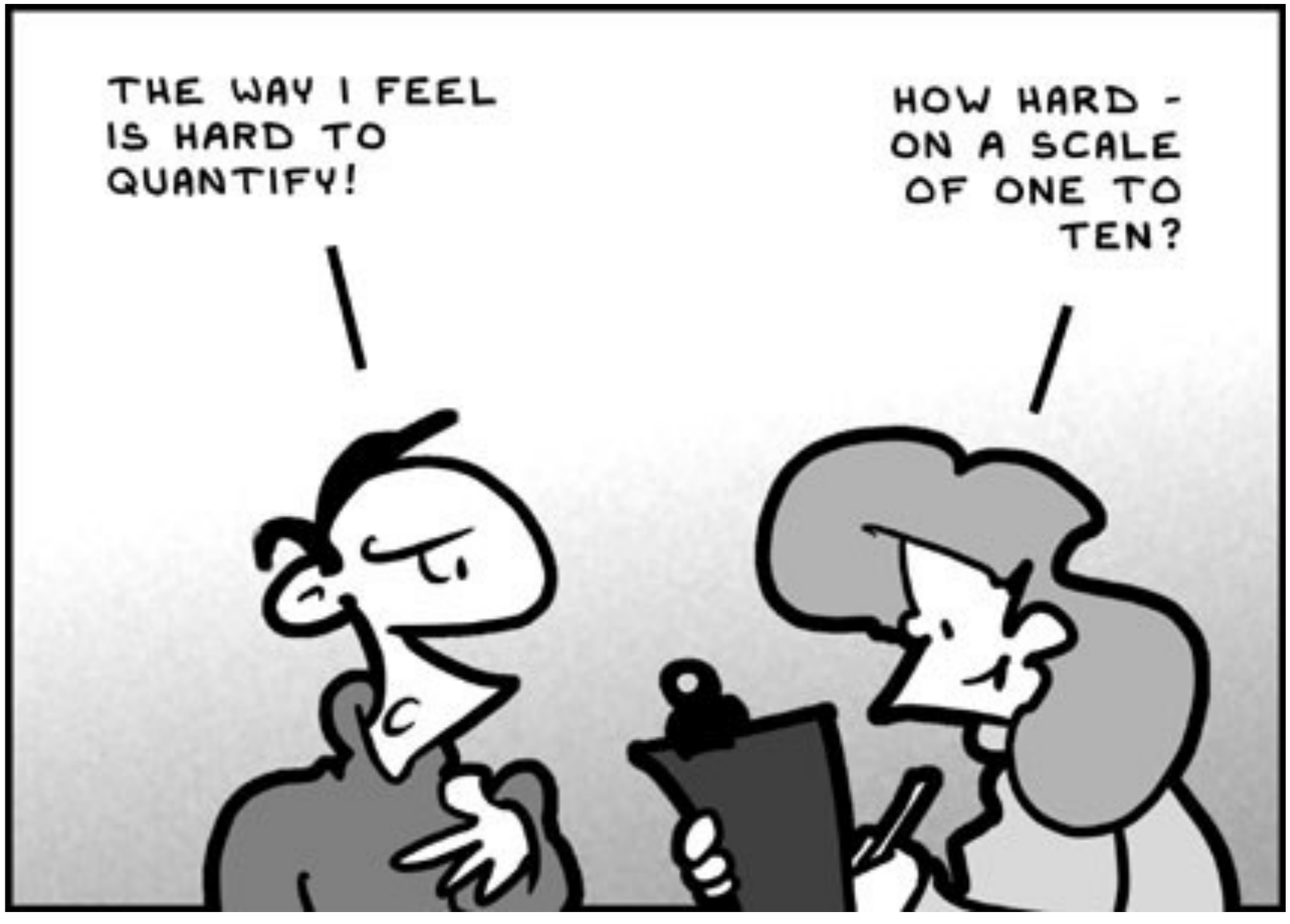
Linear mixed effects models 1



02/20/2019

Logistics

Things that came up

Reporting statistical results

Can we review how to report stats? We talked about it for different examples, but it would be really helpful to review all the different rules for reporting statistics in one thorough session.

yes, we'll have a session on this!

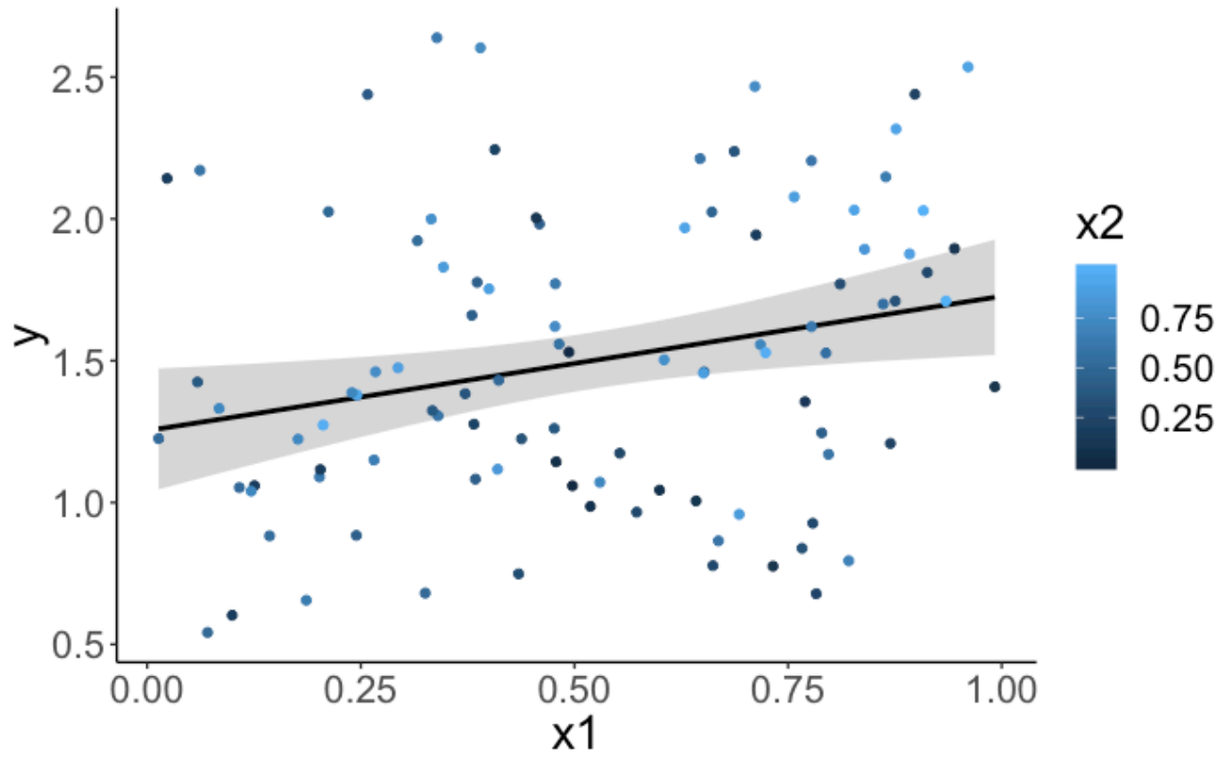
F vs. *t* in `lm()` output

One thing that did come up in the midterm though: why does **`lm()`** output a *t* value and **`anova`**() output an *F* value. And what's the difference/relationship between the two.

let me clarify

F vs. *t* in `lm()` output

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

# parameters
sample_size = 100
b0 = 1
b1 = 0.5
b2 = 0.5
sd = 0.5

# sample
df.data = tibble(
  participant = 1:sample_size,
  x1 = runif(sample_size, min = 0, max = 1),
  x2 = runif(sample_size, min = 0, max = 1),
  # simple additive model
  y = b0 + b1 * x1 + b2 * x2 + rnorm(sample_size, sd = sd)
)
```



$$Y_i = b_0 + b_1 \cdot x1_i + b_2 \cdot x2_i + e_i$$

F vs. *t* in `lm()` output

```r
# fit linear model
fit = lm(formula = y ~ 1 + x1 + x2,
    data = df.data)

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

```
Call:
lm(formula = y ~ x1 + x2, data = df.data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9290 -0.3084 -0.0716  0.2676  1.1659

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9953     0.1395   7.133 1.77e-10 ***
x1            0.4654     0.1817   2.561  0.01198 *
x2            0.5072     0.1789   2.835  0.00558 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4838 on 97 degrees of freedom
Multiple R-squared:  0.1327,  Adjusted R-squared:  0.1149
F-statistic: 7.424 on 2 and 97 DF,  p-value: 0.001
```

F vs. *t* in `lm()` output

only intercept

full model

```
1 # fit models
2 model_compact = lm(formula = y ~ 1,
3                    data = df.data)
4
5 model_augmented = lm(formula = y ~ 1 + x1 + x2,
6                      data = df.data)
7
8 # compare models using the F-test
9 anova(model_compact, model_augmented)
```

```
Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ 1 + x1 + x2
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     99 26.175
2     97 22.700  2    3.4746 7.4236  0.001 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

F vs. *t* in `lm()` output

```
Call:
lm(formula = y ~ x1 + x2, data = df.data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9290 -0.3084 -0.0716  0.2676  1.1659

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9953     0.1395   7.133 1.77e-10 ***
x1            0.4654     0.1817   2.561  0.01198 *
x2            0.5072     0.1789   2.835  0.00558 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4838 on 97 degrees of freedom
Multiple R-squared:  0.1327,  Adjusted R-squared:  0.1149
F-statistic: 7.424 on 2 and 97 DF,  p-value: 0.001
```

`summary()`

`anova(` compact, augmented)

```
Analysis of Variance Table

Model 1: y ~ 1
Model 2: y ~ 1 + x1 + x2
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     99 26.175
2     97 22.700  2    3.4746 7.4236  0.001 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

F vs. *t* in `lm()` output

```
Call:
lm(formula = y ~ x1 + x2, data = df.data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9290 -0.3084 -0.0716  0.2676  1.1659

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9953     0.1395   7.133 1.77e-10 ***
x1            0.4654     0.1817   2.561  0.01198 *
x2            0.5072     0.1789   2.835  0.00558 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4838 on 97 degrees of freedom
Multiple R-squared:  0.1327,  Adjusted R-squared:  0.1149
F-statistic: 7.424 on 2 and 97 DF,  p-value: 0.001
```

Is x1 a significant predictor, controlling for x2?

F vs. *t* in `lm()` output

without x1

with x1

```
# fit models
model_compact = lm(formula = y ~ 1 + x2,
                   data = df.data)

model_augmented = lm(formula = y ~ 1 + x1 + x2,
                     data = df.data)

# compare models using the F-test
anova(model_compact, model_augmented)
```

```
Analysis of Variance Table

Model 1: y ~ 1 + x2
Model 2: y ~ 1 + x1 + x2
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     98 24.235
2     97 22.700  1    1.5347 6.558 0.01198 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

F vs. *t* in `lm()` output

```
Call:
lm(formula = y ~ x1 + x2, data = df.data)

Residuals:
    Min      1Q  Median      3Q     Max
-0.9290 -0.3084 -0.0716  0.2676  1.1659

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.9953     0.1395   7.133 1.77e-10 ***
x1            0.4654     0.1817   2.561  0.01198 *
x2            0.5072     0.1789   2.835  0.00558 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4838 on 97 degrees of freedom
Multiple R-squared:  0.1327,  Adjusted R-squared:  0.1149
F-statistic: 7.424 on 2 and 97 DF,  p-value: 0.001
```

```
Analysis of Variance Table

Model 1: y ~ 1 + x2
Model 2: y ~ 1 + x1 + x2
  Res.Df    RSS Df Sum of Sq     F   Pr(>F)
1     98 24.235
2     97 22.700  1    1.5347 6.558 0.01198 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

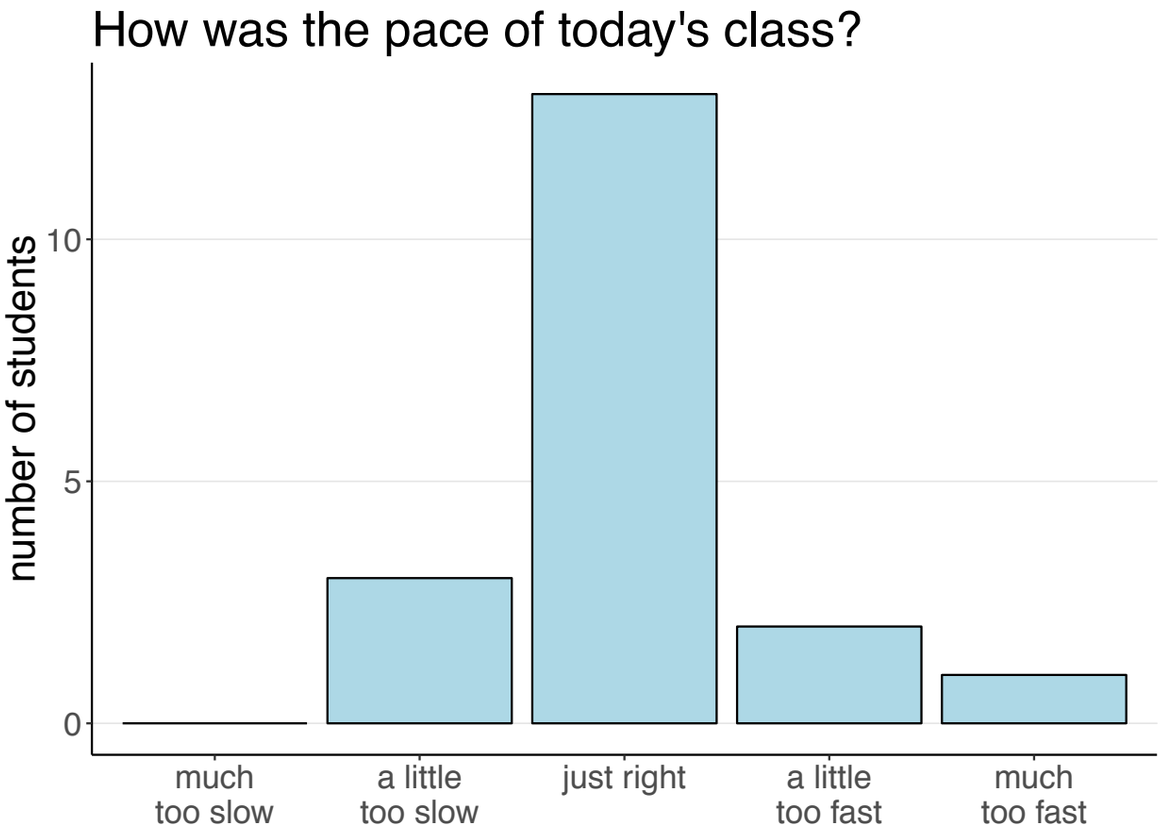
deterministic mapping between t and F

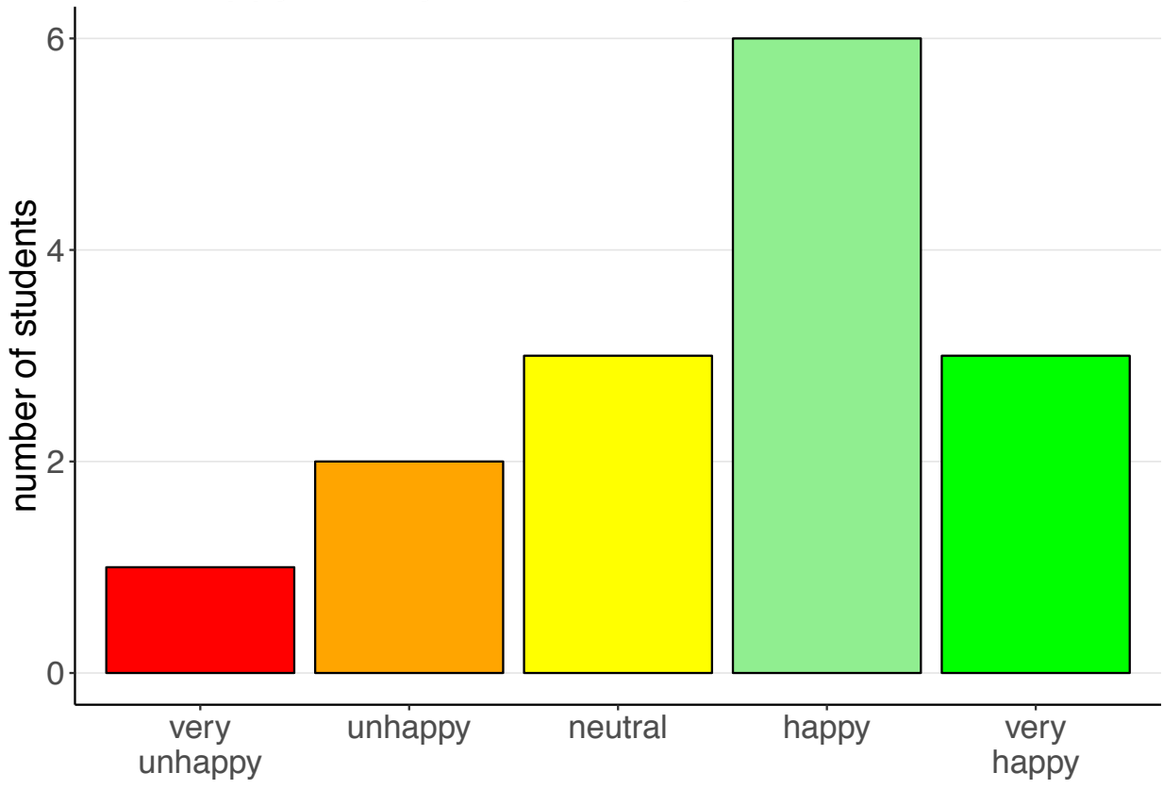
$$t = \sqrt{F}$$

$$2.561 = \sqrt{6.558}$$

Your feedback

Your feedback





lecture needs some work ...

Your feedback

So satisfying to finally understand LOO!!

I really like today's class. it makes a lot of sense to me

some positive notes

Your feedback

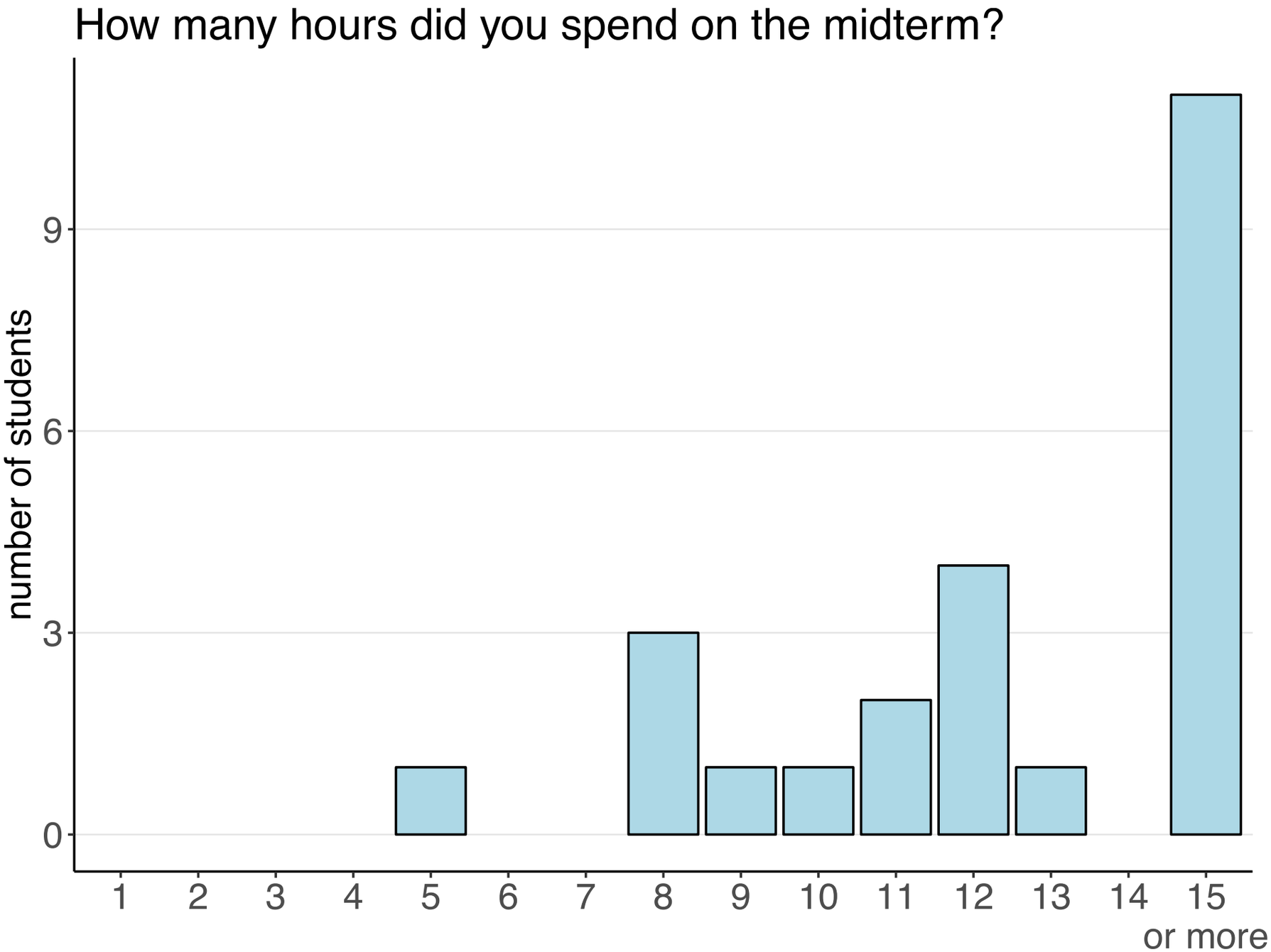
Presenting the code can be kind of dense. **There is a lot packed into each individual R function**. But I appreciate the step by step walk through. I still start to lose it at the end

A little **too much of reciting code in class today**, I think (as opposed to more conceptual content).

I can see how much you put in the class. I think the class is ambitious to teach students (who don't really have a lot of programming skills) simulations. There is way too much information over there, in addition to the goal of teaching students other important concepts. **I completely failed to follow the simulation r code**. On the other hand, I **also failed to understand other concepts, such as the leave-one-out approach**.

will try the class as a practical class using RMarkdown next time

Midterm



mean = 13.7

sorry!

Your feedback

I thought the midterm was incredibly helpful, and thoughtfully written. In particular, I had been meaning to implement the model comparison approach on my own time but hadn't gotten around to it, so **having a structured way to practice that using a straight-forward dataset was really great**. Even though it was time-consuming, it was 100% worth it considering how much I gained from it. (In fact, thanks to the midterm, in a grant proposal I wrote for another class, I felt comfortable proposing a model comparison approach for the methods section, which felt very satisfying.) Thanks for an awesome and instructive midterm!

I think this midterm was a much better way to learn and understand the material than the previous homework. **However, it was somewhat repetitive and much too long.** Additionally, in general it seems more fair to give everyone more time on an exam rather than to given certain students extensions.

Your feedback

You need to give us a chance to discuss feedback on the midterm. Clearly, it was not a 6-hour exam (please report the mean length), and even if it was, you should not have assigned a 6-hour exam to begin with. The purpose of the take-home exam is to offer an exam in a more flexible format. Stanford's exam policy specifies that (final!) exams fit within a 3-hour slot. Unless you have a substantial reason why you cannot evaluate your course material in such a format, you should not use the take-home option as a mechanism to simply assign a longer exam (Stanford even has formal channels for students to bring such practices to the attention of University administrators). **Moreover, the exam clearly did not need to be so long, as its length was not due to challenging questions but rather to a great deal of repetitive material**. If we can interpret coefficients in a linear regression once, we don't need to do it 3 times more. **The classmates with which I spoke shared the sentiment that the midterm was disrespectful of our time (in a fashion similar to that of HW2)**.

Thanks for the honest feedback! We will make sure to shorten the exam substantially.

I take full responsibility for the mistake in timing.

Project proposal

Upload the proposal as a pdf on canvas.

we will post the assignment on canvas after class today

Homework 5

Homework 5

- **Topics**:
  - power analysis
  - bootstrapping
  - cross-validation

- will be available after class today
- is due on **Tuesday, February 26th at 8pm**
- upload via canvas
- post any questions on Piazza

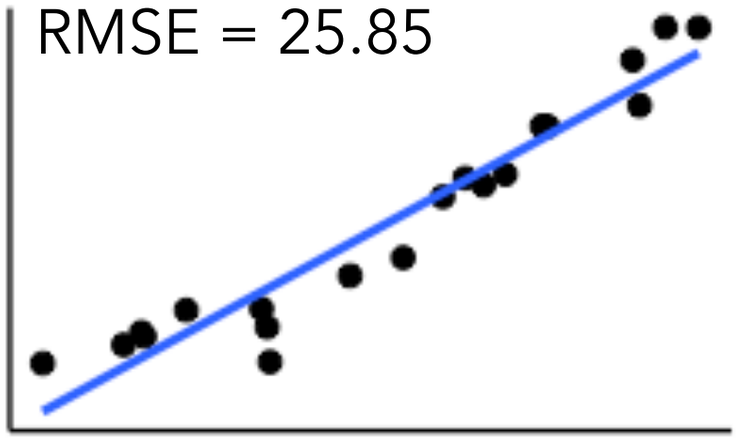
Plan for today

- Model comparison
  - Cross-validation
  - AIC and BIC
- Linear mixed effects model
  - modeling dependence in data

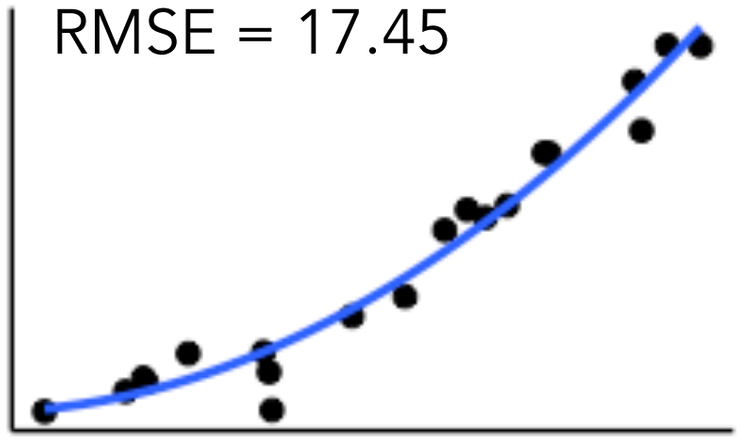
Cross-validation

Which model describes the data best?

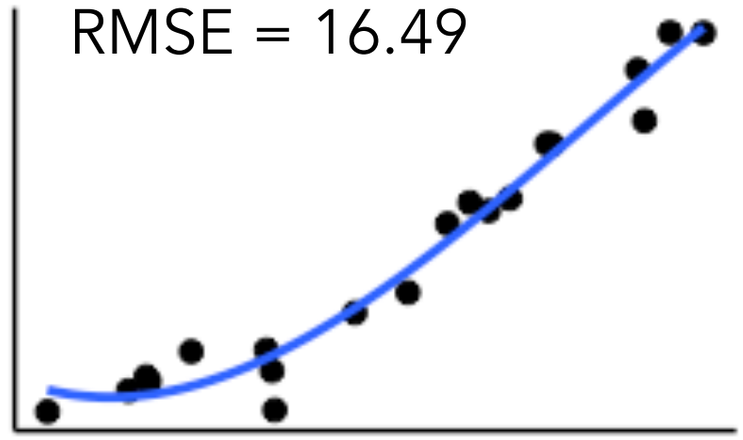
RMSE = 25.85

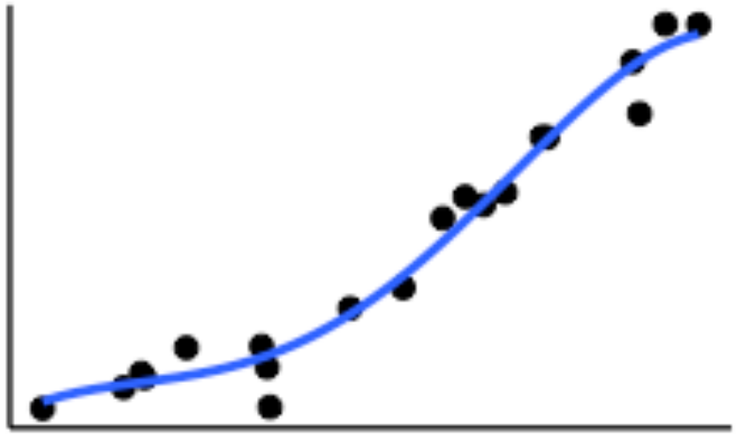


RMSE = 17.45

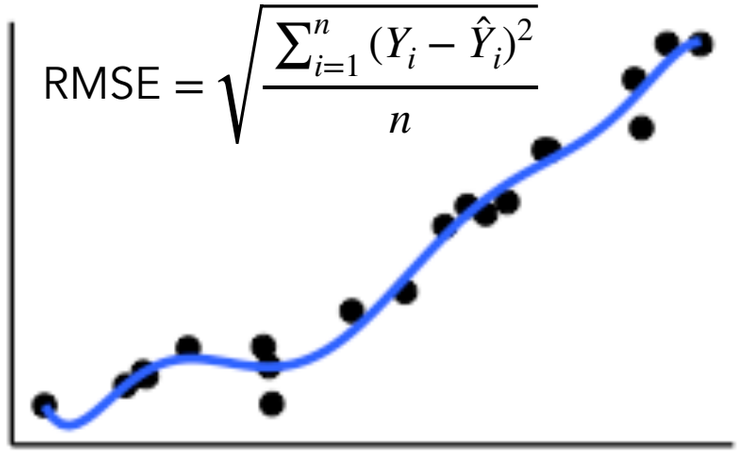


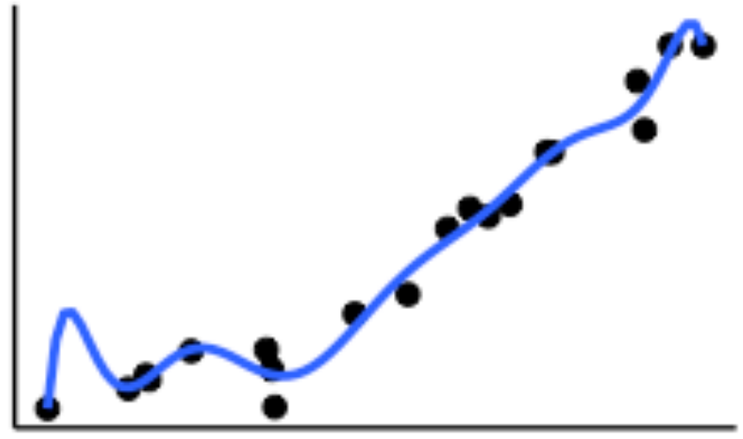
RMSE = 16.49

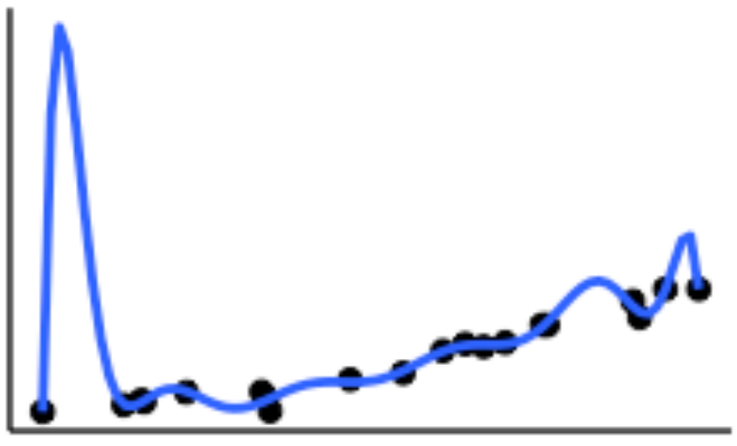


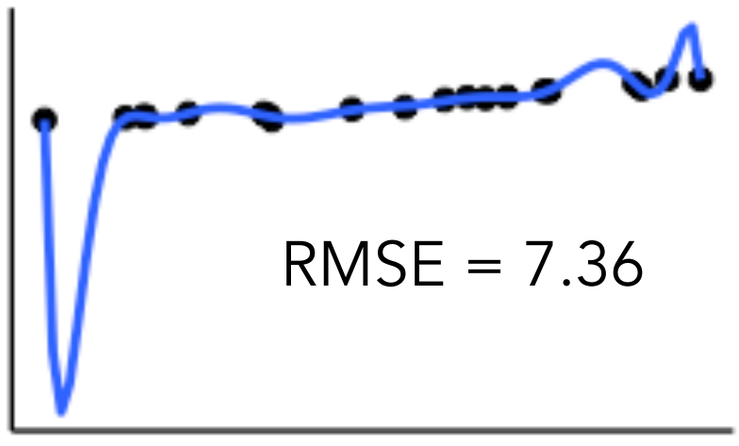


$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{n}}$$

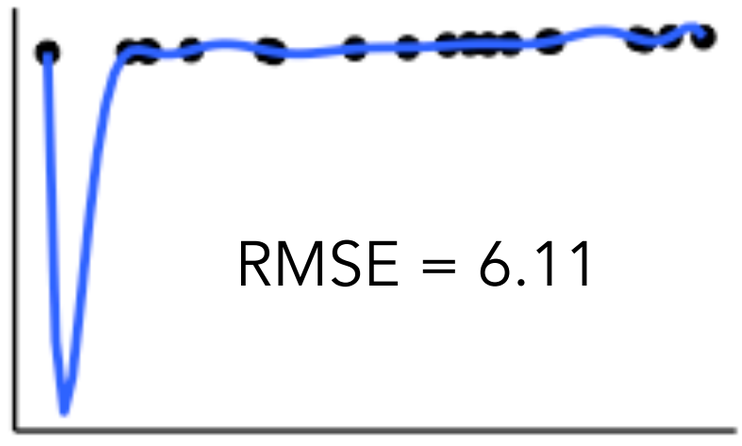








RMSE = 7.36

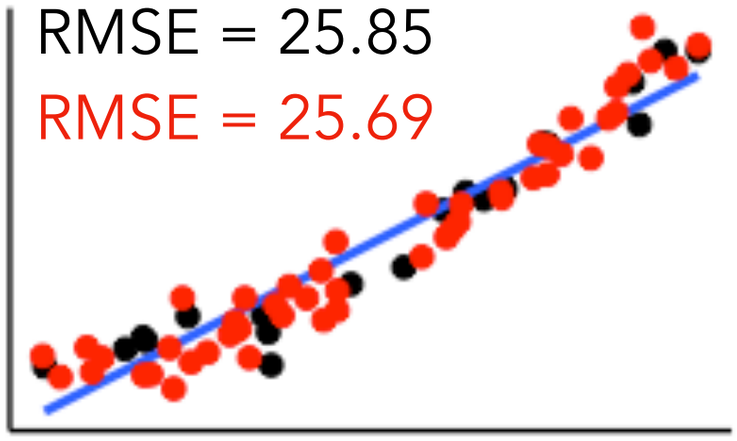


RMSE = 6.11

Which model describes the data best?

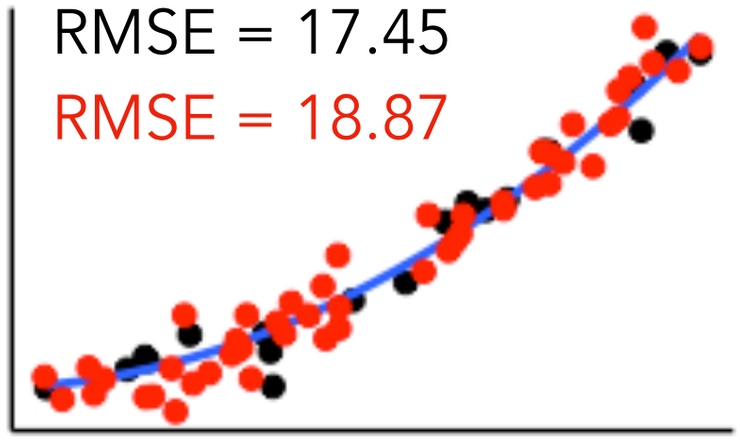
RMSE = 25.85

RMSE = 25.69



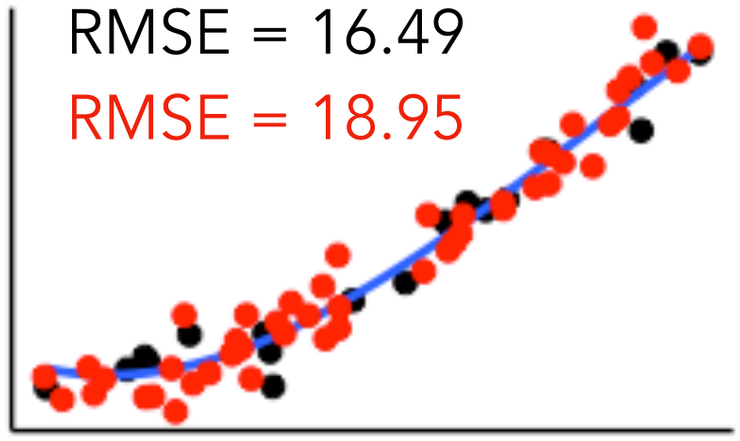
RMSE = 17.45

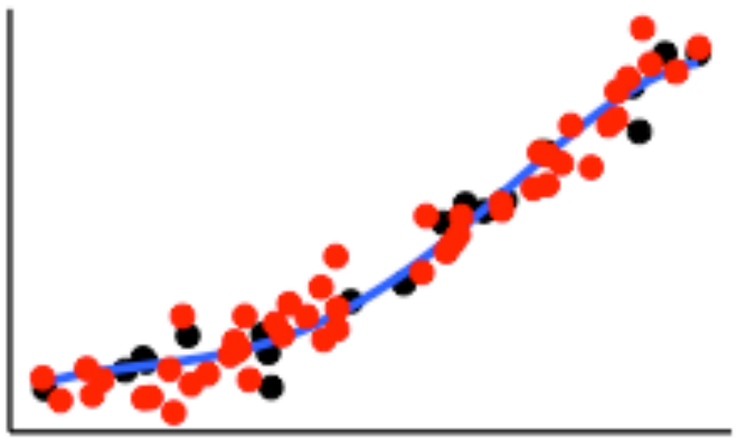
RMSE = 18.87



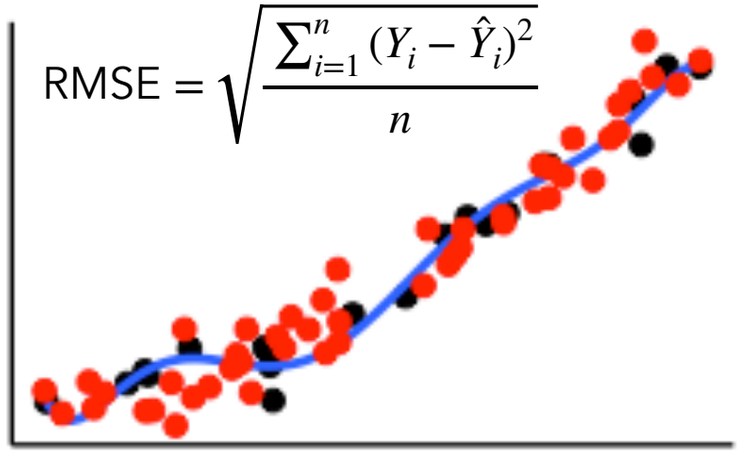
RMSE = 16.49

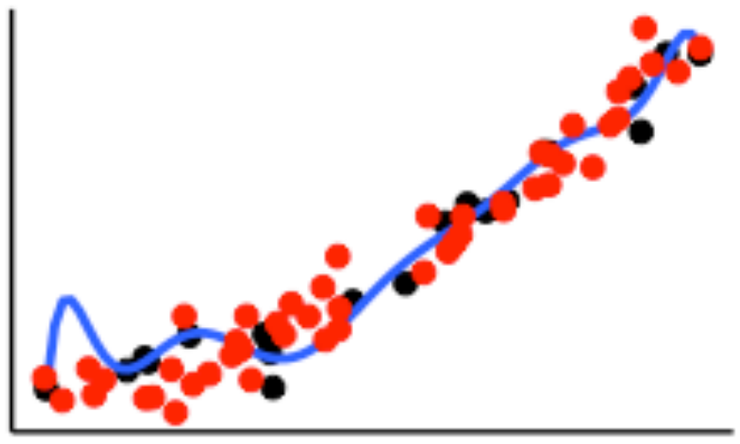
RMSE = 18.95

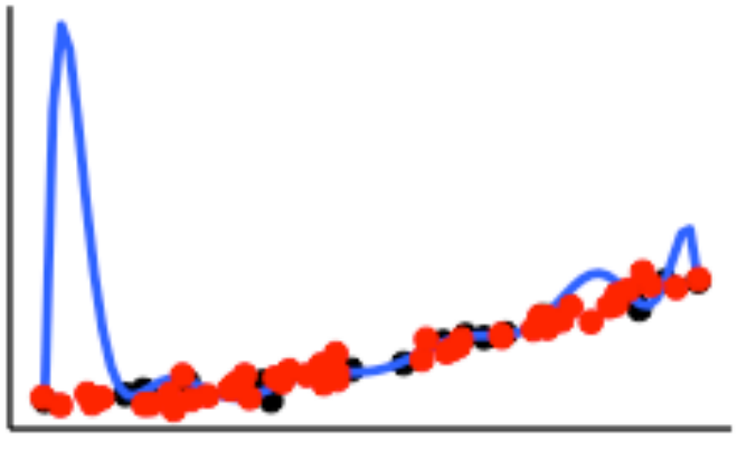


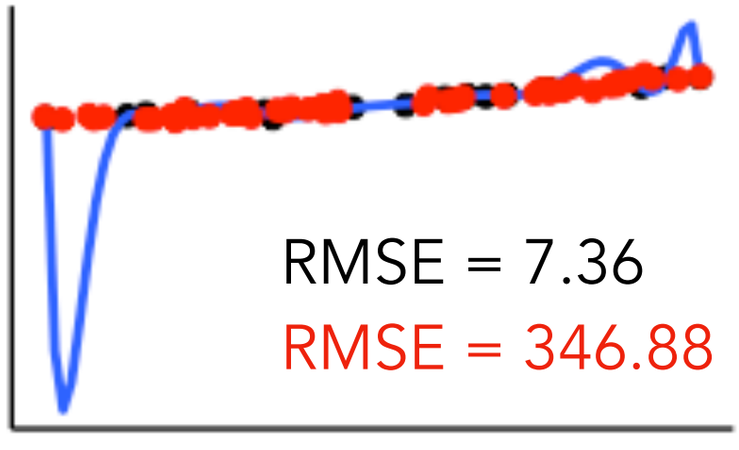


$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}{n}}$$



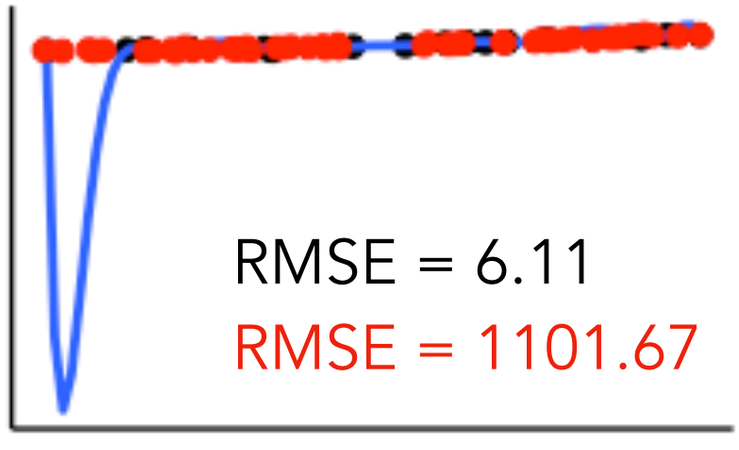






RMSE = 7.36

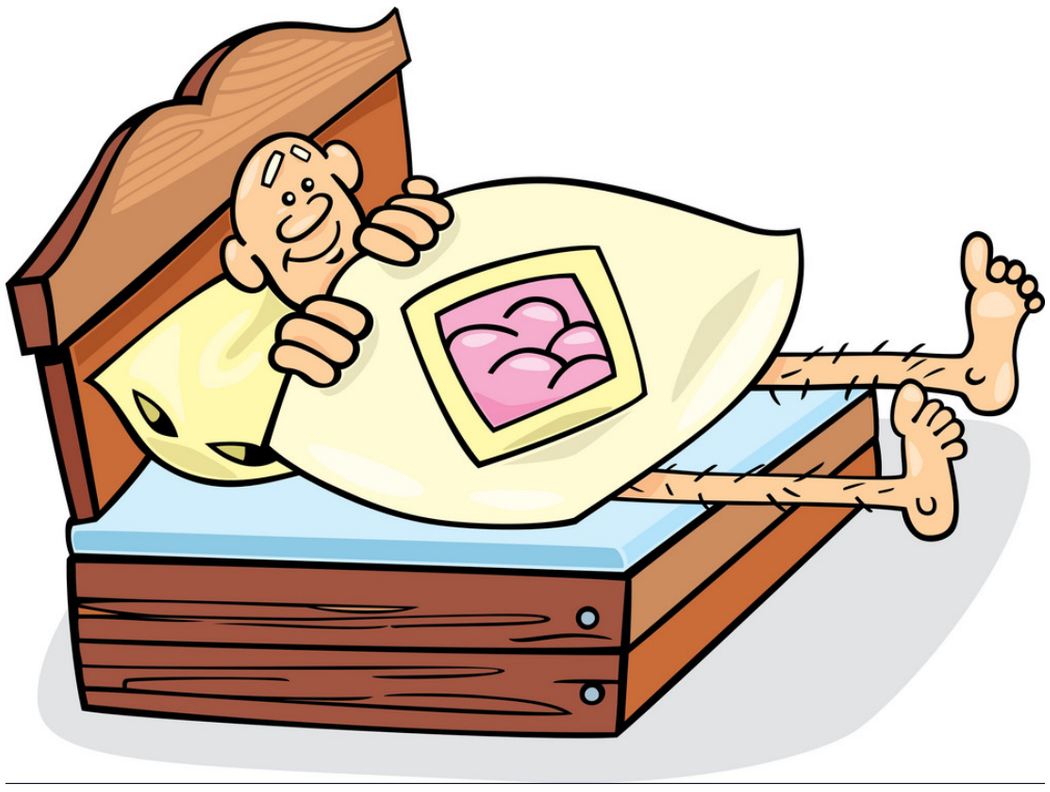
RMSE = 346.88



RMSE = 6.11

RMSE = 1101.67

Underfitting vs. Overfitting



ONE WAY TO EXPLAIN UNDERFITTING

Underfitting vs. Overfitting

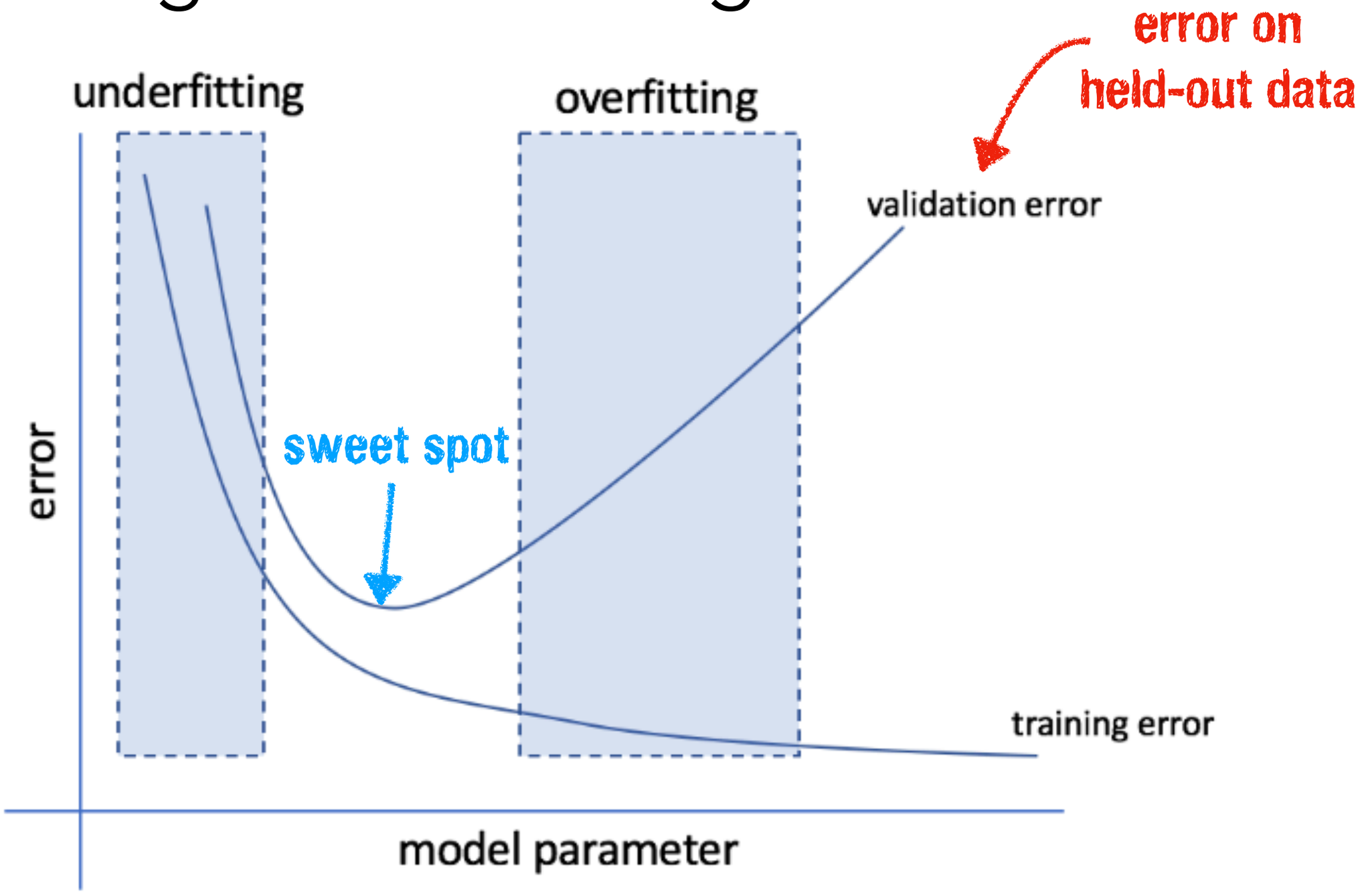
- **a good model should**:
  - explain the actual data well
  - predict future data well
- **bias-variance tradeoff**:
  - **bias** = error from erroneous assumptions in the model, high bias can cause a model to miss the relevant relations between predictors and outcome

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



in machine learning, the goal is often to find the sweet spot between underfitting and overfitting

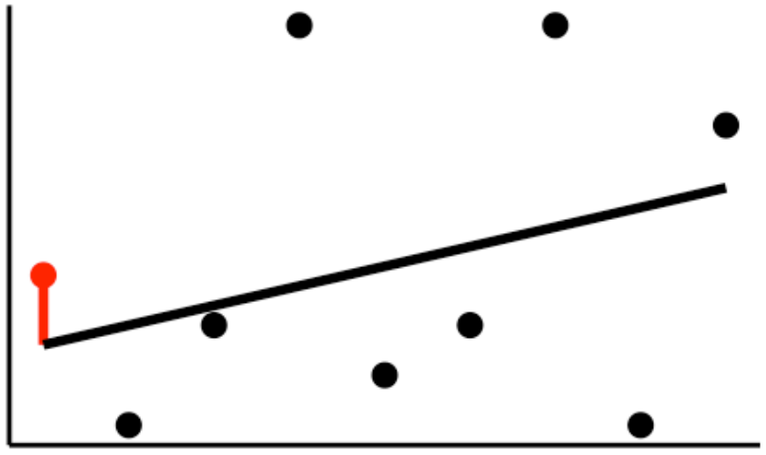
Leave-one-out crossvalidation

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

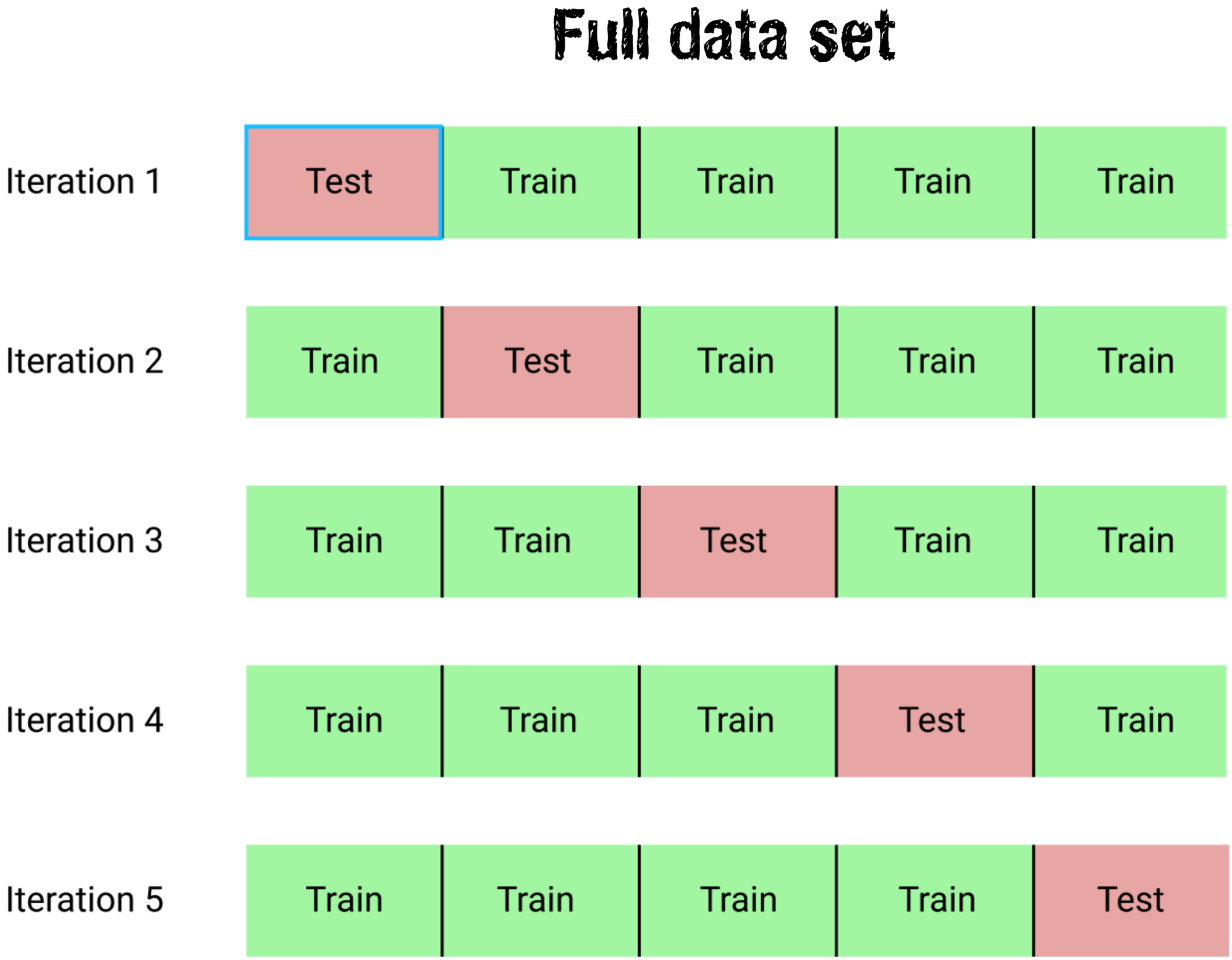
- train the model on all the data points except for one
- calculate the prediction error for the held-out data point

repeat for all data points

- **problem**: can be computationally expensive since it requires fitting the model *n* times

k-fold cross validation

k-fold crossvalidation



k-fold crossvalidation

k-fold crossvalidation

```
1 df.cross = df.data %>%
2   crossv_kfold(k = 10) %>%
3   mutate(model_simple = map(train, ~ lm(y ~ 1 + x, data = .)),
4          model_correct = map(train, ~ lm(y ~ 1 + x + I(x^2), data = .)),
5          model_complex = map(train, ~ lm(y ~ 1 + x + I(x^2) + I(x^3), data = .))) %>%
6   gather("model", "fit", contains("model")) %>%
7   mutate(rsquare = map2_dbl(fit, test, rsquare))
```

using R^2 as a measure

this wouldn't work for LOO because ...

model	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
- 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, ...

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

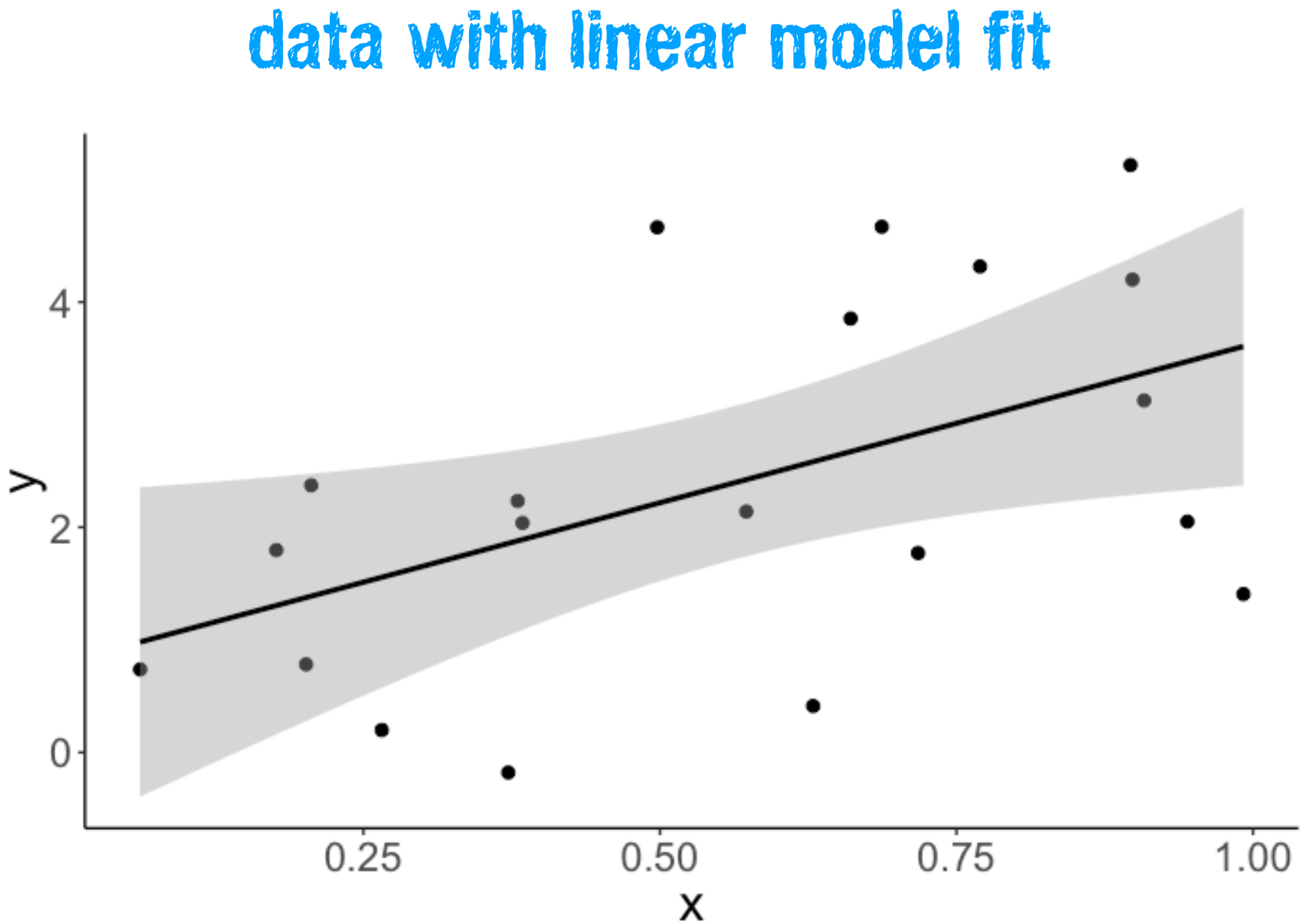
k = number of parameters in the model

n = number of observations

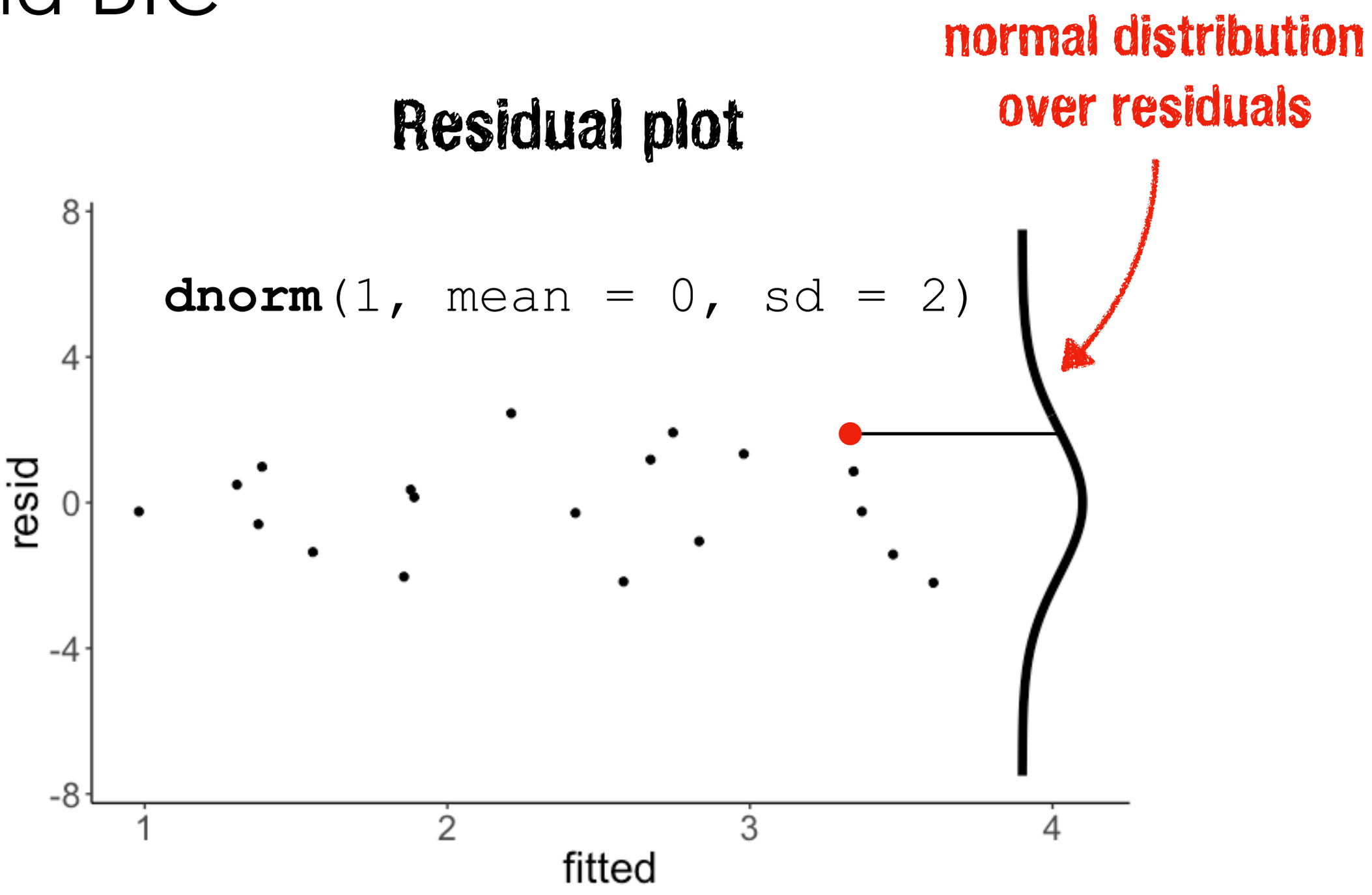
AIC and BIC

- 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

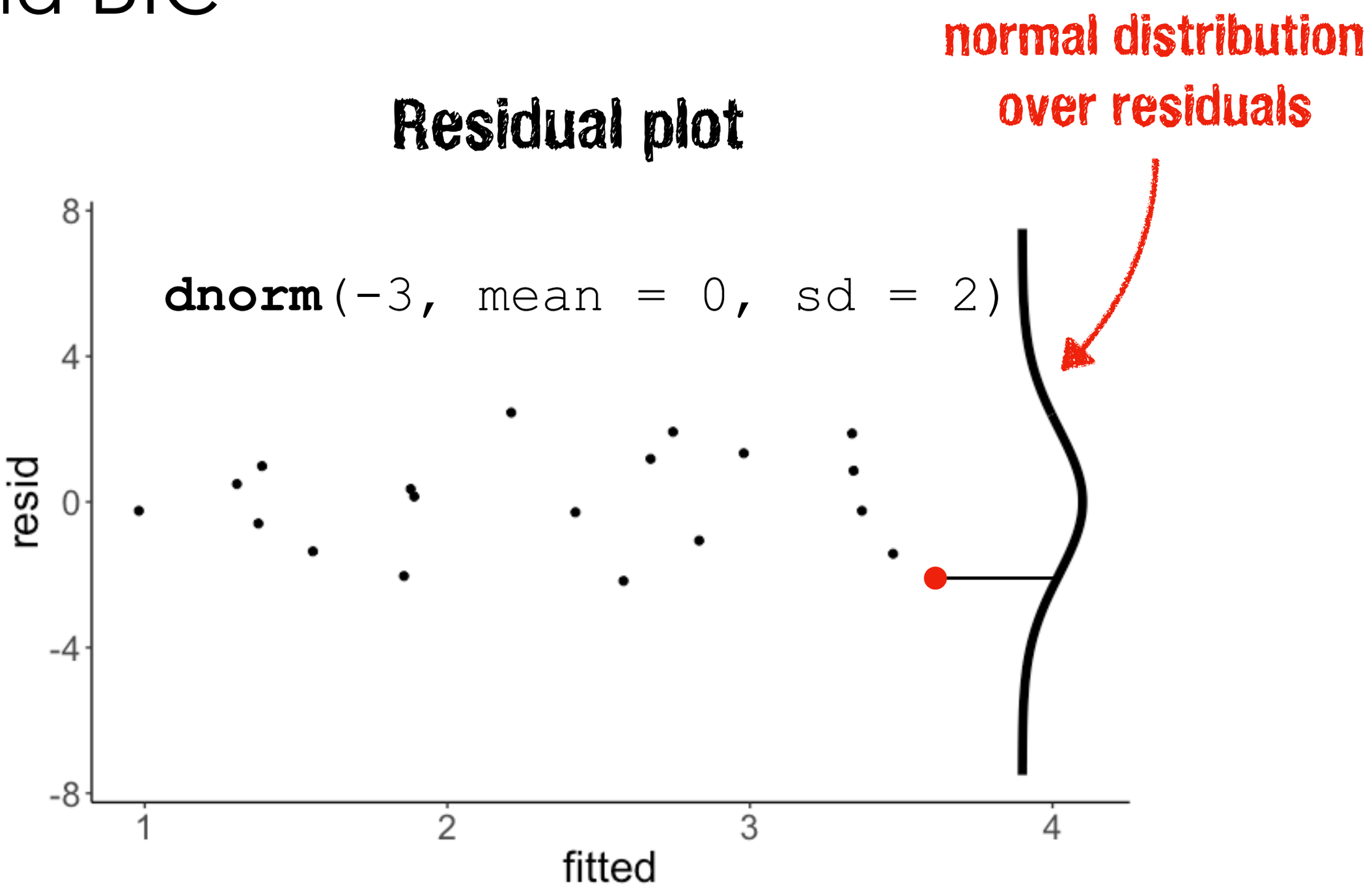
AIC and BIC



AIC and BIC


normal distribution
over residuals
Residual plot
dnorm(1, mean = 0, sd = 2)
resid
8
4
0
-4
-8
1
2
3
4
fitted

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(4.20, mean = 0, sd = 2.17) = 0.02

```
# 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
0.41	5.74	1.53	4.20	0.02
0.91	4.86	4.80	0.06	0.18
0.29	0.98	0.80	0.18	0.18
0.46	0.71	1.87	-1.16	0.16
0.33	-0.34	1.05	-1.39	0.15
0.65	0.82	3.11	-2.29	0.11
0.26	-1.35	0.57	-1.92	0.12
0.48	4.75	2.00	2.75	0.08
0.77	4.96	3.86	1.11	0.16
0.08	0.80	-0.55	1.35	0.15

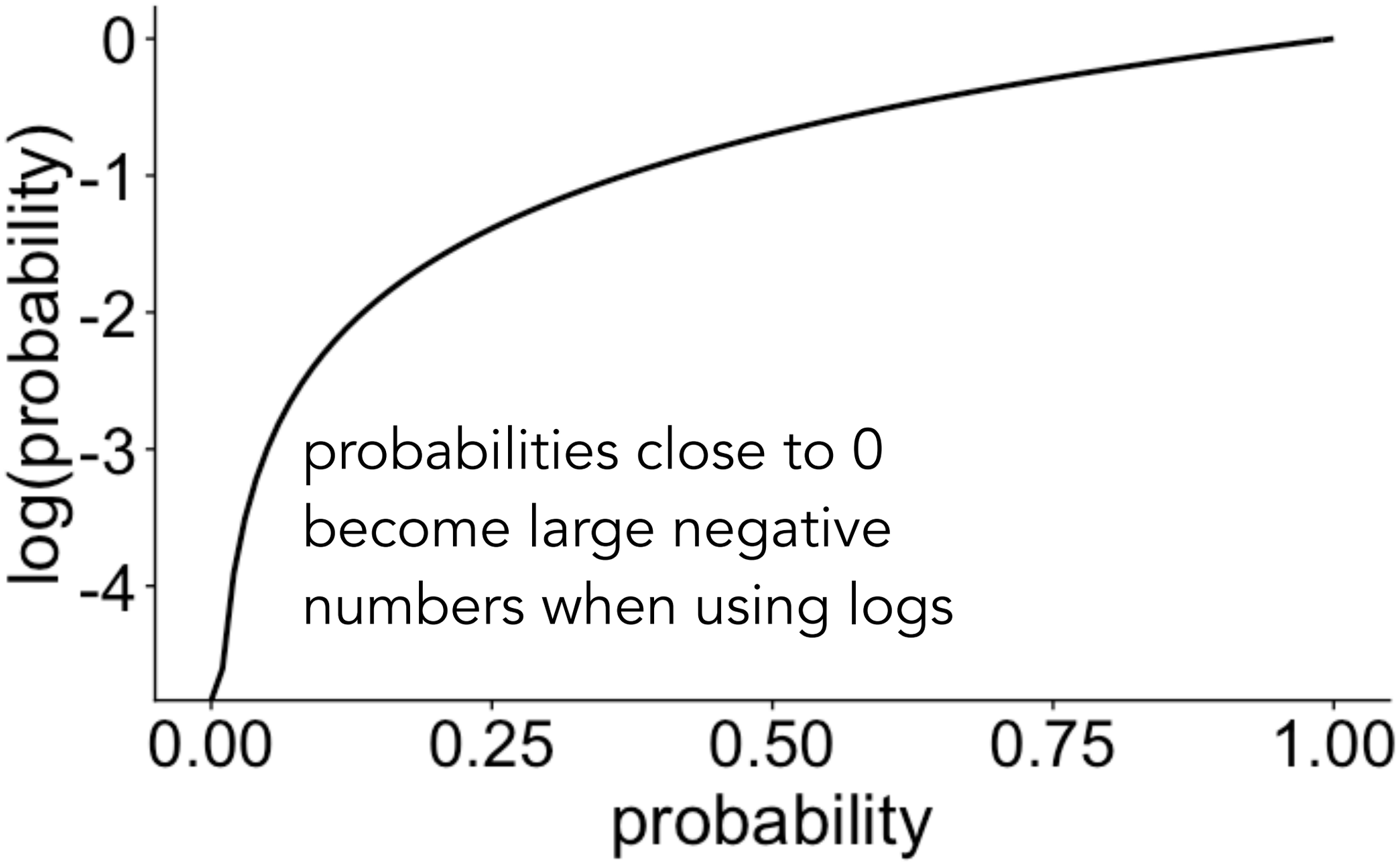
inferred standard deviation of the error

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

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual
0.38	0.35	2.17	11.2	0	2	-42.78	91.56	94.55	84.43	18

$$e \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = 2.17)$$

`log()` is your friend!



Clue guide to probability

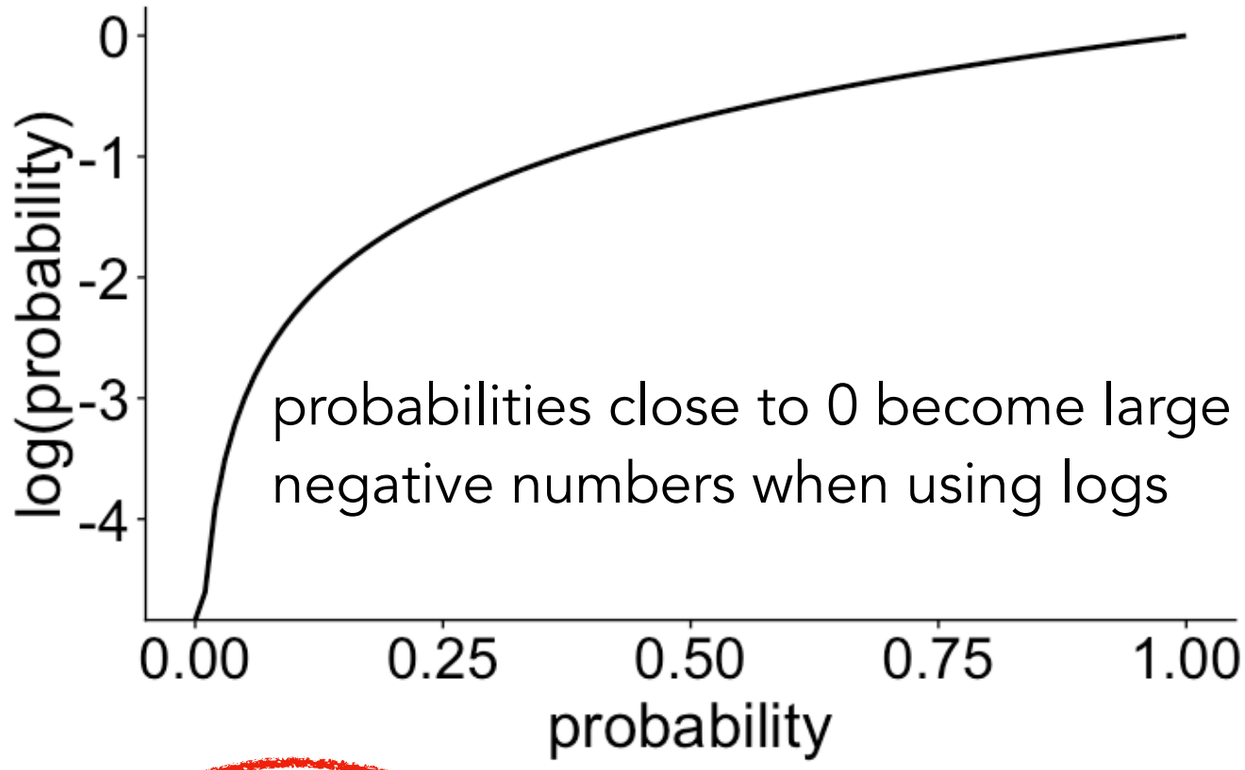
- *joint probability*:
- if A and B are independent then
- **Definition**: $p(A, B) = p(A) \cdot p(B)$

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

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`

summing logs

number becomes large but that's ok

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

transform back into probability

`exp(-18.42068)= 0.00000001`

often not necessary since we just use logLikelihood

AIC and BIC

dnorm(4.20, mean = 0, sd = 2.17) = 0.02

```
# 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
0.41	5.74	1.53	4.20	0.02
0.91	4.86	4.80	0.06	0.18
0.29	0.98	0.80	0.18	0.18
0.46	0.71	1.87	-1.16	0.16
0.33	-0.34	1.05	-1.39	0.15
0.65	0.82	3.11	-2.29	0.11
0.26	-1.35	0.57	-1.92	0.12
0.48	4.75	2.00	2.75	0.08
0.77	4.96	3.86	1.11	0.16
0.08	0.80	-0.55	1.35	0.15

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
0.38	0.35	2.17	11.2	0	2	-42.78	91.56	94.55	84.43	18

$$e \sim \mathcal{N}(\text{mean} = 0,\ \text{sd} = 2.17)$$

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
- The BIC generally penalizes free parameters more strongly than the Akaike information criterion, though it depends on the size of *n* and relative magnitude of *n* and *k*.

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).

Linear mixed effects models

Dependence

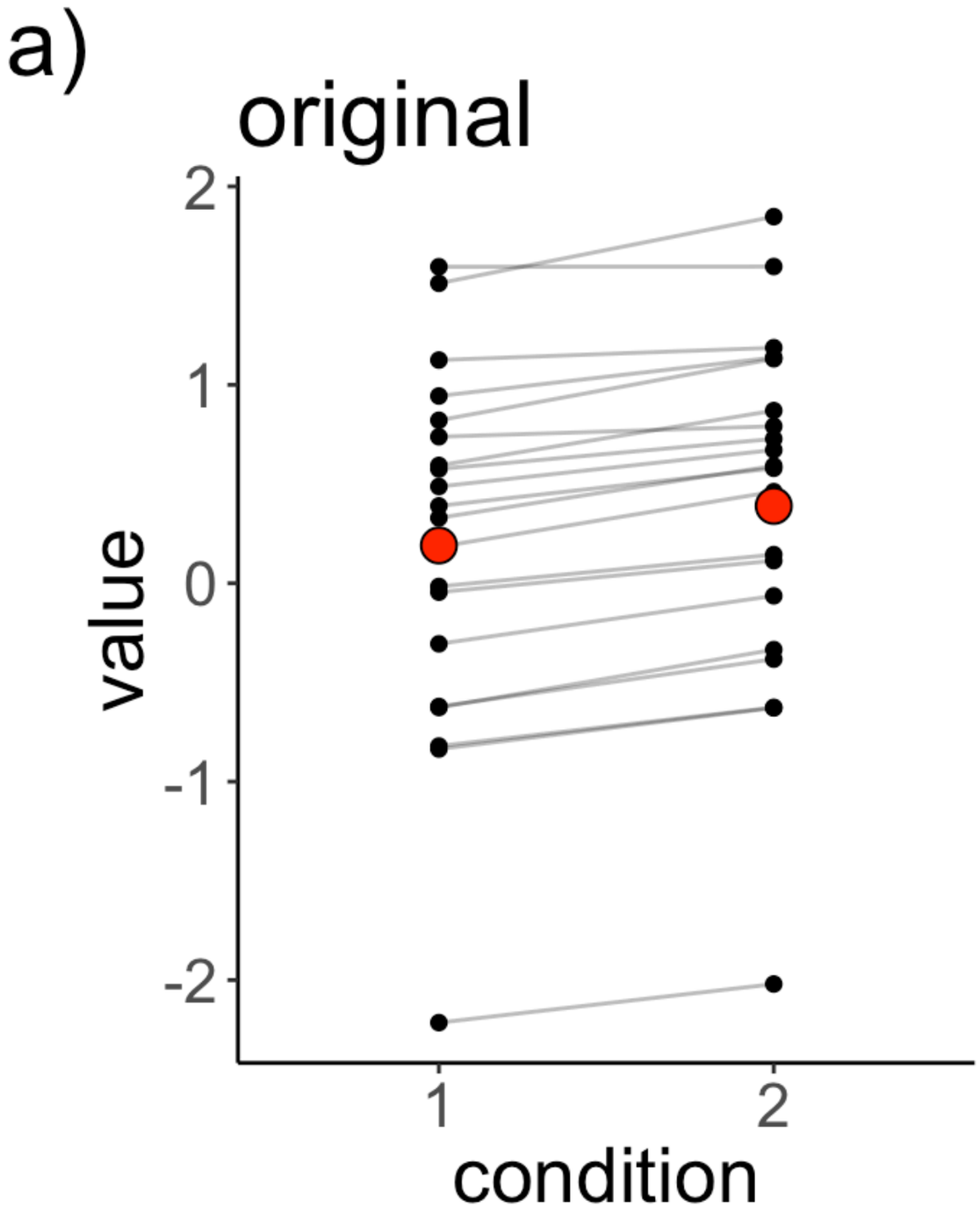
- so far, all the models that we've discussed (linear model with different kinds of predictors and contrasts) make the assumption that the data are **iid** (independent, and identically distributed)

- 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

Dependence

Does it really matter?

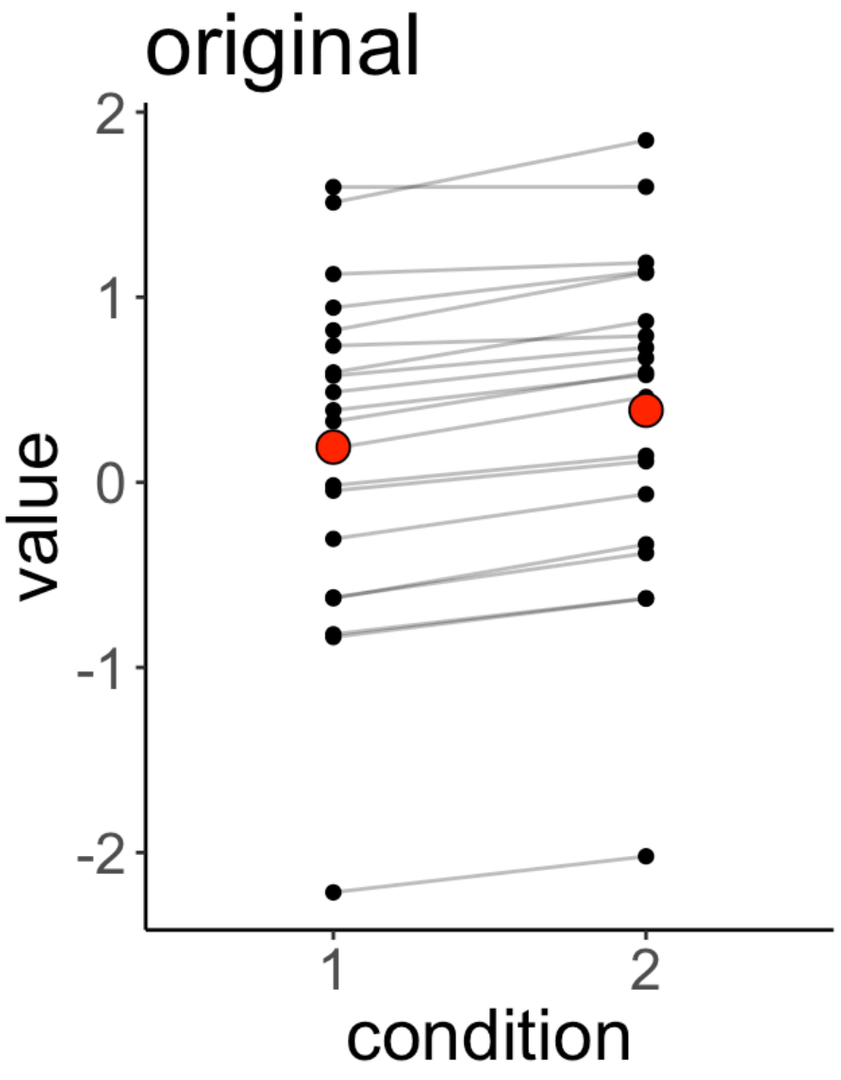
Is there a significant difference between conditions 1 and 2?



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
```

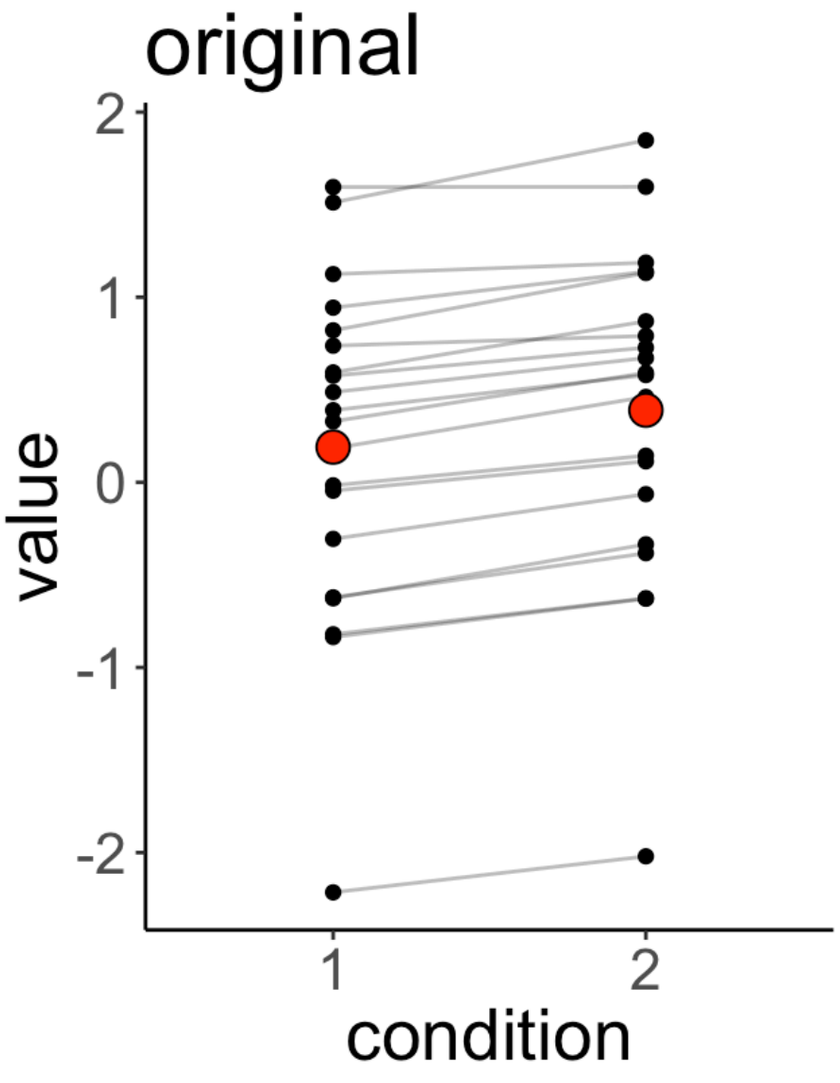
- we ignore the fact that we have repeated observations from the same participants
- in the data it looks like there is a small but consistent effect of condition

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

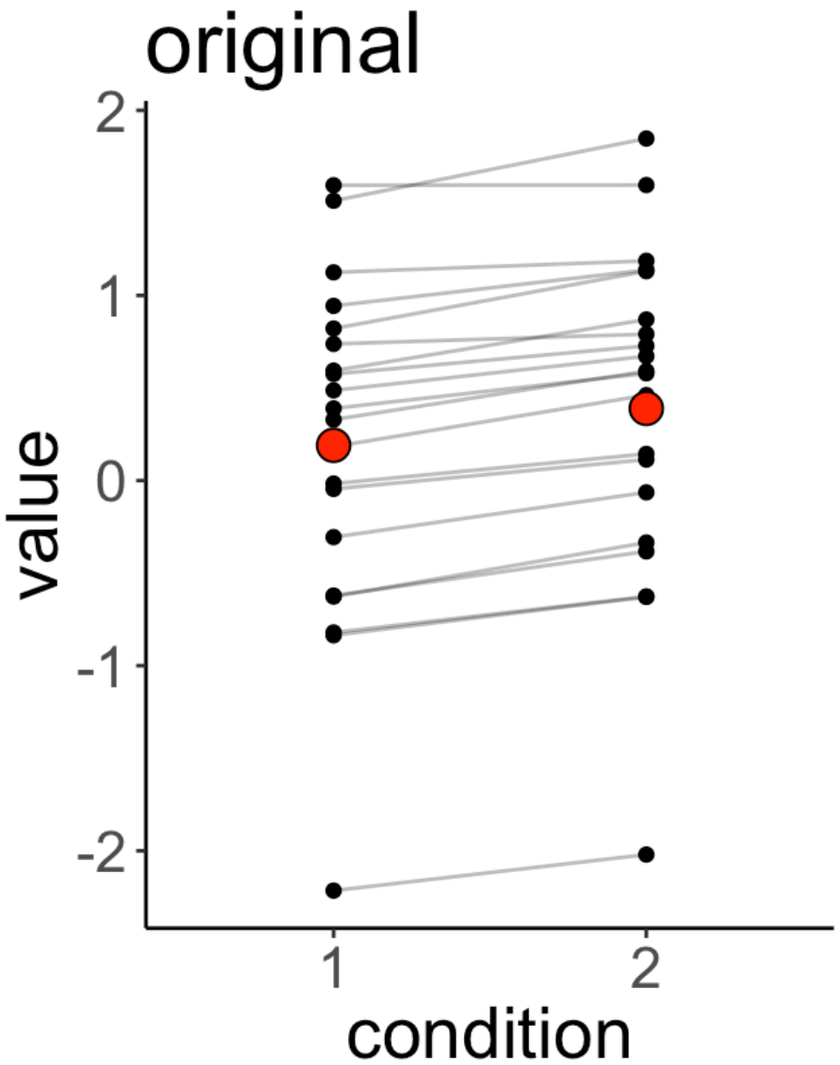
```r
# fit models
fit.compact = lmer(formula = value ~ 1 + (1 | participant),
                   data = df.original)

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

# compare via Chisq-test
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
```

Why Chisq?

Dependence

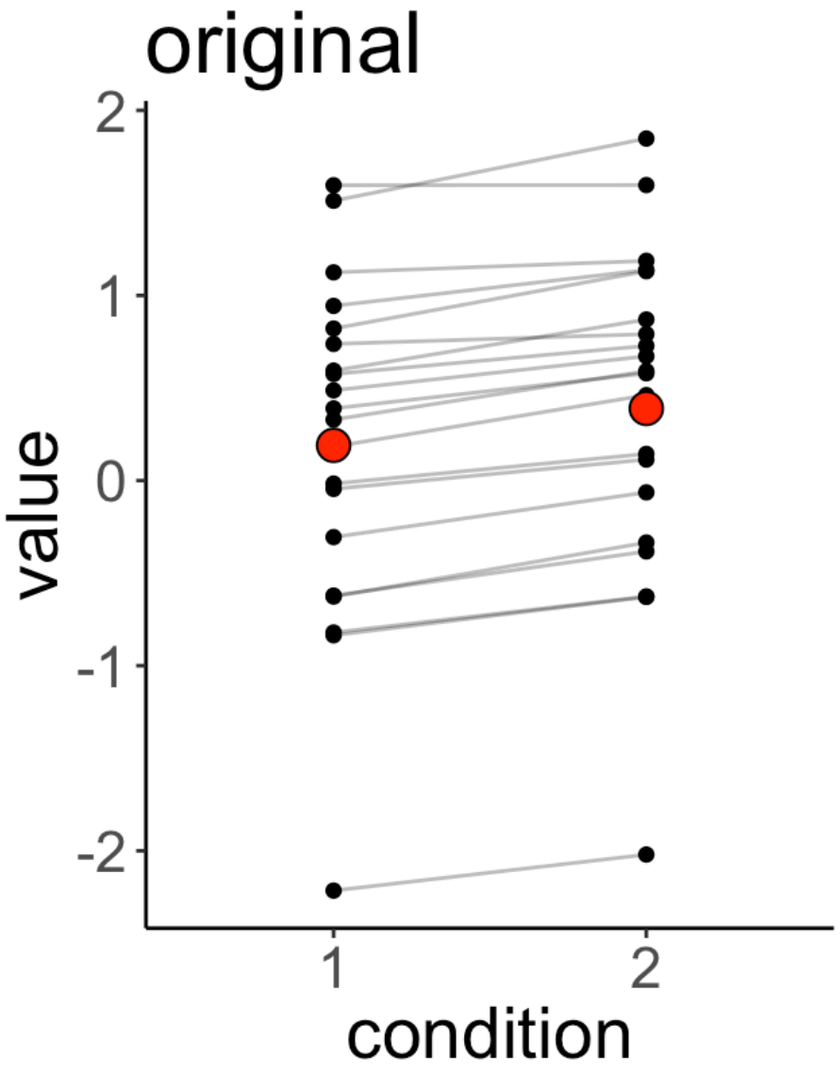
Why is the effect of condition significant when we account for the dependence in the data?



- there are large interindividual differences in the baseline
- 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?



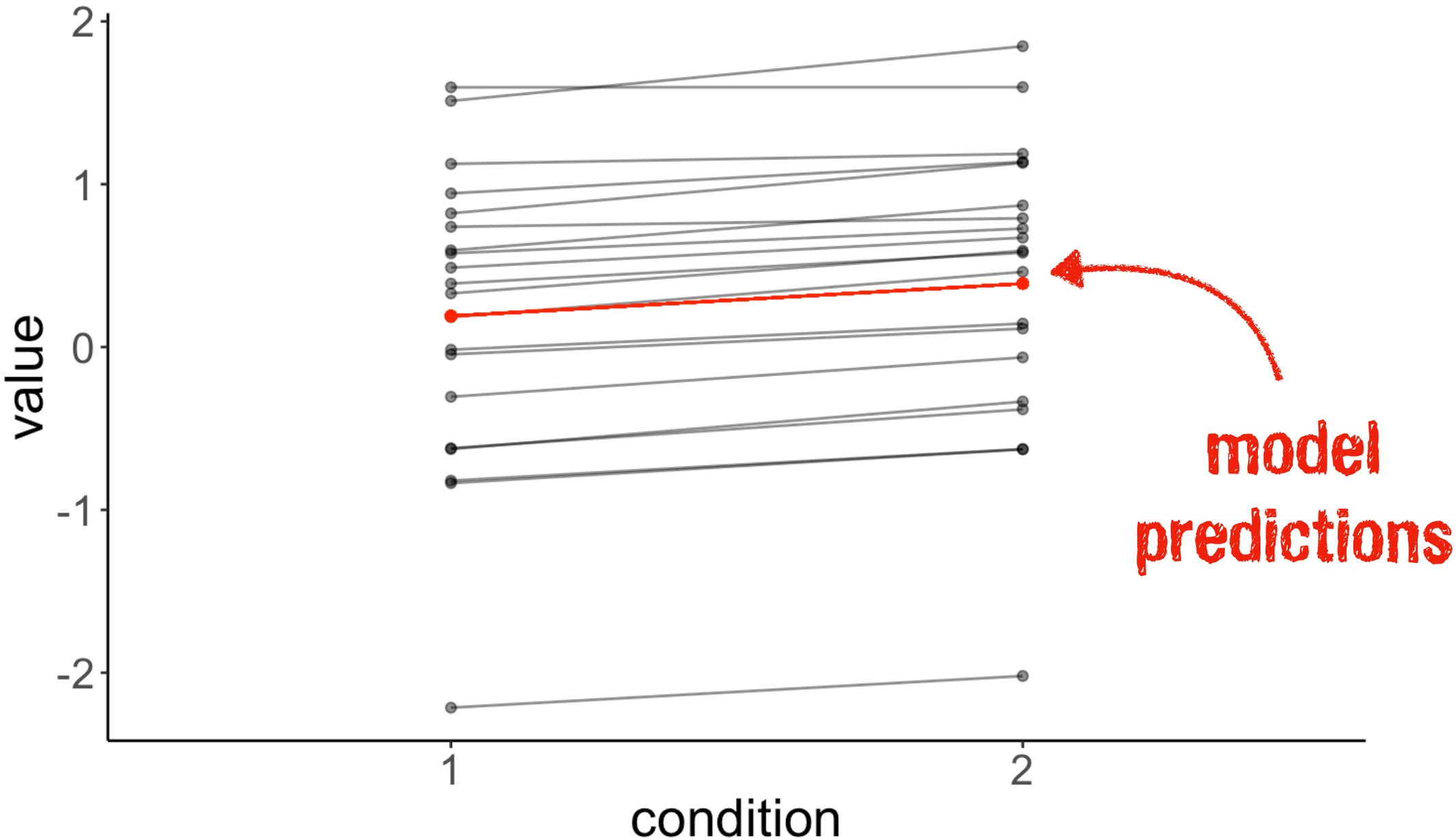
- by explicitly modeling the dependence in the data, we account for the interindividual differences

let's visualize the model predictions!

Linear model (assuming independence)

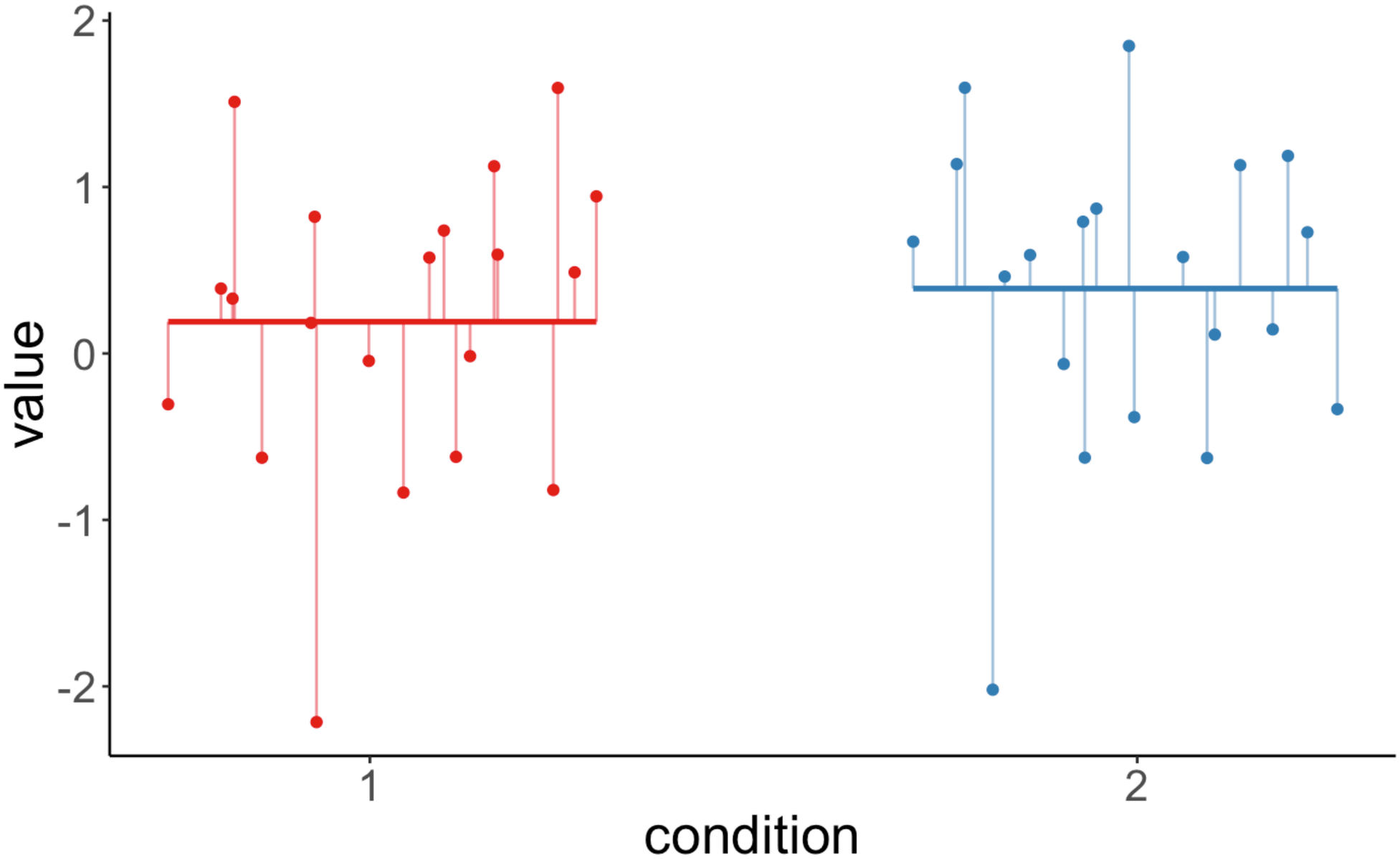
Predictions by the linear model which assumes independence

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



Linear model (assuming independence)

Residuals of the model

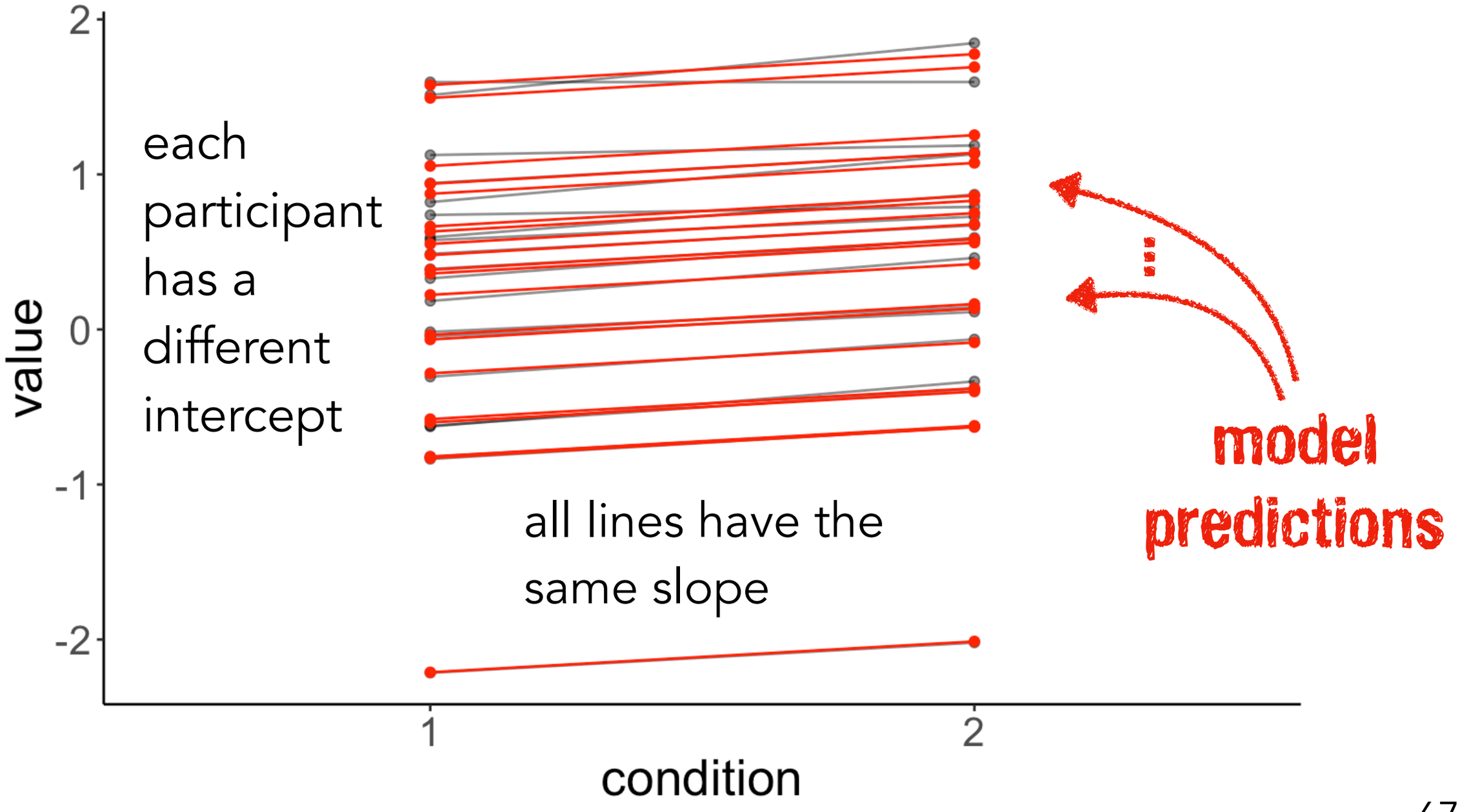


This is not much better than fitting a single line (point).

Linear mixed effects model (accounting for dependence)

Predictions by the linear mixed effects model

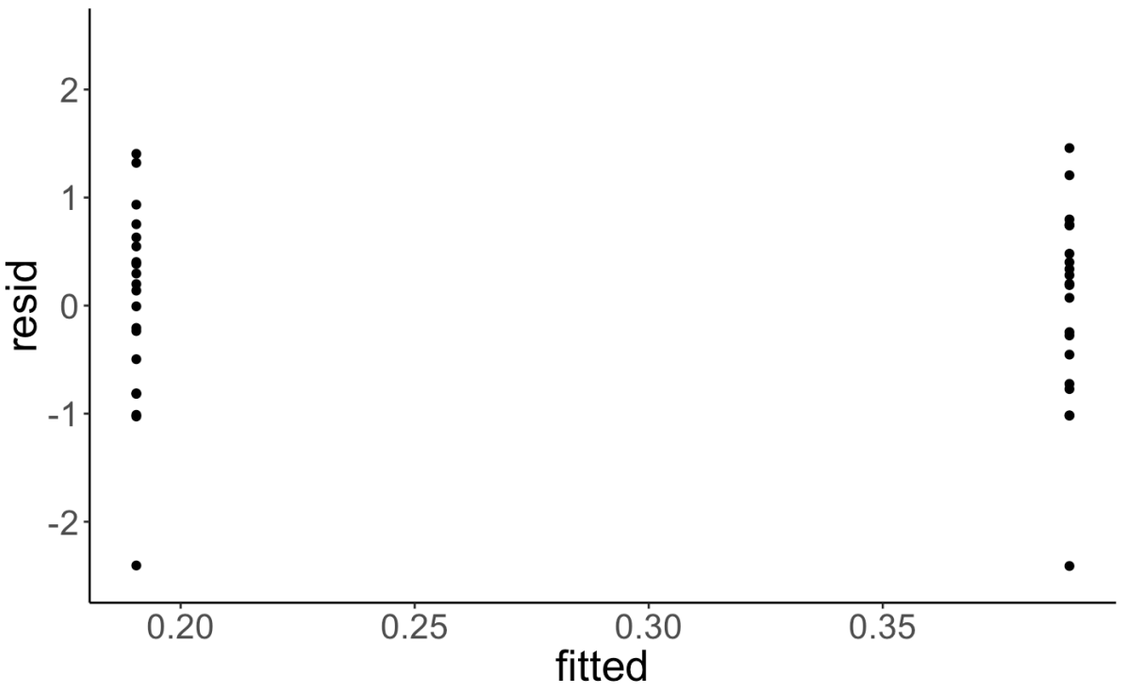
```
lmer(formula = value ~ condition + (1 | participant),
     data = df.original)
```



Model comparison

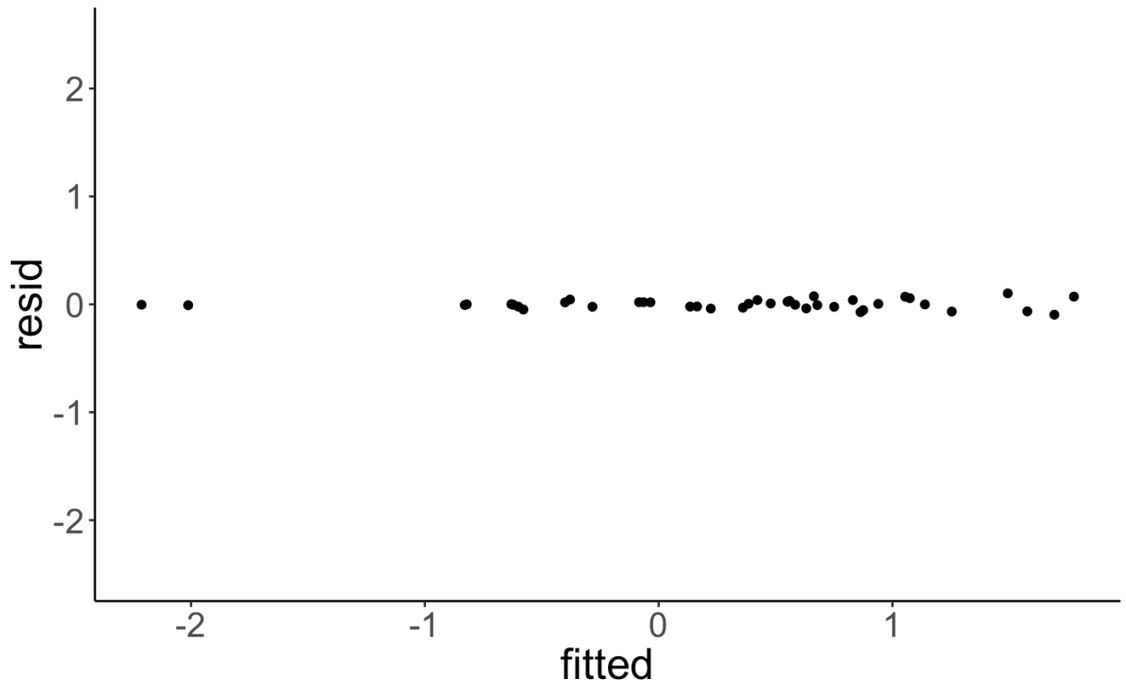
Residual plots

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



much variance left to be explained

```
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)
```

$$\text{value}_{i,j} = b_0 + b_1 \cdot \text{condition}_{i,j} + U_i + e_i$$

i = participant,

j = time point

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

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

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 -0.36399 -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 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

```
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
```

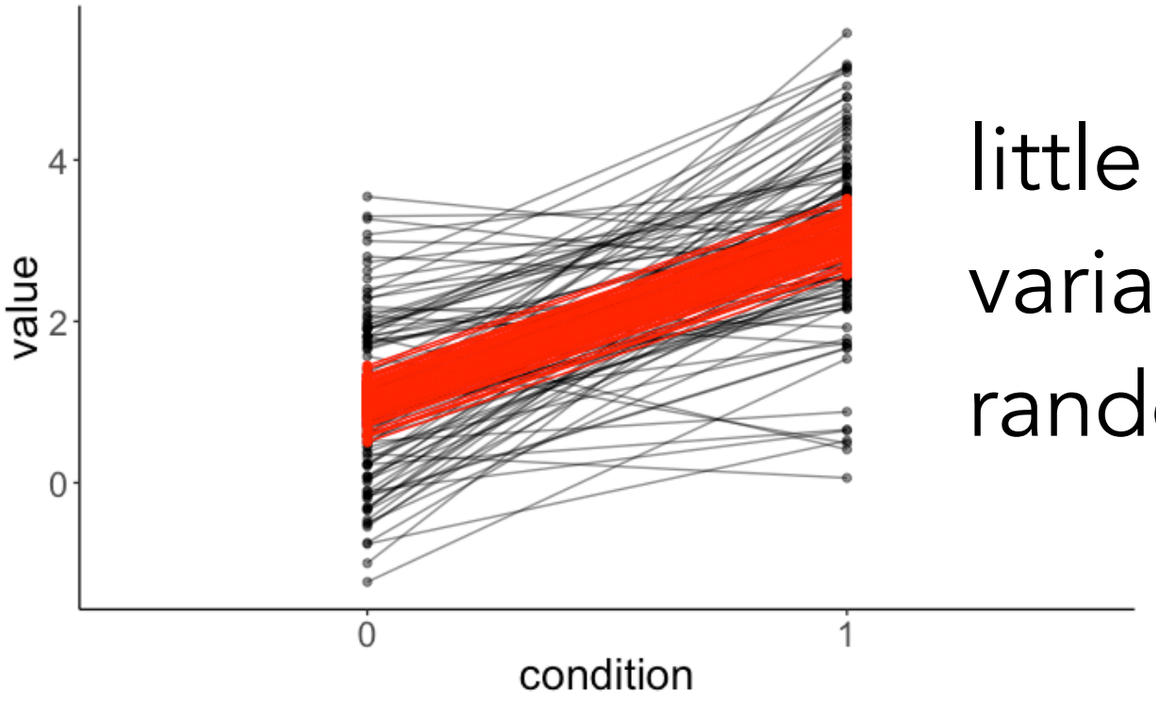
correlation between intercept and condition2

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 (as OP notes). It is in fact about the expected correlation of the regression coefficients.

Let's simulate an **lmer**()

```r
# parameters
sample_size = 100
b0 = 1
b1 = 2
sd_residual = 1
sd_participant = 0.5

# randomly draw intercepts for each participant
intercepts = rnorm(sample_size, sd = sd_participant)

# generate the data
df.test = tibble(
  condition = rep(0:1, each = sample_size),
  participant = rep(1:sample_size, 2)) %>%
  group_by(condition) %>%
  mutate(value = b0 + b1 * condition + intercepts + rnorm(n(), sd = sd_residual))

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

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

REML criterion at convergence: 606

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.53710 -0.62295 -0.04364  0.67035  2.19899

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.1607   0.4009
 Residual                1.0427   1.0211
Number of obs: 200, groups:  participant, 100

Fixed effects:
            Estimate Std. Error t value
(Intercept)   1.0166     0.1097   9.267
condition1    2.0675     0.1444  14.317

Correlation of Fixed Effects:
           (Intr)
condition1 -0.658
```



little interindividual variation explained by random intercepts ...

Model comparison

Hypothesis test

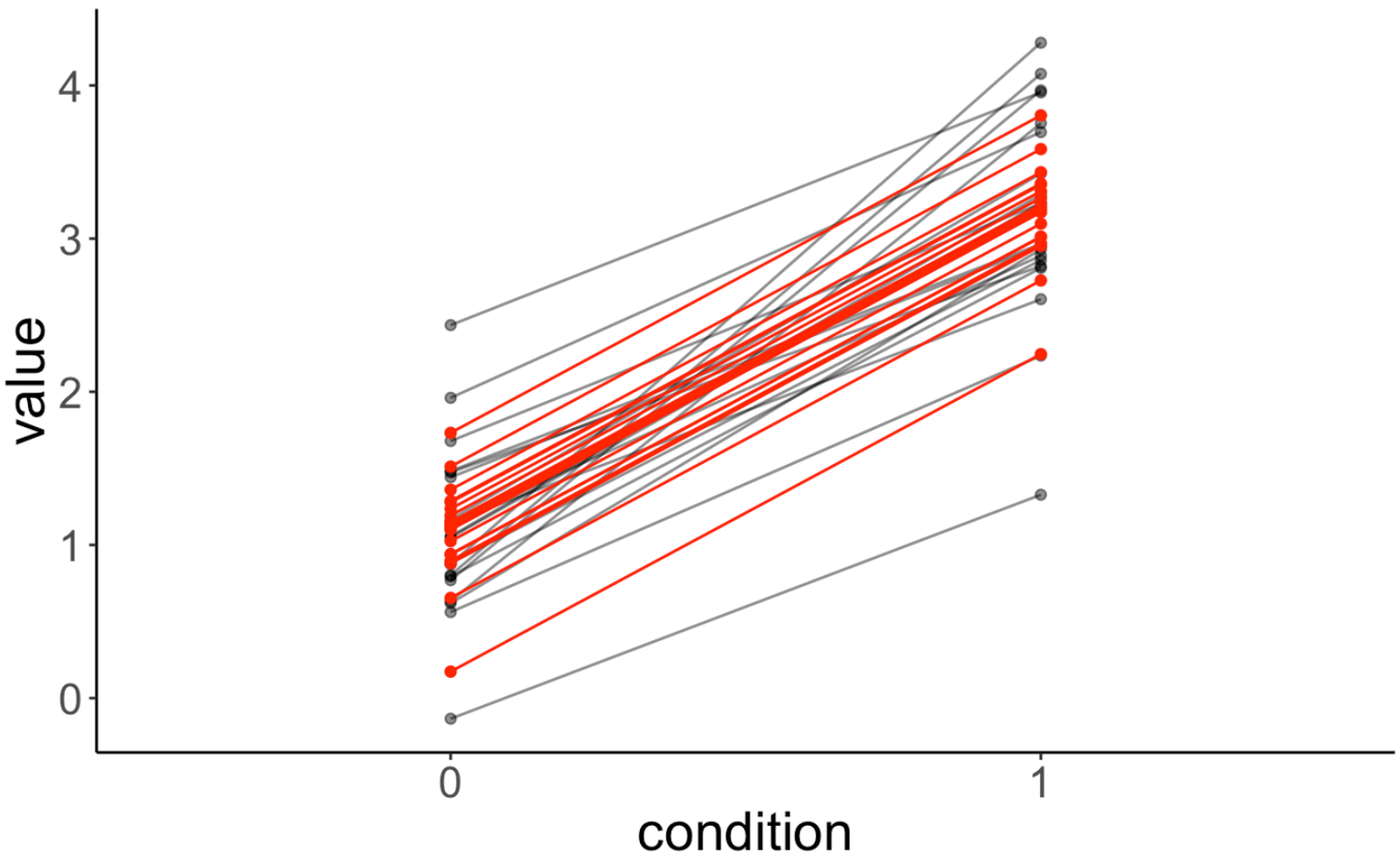
Is taking into account individual differences worth it?

```
# fit models (without and with dependence)
fit.compact = lm(formula = value ~ 1 + condition,
                 data = df.test)

fit.augmented = lmer(formula = value ~ 1 + condition + (1 | participant),
                     data = df.test)

# compare models
# note: the lmer model has to be supplied first
anova(fit.augmented, fit.compact)
```

```
refitting model(s) with ML (instead of REML)
Data: df.test
Models:
fit.lm: value ~ 1 + condition
fit.test: value ~ 1 + condition + (1 | participant)
          Df   AIC    BIC logLik deviance  Chisq Chi Df Pr(>Chisq)
fit.lm     3 608.6 618.49 -301.3    602.6
fit.test   4 608.8 621.99 -300.4    600.8 1.7999      1     0.1797
```

not worth it

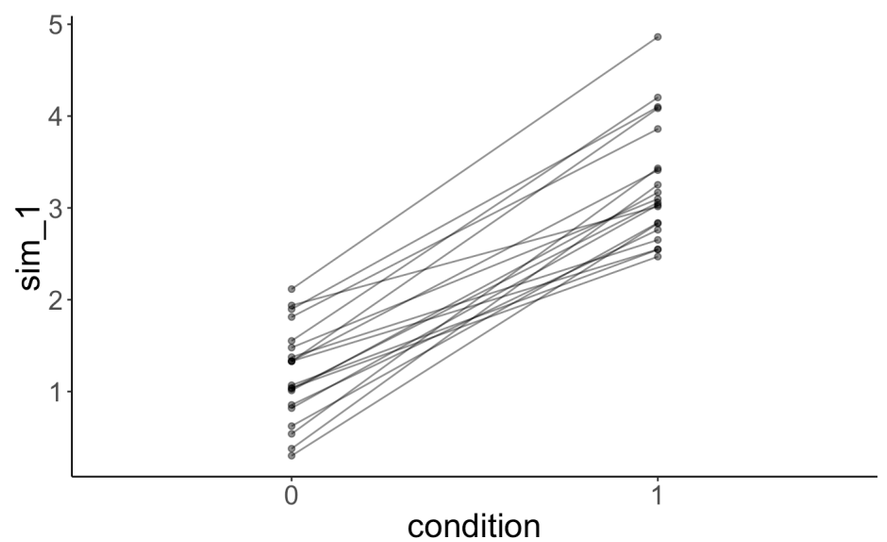
Outliers



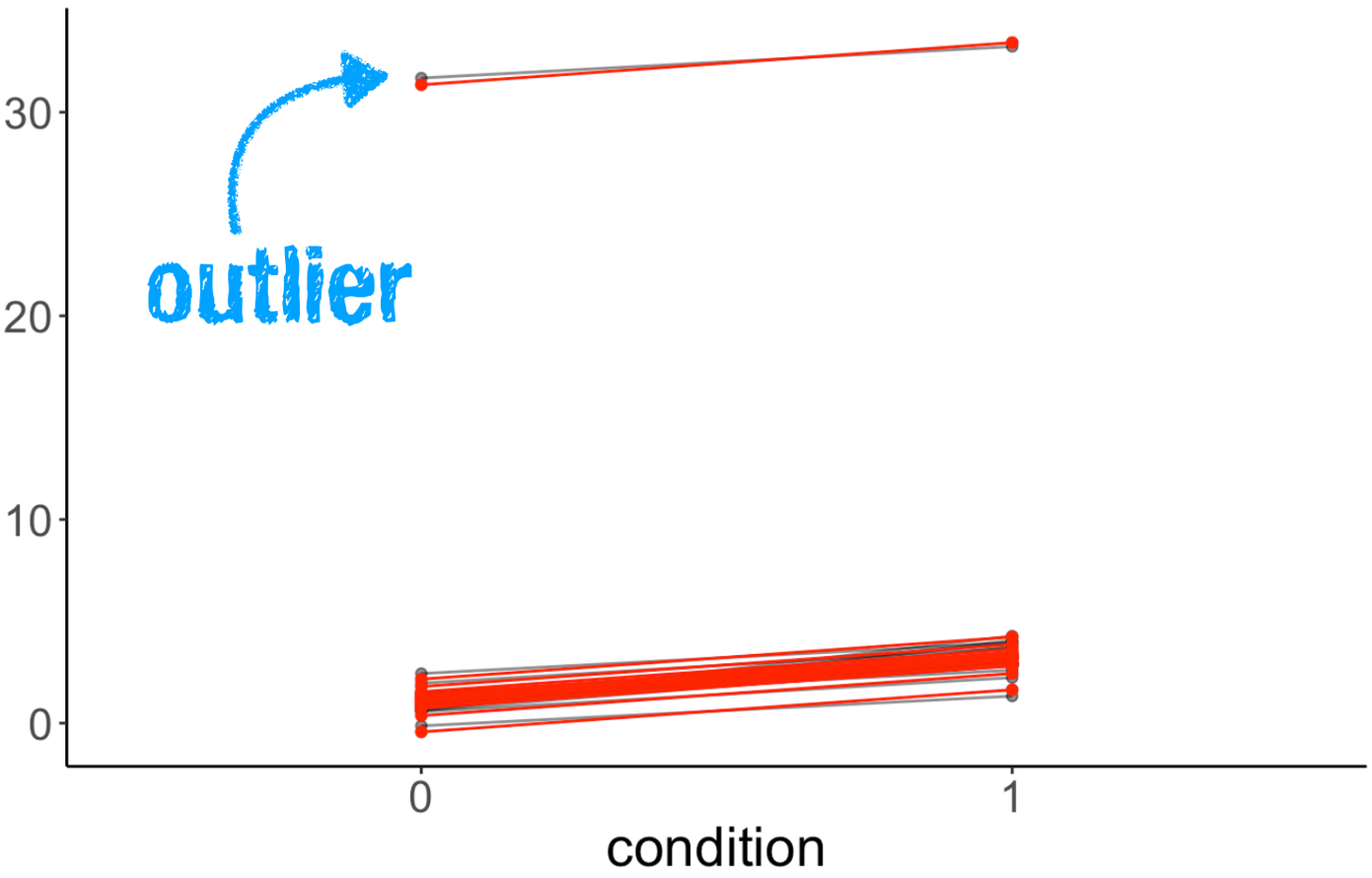

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + condition + (1 | participant)
   Data: df.test

REML criterion at convergence: 74.9

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.9268 -0.5412 -0.1103  0.4868  1.7747

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.1702   0.4125
 Residual                0.2270   0.4764
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)   1.0920     0.1409    7.75
condition1    2.0726     0.1507   13.76

Correlation of Fixed Effects:
           (Intr)
condition1 -0.535
```

```r
# fit model
fit.test = lmer(formula = value ~ 1 + condition + (1 | participant),
                data = df.test)

# simulate data
fit.test %>%
  simulate()
```

simulated data



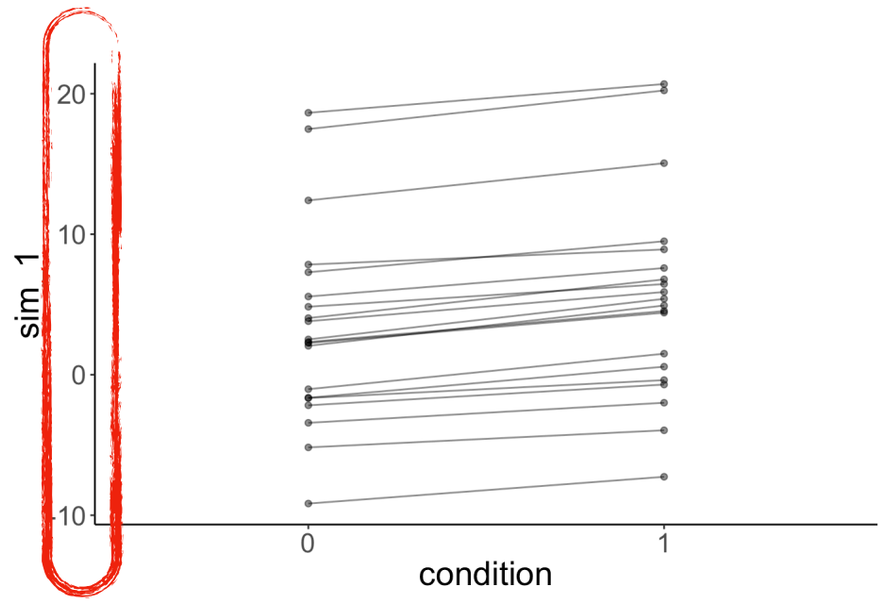
Outliers




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

REML criterion at convergence: 171.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.4038 -0.4678 -0.0094  0.5800  1.3930

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 46.198   6.7969
 Residual                 0.227   0.4764
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)   2.5920     1.5236   1.701
condition1    2.0726     0.1507  13.758

Correlation of Fixed Effects:
           (Intr)
condition1 -0.049
```

```r
# fit model
fit.test = lmer(formula = value ~ 1 + condition + (1 | participant),
                data = df.test)

# simulate data
fit.test %>%
  simulate()
```

simulated data



general points about **`lmer()`**

- **fixed effects**:
  - parameters are estimated
  - often: factors that we manipulate experimentally
- **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

general points about **`lmer`**()

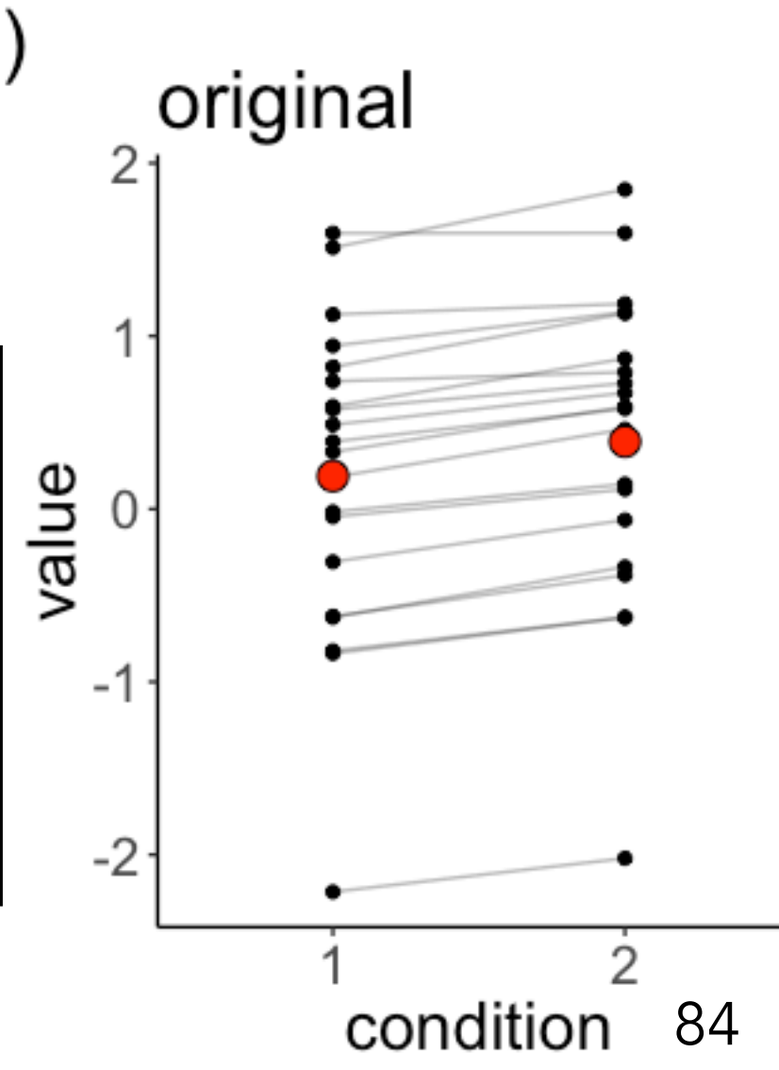
- Why don't we just run individual regressions?
  - overfitting ...
  - inflating type 1 error
  - larger uncertainty in parameter estimates because only few data points are used
- Why don't we just run a regression on the means?
  - we throw away a lot of information
- Mixed effects models:
  - make use of all available information
  - more next time ...

we just performed a paired t-test ...

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

```
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
```



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)
```

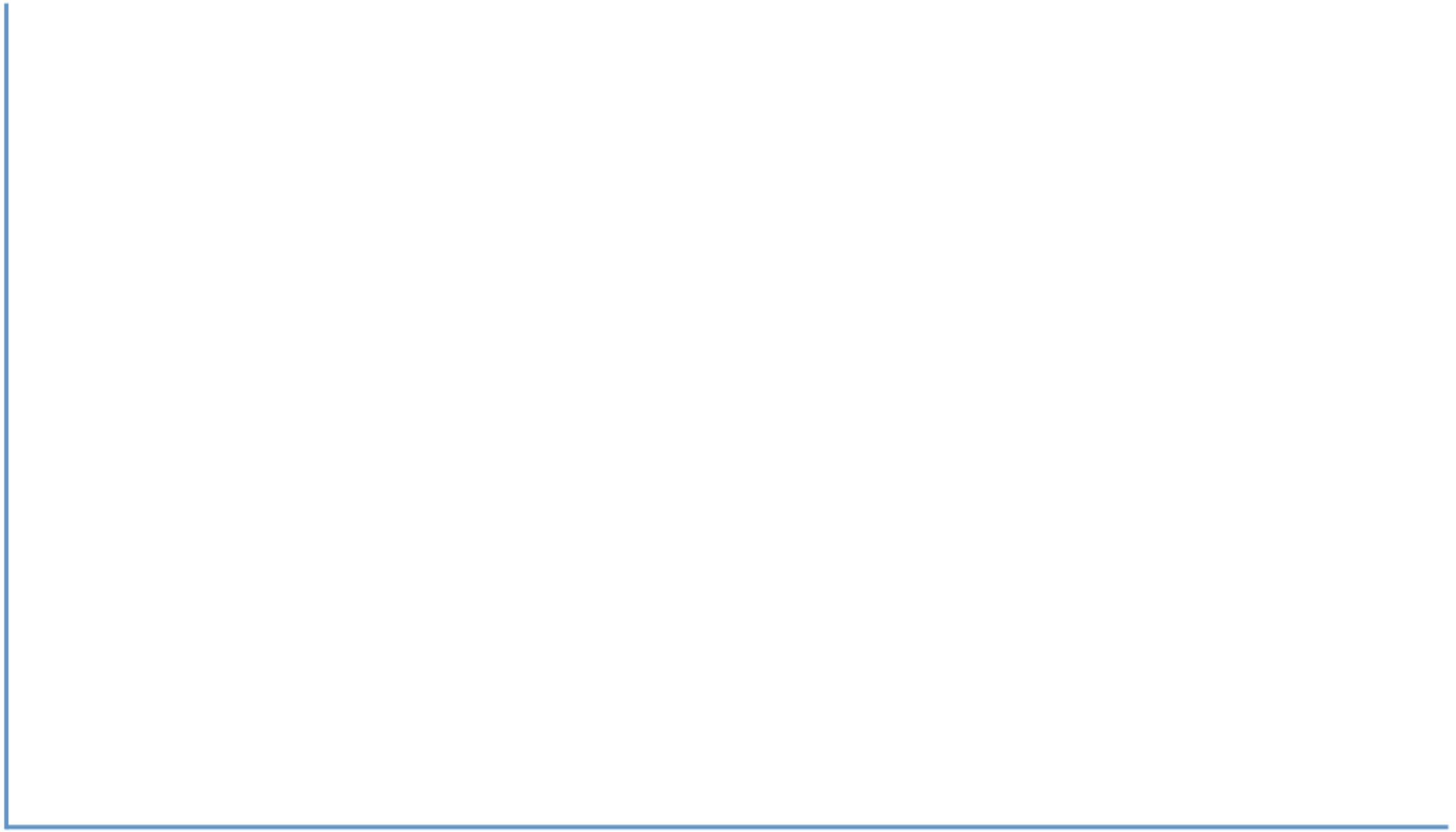
```
Paired t-test

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
```

Summary

- Model comparison
  - Cross-validation
    - different methods
    - the power of `library("purrr")`
  - AIC and BIC
    - trading off maximum likelihood and model complexity
    - least squares and maximum likelihood
- Linear mixed effects model
  - model dependence in data

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!