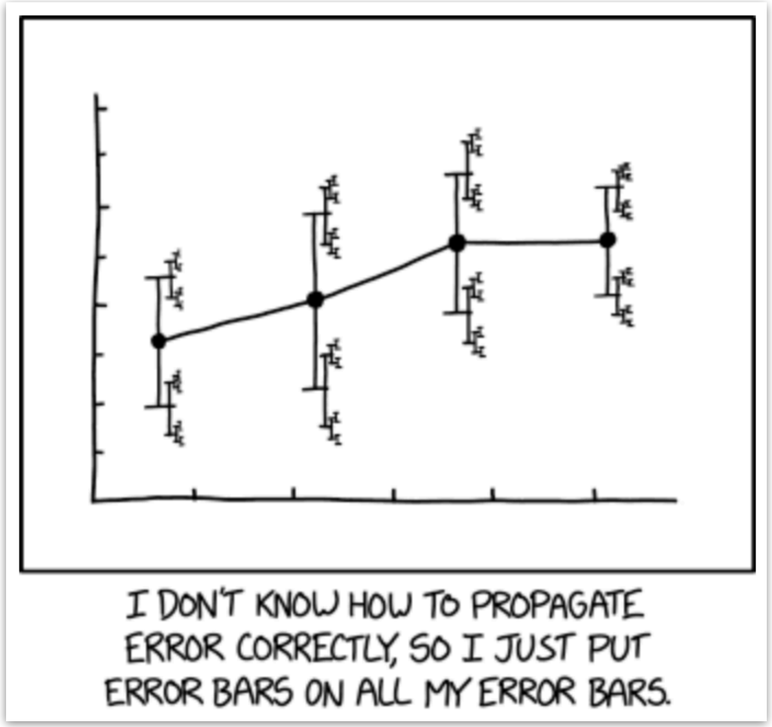
PSYCH 252: Statistical Methods for Behavioral and Social Sciences

Linear mixed effects models 1,2



02/21/2025

Logistics

Homework 5

available after class today

due Thursday, February 27th, 8pm

Power Analysis

Model Comparison

3.1 (3 points)

Since there are so many different models we could make with all those variables, we would like to use some measure to compare how well they fit the data. Some methods are AIC/BIC and cross-validation.

Create a data frame with 4 rows and the following 5 columns: `model_name`, `rsquared`, `logLik`, `AIC`, `BIC`. Each row should represent the following 4 models

1. **`model_med_age`**: mean_income ~ median_age,
2. **`model_med_age10`**: mean_income ~ median_age + median_age^2 + median_age^3 + ... + median_age^10
3. **`model_edu`**: mean_income ~ less9thgrade + grade9to12 + highschool + somecollege + assoc + bachelors + grad
4. **`model_race`**: mean_income ~ percent_white + percent_black + percent_amindian_alaskan + percent_asian + percent_nativeandother + percent_other_nativeandother + percent_hispanicorlatino + percent_race_other

To create the data frame, make sure to use `map()` for fitting the linear models, and you can use `glance()` from the `broom` package to get the model summaries in tidy format. For defining model 2. the `poly()` function is helpful.

Which model is the best model using the different measures? Are the different model comparison measures consistent?

```
### YOUR CODE HERE
df.compare =

####################
df.compare
```

YOUR ANSWER HERE

Plan for today

- Quick recap
- Linear Mixed Model
  - Accommodating non-idependence in data
  - Understanding lmer() syntax
  - A worked example
  - Reporting results
  - Simpsons Paradox

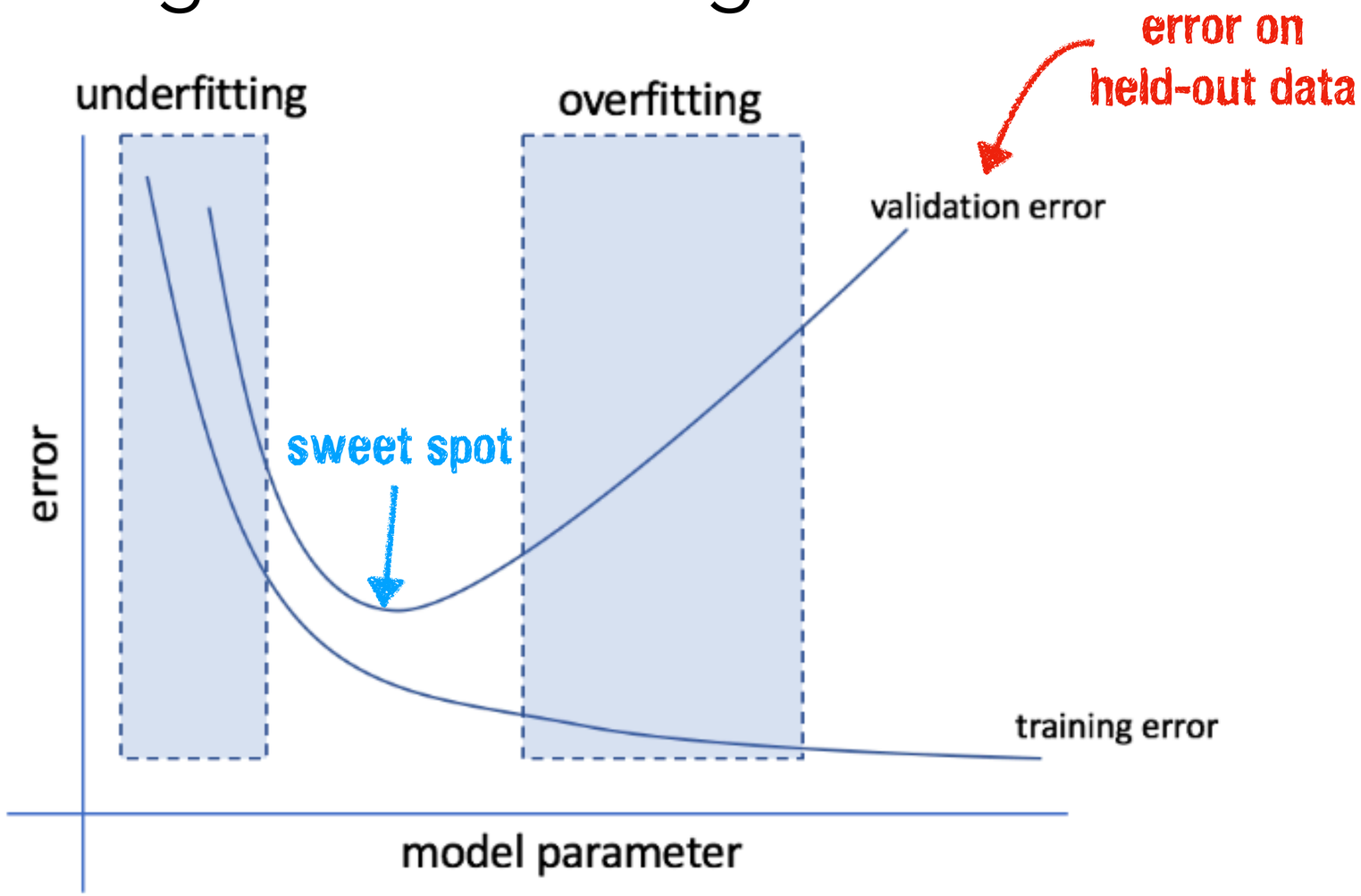
Quick recap

Quick recap: Model comparison

More complex models fit the data better.

We need to trade-off **model fit** and model **complexity**.

Underfitting vs. Overfitting



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

Quick recap: Model comparison

1. compare the proportional reduction in error using `anova()`
    - only works for nested models
2. cross-validation
    - works generally, but can take some time ...
    - different cross-validation procedures (LOO, k-fold, Monte Carlo)
3. AIC, BIC
    - works as long as we can compute the likelihood of the data
    - some discussion whether number of parameters is a good measure of model complexity
4. Bayesian model comparison

Quick recap: AIC and BIC

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

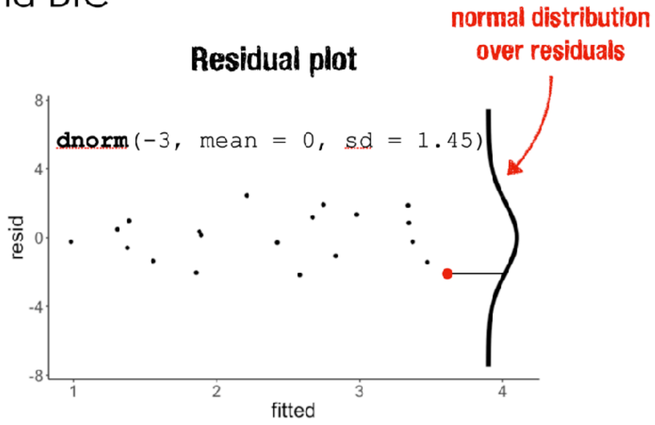
not that much Bayesian about it ...

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

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

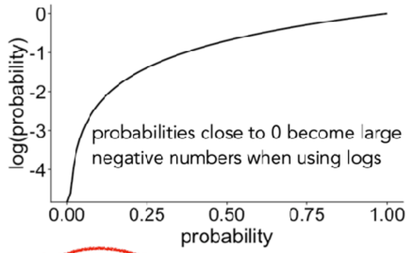
$\hat{L}$ = maximized value of the likelihood function of the model
k = number of parameters in the model
n = number of observations

AIC and BIC



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

`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

27

AIC and BIC

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

```
1  # generate some data
2  df.like = tibble(
3    x = runif(20, min = 0, max = 1),
4    y = 1 + 3 * x + rnorm(20, sd = 2)
5  )
6
7  # fit the model
8  fit = lm(formula = y ~ x,
9           data = df.like)
10
11 # model summary
12 fit %>%
13   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

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

28

Linear mixed effects models

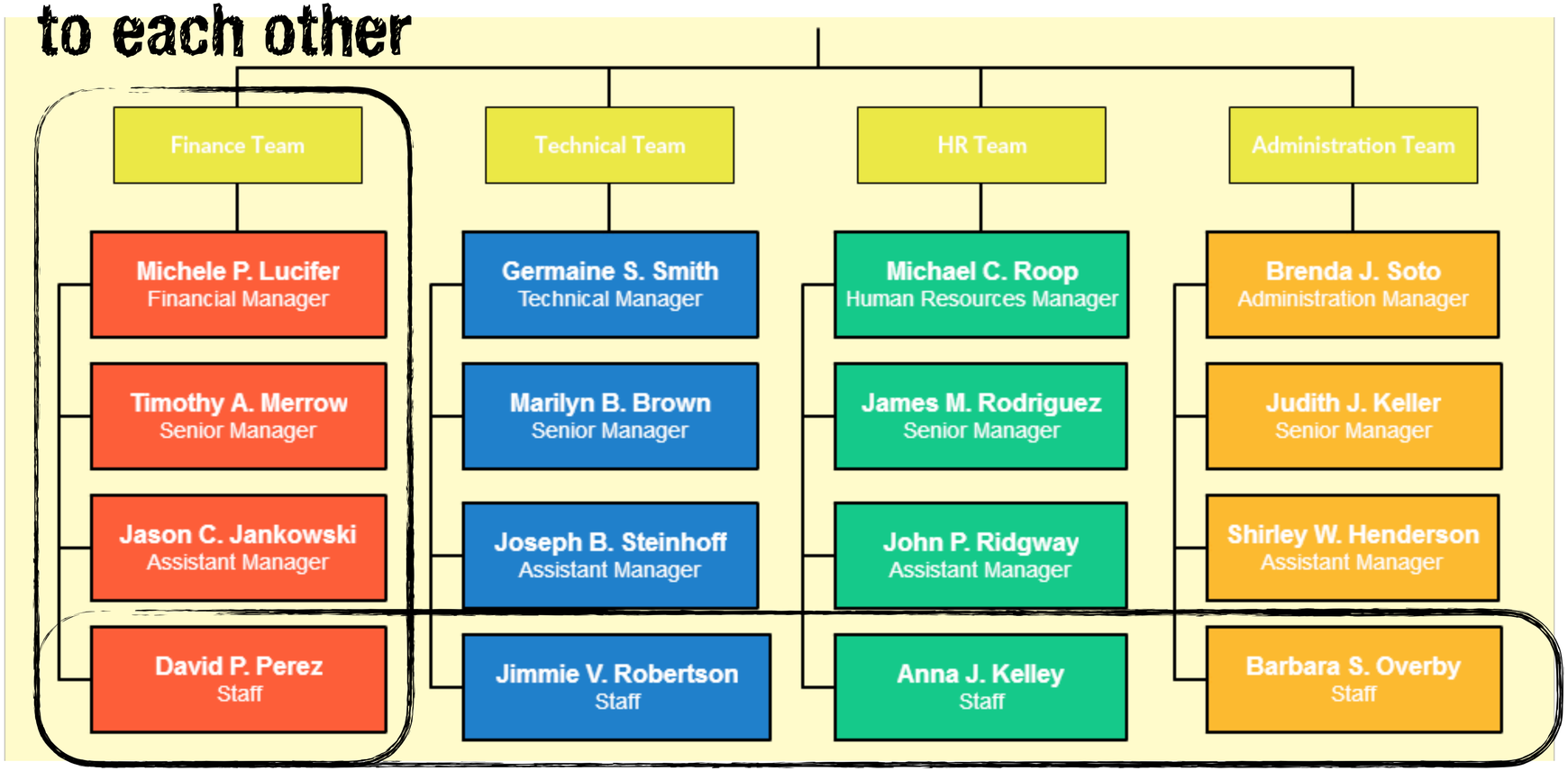
Dependence (vs iid)

- 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

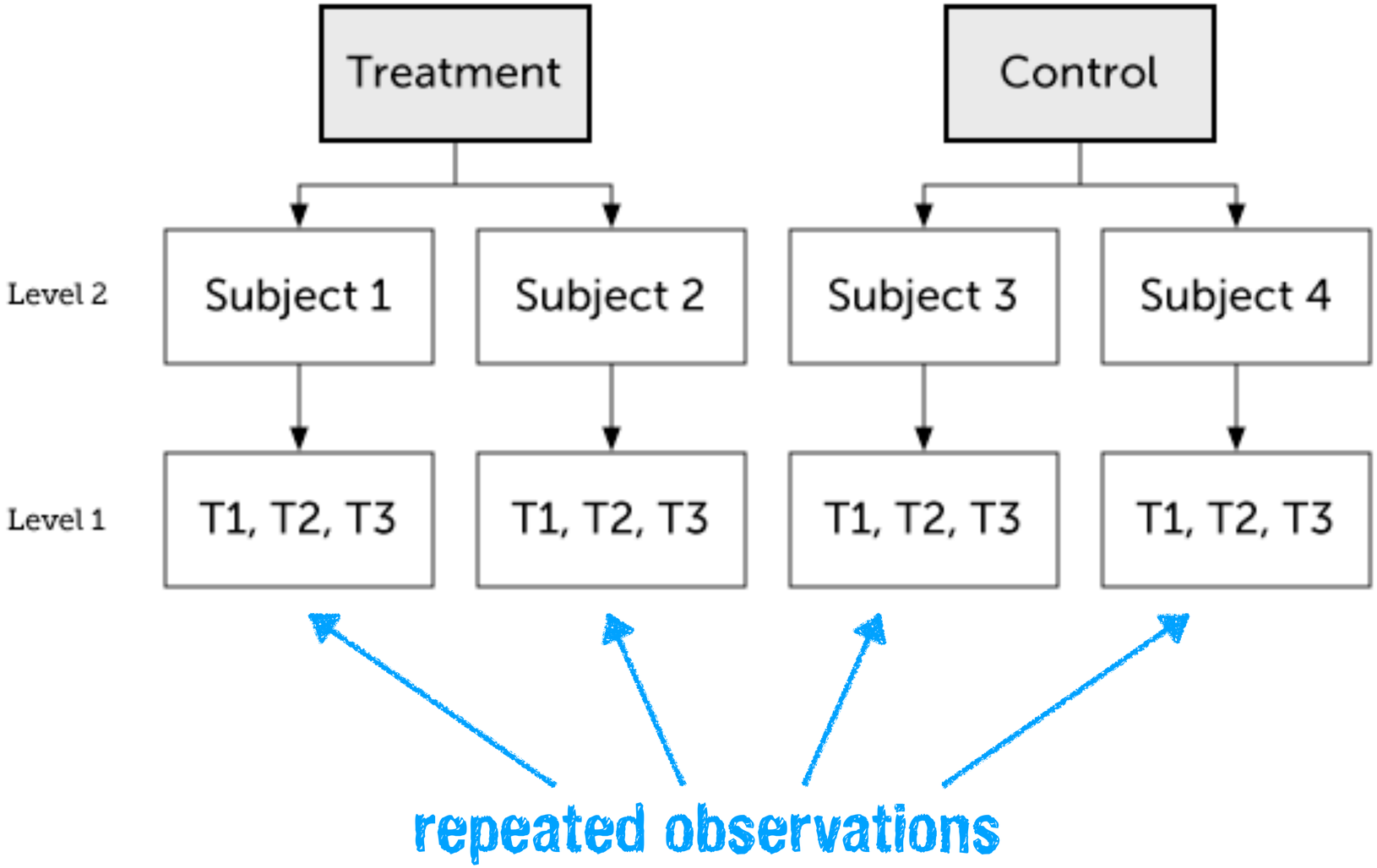
Main use cases: **Hierarchical models**

more similar to each other



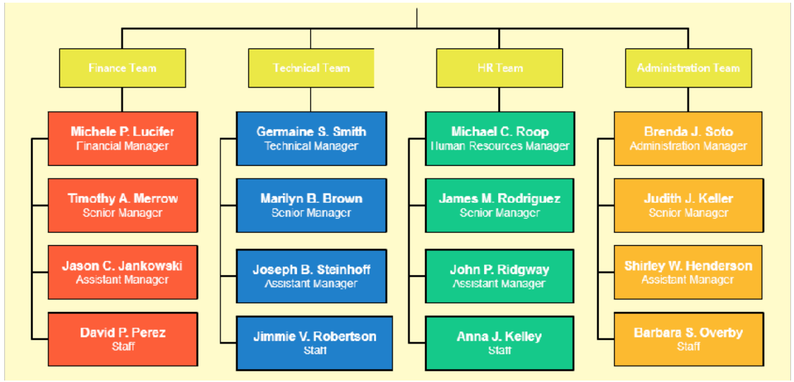
less similar to each other

Main use cases: **Longitudinal models**

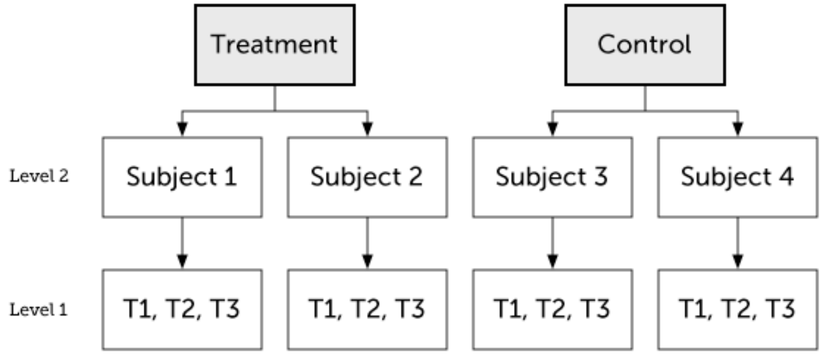


Linear mixed effects models

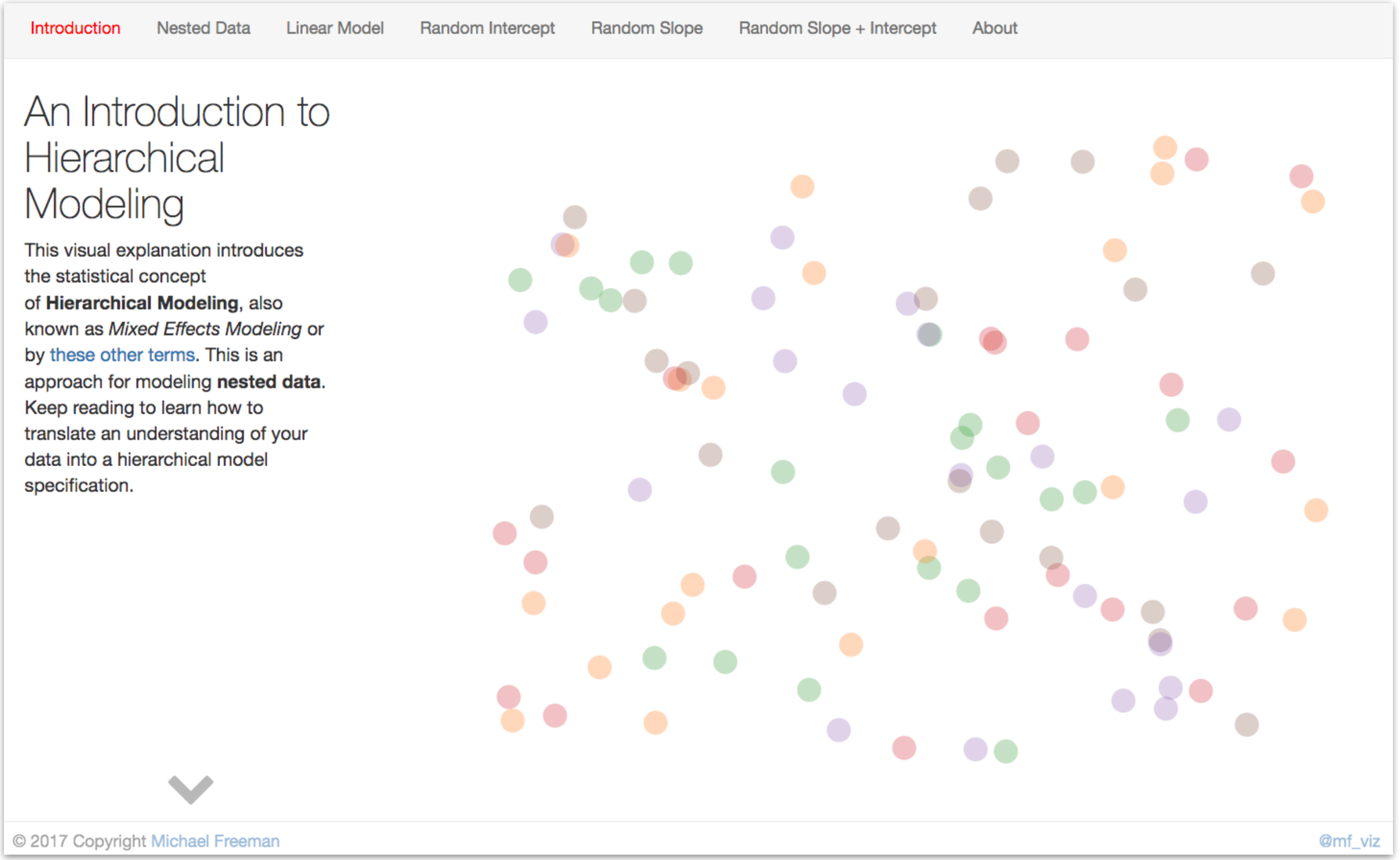
Hierarchical models



Longitudinal models



- allow us to account for dependencies in our data
- **hierarchical models**: schools > teachers > students
- **longitudinal models**: repeated observations from the same people



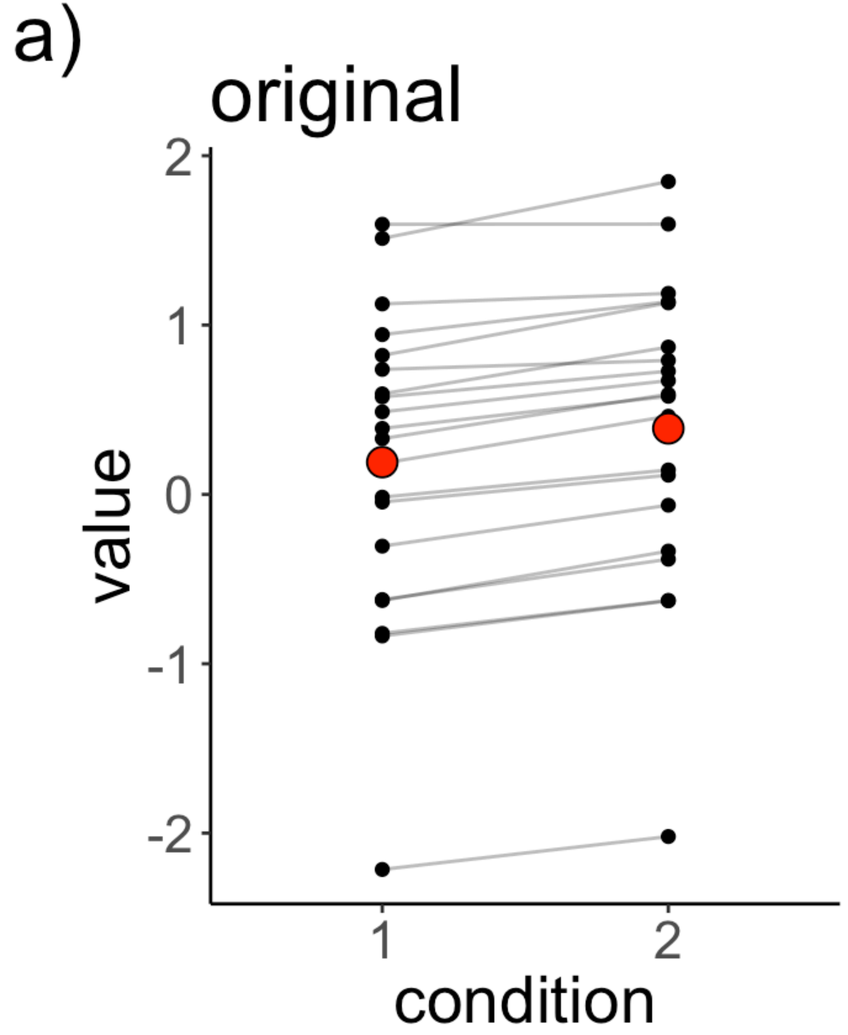
http://mfviz.com/hierarchical-models/

Modeling dependence in data

Dependence

Does it really matter?

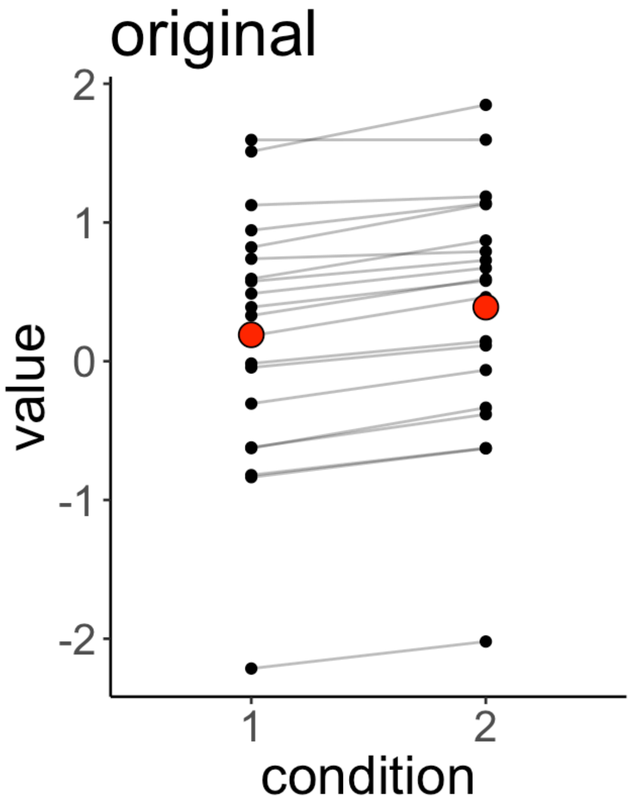
Is there a significant difference between conditions 1 and 2?



Dependence

assuming independence!

```r
# linear model
lm(formula = value ~ condition,
   data = df.original) %>%
  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
```

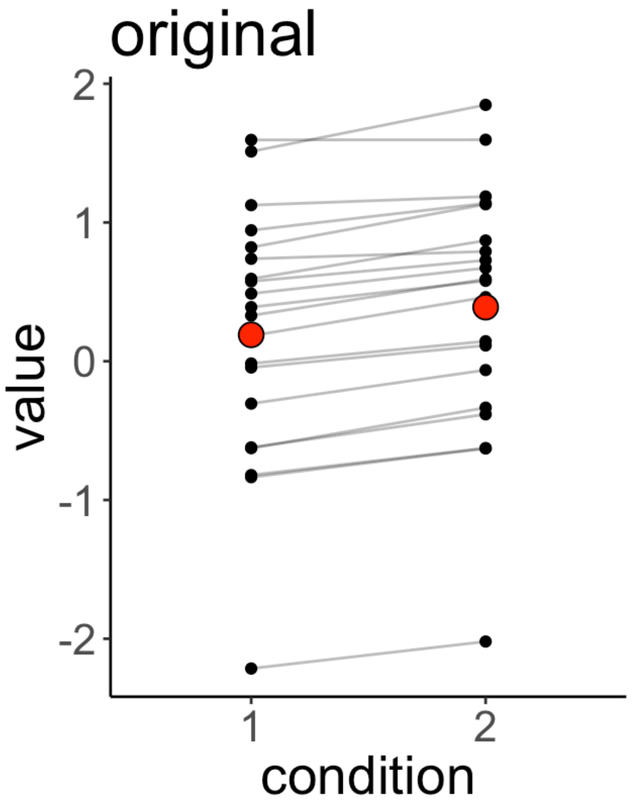
- we ignore the fact that we have repeated observations from the same participants
- in the data it looks like there is a small but consistent effect of condition

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

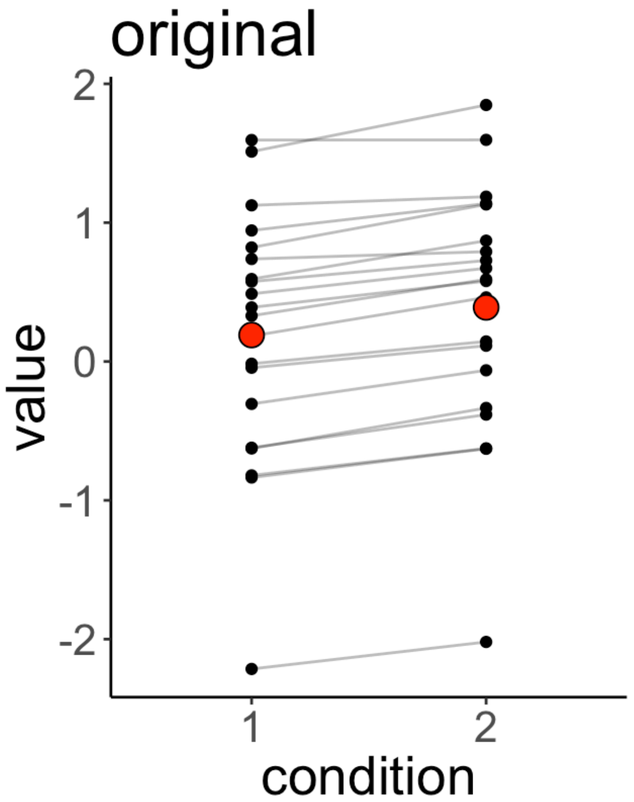
we can still do our good ol model comparison trick

```
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

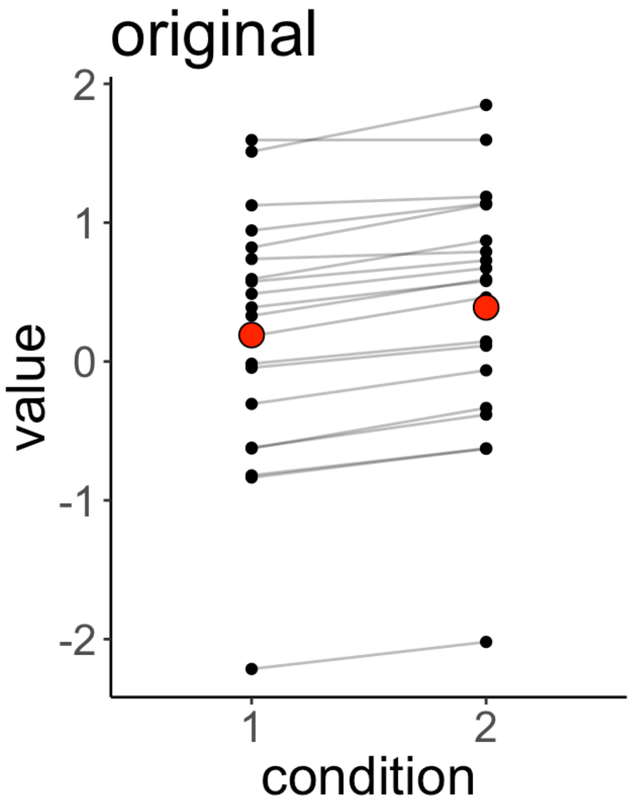
Why is the effect of condition significant when we account for the dependence in the data?



- there are large interindividual differences 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?



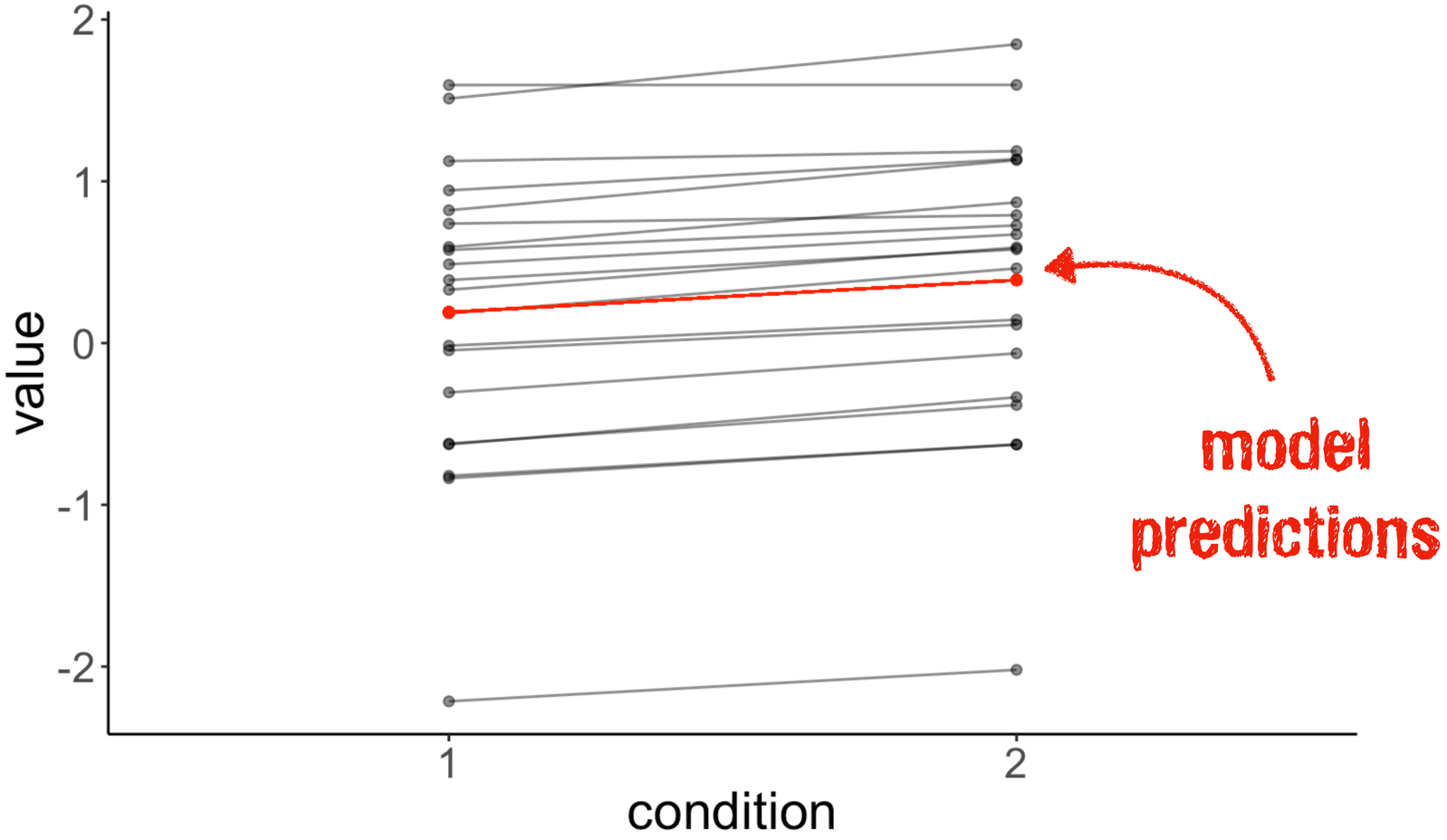
- by explicitly modeling the dependence in the data, we account for the interindividual differences

let's visualize the model predictions!

Linear model (assuming independence)

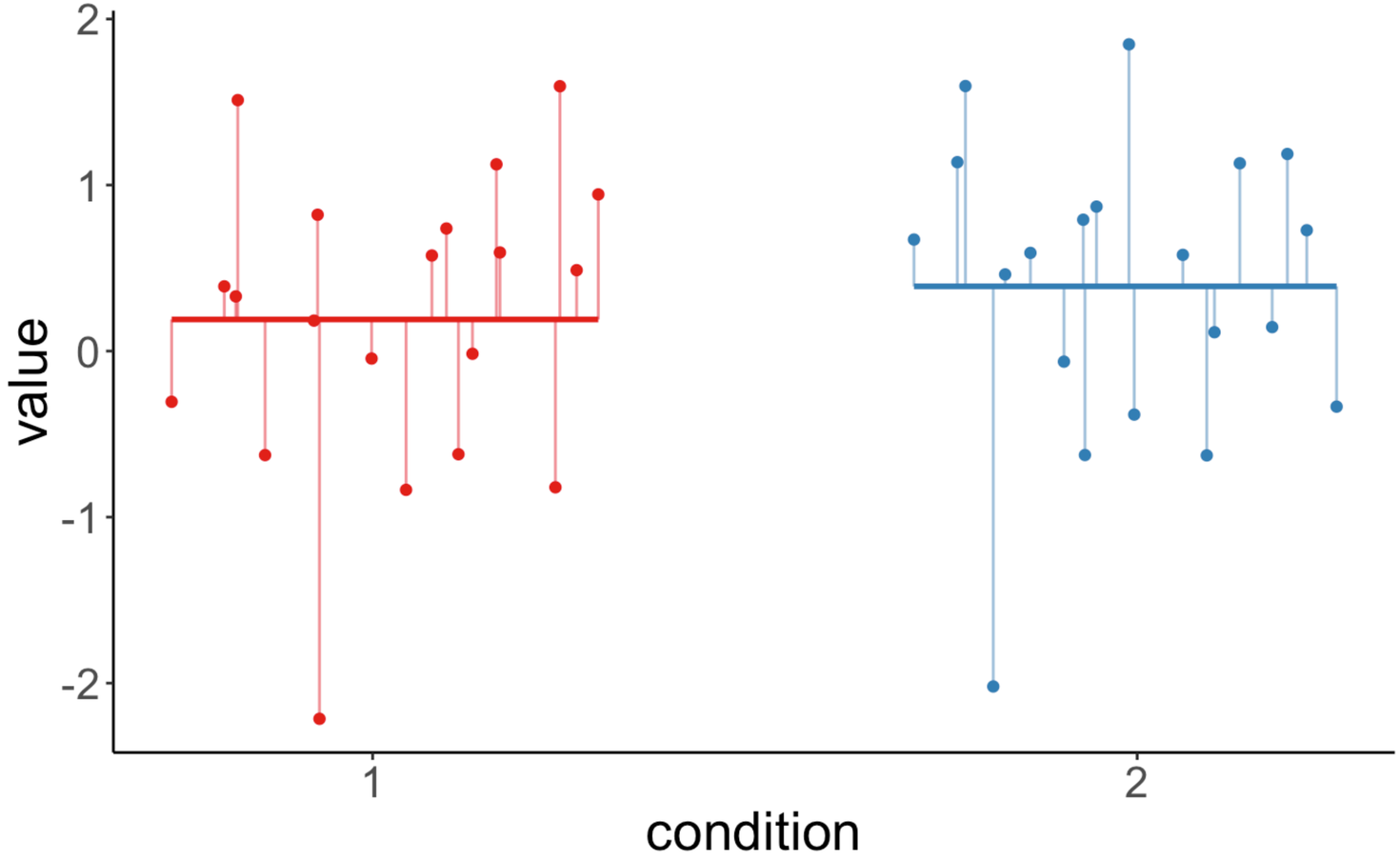
Predictions by the linear model which assumes independence

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



Linear model (assuming independence)

Residuals of the model

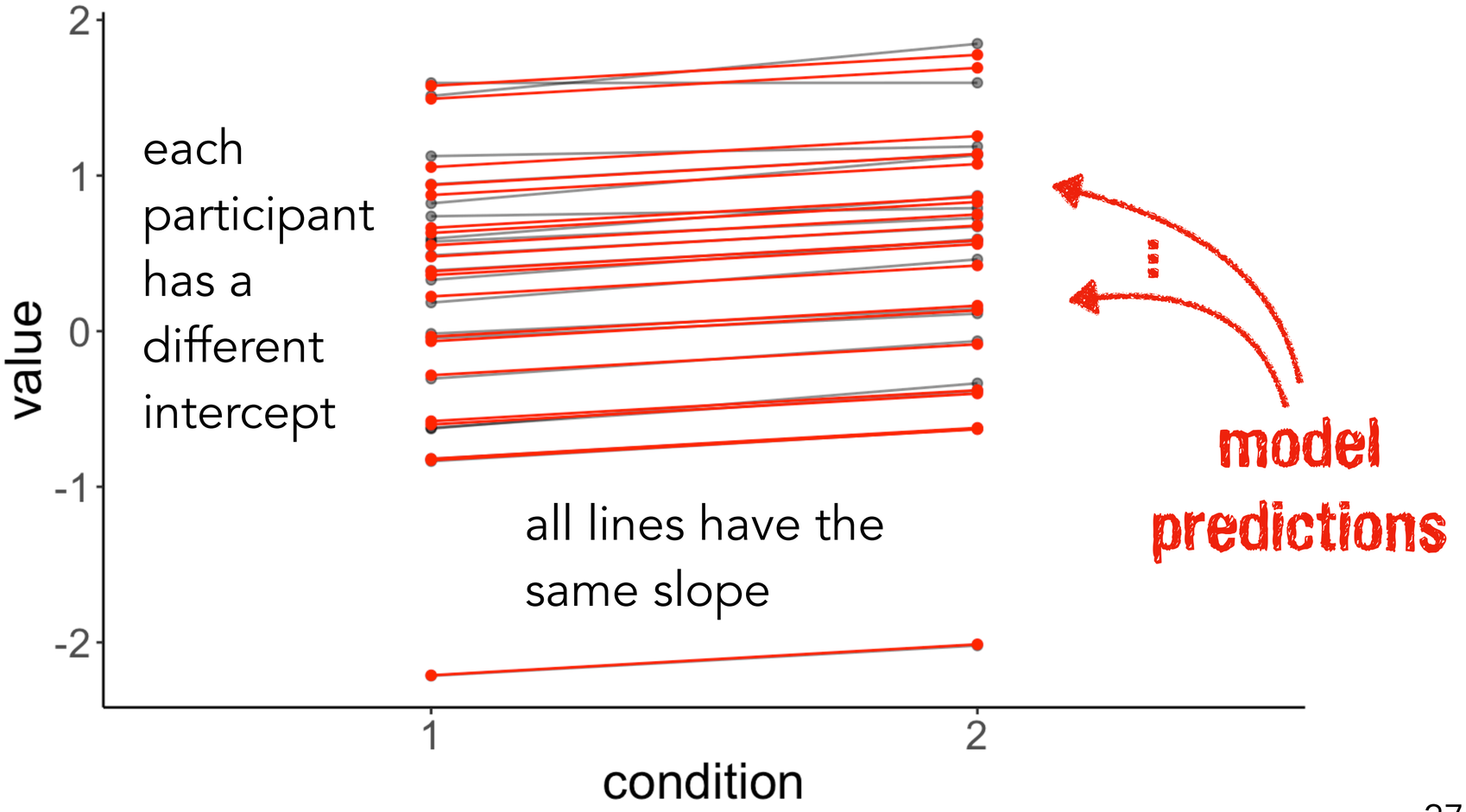


This is not much better than fitting a single line (point).

Linear mixed effects model (accounting for dependence)

Predictions by the linear mixed effects model

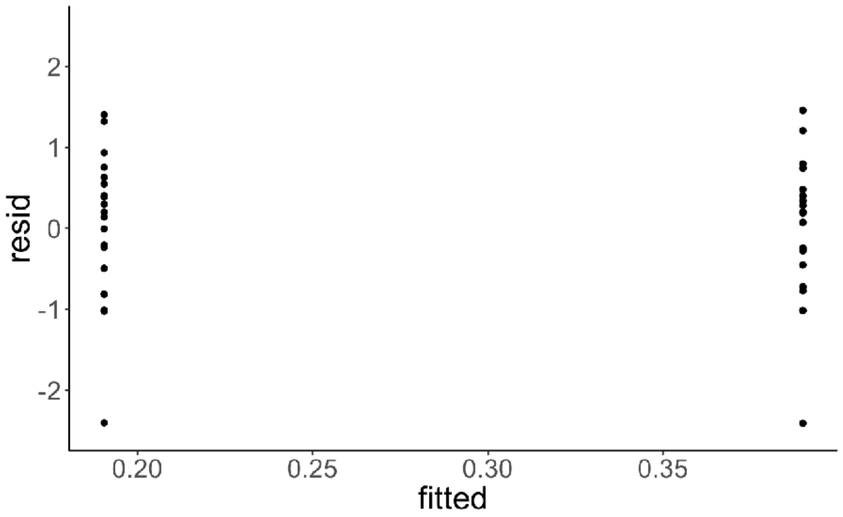
```
lmer(formula = value ~ 1 + condition + (1 | participant),
     data = df.original)
```



Model comparison

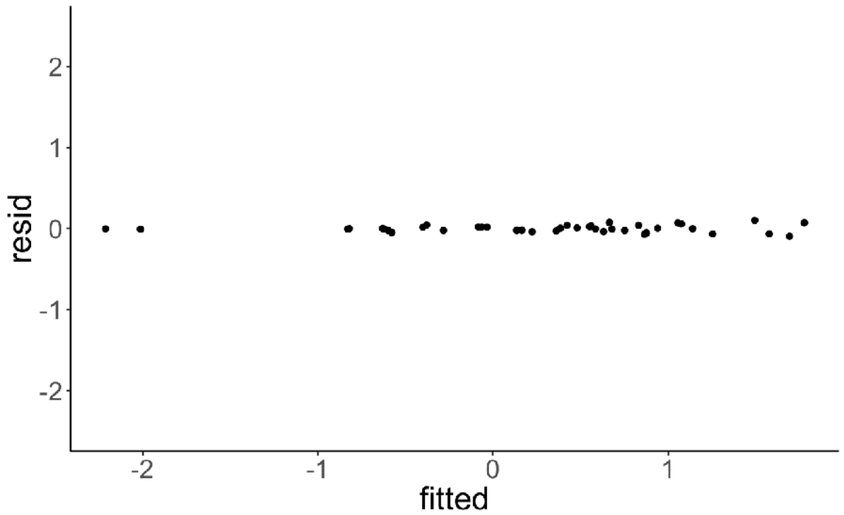
Residual plots

```
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?

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

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

# 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.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

Understanding the **lmer()** summary

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

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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 ~ 1 + condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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 ~ 1 + condition + (1 | participant),
3                     data = df.original) %>%
4   summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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 ~ 1 + condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

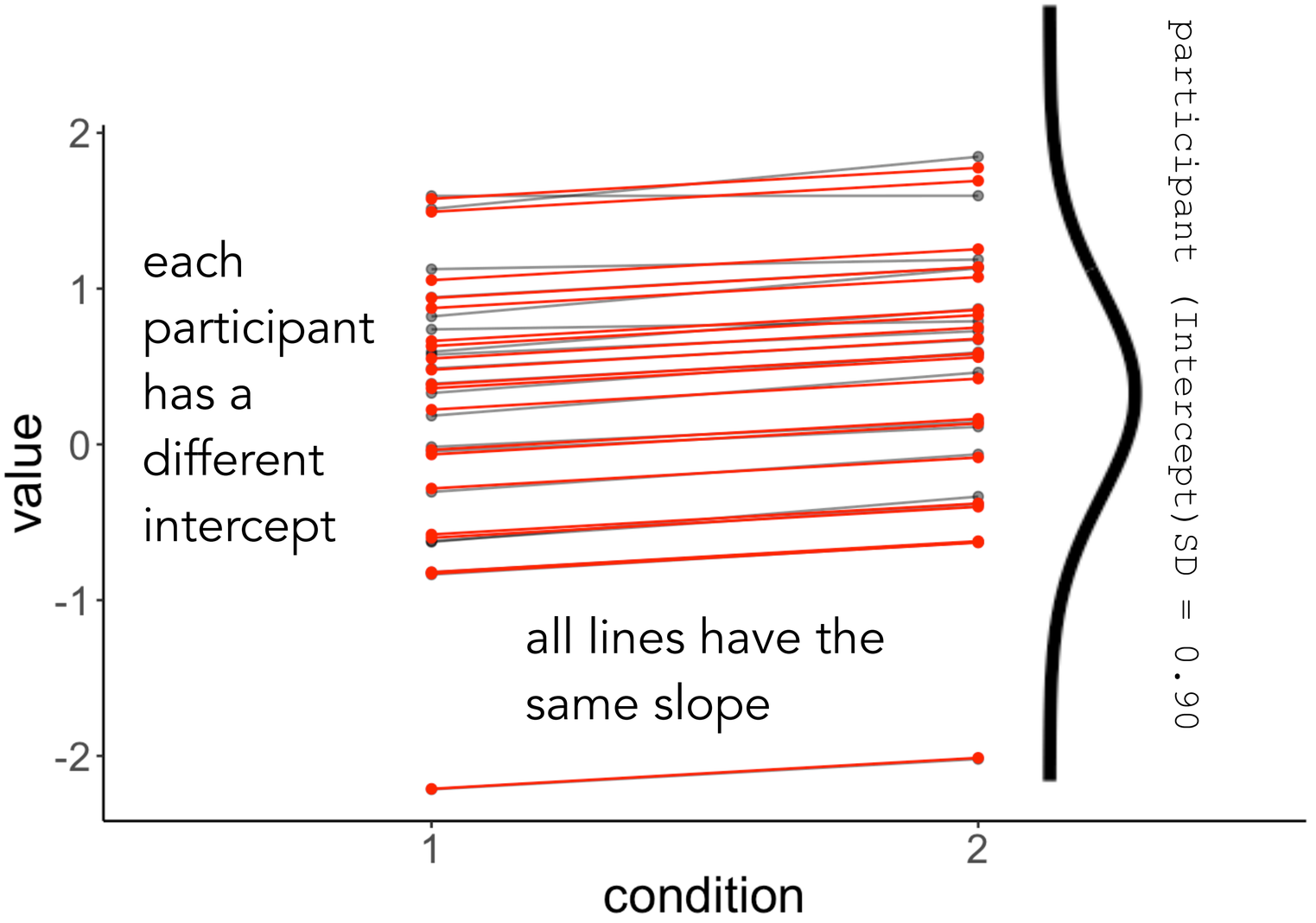
```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + condition + (1 | participant)
   Data: df.original

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
```

Random effects

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 ~ 1 + condition + (1 | participant),
3                    data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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
```

Fixed effects

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 ~ 1 + condition + (1 | participant),
3                     data = df.original) %>%
4    summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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

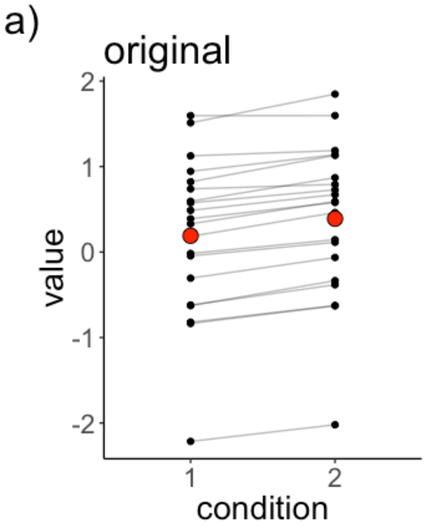
The "correlation of fixed effects" output doesn't have the intuitive meaning that most would ascribe to it. Specifically, is not about the correlation of the variables. It is in fact about the expected correlation of the regression coefficients.

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

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

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + 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
```

general points about `lmer()`

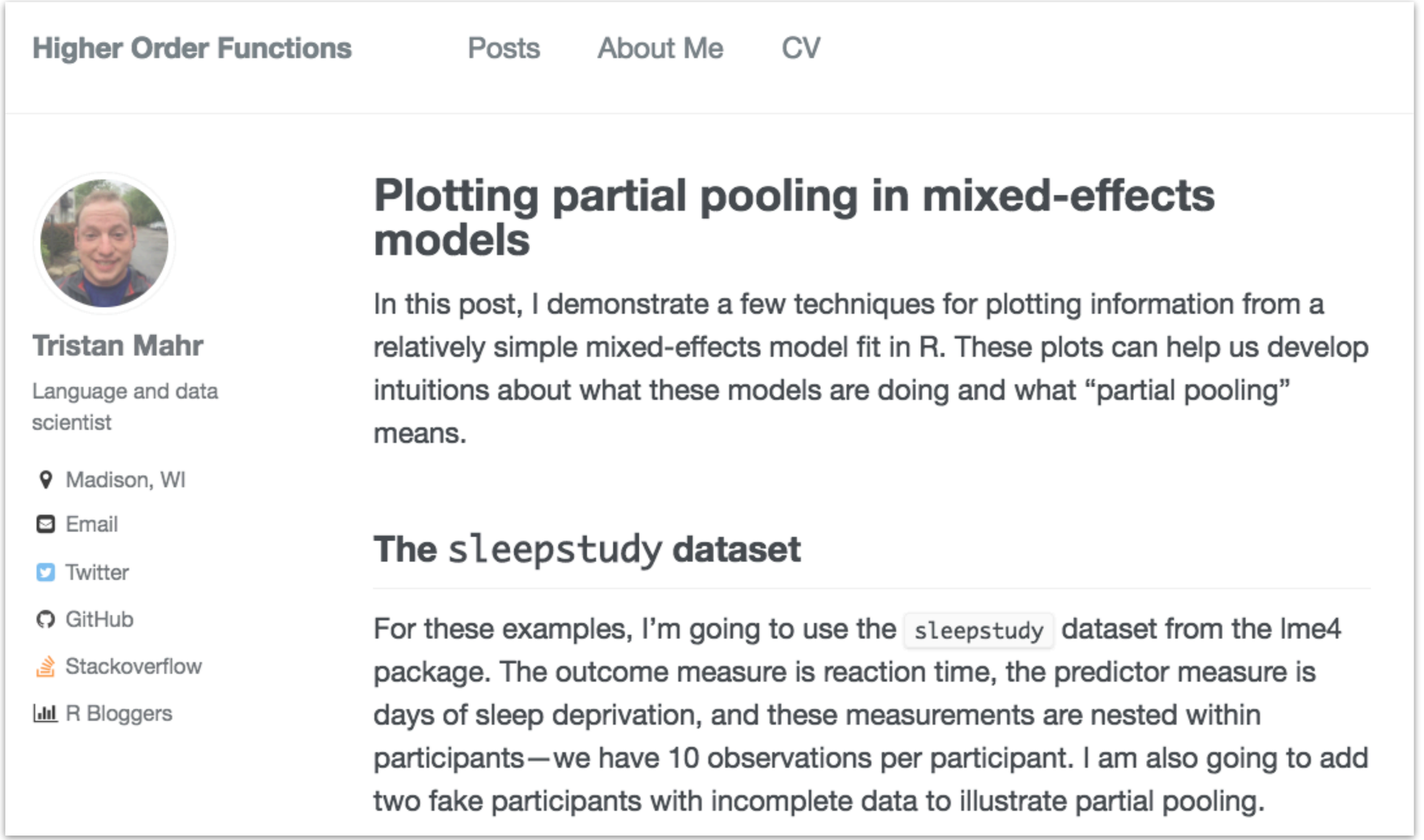
- **fixed effects**:
  - parameters are estimated
  - often: factors that we manipulate experimentally
- **random effects**:
  - variation we want to take into account
  - often: differences between participants (or items) in our experiment
  - sampling viewpoint: we explicitly model the variation in 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 for each model
  - unclear how to aggregate the results to make an overall statement
- Why don't we just run a regression on the means?
  - we throw away a lot of information
  - what to do when the design is unbalanced?
- Mixed effects model:
  - makes use of all available information
  - addresses the main problems of the other two approaches

let's take a look at an example

A worked example



https://www.tjmahr.com/plotting-partial-pooling-in-mixed-effects-models/

Data set

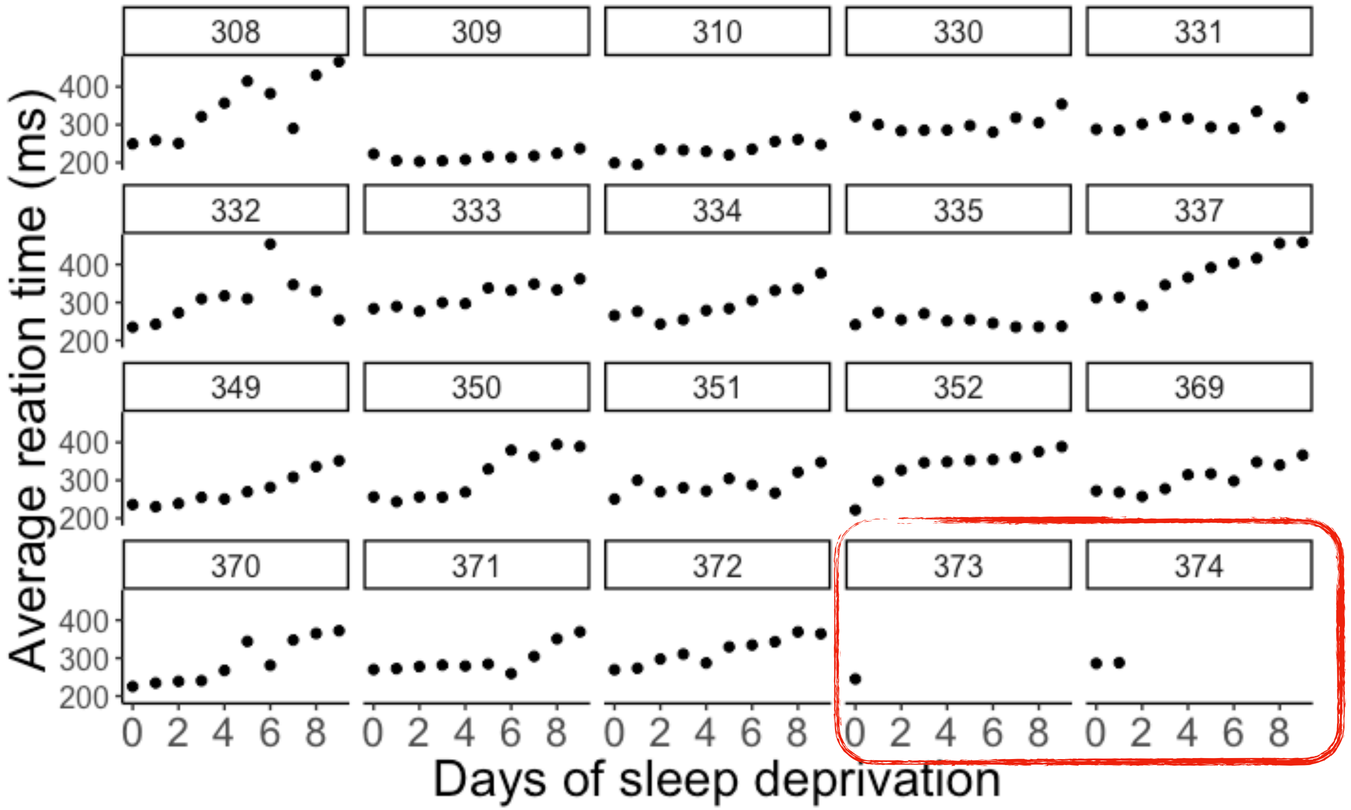
How does sleep deprivation affect reaction time?

subject	days	reaction
308	0	249.56
308	1	258.70
308	2	250.80
308	3	321.44
308	4	356.85
309	0	222.73
309	1	205.27
309	2	202.98
309	3	204.71
309	4	207.72

Data set

How does sleep deprivation affect reaction time?

subject	days	reaction
308	0	249.56
308	1	258.70
308	2	250.80
308	3	321.44
308	4	356.85
309	0	222.73
309	1	205.27
309	2	202.98
309	3	204.71
309	4	207.72



20 participants

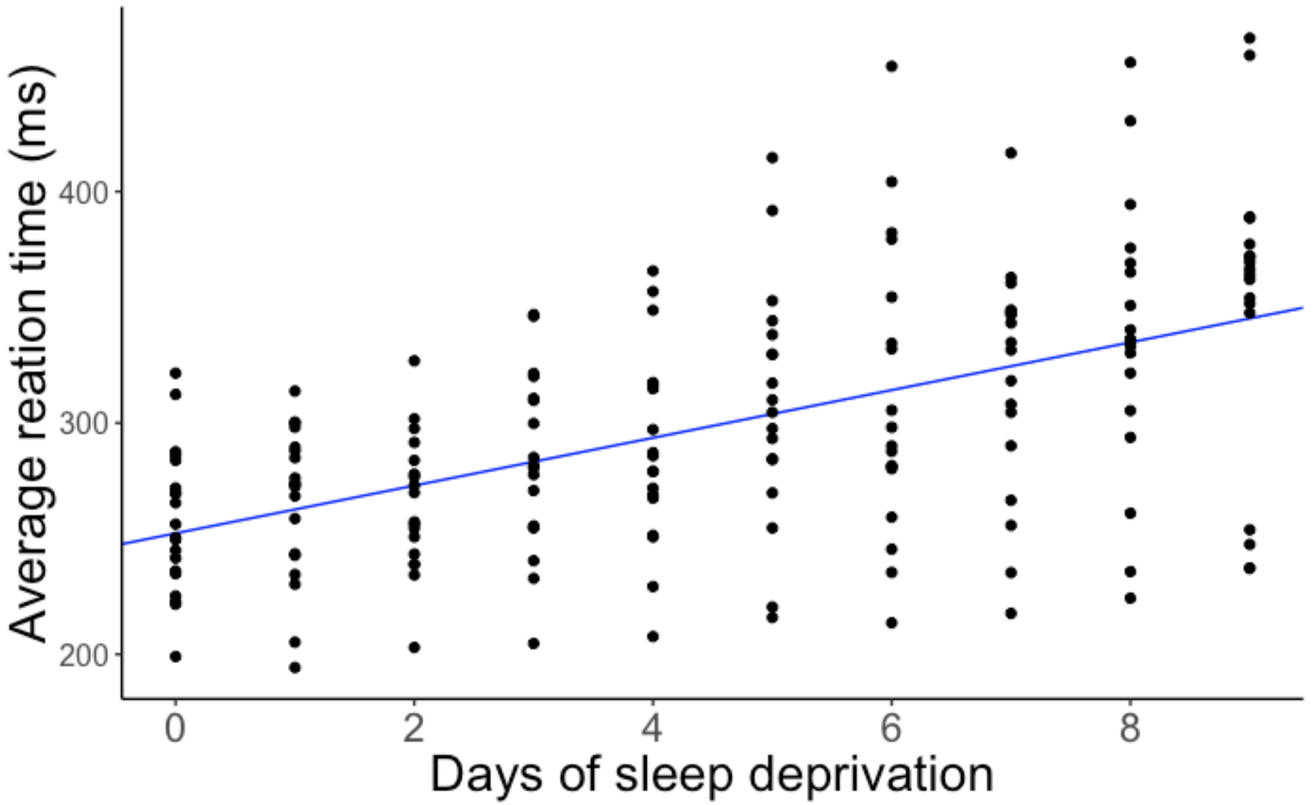
2 with incomplete information

Pooling information

- **complete pooling**
  - combine data from all participants and fit one global regression
- **no pooling**
  - don't combine any of the data and fit a separate regression to each individual participant
- **partial pooling**
  - take into account all information by explicitly modeling the variation between participants

Complete pooling: Fit one global regression

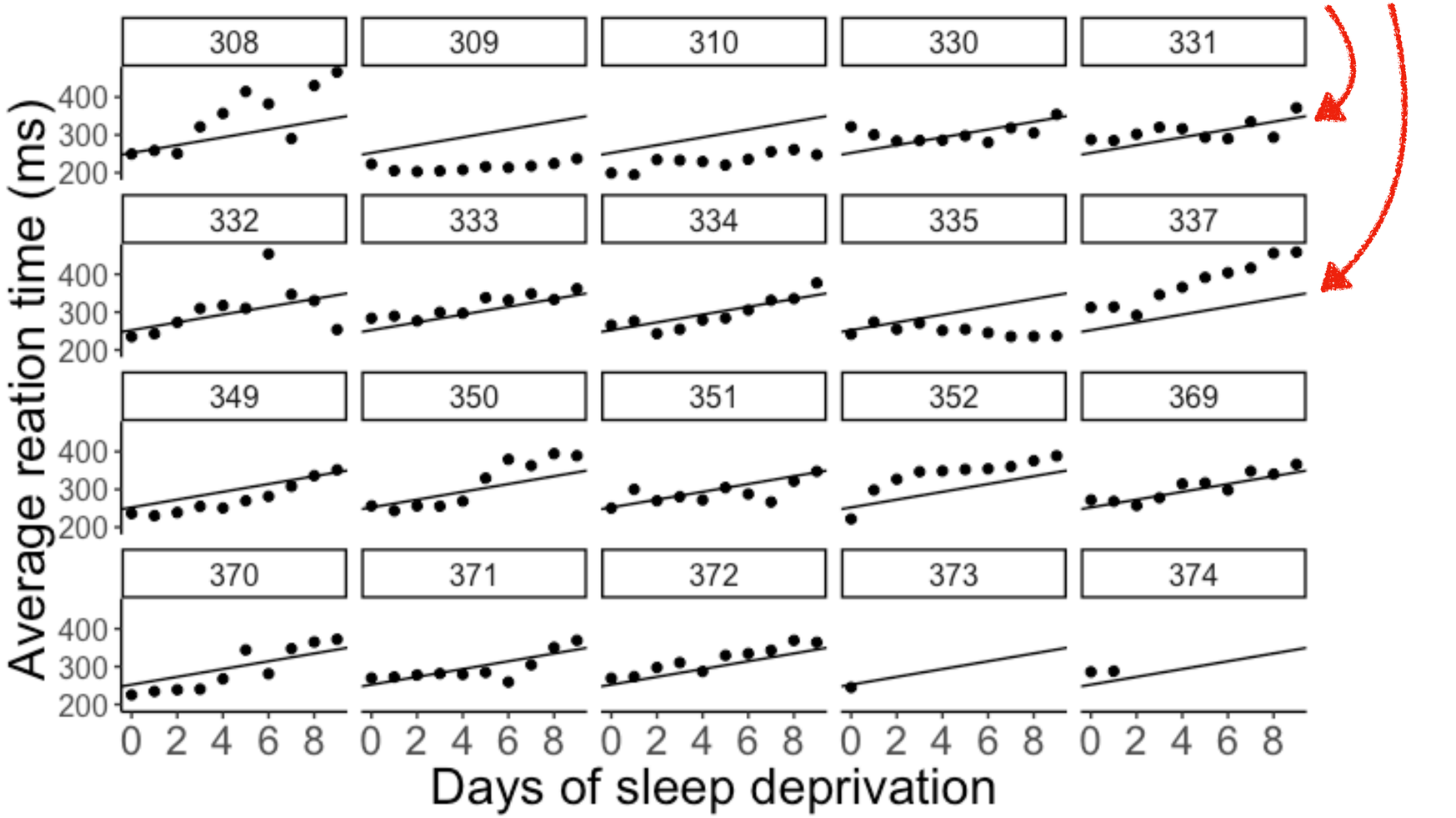
```
lm(formula = reaction ~ 1 + days,
   data = df.sleep)
```



Complete pooling: Fit one global regression

```
lm(formula = reaction ~ 1 + days,
    data = df.sleep)
```

same line for each participant



No pooling: Fit separate regressions

```
df.no_pooling = df.sleep %>%
  group_by(subject) %>%
  nest(data = c(days, reaction)) %>%
  mutate(fit = map(data, ~ lm(reaction ~ 1 + days, data = .)),
         params = map(fit, tidy)) %>%
  unnest(c(params)) %>%
  select(subject, term, estimate) %>%
  complete(subject, term, fill = list(estimate = 0)) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  clean_names()
```

data | regression fit | extracted parameters

	subject	data	fit	params
1	308	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 244.1926690909...	list(term = c("(Intercept)", "days"), estimate = c(244.1...
2	309	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 205.0549454545...	list(term = c("(Intercept)", "days"), estimate = c(205.0...
3	310	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(1...	list(coefficients = c(`(Intercept)` = 203.4842254545...	list(term = c("(Intercept)", "days"), estimate = c(203.4...
4	330	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(3...	list(coefficients = c(`(Intercept)` = 289.6850927272...	list(term = c("(Intercept)", "days"), estimate = c(289.6...
5	331	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 285.7389654545...	list(term = c("(Intercept)", "days"), estimate = c(285.7...
6	332	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 264.2516145454...	list(term = c("(Intercept)", "days"), estimate = c(264.2...
7	333	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 275.0191054545...	list(term = c("(Intercept)", "days"), estimate = c(275.0...
8	334	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 240.1629145454...	list(term = c("(Intercept)", "days"), estimate = c(240.1...
9	335	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 263.0346927272...	list(term = c("(Intercept)", "days"), estimate = c(263.0...
10	337	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(3...	list(coefficients = c(`(Intercept)` = 290.1041272727...	list(term = c("(Intercept)", "days"), estimate = c(290.1...
11	349	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 215.1117727272...	list(term = c("(Intercept)", "days"), estimate = c(215.1...
⋮				
19	374	list(days = c(0, 1), reaction = c(286, 288))	list(coefficients = c(`(Intercept)` = 286, days = 2.000...	list(term = c("(Intercept)", "days"), estimate = c(286, 2...
20	373	list(days = 0, reaction = 245)	list(coefficients = c(`(Intercept)` = 245, days = NA), r...	list(term = "(Intercept)", estimate = 245, std.error = ...

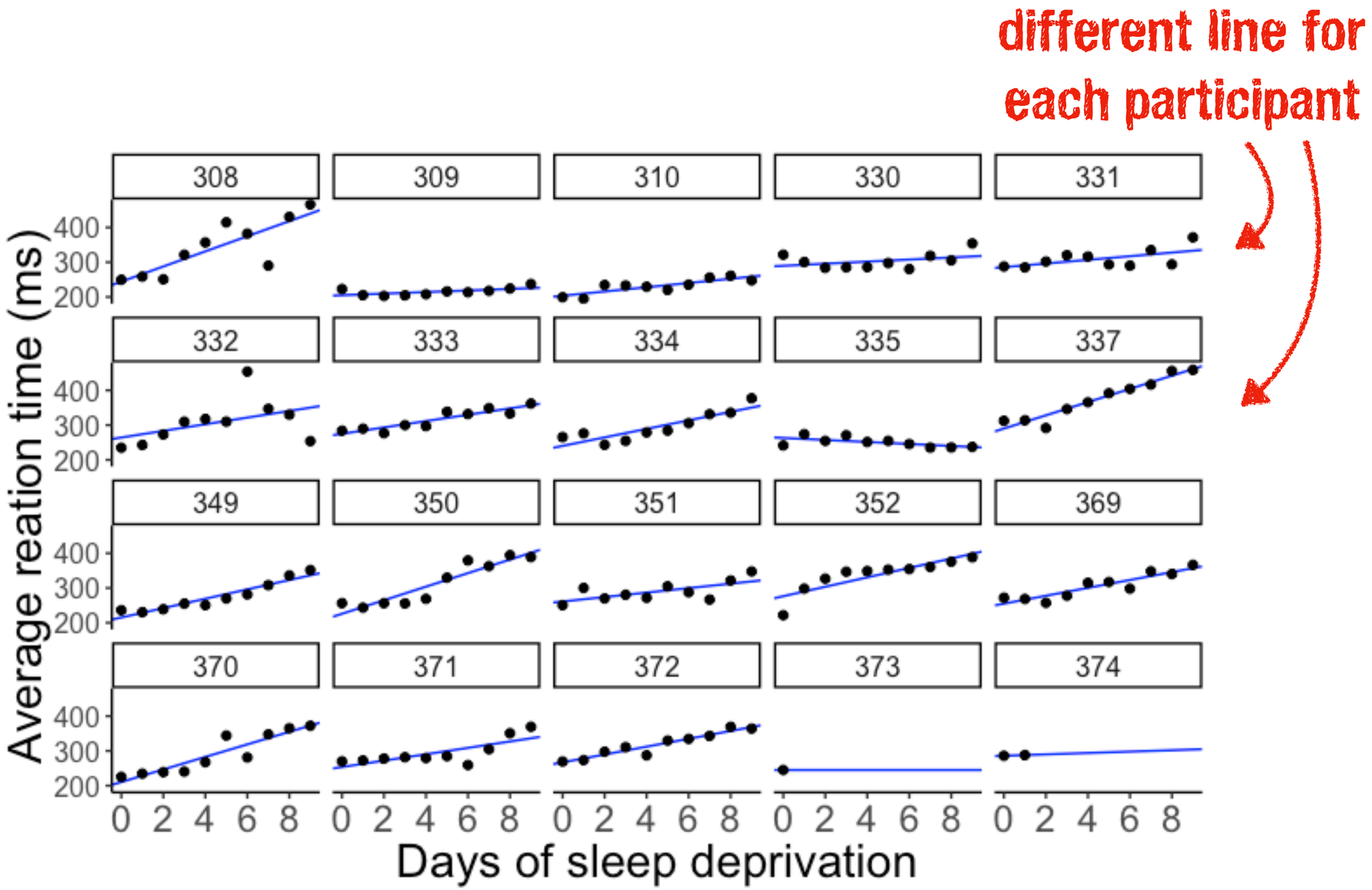
No pooling: Fit separate regressions

```
df.no_pooling = df.sleep %>%
  group_by(subject) %>%
  nest(data = c(days, reaction)) %>%
  mutate(fit = map(data, ~ lm(reaction ~ 1 + days, data = .)),
         params = map(fit, tidy)) %>%
  unnest(c(params)) %>%
  select(subject, term, estimate) %>%
  complete(subject, term, fill = list(estimate = 0)) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  clean_names()
```

separate intercept and slope for each participant

	subject	intercept	days
1	308	244.1927	21.764702
2	309	205.0549	2.261785
3	310	203.4842	6.114899
4	330	289.6851	3.008073
5	331	285.7390	5.266019
6	332	264.2516	9.566768
7	333	275.0191	9.142045
8	334	240.1629	12.253141
9	335	263.0347	-2.881034
10	337	290.1041	19.025974
11	349	215.1118	13.493933
12	350	225.8346	19.504017
13	351	261.1470	6.433498
14	352	276.3721	13.566549
15	369	254.9681	11.348109
16	370	210.4491	18.056151
17	371	253.6360	9.188445
18	372	267.0448	11.298073
19	373	245.0000	0.000000
20	374	286.0000	2.000000

No pooling: Fit separate regressions



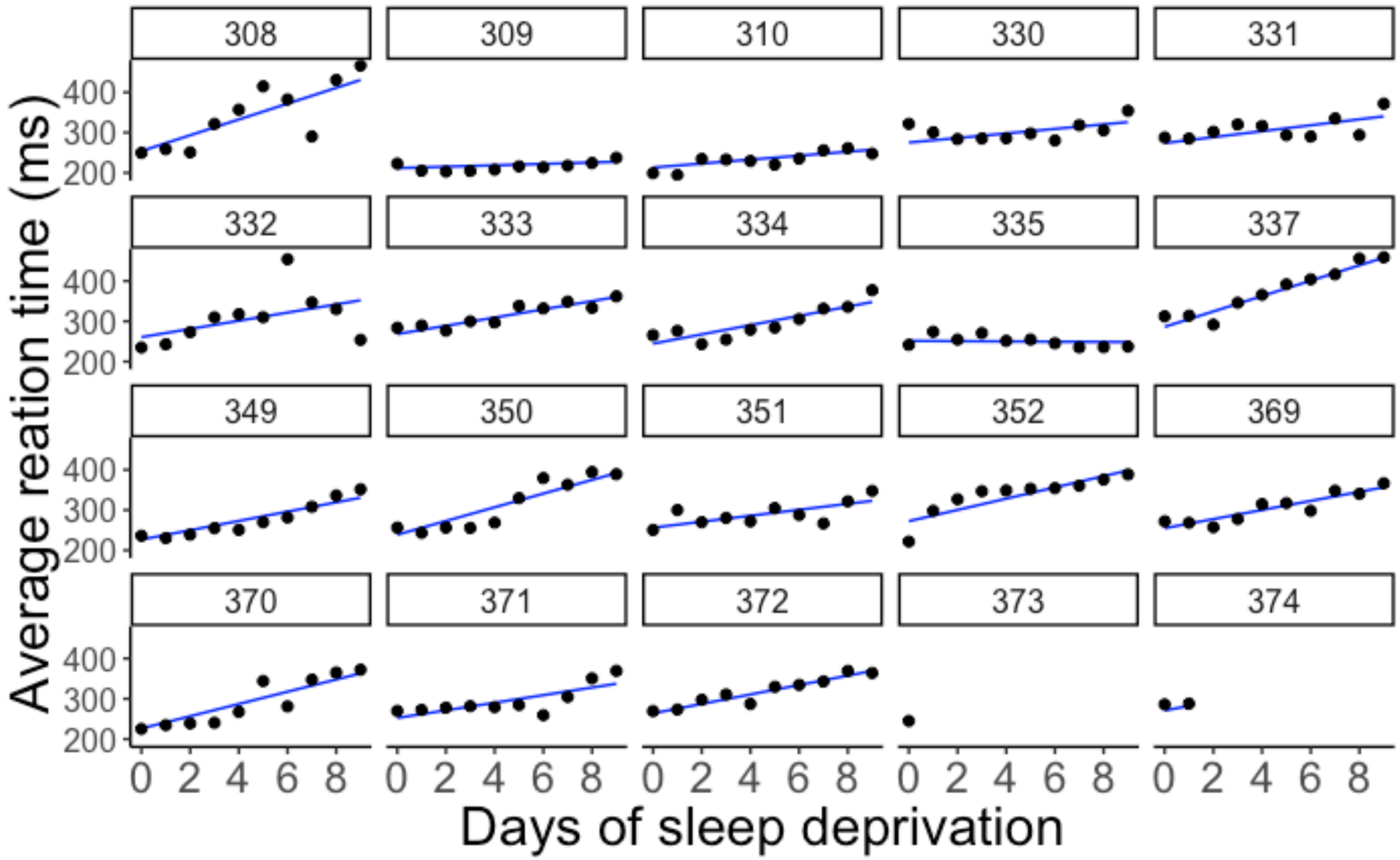
Partial pooling: Fit mixed effects model

intercepts and slopes differ between participants

random intercept

random slope

```
lmer(formula = reaction ~ 1 + days + (1 + days | subject),
     data = df.sleep)
```

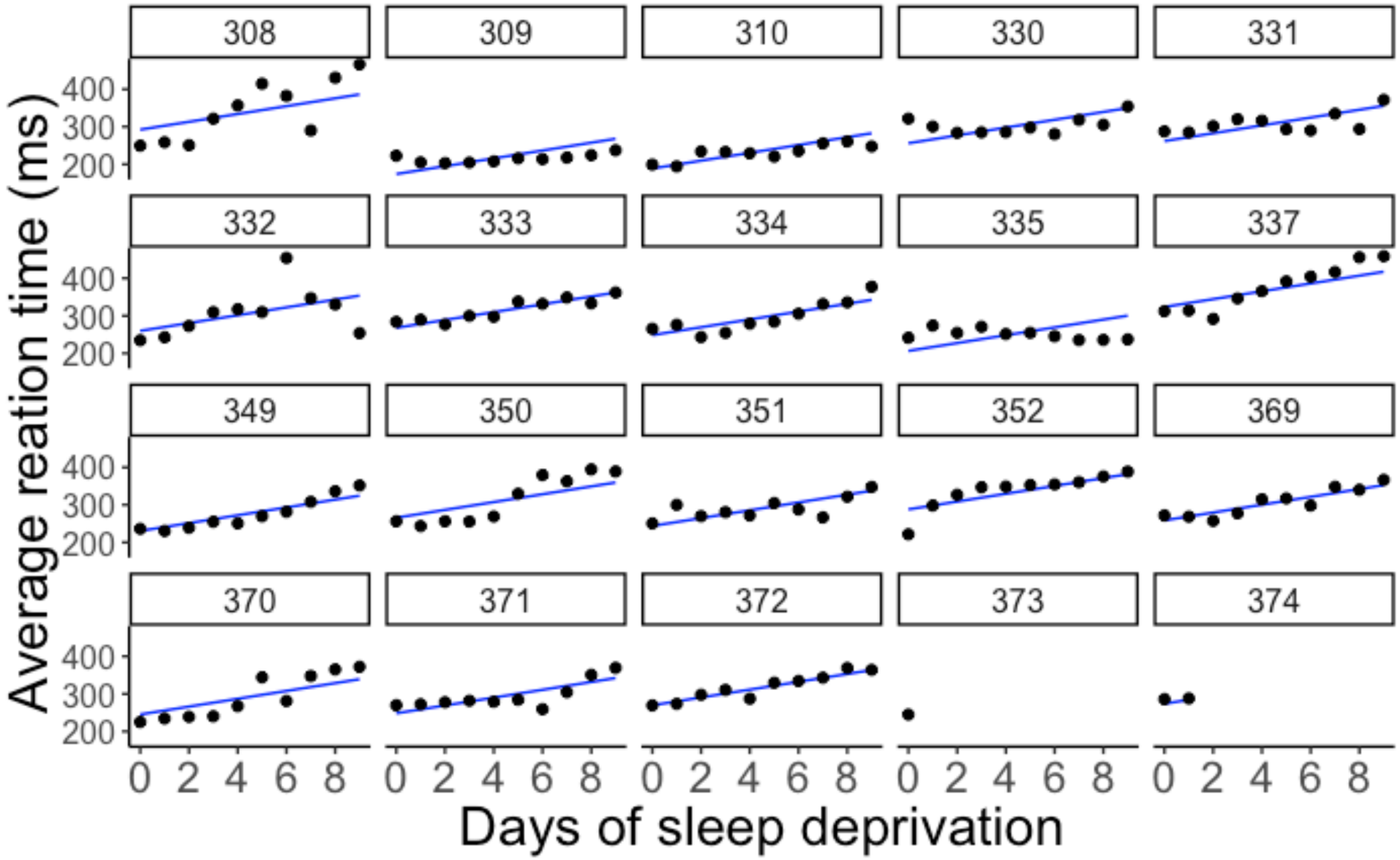


Partial pooling: Fit mixed effects model

only intercepts differ between participants

random intercept

```
lmer(formula = reaction ~ 1 + days + (1 | subject),
     data = df.sleep)
```

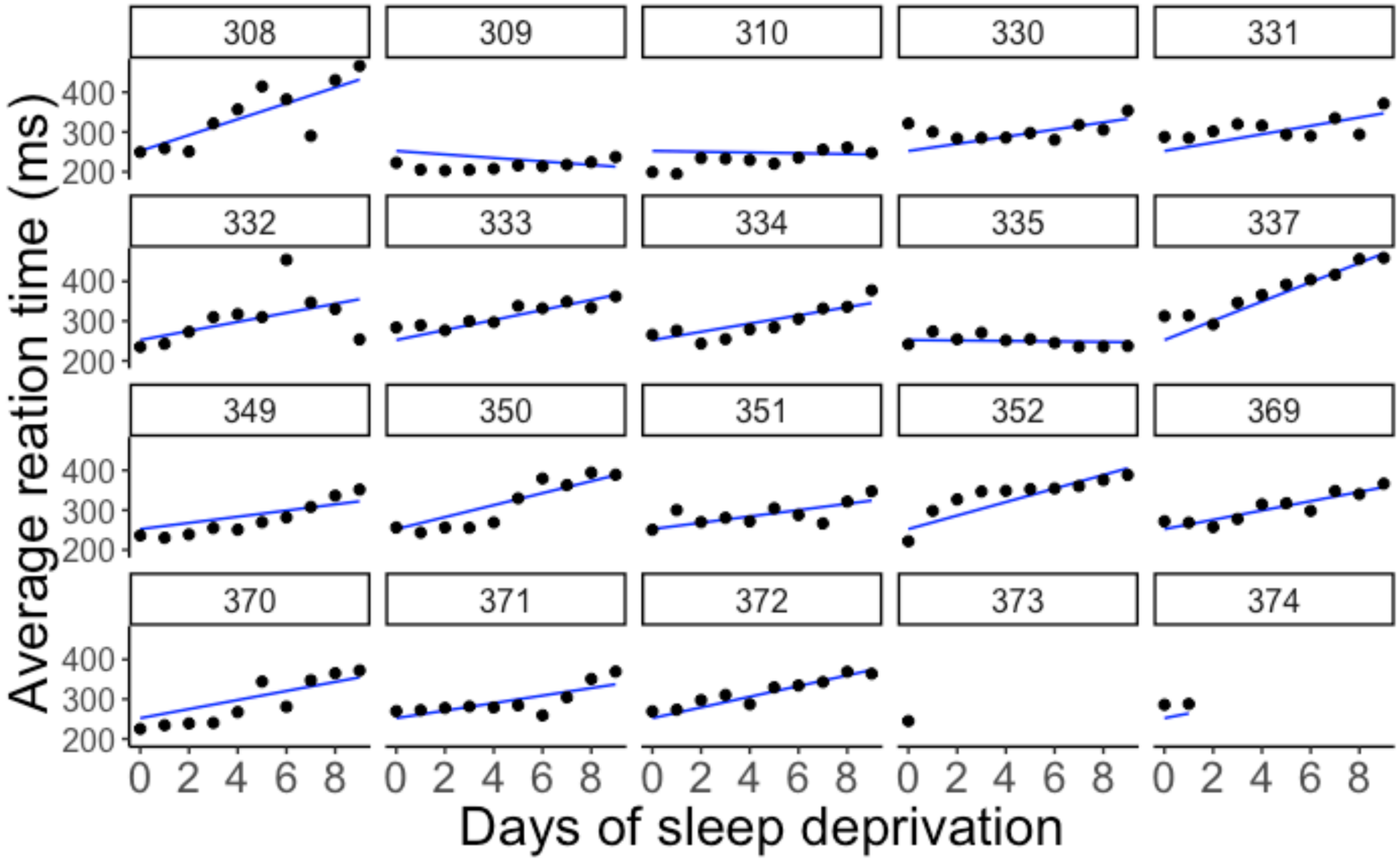


Partial pooling: Fit mixed effects model

only slopes differ between participants

random slope

```
lmer(formula = reaction ~ 1 + days + (0 + days | subject),
     data = df.sleep)
```



Coefficients

```
lmer(formula = reaction ~ 1 + days + ...,
     data = df.sleep)
```

`(1 | subject)`

random intercepts

```
$subject
    (Intercept)     days
308    292.2749 10.43191
309    174.0559 10.43191
310    188.7454 10.43191
330    256.0247 10.43191
331    261.8141 10.43191
332    259.8262 10.43191
333    268.0765 10.43191
334    248.6471 10.43191
335    206.5096 10.43191
337    323.5643 10.43191
349    230.5114 10.43191
350    265.6957 10.43191
351    243.7988 10.43191
352    287.8850 10.43191
369    258.6454 10.43191
370    245.2931 10.43191
371    248.3508 10.43191
372    269.6861 10.43191
373    248.2086 10.43191
374    273.9400 10.43191
```

`(0 + days | subject)`

random slopes

```
$subject
    (Intercept)       days
308    252.2965 19.9526801
309    252.2965 -4.3719650
310    252.2965 -0.9574726
330    252.2965  8.9909957
331    252.2965 10.5394285
332    252.2965 11.3994289
333    252.2965 12.6074020
334    252.2965 10.3413879
335    252.2965 -0.5722073
337    252.2965 24.2246485
349    252.2965  7.7702676
350    252.2965 15.0661415
351    252.2965  7.9675415
352    252.2965 17.0002999
369    252.2965 11.6982767
370    252.2965 11.3939807
371    252.2965  9.4535879
372    252.2965 13.4569059
373    252.2965 10.4142695
374    252.2965 11.9097917
```

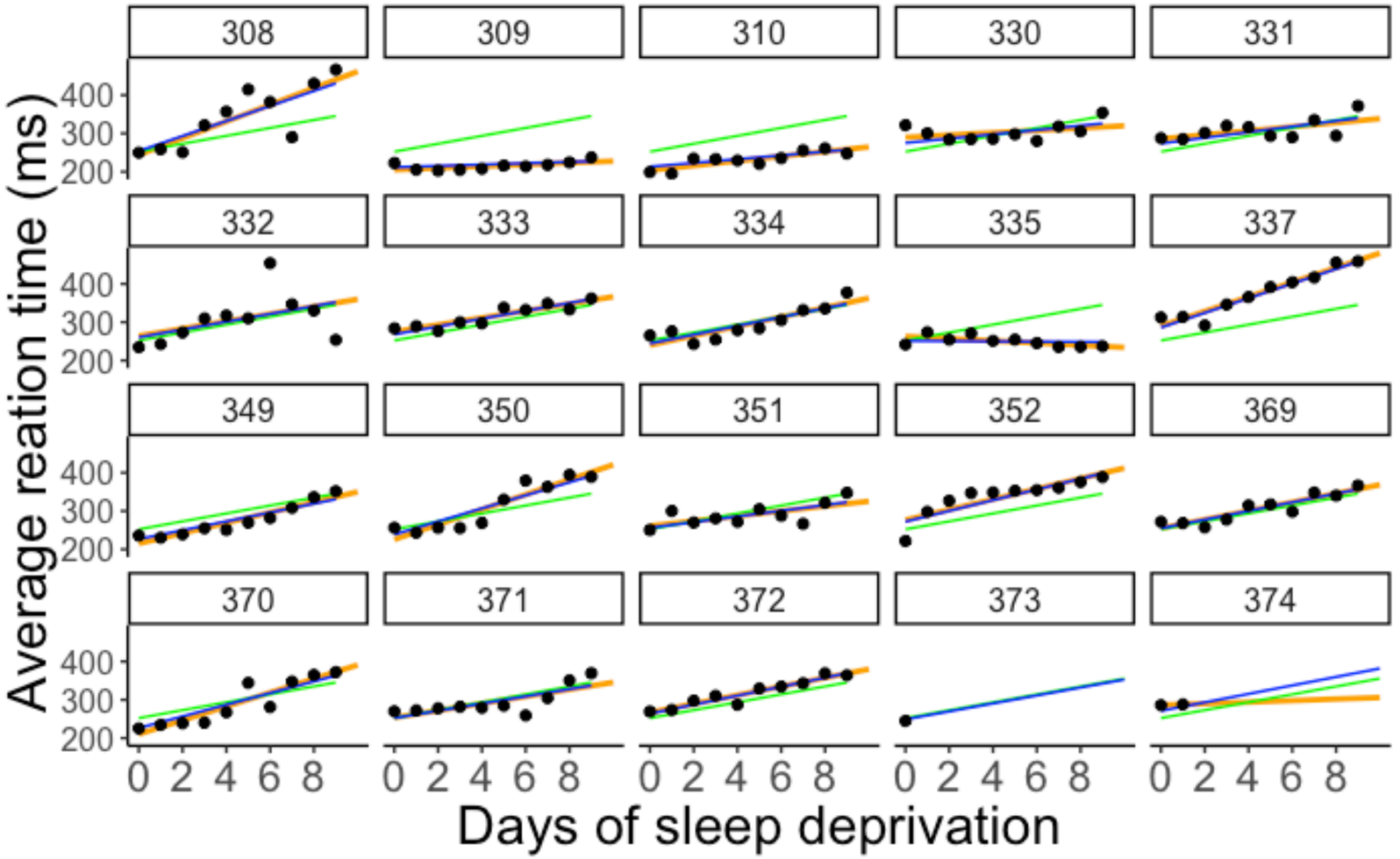
`(1 + days | subject)`

random intercepts and slopes (+ correlation)

```
$subject
    (Intercept)       days
308    253.9479 19.6264139
309    211.7328  1.7319567
310    213.1579  4.9061843
330    275.1425  5.6435987
331    273.7286  7.3862680
332    260.6504 10.1632535
333    268.3684 10.2245979
334    244.5523 11.4837825
335    251.3700 -0.3355554
337    286.2321 19.1090061
349    226.7662 11.5531963
350    238.7807 17.0156766
351    256.2344  7.4119501
352    272.3512 13.9920698
369    254.9484 11.2985741
370    226.3701 15.2027922
371    252.5051  9.4335432
372    263.8916 11.7253342
373    248.9752 10.3915245
374    271.1451 11.0782697
```

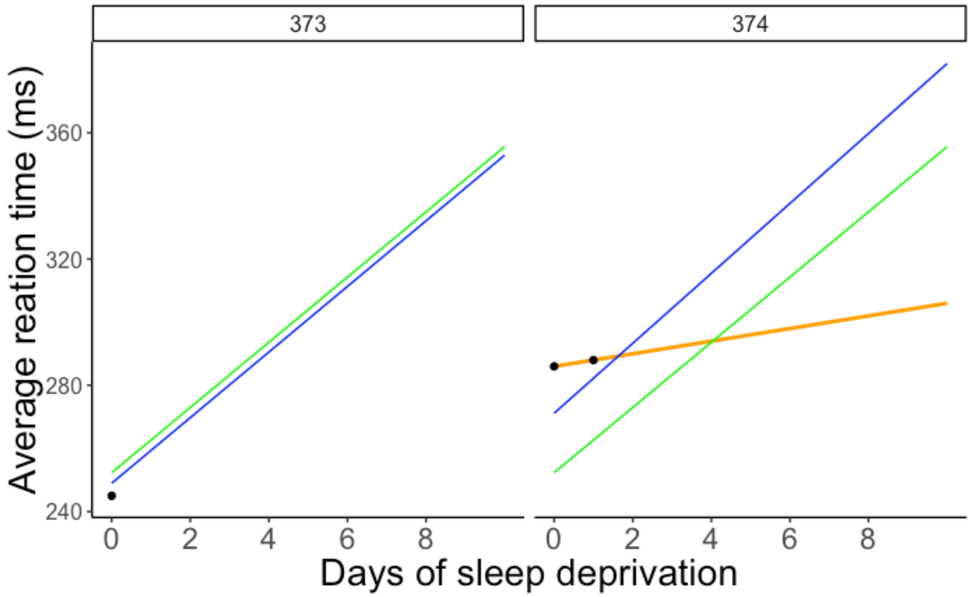
Comparison

complete pooling
no pooling
partial pooling



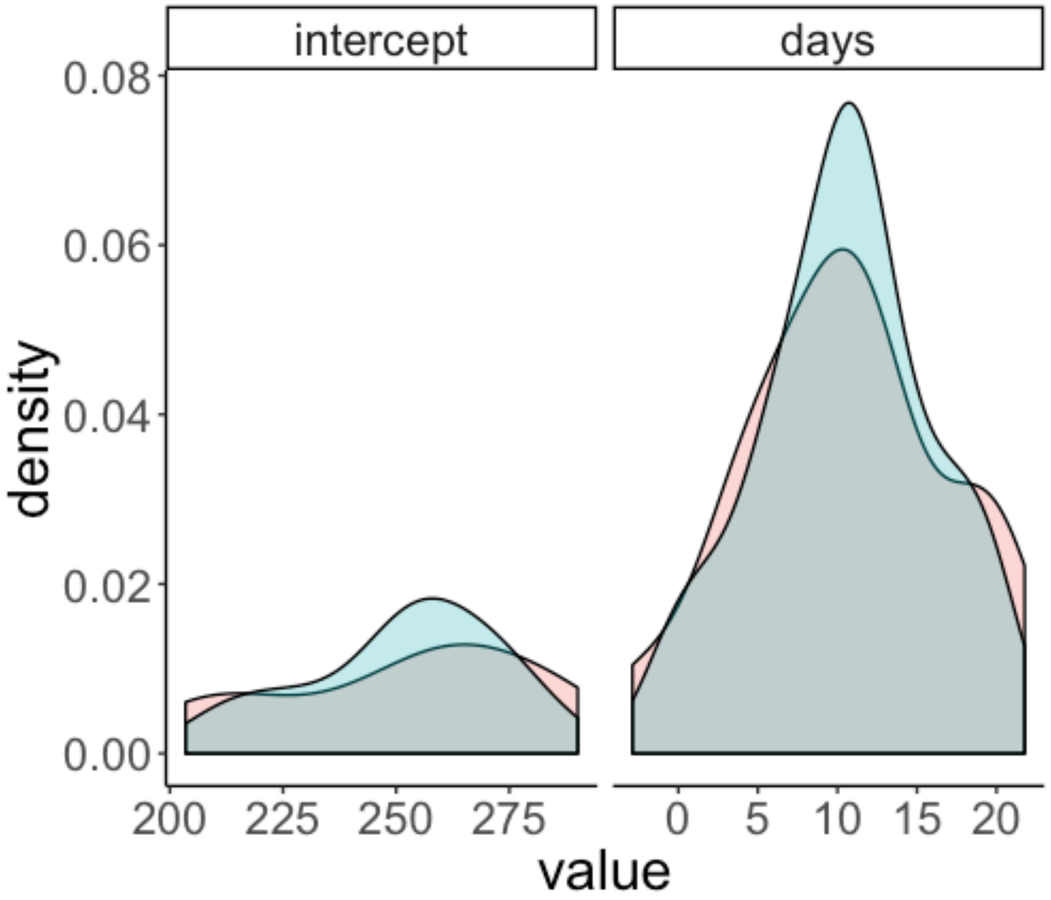
Comparison

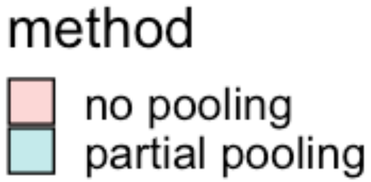
complete pooling
no pooling
partial pooling



- **complete pooling**:
  - doesn't account for any individual variation
- **no pooling**:
  - doesn't yield predictions when we only have one observation
  - doesn't consider the general effect of sleep deprivation when making predictions
- **partial pooling**:
  - draws on all the information in the data
  - extrapolates based on information about the individual participants, as well as information based on the whole sample

Shrinkage



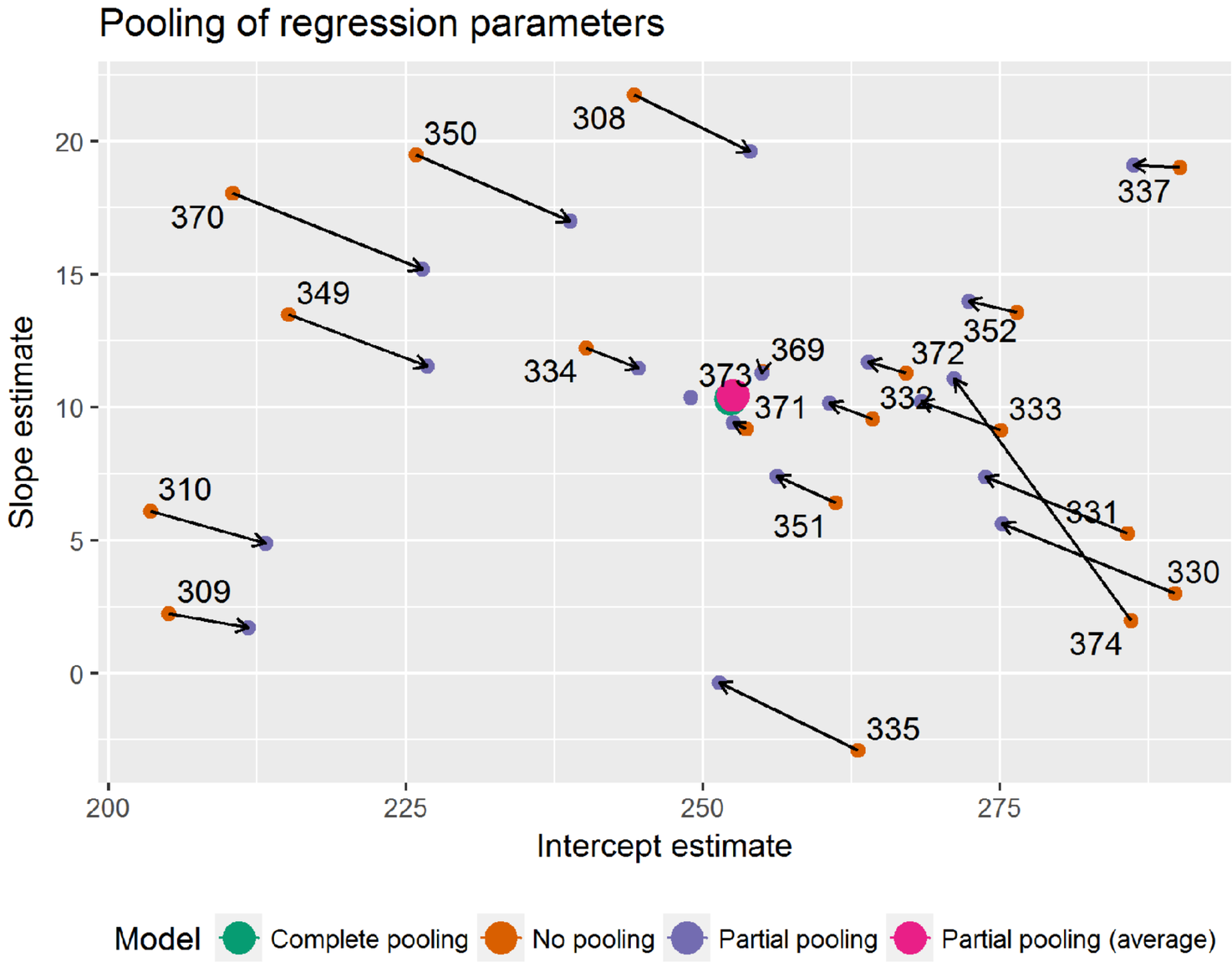


standard deviation

method	intercept	days
no pooling	28.95	6.56
partial pooling	21.59	5.46

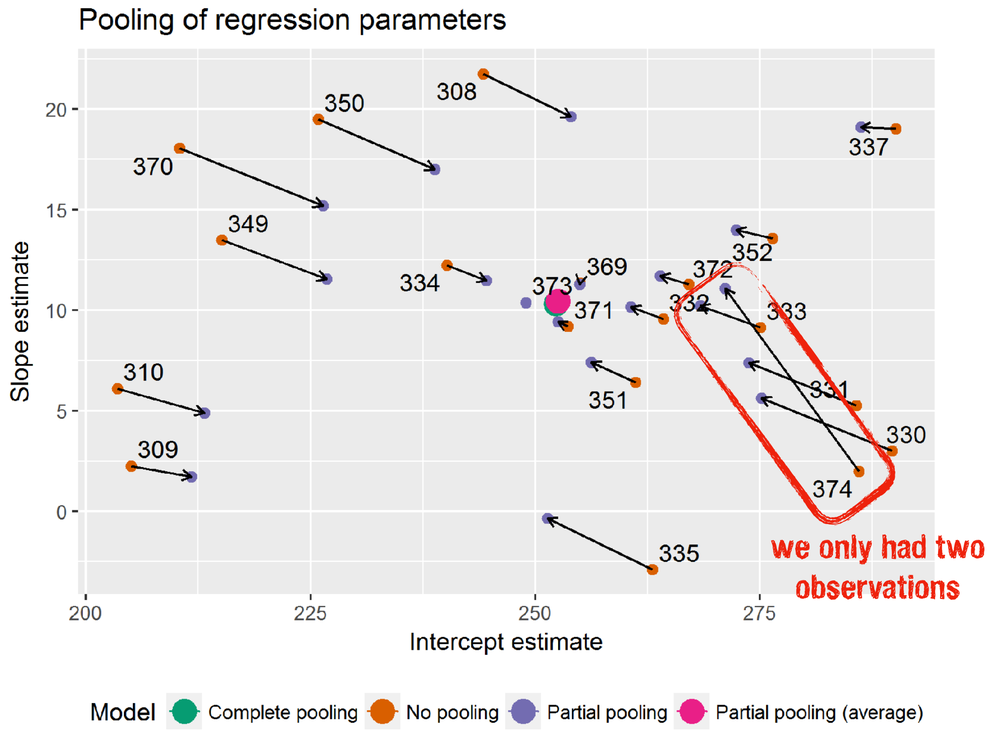
variance "shrinks"

Shrinkage


Pooling of regression parameters
Slope estimate
Intercept estimate
20
15
10
5
0
200
225
250
275
308
350
370
337
349
352
334
369
372
373
332
371
333
310
351
331
330
309
374
335
Model
Complete pooling
No pooling
Partial pooling
Partial pooling (average)

Shrinkage

- more shrinkage when estimate is further from the average
- more shrinkage when estimate is more uncertain (based on fewer observations); more information "borrowed" from other clusters

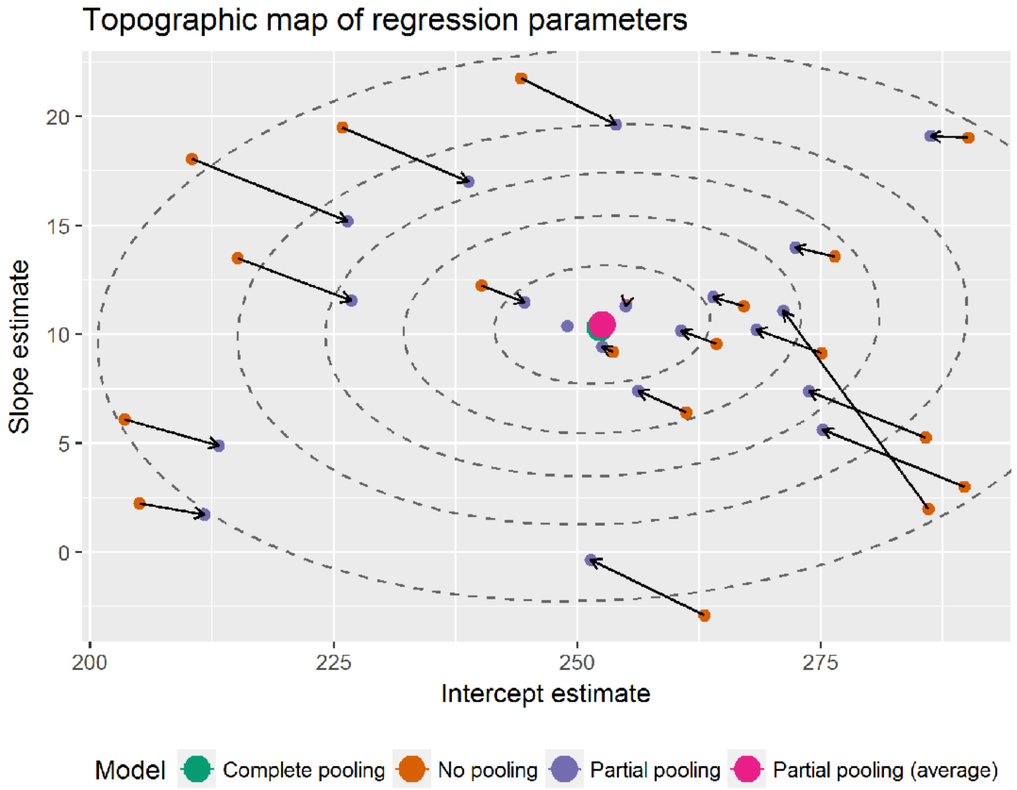


In the lme4 book, Douglas Bates provides an alternative to *shrinkage*:

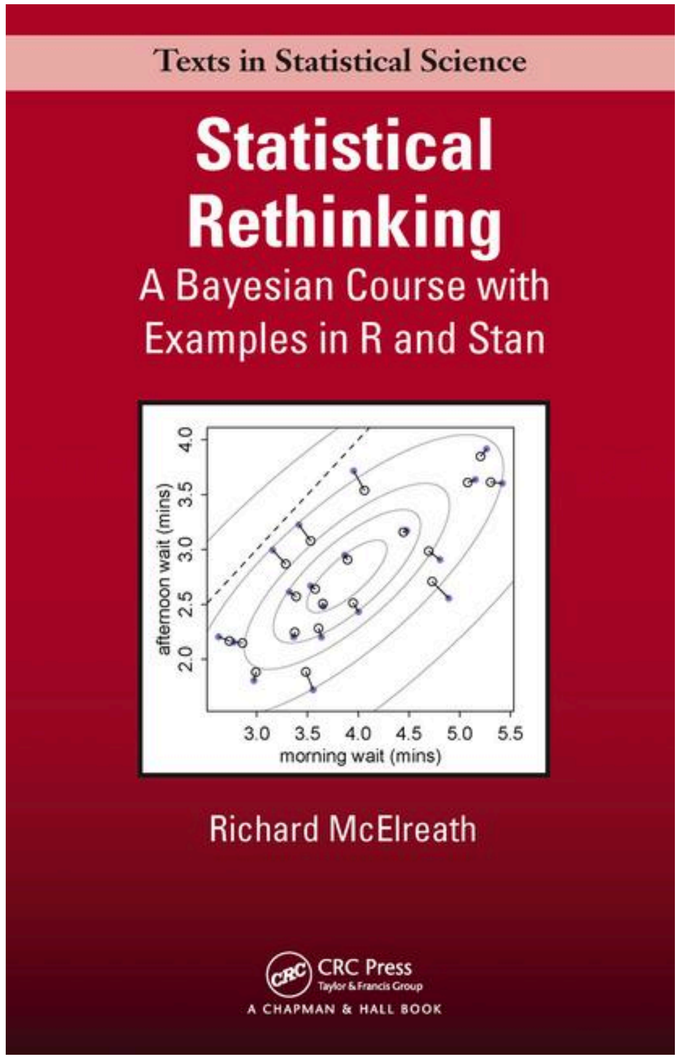
The term "shrinkage" may have negative connotations. John Tukey preferred to refer to the process as the estimates for individual subjects **"borrowing strength" from each other**.

This is a fundamental difference in the models underlying mixed-effects models versus strictly fixed effects models. In a mixed-effects model we assume that the levels of a grouping factor are a selection from a population and, as a result, can be expected to share characteristics to some degree. Consequently, the predictions from a mixed-effects model are attenuated relative to those from strictly fixed-effects models.

Shrinkage

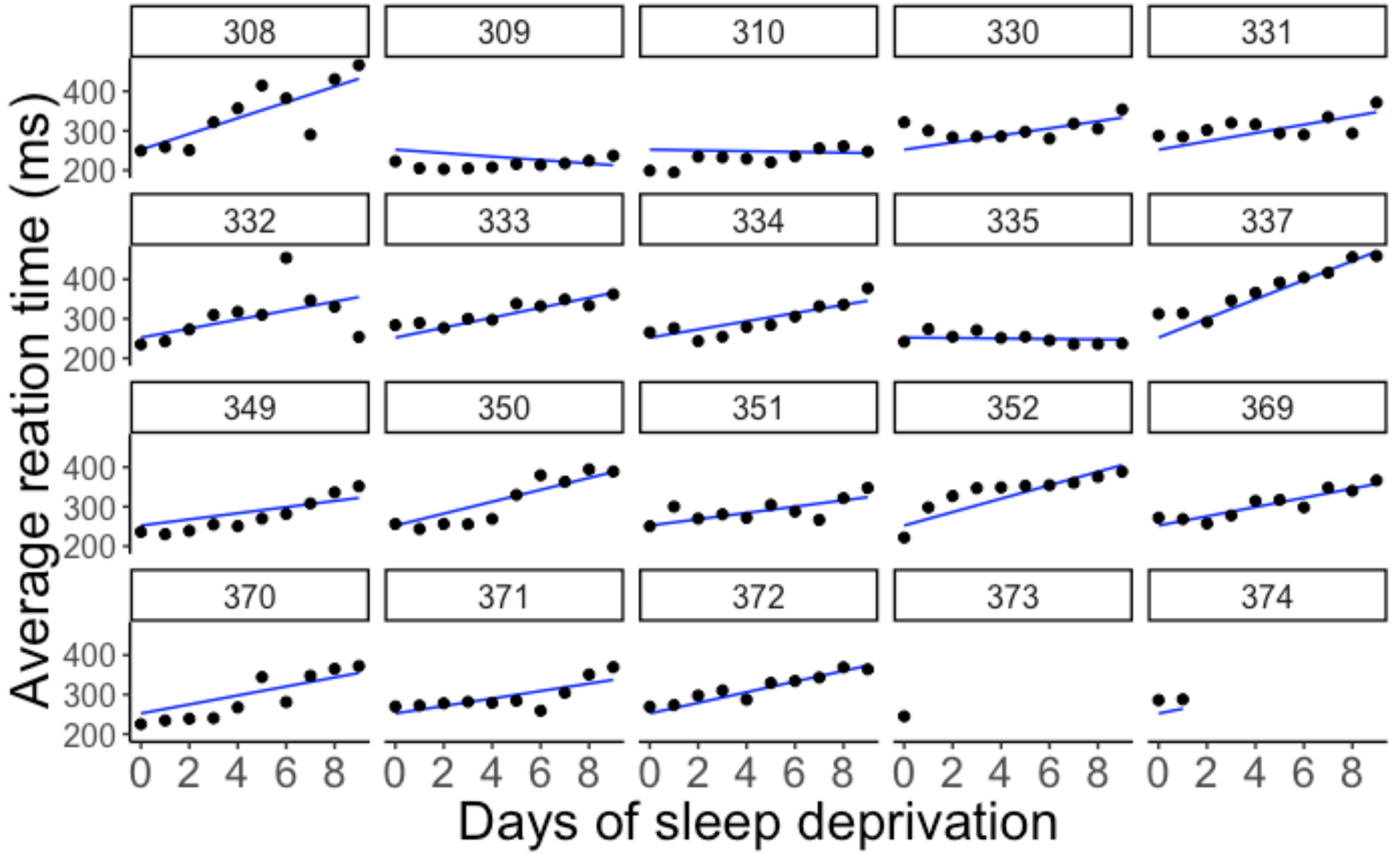


mixed effects model estimates a multi-variate Gaussian to account for (possible) correlations between intercepts and slopes



Reporting results

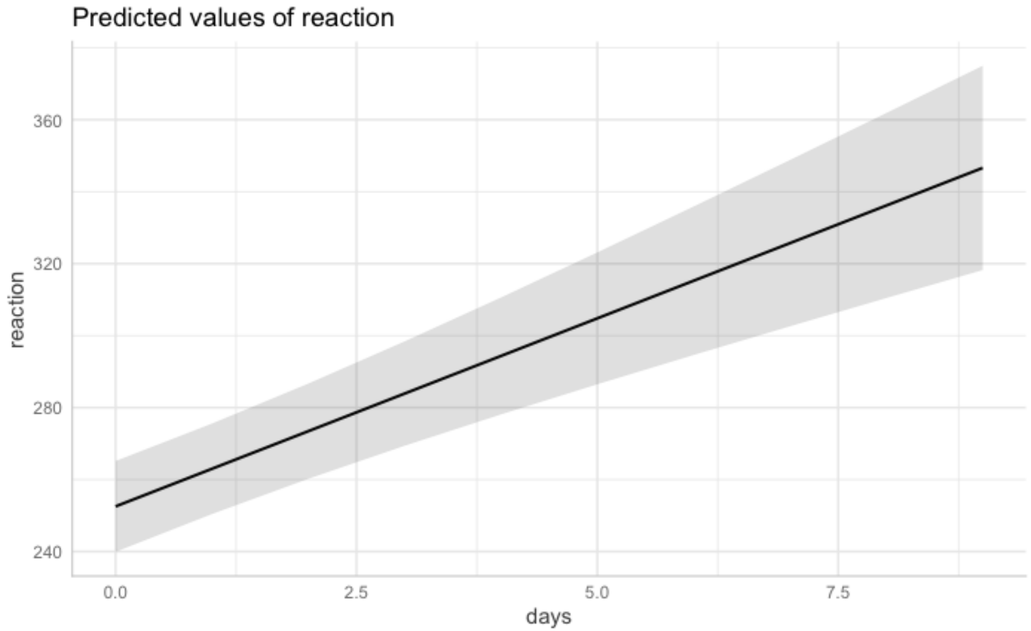
Visualization



show the data together with the model predictions

Visualization


```
1 library("ggeffects")
2
3 ggpredict(model = fit.random_intercept_slope,
4            terms = "days",
5            type = "fe") %>%
6   plot()
```



- the relationship between the variables of interest (marginalizing over other variables)

show the (marginalized) model prediction

Reporting results

7.1. In Writing

Our reports include a description of the following parts (also see Meteyard & Davies, 2019; Barr et al., 2013):

- Model specification, including:
  - Dependent variable, and all fixed and random effects (intercepts, slopes, correlations), both in words and possibly also by providing the model equation/ R-pseudo code (so-called Wilkinson notation)
  - Transformation of variables, e.g., standardizing or centering variables
  - Contrast coding (typically sum-to-zero coding)
- Inference:
  - Description of how *p*-values were obtained (in case of a frequentist approach) or what other (Bayesian) decision rule was used for inference.
  - Description of what post-hoc or follow-up tests were performed
  - Any convergence issues that may arise while running the model (in particular if they require adjustments in the model specification) and how they were dealt with should be described, as well as the subsequent adjustments that were made.
- Model output, at minimum the following:
  - Model results: (un)standardized regression coefficients, standard errors and/or confidence / credible intervals, test statistics, degrees of freedom, *p*-values

Meteyard, L. & Davies, R. A. I. (2020). Best practice guidance for linear mixed-effects models in psychological science. *Journal of Memory and Language*.

Reporting results

Table 2

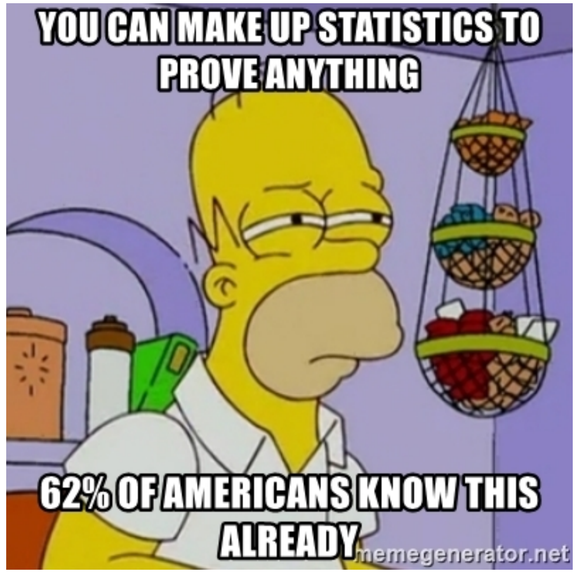
***Experiment 1 – Normality inference**: Estimates of the posterior mean and 95% highest density intervals (HDIs) for the different predictors in the Bayesian regression model. Note: For the dependent variable (normality rating), 100 = abnormal and 0 = normal.*

model specification: `normality rating ~ 1 + structure * norm`

term	estimate	lower 95% CI	upper 95% CI
intercept	62.83	57.57	68.11
structure	21.22	16.27	26.35
norm	−1.47	−6.37	3.83
structure:norm	−1.46	−6.41	3.53

[7] All categorical predictors were coded using sum contrasts. We adopt the convention of calling something an effect if the 95% highest density interval (HDI) of the estimated parameter in the Bayesian model excludes 0.

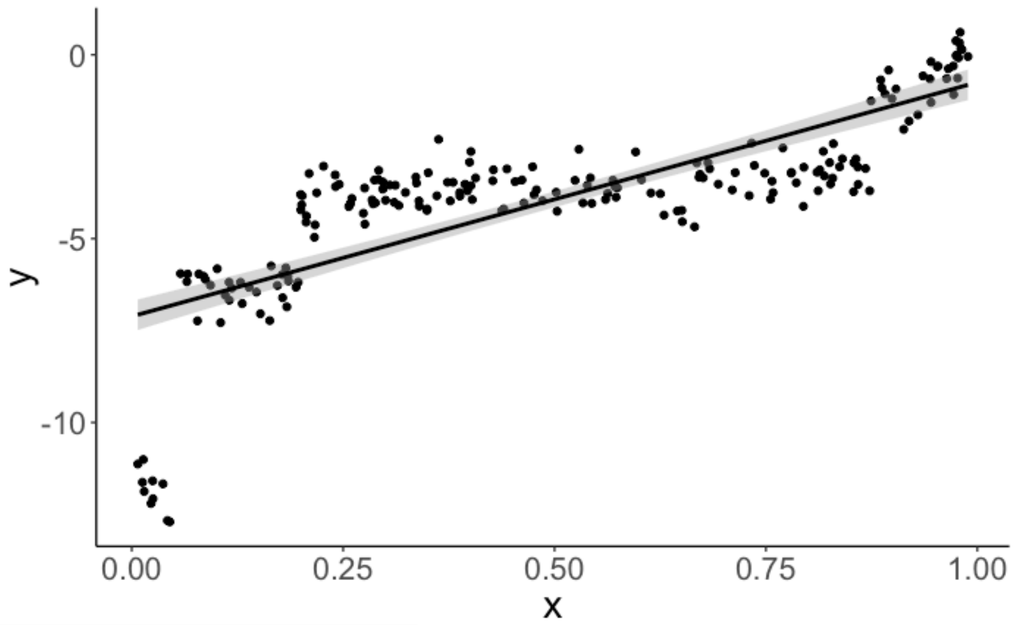
Where are the Random Effects??



Simpsons Paradox

Simpson's paradox

```r
lm(formula = y ~ 1 + x,
   data = df.simpson) %>%
  summary()
```



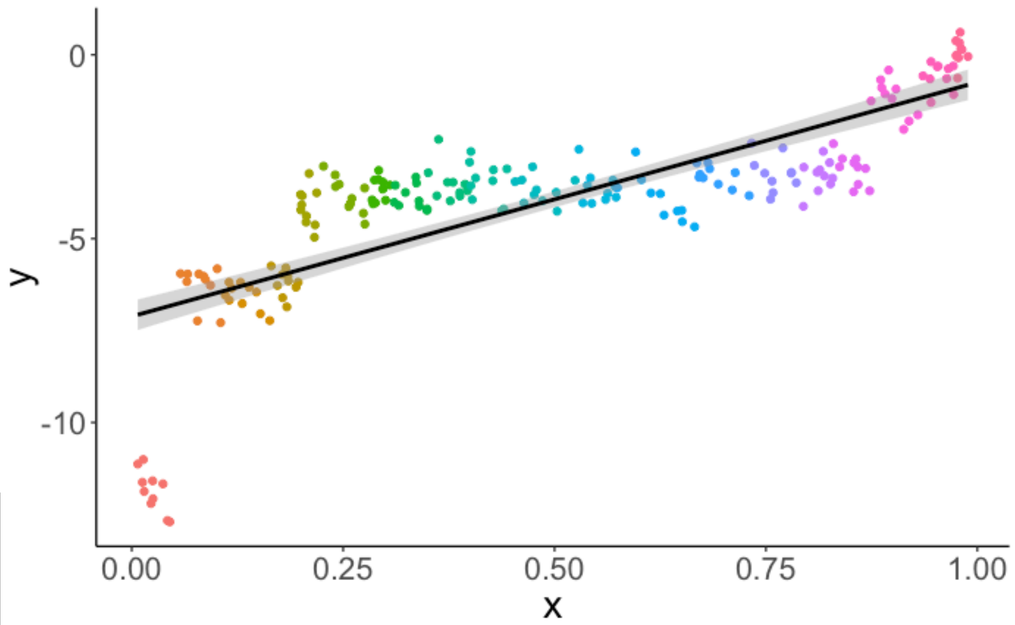

```
Call:
lm(formula = y ~ x, data = df.simpson)

Residuals:
    Min      1Q  Median      3Q     Max
-5.8731 -0.6362  0.2272  1.0051  2.6410

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.1151     0.2107  -33.76   <2e-16 ***
x             6.3671     0.3631   17.54   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.55 on 198 degrees of freedom
Multiple R-squared:  0.6083,    Adjusted R-squared:  0.6064
F-statistic: 307.5 on 1 and 198 DF,  p-value: < 2.2e-16
```

positive relationship between x and y

Simpson's paradox

```r
lmer(formula = y ~ 1 + x + (1 | participant),
     data = df.simpson) %>%
  summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ 1 + x + (1 | participant)
   Data: df.simpson

REML criterion at convergence: 345.1

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.43394 -0.59687  0.04493  0.62694  2.68828

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 21.4898  4.6357
 Residual                 0.1661  0.4075
Number of obs: 200, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -0.1577     1.3230  -0.119
x            -7.6678     1.6572  -4.627

Correlation of Fixed Effects:
  (Intr)
x -0.621
```

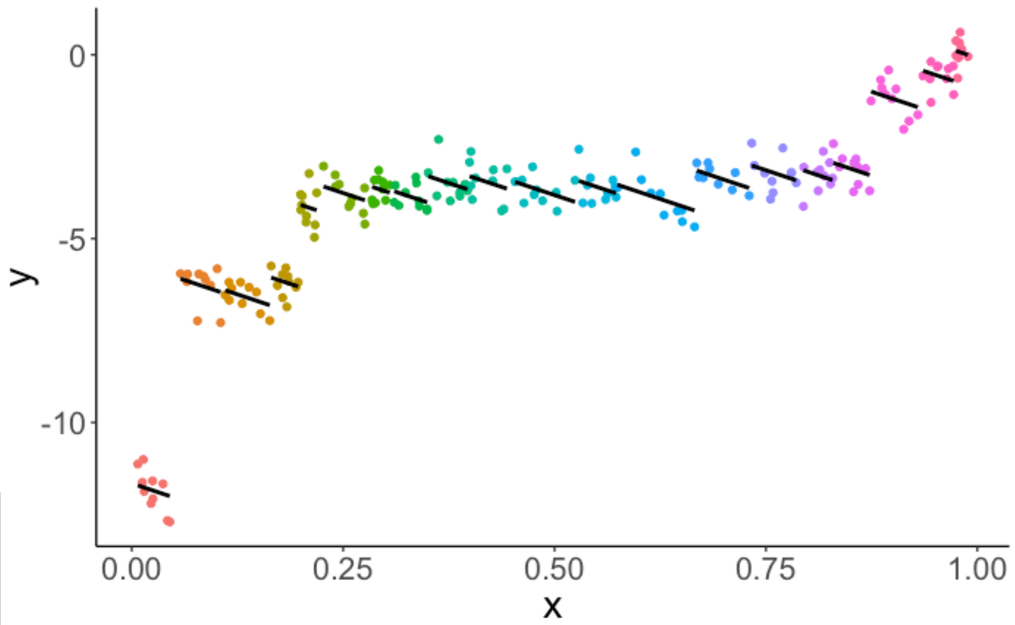


negative (!)
relationship between
x and y

Simpson's paradox

```
1 lmer(formula = y ~ 1 + x + (1 | participant),
2       data = df.simpson) %>%
3   summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ 1 + x + (1 | participant)
   Data: df.simpson

REML criterion at convergence: 345.1

Scaled residuals:
     Min       1Q   Median       3Q      Max
-2.43394 -0.59687  0.04493  0.62694  2.68828

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 21.4898  4.6357
 Residual                 0.1661  0.4075
Number of obs: 200, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  -0.1577     1.3230  -0.119
x            -7.6678     1.6572  -4.627

Correlation of Fixed Effects:
  (Intr)
x -0.621
```



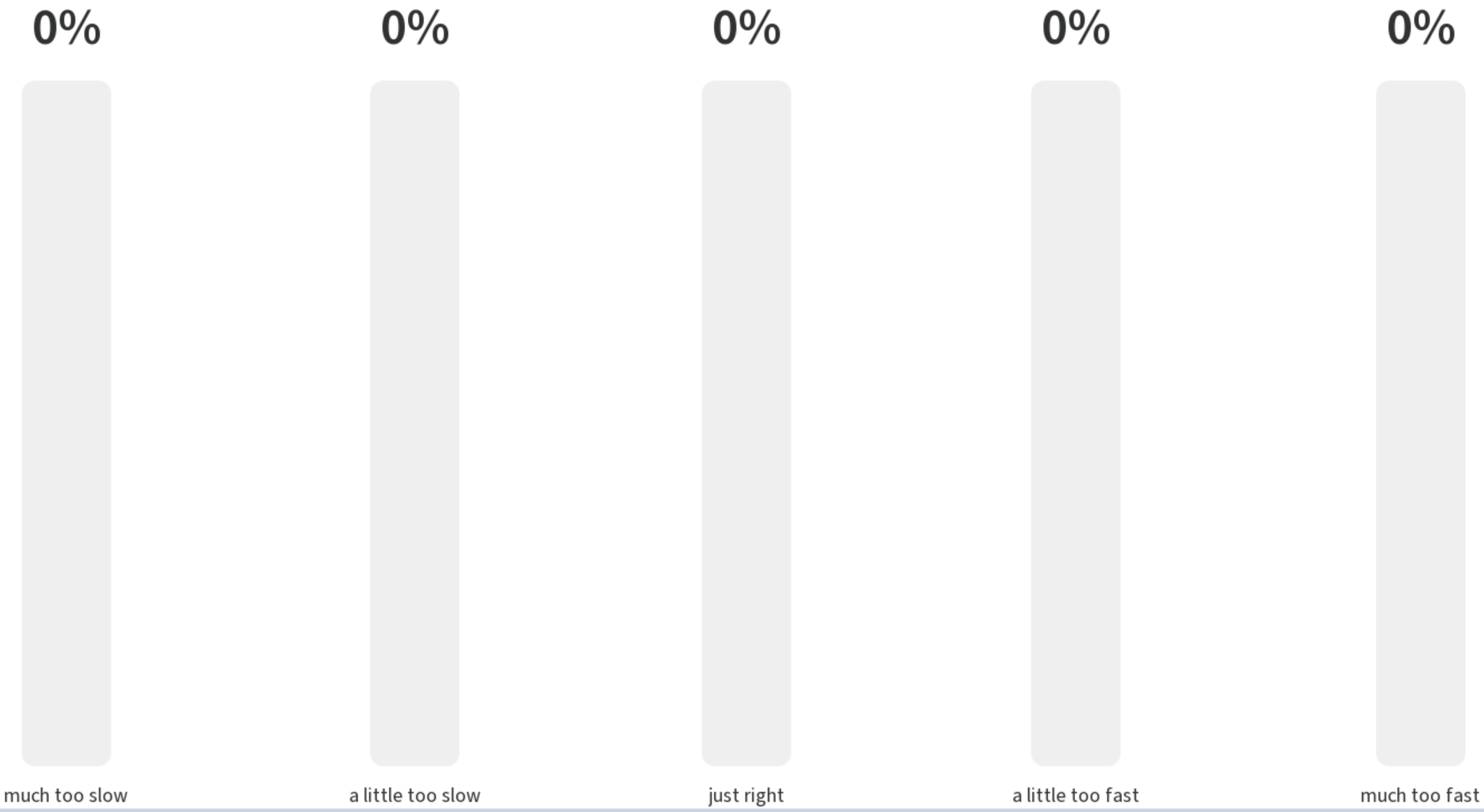
negative (!) relationship between x and y

(once we take into account individual differences)

Plan for today

- Quick recap
- Linear Mixed Model
  - Accommodating non-idependence in data
  - Understanding lmer() syntax
  - A worked example
  - Reporting results
  - Simpsons Paradox

0



Thank you!