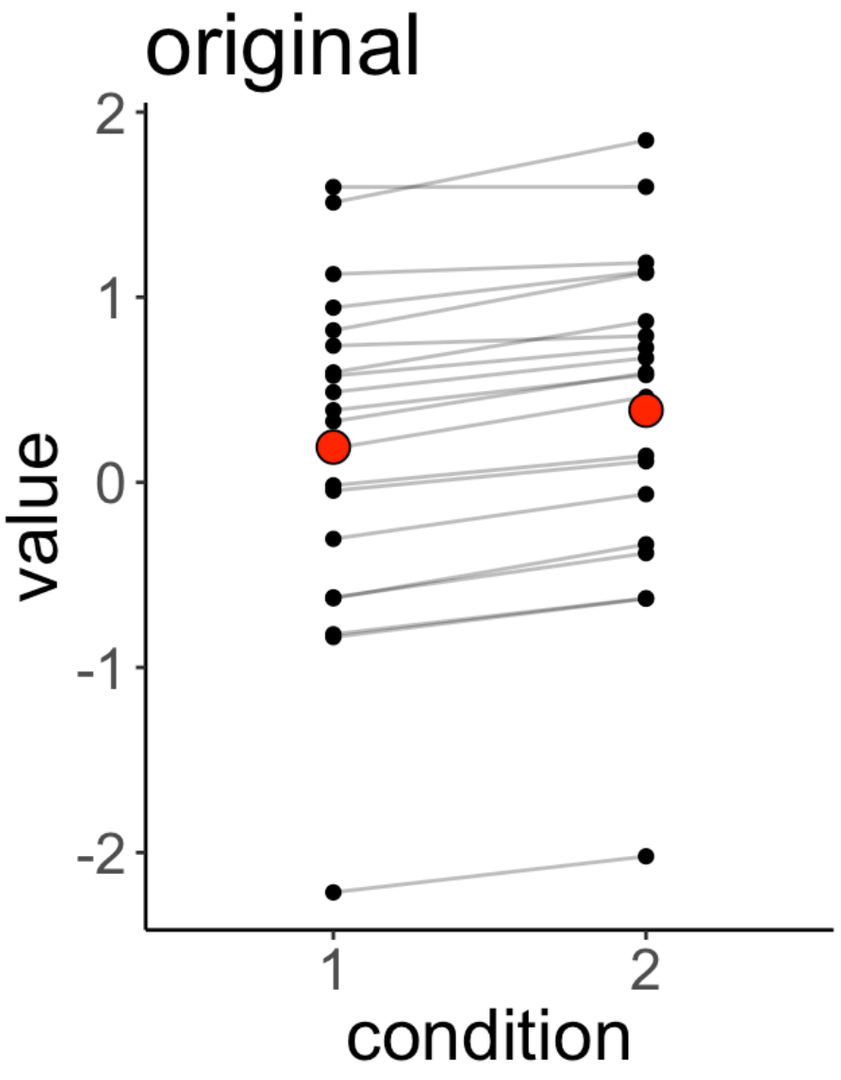
Is there a significant difference between conditions 1 and 2?



Dependence

assuming independence!

```
1 # linear model
2 lm(formula = value ~ condition,
3     data = df.original) %>%
4   summary()
```




```
Call:
lm(formula = value ~ condition, data = df.original)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4100 -0.5530  0.1945  0.5685  1.4578

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.1905     0.2025   0.941    0.353
condition2    0.1994     0.2864   0.696    0.491

Residual standard error: 0.9058 on 38 degrees of freedom
Multiple R-squared:  0.01259,    Adjusted R-squared:  -0.0134
F-statistic: 0.4843 on 1 and 38 DF,  p-value: 0.4907
```

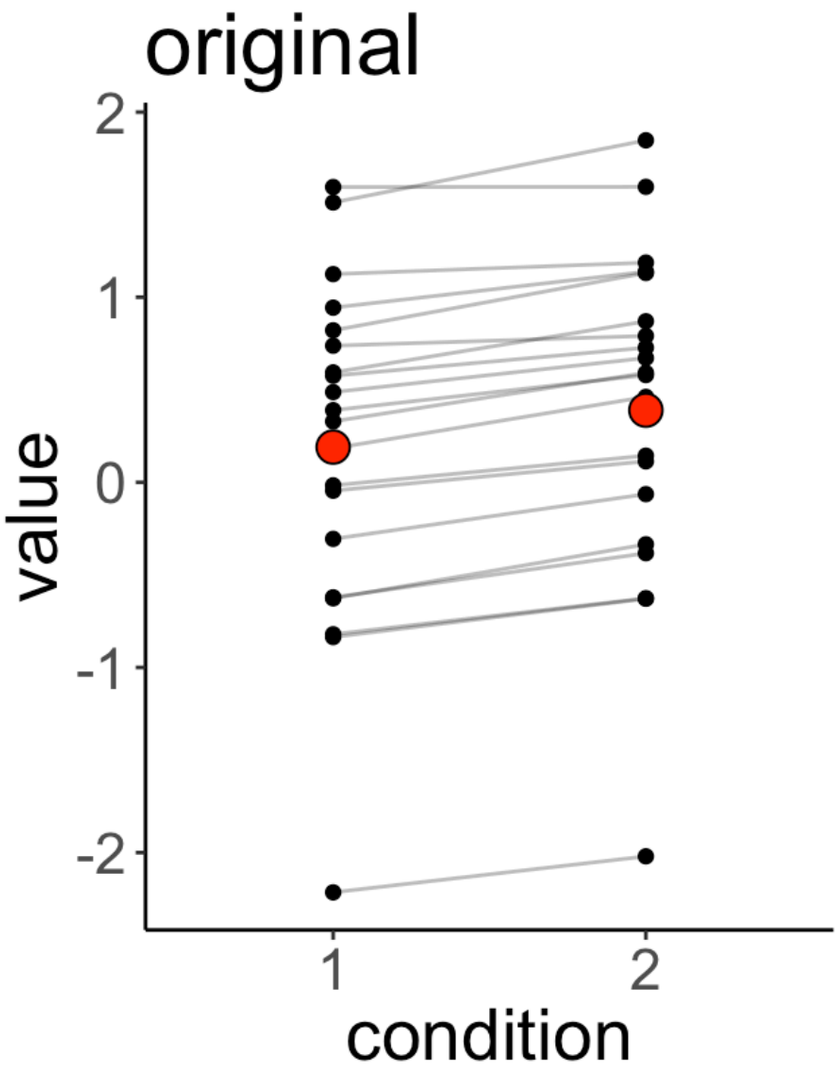
- we ignore the fact that we have repeated observations from the same participants
- in the data it looks like there is a small but consistent effect of condition

meet **`lmer()`**

Dependence

new syntax

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```




```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups       Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                 0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
            (Intr)
condition2 -0.048
```

no p-value!

NO P-VALUE

Dependence

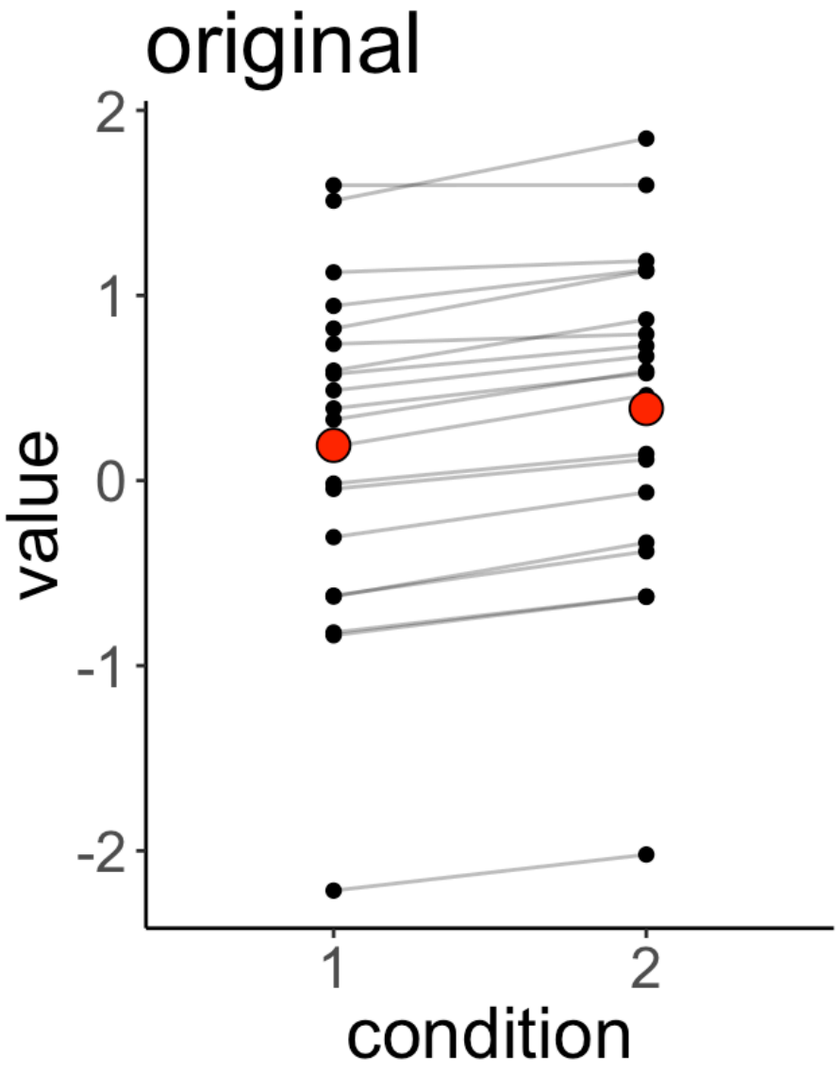
we can still do our good ol model comparison trick

```
1 # fit models
2 fit.compact = lmer(formula = value ~ 1 + (1 | participant),
3                    data = df.original)
4 fit.augmented = lmer(formula = value ~ 1 + condition + (1 | participant),
5                    data = df.original)
6
7 # compare via Chisq-test
8 anova(fit.compact, fit.augmented)
```

```
refitting model(s) with ML (instead of REML)
Data: df.original
Models:
fit.compact: value ~ 1 + (1 | participant)
fit.augmented: value ~ 1 + condition + (1 | participant)
                Df    AIC    BIC   logLik deviance  Chisq Chi Df Pr(>Chisq)
fit.compact      3 53.315 58.382 -23.6575   47.315
fit.augmented    4 17.849 24.605  -4.9247    9.849 37.466      1  9.304e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Dependence

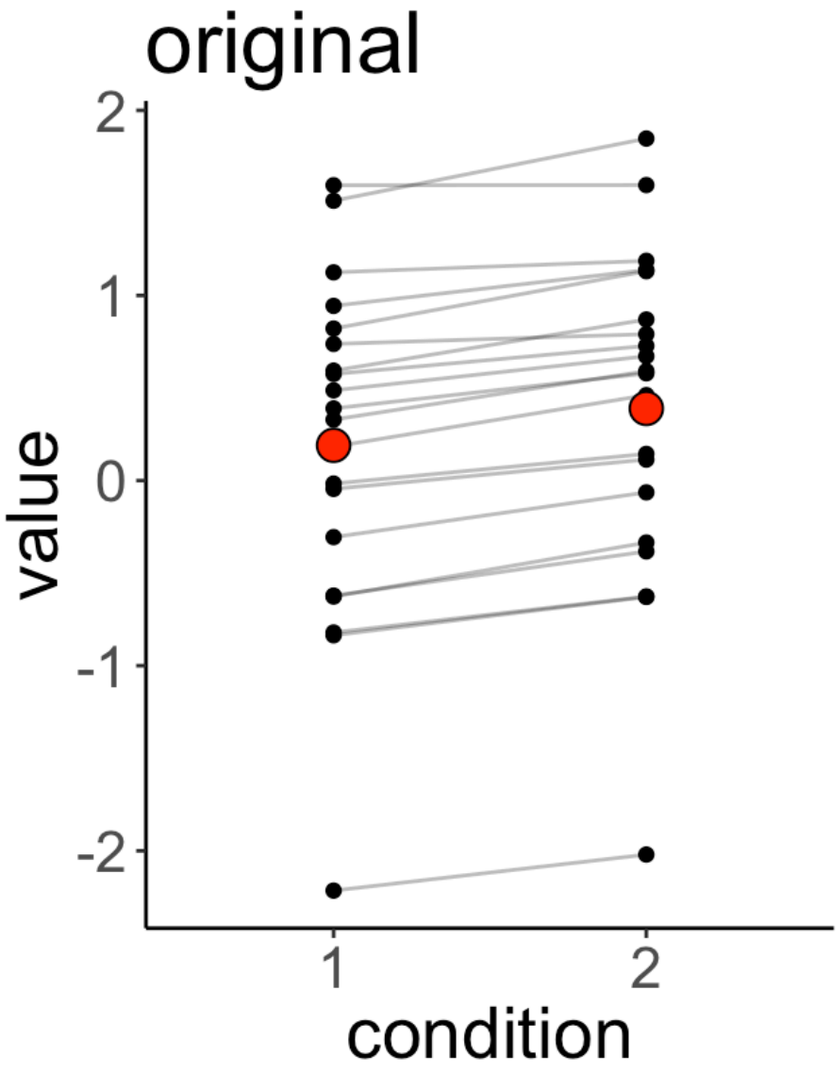
Why is the effect of condition significant when we account for the dependence in the data?



- there are large interindividual differences
- the variance explained by the effect of condition is (much) smaller than the interindividual variance
- **but**: the effect of condition is highly consistent

Dependence

Why is the effect of condition significant when we account for the dependence in the data?



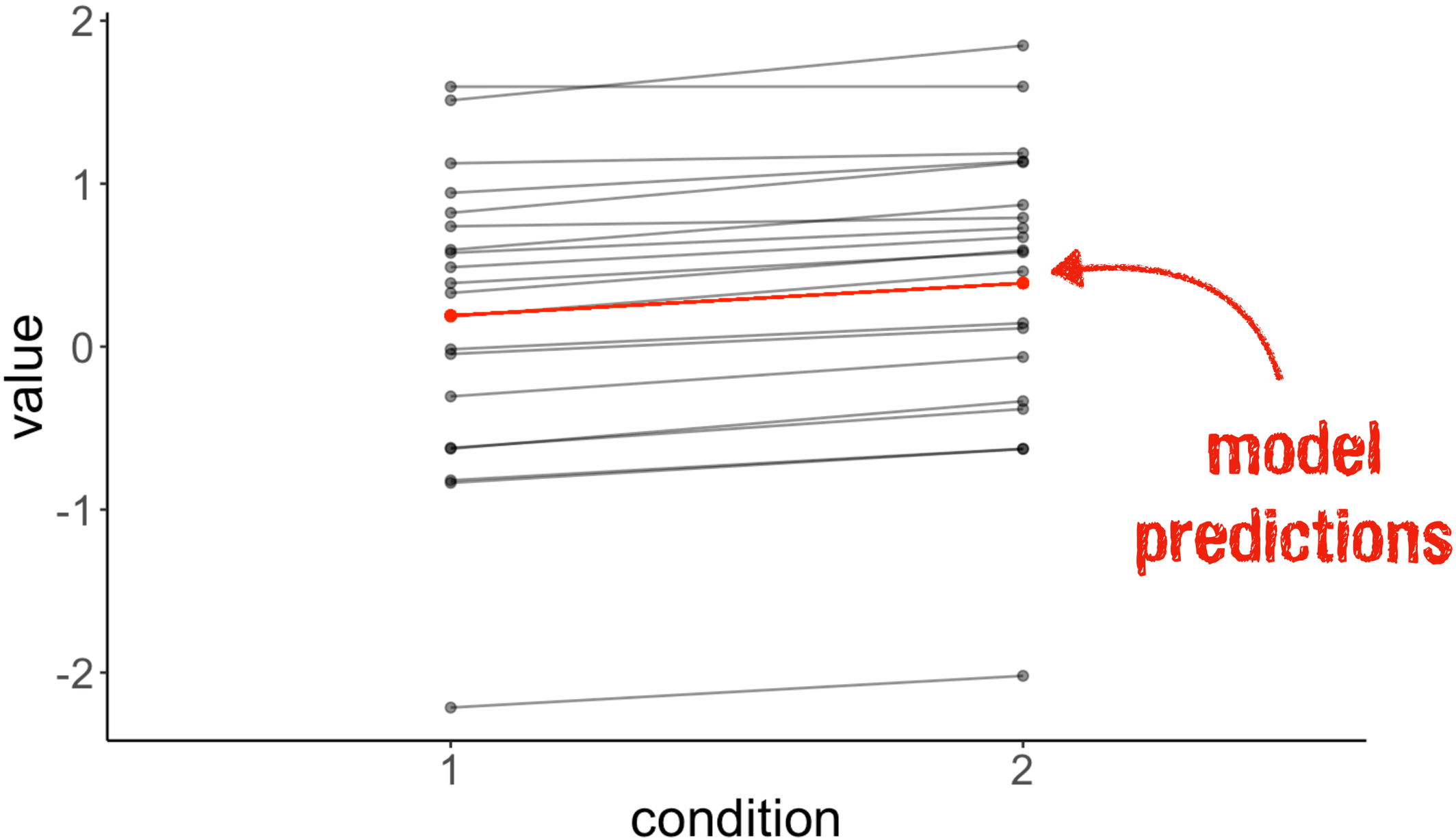
- by explicitly modeling the dependence in the data, we account for the interindividual differences

let's visualize the model predictions!

Linear model (assuming independence)

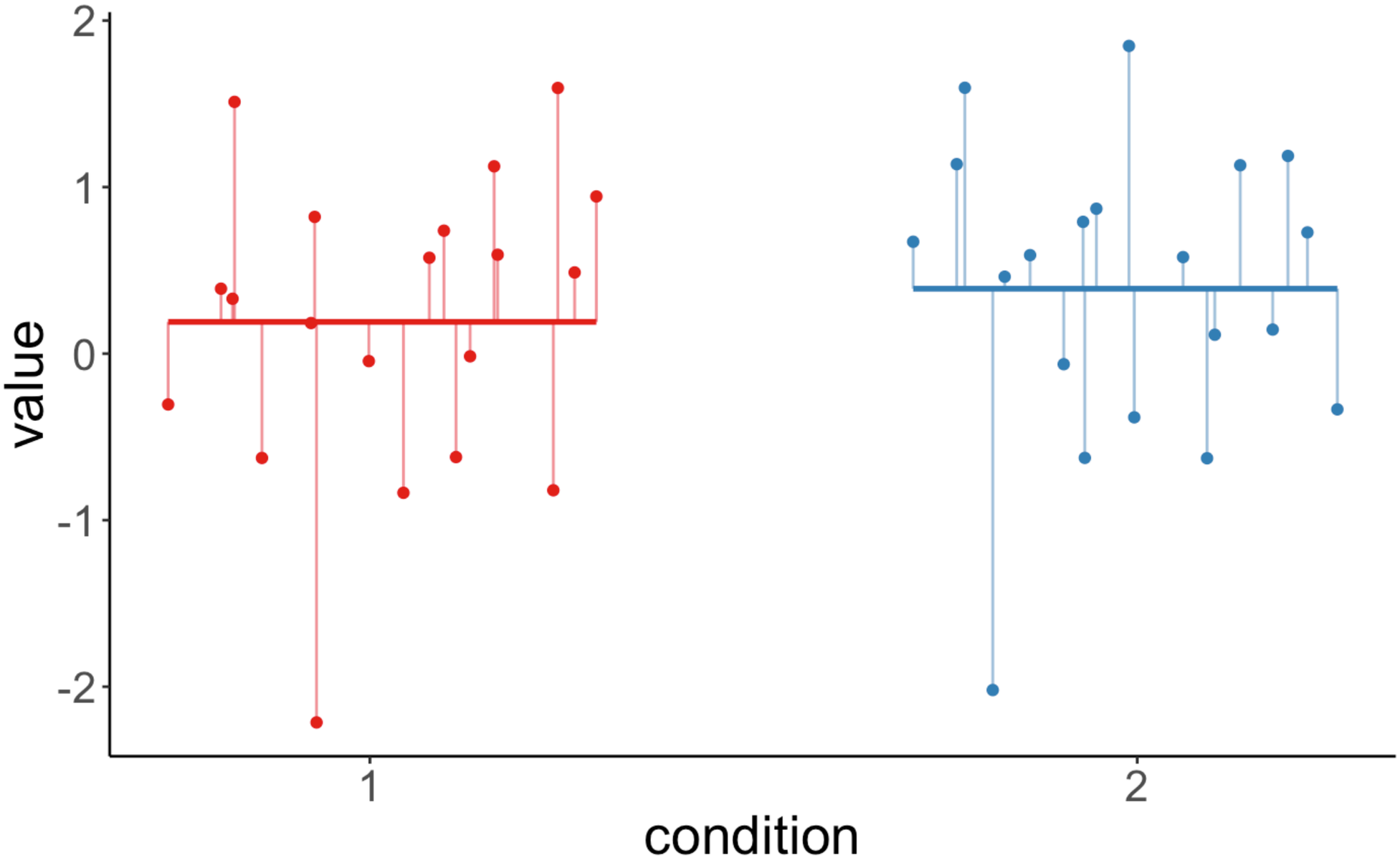
Predictions by the linear model which assumes independence

```
lm(formula = value ~ 1 + condition,
   data = df.original)
```



Linear model (assuming independence)

Residuals of the model

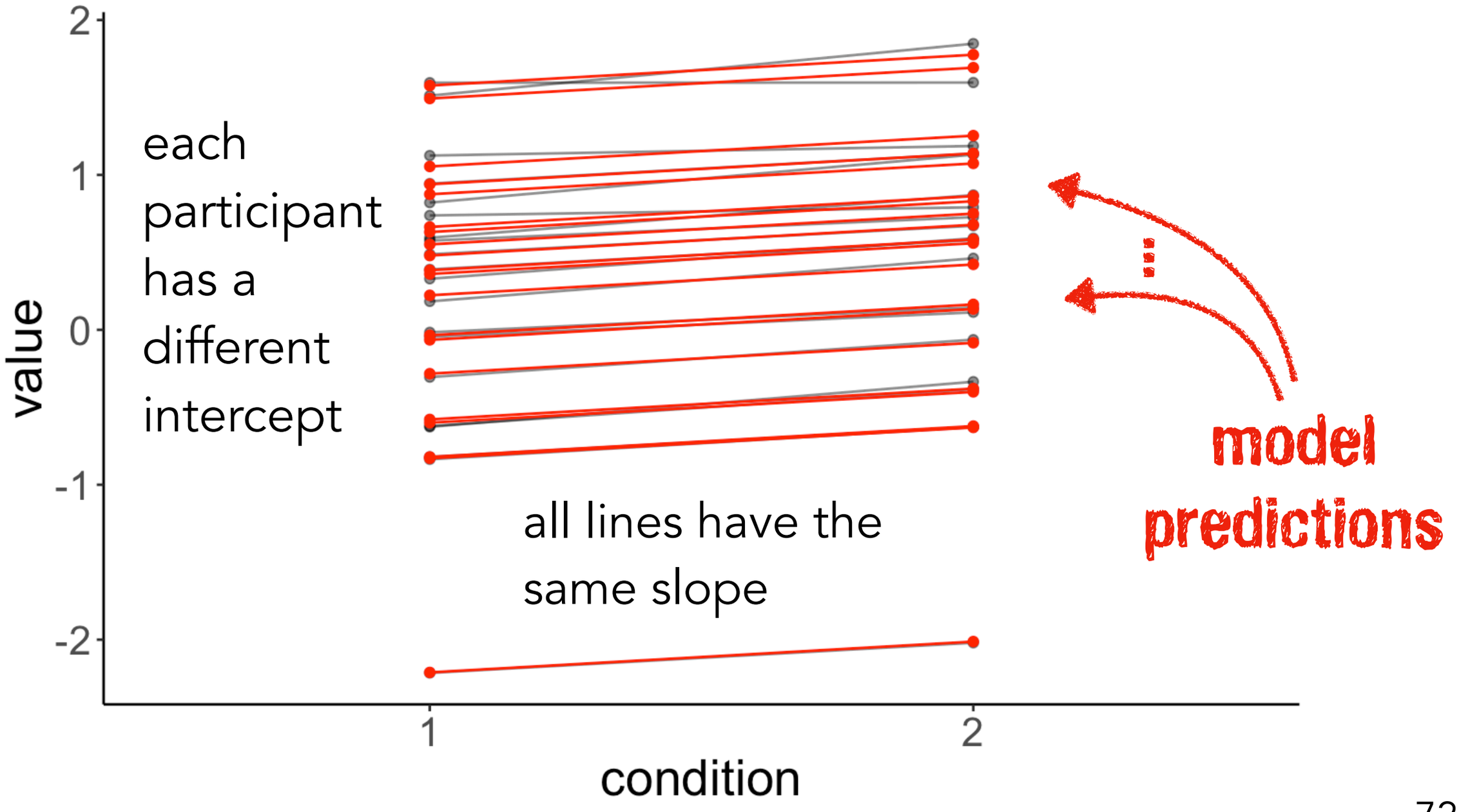


This is not much better than fitting a single line (point).

Linear mixed effects model (accounting for dependence)

Predictions by the linear mixed effects model

```
lmer(formula = value ~ 1 + condition + (1 | participant),
     data = df.original)
```

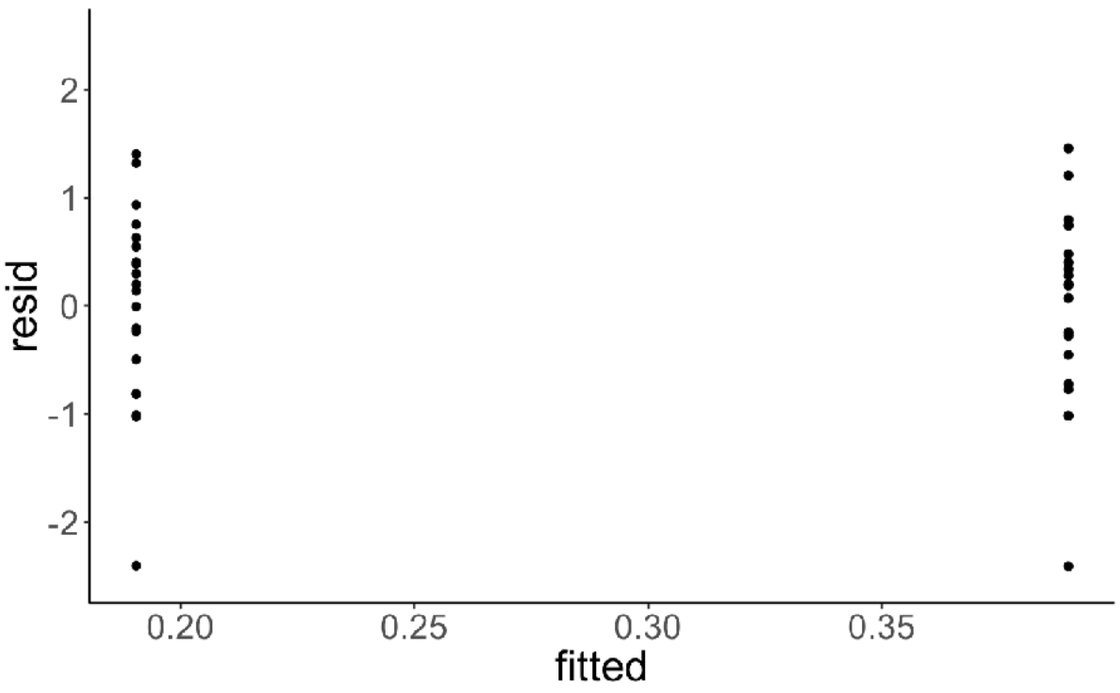


Model comparison

Residual plots

linear model

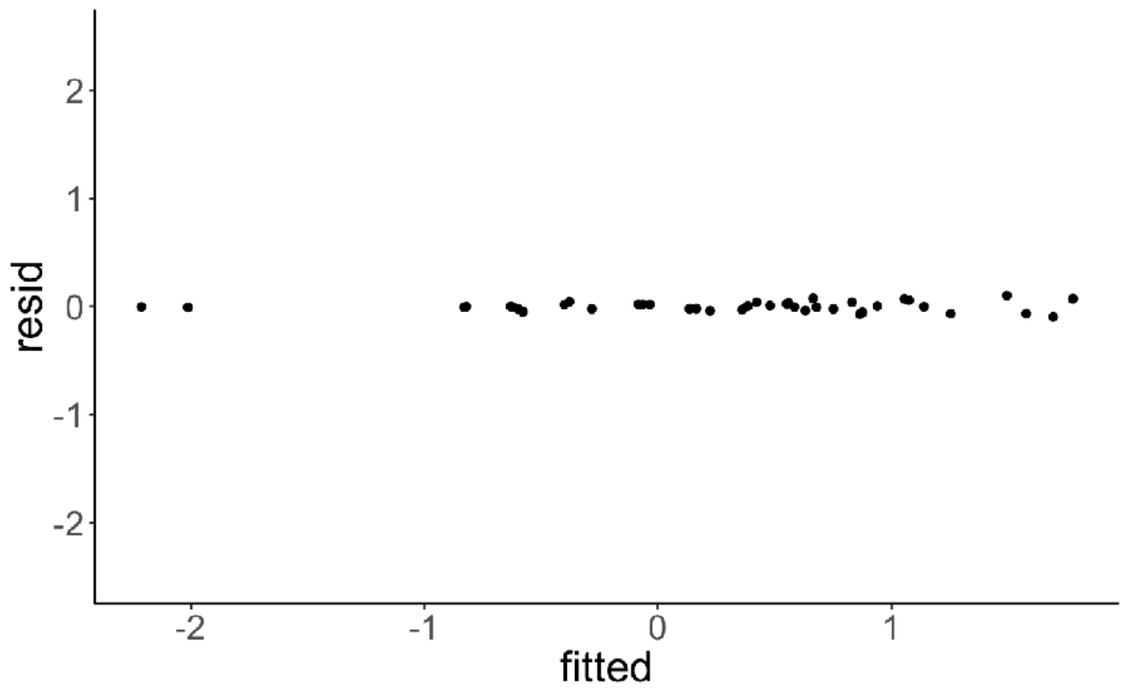
```
lm(formula = value ~ 1 + condition,
   data = df.original)
```



much variance left to be explained

linear mixed effects model

```
lmer(formula = value ~ 1 + condition +
                (1 | participant),
     data = df.original)
```



almost all variance explained

Model comparison

Hypothesis test

Is taking into account individual differences worth it?

```
1  # fit models (without and with dependence)
2  fit.compact = lm(formula = value ~ 1 + condition,
3                   data = df.original)
4
5  fit.augmented = lmer(formula = value ~ 1 + condition + (1 | participant),
6                       data = df.original)
7
8  # compare models
9  # note: the lmer model has to be supplied first
10 anova(fit.augmented, fit.compact)
```

```
refitting model(s) with ML (instead of REML)
Data: df.original
Models:
fit.compact: value ~ 1 + condition
fit.augmented: value ~ 1 + condition + (1 | participant)
              Df     AIC     BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
fit.compact    3 109.551 114.617 -51.775  103.551
fit.augmented  4  17.849  24.605  -4.925    9.849 93.701      1  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Linear model

```
lm(formula = value ~ 1 + condition,
   data = df.original)
```

$$\text{value}_i = b_0 + b_1 \cdot \text{condition}_i + e_i$$

i = observation

$$e_i \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = s_{\text{error}})$$

3 parameters: b_0, b_1, s_{error}

Linear mixed effects model

```
lmer(formula = value ~ 1 + condition +
                (1 | participant),
     data = df.original)
```

$$\text{value}_{ij} = b_0 + b_1 \cdot \text{condition}_{ij} + U_i + e_{ij}$$

i = participant,
j = time point

$$e_{ij} \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = s_{\text{error}})$$

$$U_i \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = s_{\text{U}})$$

b_0, b_1 = fixed effects
U_i = random effect

here: random intercept

4 parameters: b_0, b_1, s_{error}, s_{U}

Model coefficients

Linear model

```
fit = lm(formula = value ~ 1 + condition,
         data = df.original)
coef(fit)
```

```
(Intercept)  condition2
  0.1905239   0.1993528
```

- one intercept
- one slope for condition

Linear mixed effects model

```
fit = lmer(formula = value ~ 1 + condition +
                     (1 | participant),
           data = df.original)
coef(fit)
```

```
$participant
   (Intercept) condition2
1  -0.57839428  0.1993528
2   0.22299824  0.1993528
3  -0.82920677  0.1993528
4   1.49310938  0.1993528
5   0.36042775  0.1993528
6  -0.82060123  0.1993528
7   0.47929171  0.1993528
8   0.66401020  0.1993528
9   0.55135879  0.1993528
10 -0.28306703  0.1993528
11  1.57681676  0.1993528
12  0.38457642  0.1993528
13 -0.59969682  0.1993528
14 -2.21148391  0.1993528
15  1.05439374  0.1993528
16 -0.06476643  0.1993528
17 -0.03505690  0.1993528
18  0.93945348  0.1993528
19  0.87495531  0.1993528
20  0.63135911  0.1993528

attr(,"class")
[1] "coef.mer"
```

- different intercept for each participant
- one slope for condition

Understanding the **lmer()** summary

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
           (Intr)
condition2 -0.048
```

REML = restricted maximum likelihood

method for fitting models with **random effects**

Understanding the **lmer()** summary

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
           (Intr)
condition2 -0.048
```

fitting **lmer()** doesn't always work ...

lmer() complains when it didn't work

Understanding the **lmer()** summary

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
           (Intr)
condition2 -0.048
```

summary information about residuals

Understanding the **lmer()** summary

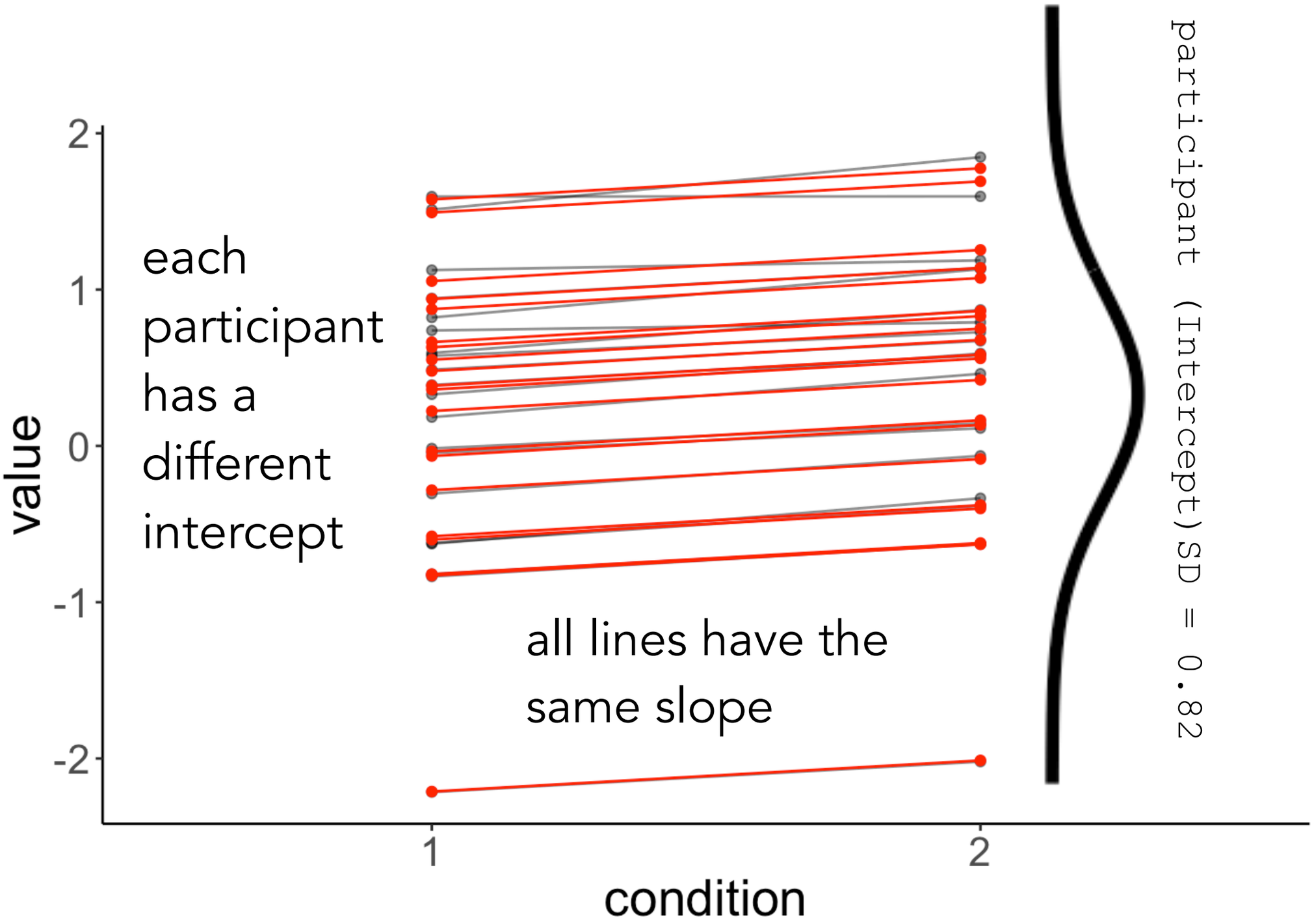
```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                     data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36309 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
            (Intr)
condition2 -0.048
```

one parameter to capture the variance between participants (gives us a sense for whether there are interindividual differences)

one parameter to capture the residual variance (just like sigma in an **lm()**)

Understanding the **`lmer()`** summary



Understanding the **lmer()** summary

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
            (Intr)
condition2 -0.048
```

one parameter for the global intercept (value for the baseline condition)

one parameter for the condition effect (difference between the two conditions)

interpretation the same as for **lm()**, also: we can use contrasts!

Understanding the `lmer()` summary

```
1 # fit a linear mixed effects model
2 lmer(formula = value ~ condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
           (Intr)
condition2 -0.048
```

correlation between intercept and condition2

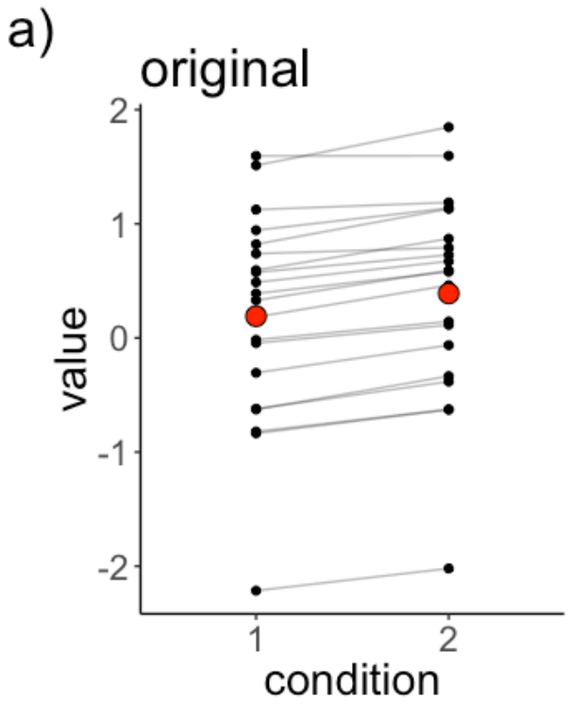
The "correlation of fixed effects" output doesn't have the intuitive meaning that most would ascribe to it. Specifically, is not about the correlation of the variables. It is in fact about the expected correlation of the regression coefficients.

we just performed a paired t-test ...

```r
t.test(df.original$value[df.original$condition == "1"],
       df.original$value[df.original$condition == "2"],
       alternative = "two.sided",
       paired = T)
```

```
Paired t-test

data:  df.original$value[df.original$condition == "1"] and
df.original$value[df.original$condition == "2"]
t = -10.231, df = 19, p-value = 3.636e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2401340 -0.1585717
sample estimates:
mean of the differences
             -0.1993528
```

If we take the differences for each participant between condition 1 and condition 2, are these difference scores significantly different from 0?



we just performed a paired t-test ...

```
lmer(formula = value ~ condition + (1 | participant),
     data = df.original)
```

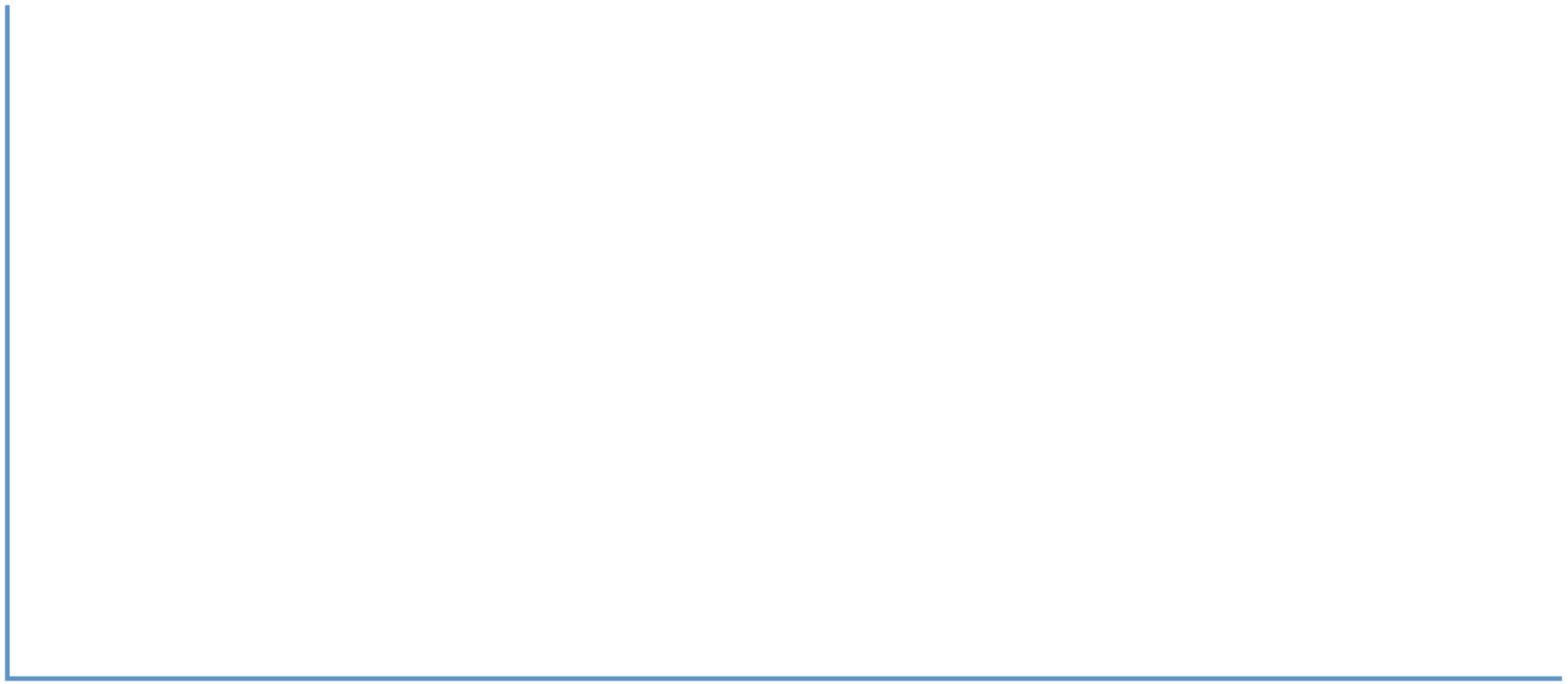
- explicitly models the interindividual variation
- much more flexible ...

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ condition + (1 | participant)
   Data: df.original

REML criterion at convergence: 17.3

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.55996 -0.36399 -0.03341  0.34400  1.65823

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.816722 0.90373
 Residual                0.003796 0.06161
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.19052    0.20255   0.941
condition2   0.19935    0.01948  10.231

Correlation of Fixed Effects:
            (Intr)
condition2 -0.048
```

```
1 t.test(df.original$value[df.original$condition == "1"],
2        df.original$value[df.original$condition == "2"],
3        alternative = "two.sided",
4        paired = T)
```

```
Paired t-test

data:  df.original$value[df.original$condition == "1"] and
df.original$value[df.original$condition == "2"]
t = -10.231, df = 19, p-value = 3.636e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.2401340 -0.1585717
sample estimates:
mean of the differences
             -0.1993528
```

How was the pace of today's class?



much too slow

a little too slow

just right

a little too fast

much too fast

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!