

Linear mixed effects models 3



"Your recent Amazon purchases, Tweet score and location history makes you 23.5% welcome here."

Logistics

Things that came up

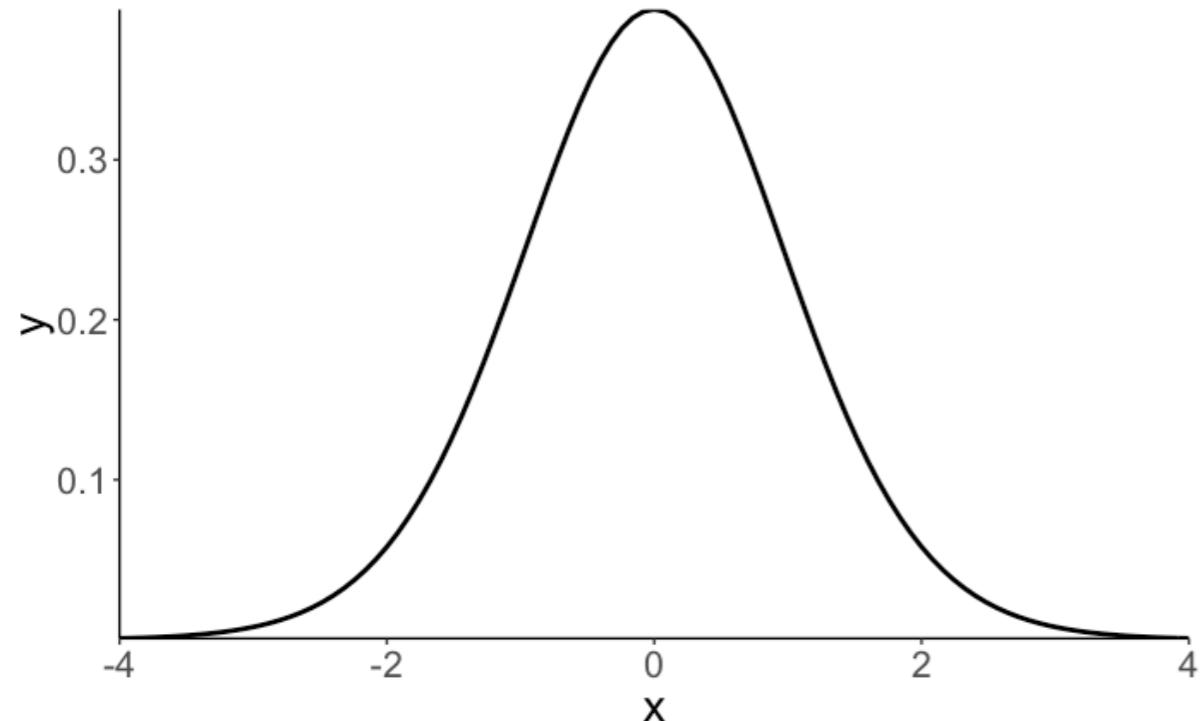
Things that came up

Everything's great, but I feel like **I don't have a good sense of how to tell whether I need a test that is one-sided or alternatively two-sided.** I'm not sure how others feel about this too. It'd be great if there could be some exercises on this, whether it's as homework or as one of your fun in-class games which we could all participate in.

let me clarify

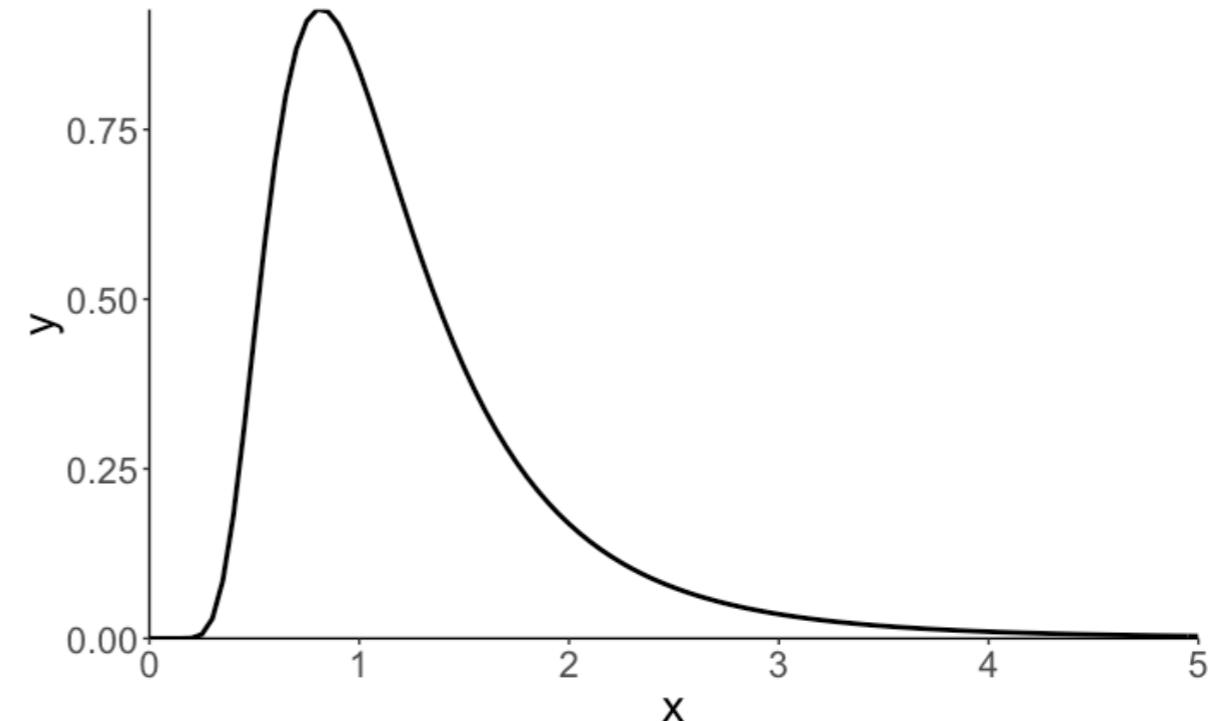
one vs. two tailed tests

two tails



t distribution

one tail



F distribution

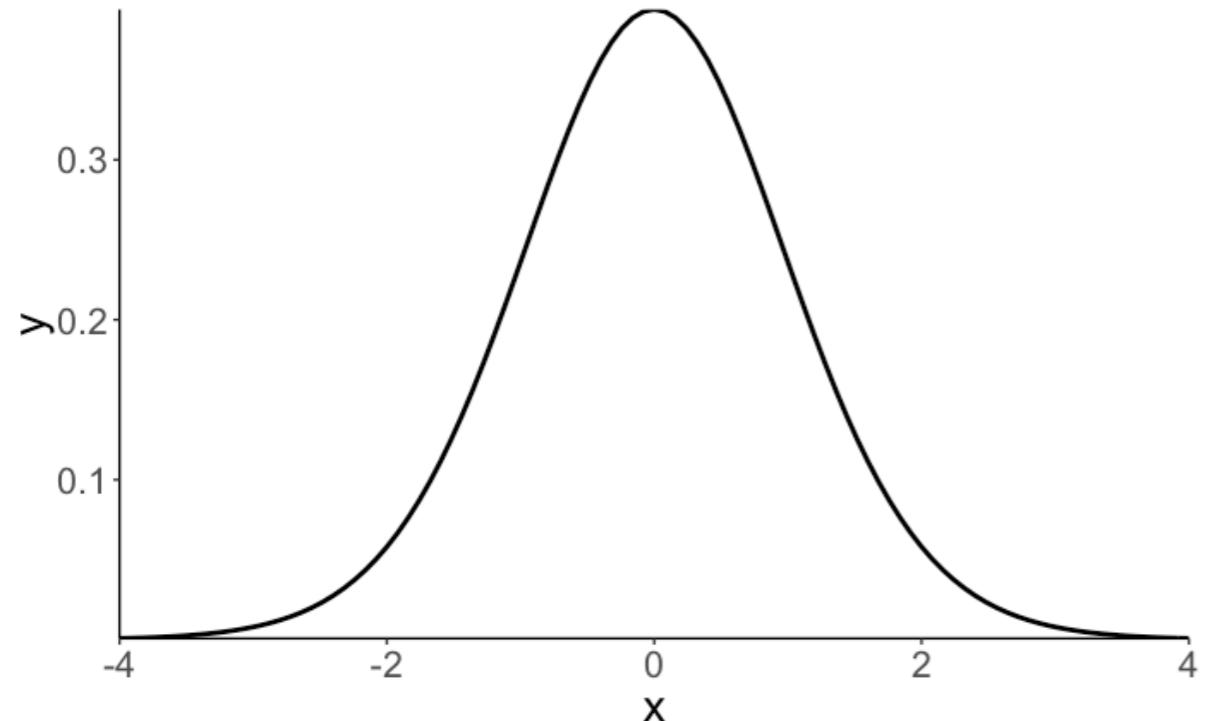
t test

ANOVA

Proportion of reduction in error

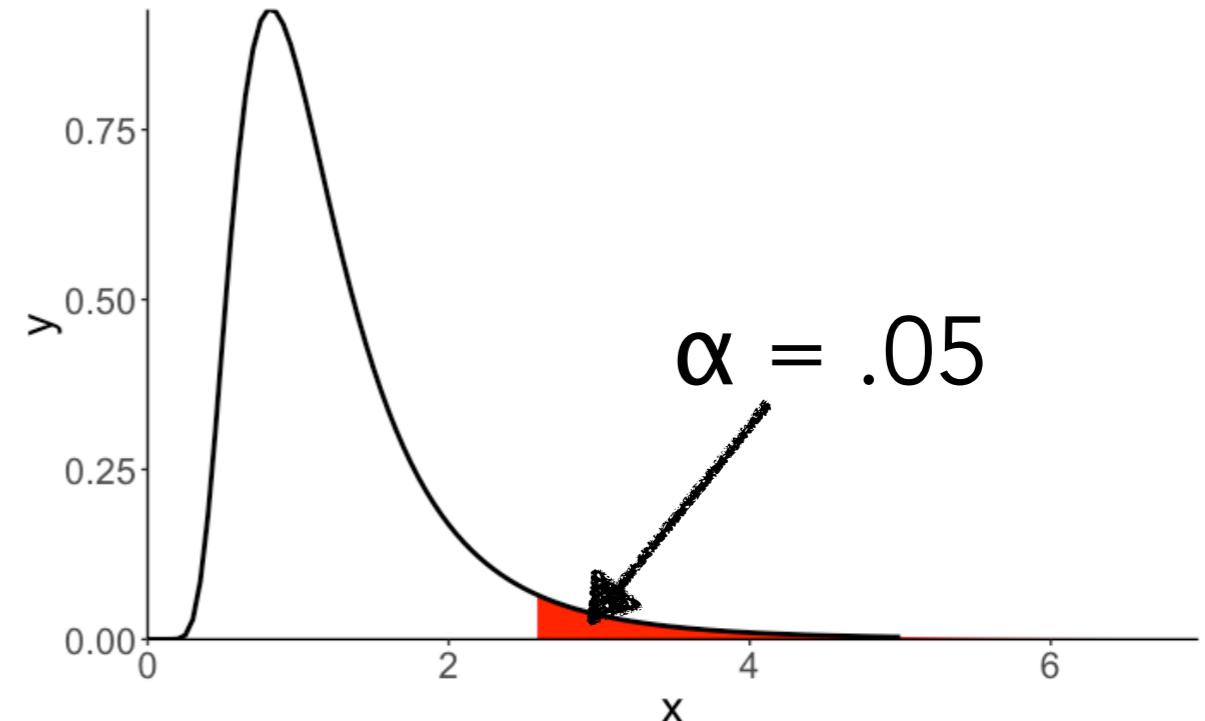
one vs. two tailed tests

two tails



t distribution

one tail



F distribution

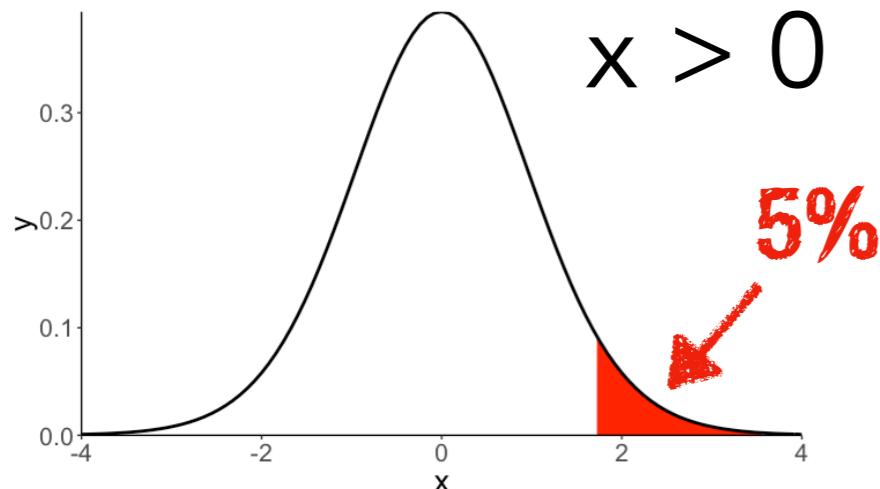
t test

ANOVA

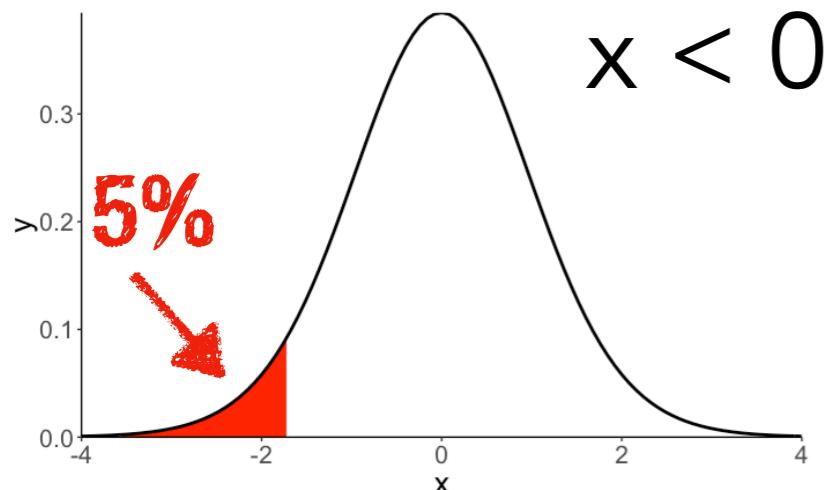
Proportion of reduction in error

one vs. two tailed tests

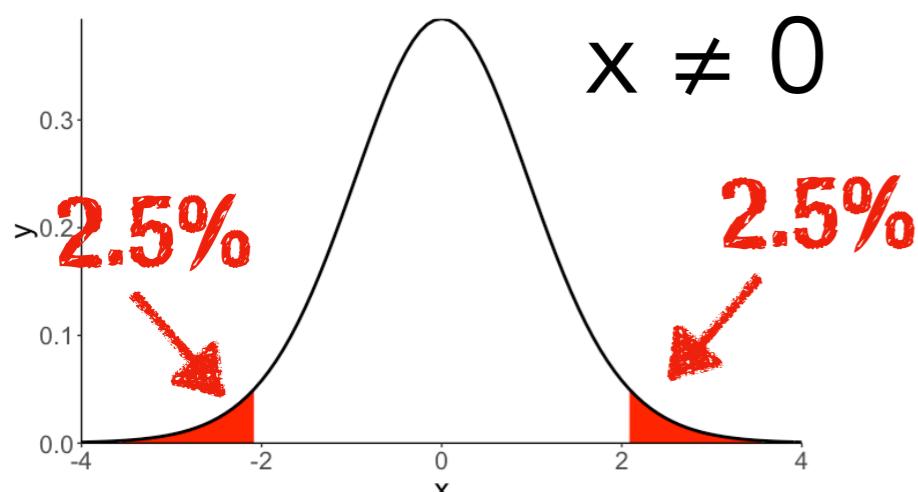
$x = \text{Performance}_{\text{Treatment}} - \text{Performance}_{\text{Control}}$



$H_1: \text{Performance in the treatment group is} \mathbf{\text{better than}} \text{ performance in the control group.}$

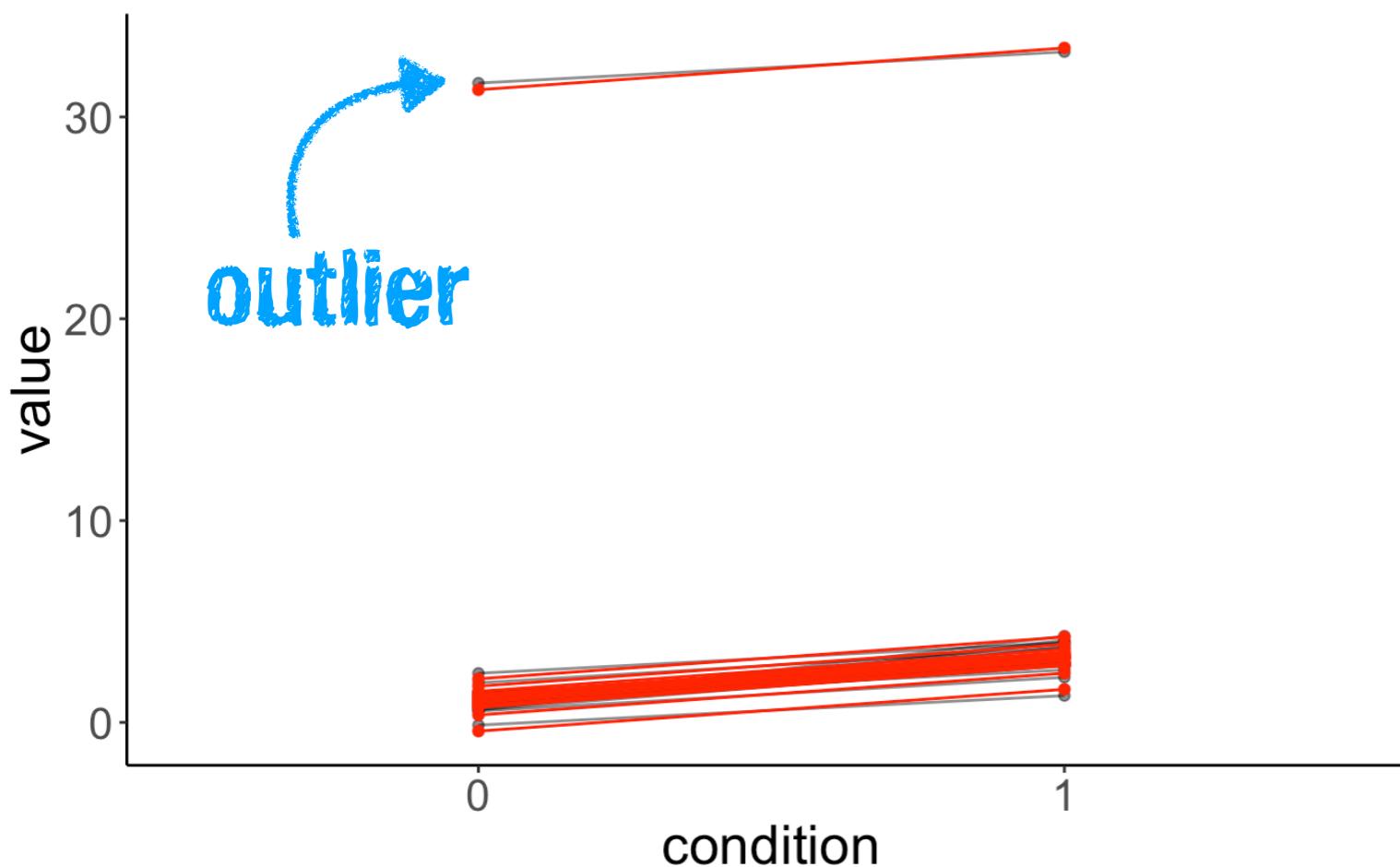


$H_1: \text{Performance in the treatment group is} \mathbf{\text{worse than}} \text{ performance in the control group.}$



$H_1: \text{Performance in the treatment group is} \mathbf{\text{different from}} \text{ performance in the control group.}$

Outliers



```
1 # fit model
2 fit.test = lmer(formula = value ~ 1 + condition + (1 | participant),
3                  data = df.test)
4
5 # simulate data
6 fit.test %>%
7   simulate()
```

simulated data

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

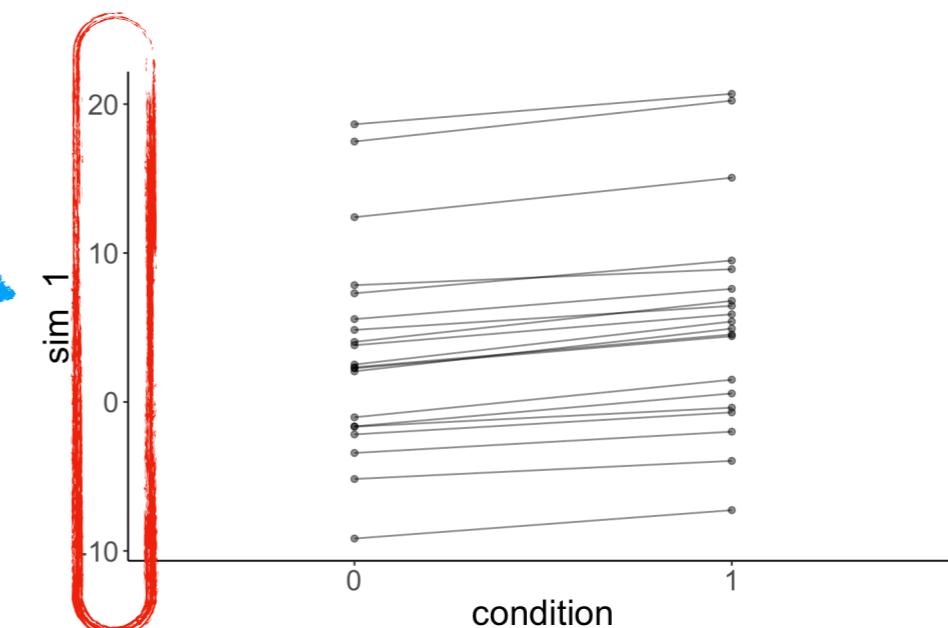
REML criterion at convergence: 171.7

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

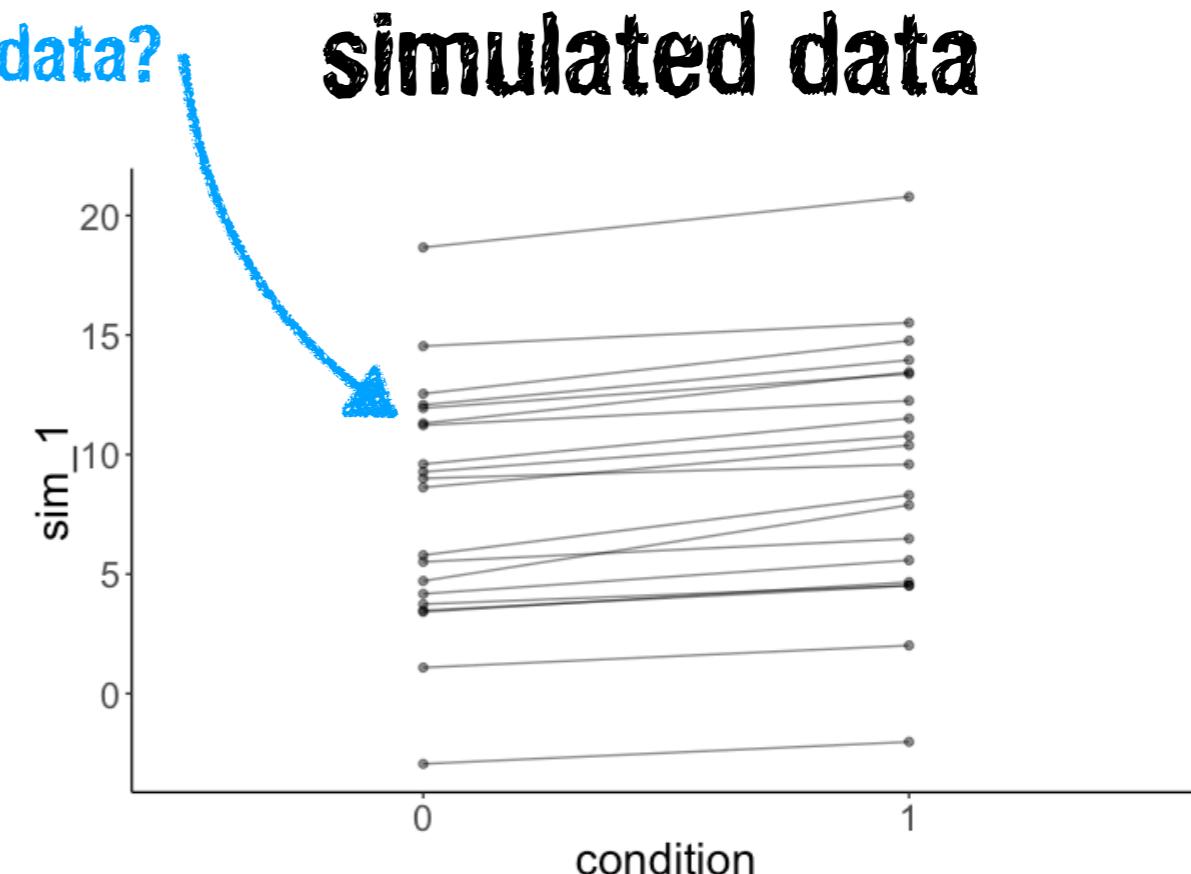
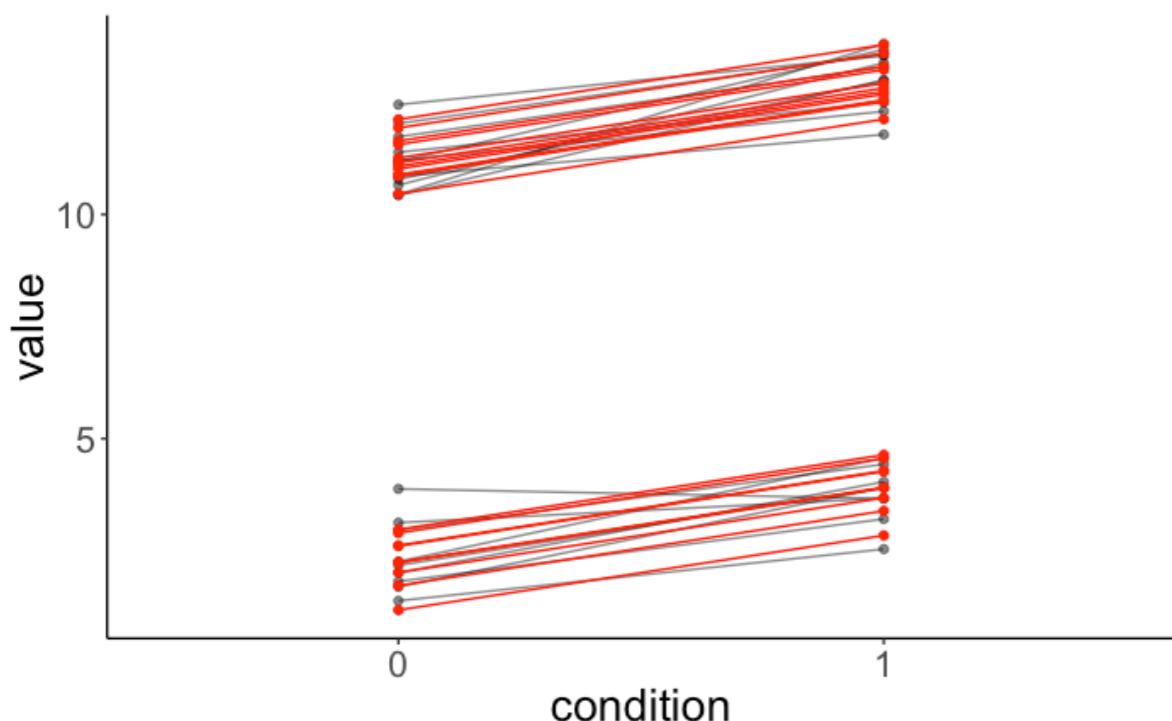
Correlation of Fixed Effects:
          (Intr) condition1 
condition1 -0.049
```



Mixture of participants

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

why are the slopes
not parallel for the
actual data simulated data?



```
Linear mixed model fit by REML ['lmerMod']  
Formula: value ~ 1 + condition + (1 | participant)  
Data: df.mixed  
  
REML criterion at convergence: 165.6  
  
Scaled residuals:  
    Min     1Q   Median     3Q    Max  
-1.6437 -0.4510 -0.0246  0.4987  1.5265  
  
Random effects:  
Groups   Name        Variance Std.Dev.  
participant (Intercept) 21.5142  4.6383  
Residual             0.3521  0.5934  
Number of obs: 40, groups: participant, 20  
  
Fixed effects:  
            Estimate Std. Error t value  
(Intercept) 7.2229    1.0456  6.908  
condition1  1.6652    0.1876  8.875  
  
Correlation of Fixed Effects:  
  (Intr)  
condition1 -0.090
```

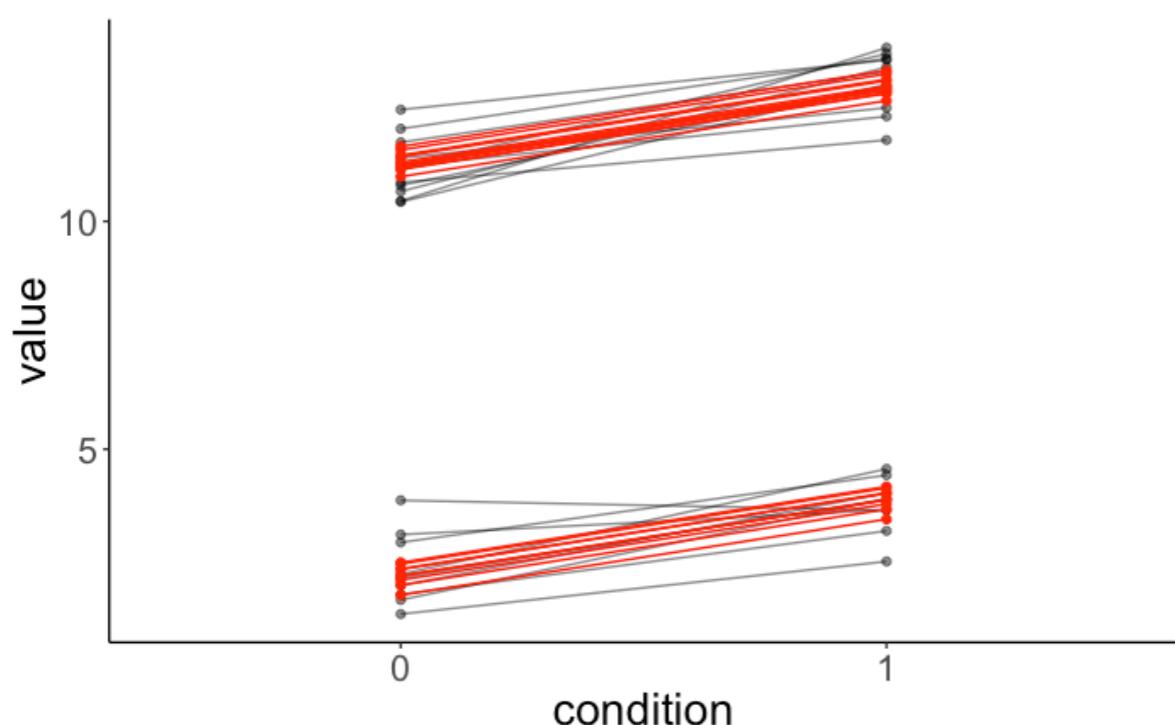
Mixture of participants

explicitly model the different groups

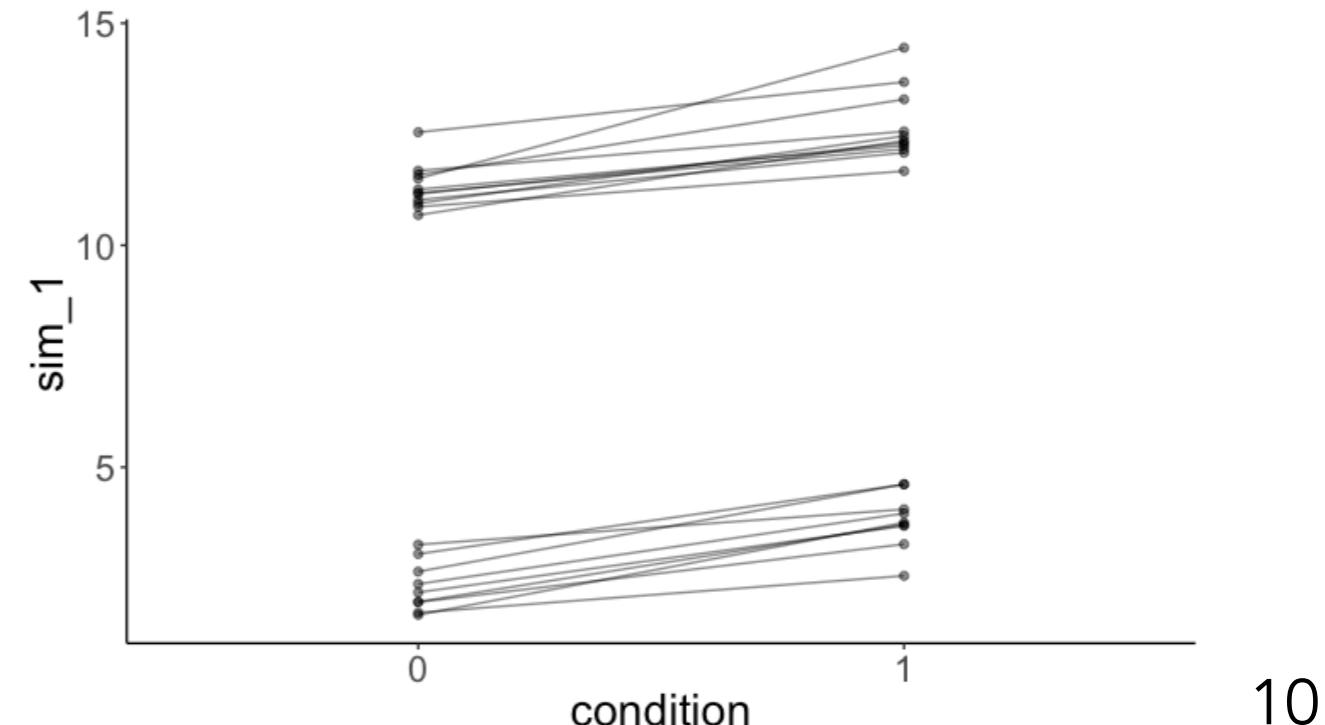
```
lmer(formula = value ~ 1 + group +  
      condition +  
      (1 | participant),  
      data = df.mixed)
```

```
Linear mixed model fit by REML ['lmerMod']  
Formula: value ~ 1 + group + condition + (1 | participant)  
Data: df.mixed  
  
REML criterion at convergence: 83.7  
  
Scaled residuals:  
    Min     1Q Median     3Q    Max  
-1.56168 -0.69876  0.05887  0.50419  2.30259  
  
Random effects:  
Groups   Name        Variance Std.Dev.  
participant (Intercept) 0.1147  0.3387  
Residual             0.3521  0.5954  
Number of obs: 40, groups: participant, 20  
  
Fixed effects:  
            Estimate Std. Error t value  
(Intercept) -6.8299    0.4055 -16.842  
group        9.0663    0.2424  37.409  
condition1   1.6652    0.1876   8.875  
  
Correlation of Fixed Effects:  
          (Intr) group  
group     -0.926  
condition1 -0.231  0.000
```

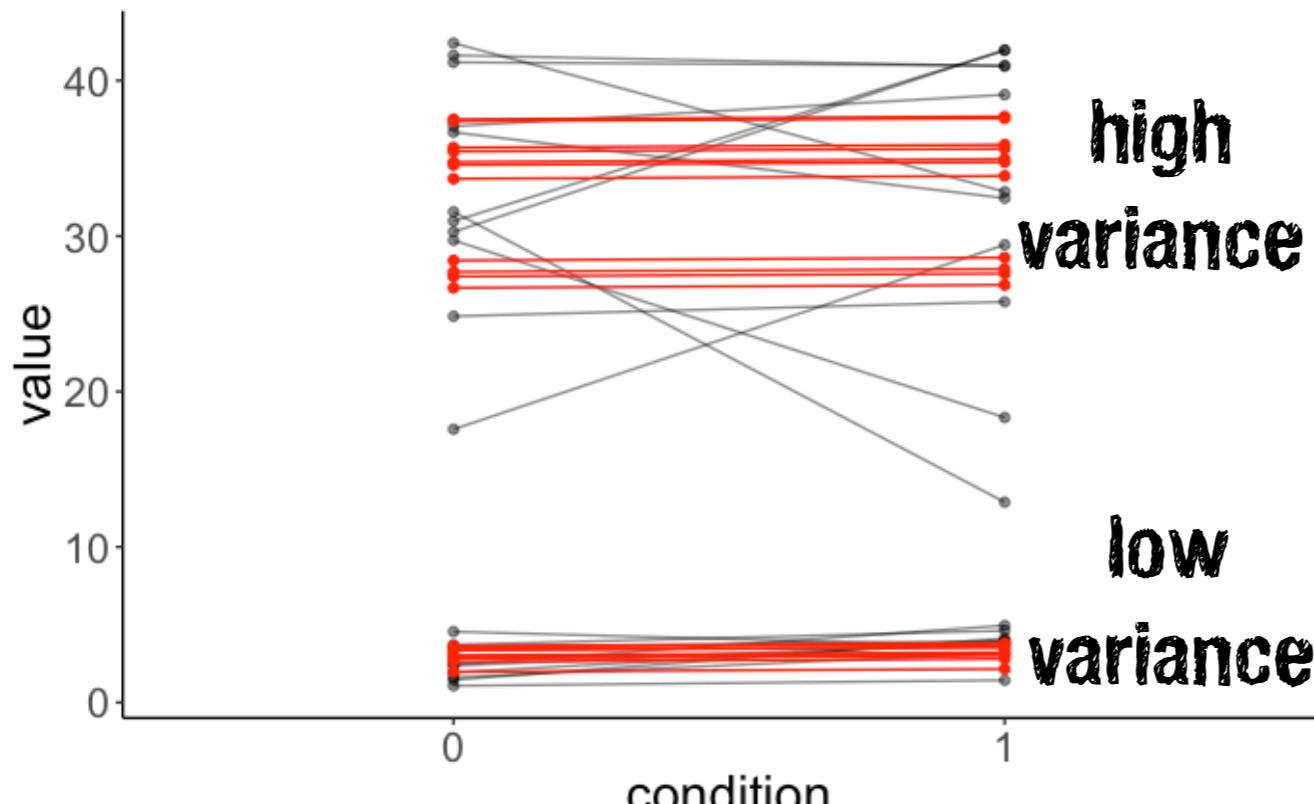
actual data



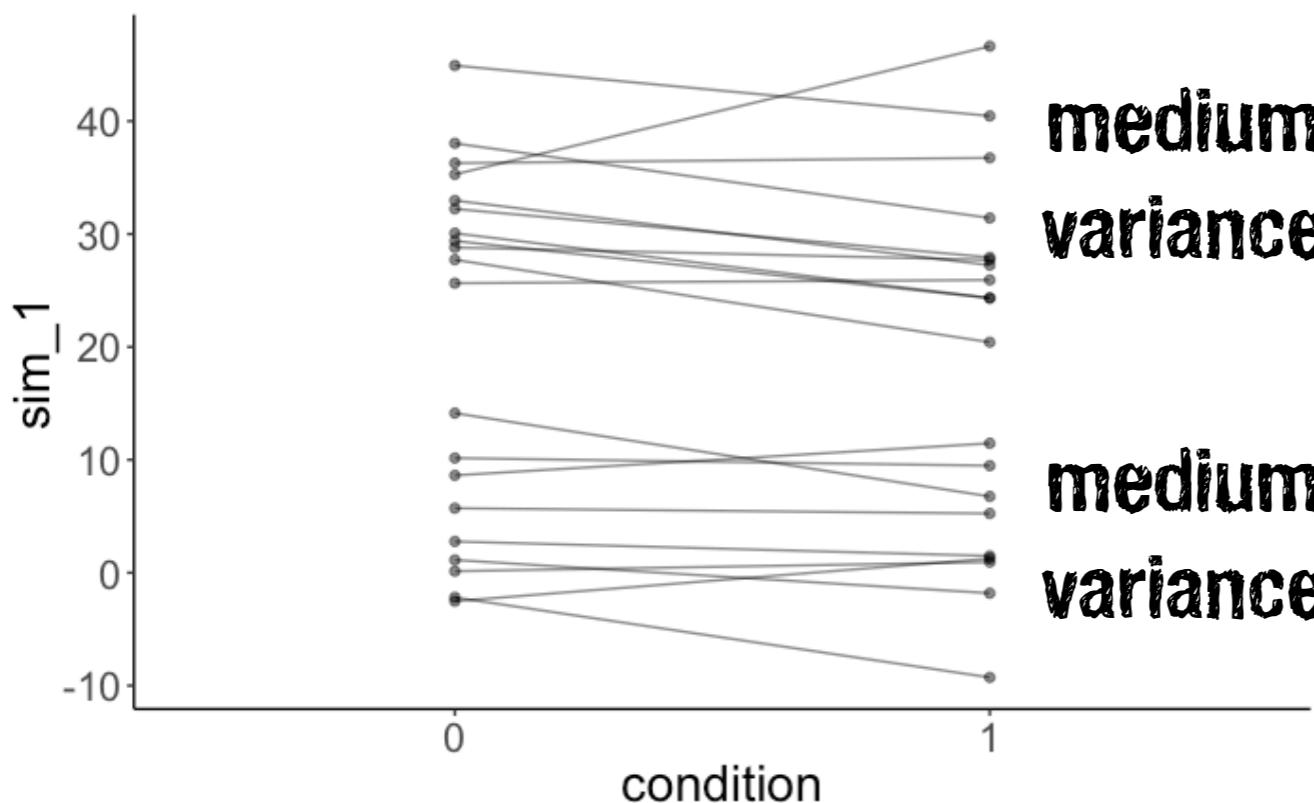
simulated data



Problem case: Heterogeneity in variance between groups



simulated data



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

REML criterion at convergence: 250

Scaled residuals:

Min	1Q	Median	3Q	Max
-2.70344	-0.21278	0.07355	0.43873	1.39493

Random effects:

Groups	Name	Variance	Std.Dev.
participant	(Intercept)	17.60	4.196
Residual		26.72	5.169

Number of obs: 40, groups: participant, 20

Fixed effects:

	Estimate	Std. Error	t value
(Intercept)	-26.5805	4.1525	-6.401
group	29.6200	2.5010	11.843
condition1	0.1853	1.6346	0.113

Correlation of Fixed Effects:

(Intr)	group
group	-0.934
condition1	-0.197
	0.000

only one normal distribution is used to represent the variance in intercepts

Plan for today

- Linear mixed effects model: A worked example
 - pooling:
 - complete pooling
 - no pooling
 - partial pooling
 - shrinkage
- Bootstrapping linear mixed effects models
- Getting p-values
- Pitfalls in fitting `lmer()`s (and what to do about it)
- Understanding `lmer()` syntax

A worked example

general points about `lmer()`

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

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

let's take a look
at an example

**Tristan Mahr**

Language and data scientist

 [Madison, WI](#) [Email](#) [Twitter](#) [GitHub](#) [Stackoverflow](#) [R Bloggers](#)

Plotting partial pooling in mixed-effects models

In this post, I demonstrate a few techniques for plotting information from a relatively simple mixed-effects model fit in R. These plots can help us develop intuitions about what these models are doing and what “partial pooling” means.

The sleepstudy dataset

For these examples, I’m going to use the `sleepstudy` dataset from the `lme4` package. The outcome measure is reaction time, the predictor measure is days of sleep deprivation, and these measurements are nested within participants—we have 10 observations per participant. I am also going to add two fake participants with incomplete data to illustrate partial pooling.

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

Data set

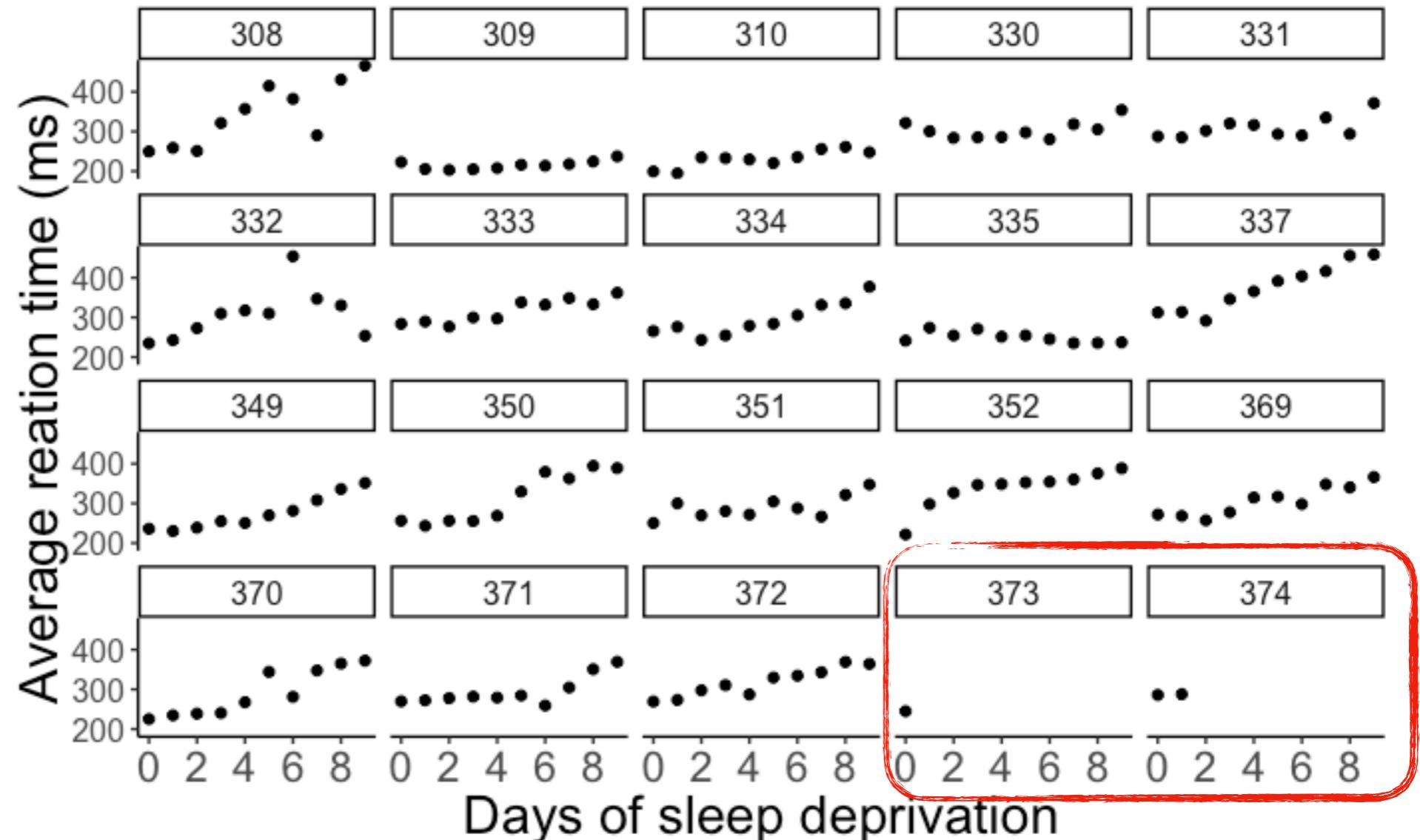
How does sleep deprivation affect reaction time?

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

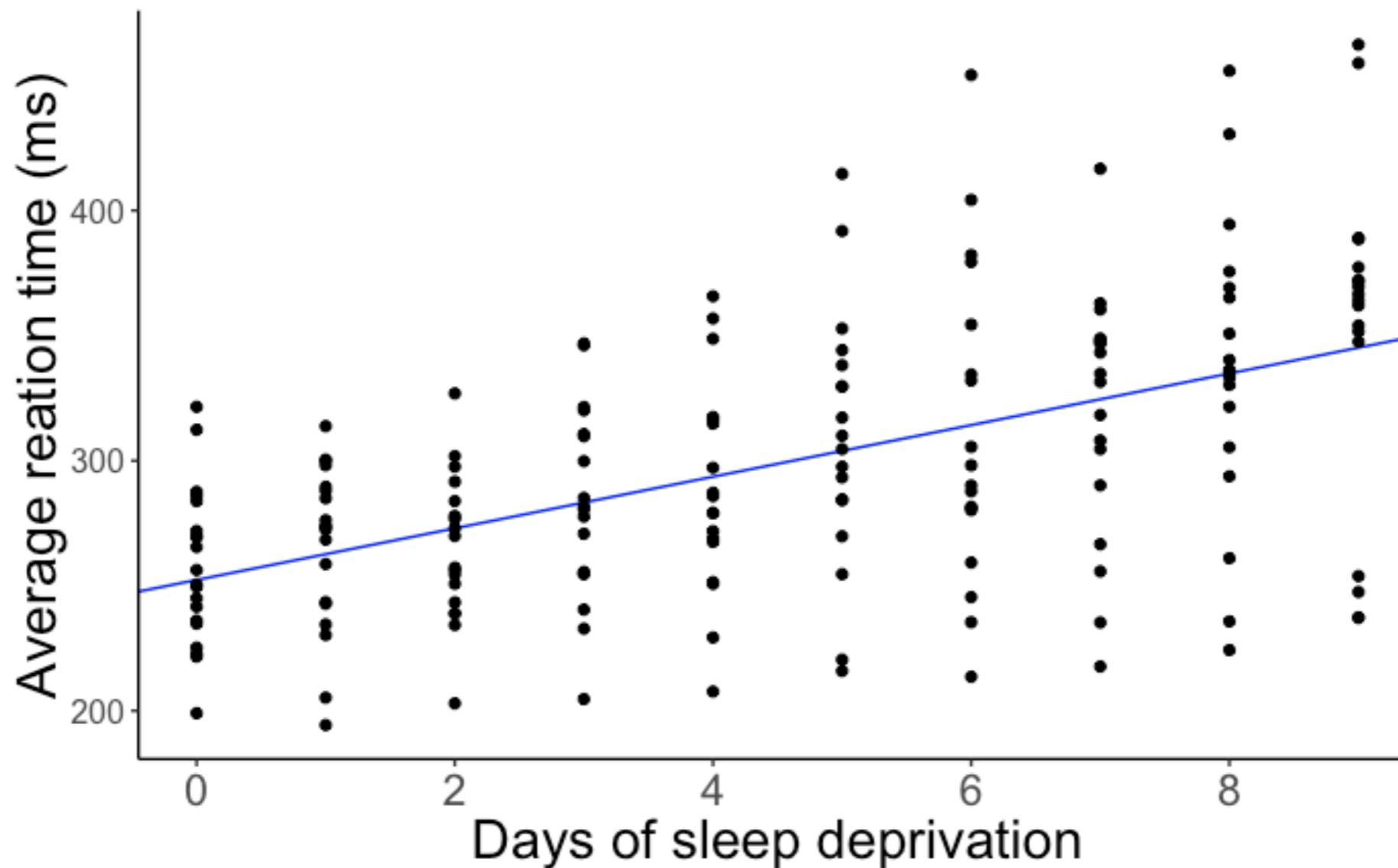
2 with incomplete information

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

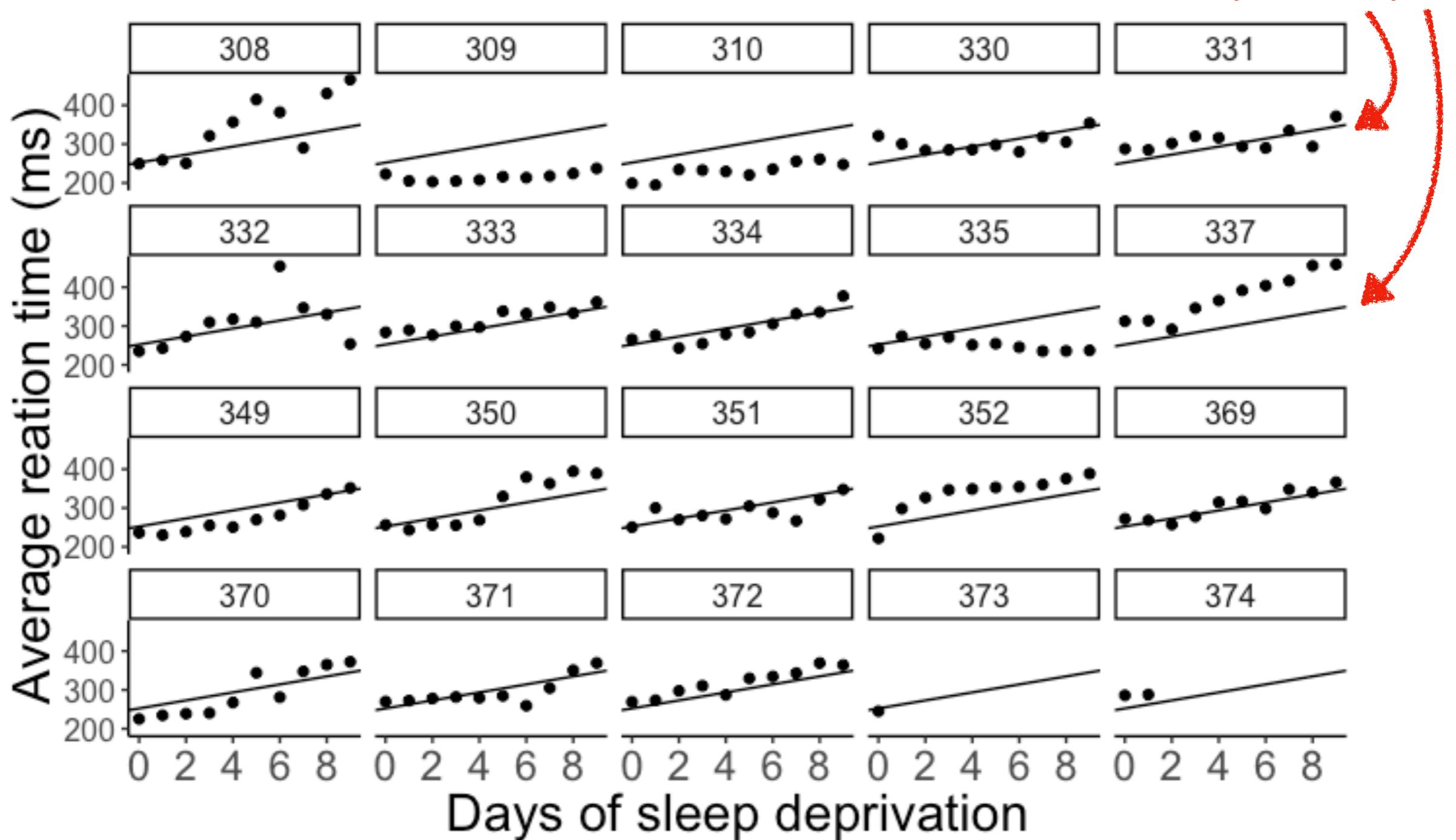
```
lm(formula = reaction ~ days,  
  data = df.sleep)
```



Complete pooling: Fit one global regression

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

same line for
each participant



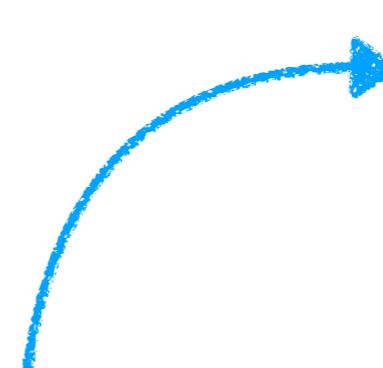
No pooling: Fit separate regressions

```
1 df.no_pooling = df.sleep %>%
2   group_by(subject) %>%
3   nest(days, reaction) %>%
4   mutate(fit = map(data, ~ lm(reaction ~ days, data = .)),
5         params = map(fit, tidy)) %>%
```

	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...
12	350	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 225.8346036363...	list(term = c("(Intercept)", "days"), estimate = c(225.8...
13	351	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 261.1470109090...	list(term = c("(Intercept)", "days"), estimate = c(261.1...
14	352	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 276.3720690909...	list(term = c("(Intercept)", "days"), estimate = c(276.3...
15	369	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 254.9681490909...	list(term = c("(Intercept)", "days"), estimate = c(254.9...
16	370	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 210.4490909090...	list(term = c("(Intercept)", "days"), estimate = c(210.4...
17	371	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 253.6360381818...	list(term = c("(Intercept)", "days"), estimate = c(253.6...
18	372	list(days = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9), reaction = c(2...	list(coefficients = c(`(Intercept)` = 267.0448, days = ...	list(term = c("(Intercept)", "days"), estimate = c(267.0...
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

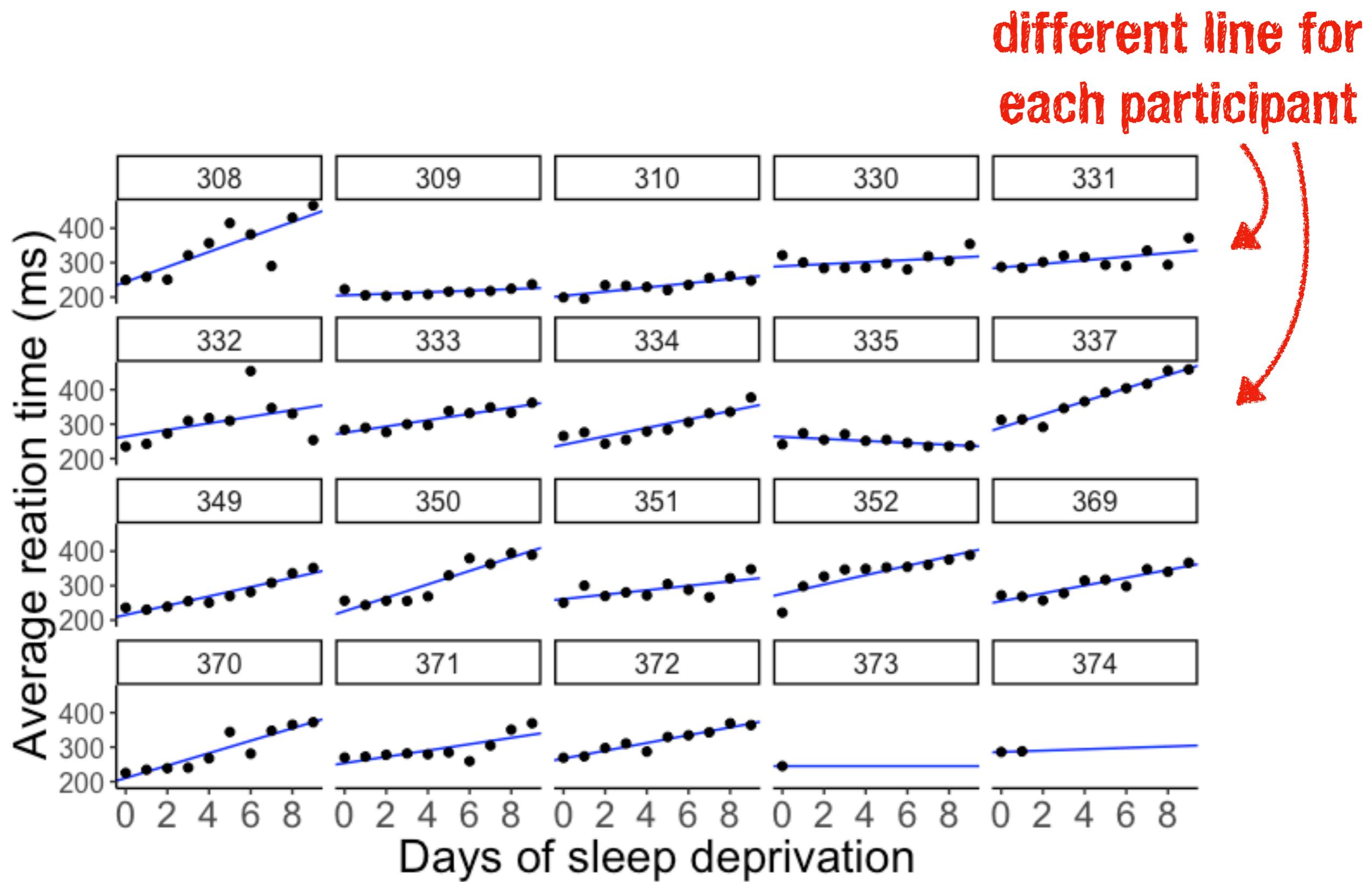
```
1 df.no_pooling = df.sleep %>%
2   group_by(subject) %>%
3   nest(days, reaction) %>%
4   mutate(fit = map(data, ~ lm(reaction ~ days, data = .)),
5         params = map(fit, tidy)) %>%
6   unnest(params) %>%
7   select(subject, term, estimate) %>%
8   complete(subject, term, fill = list(estimate = 0)) %>%
9   spread(term, estimate) %>%
10  clean_names()
```



	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

separate intercept and
slope for each participant

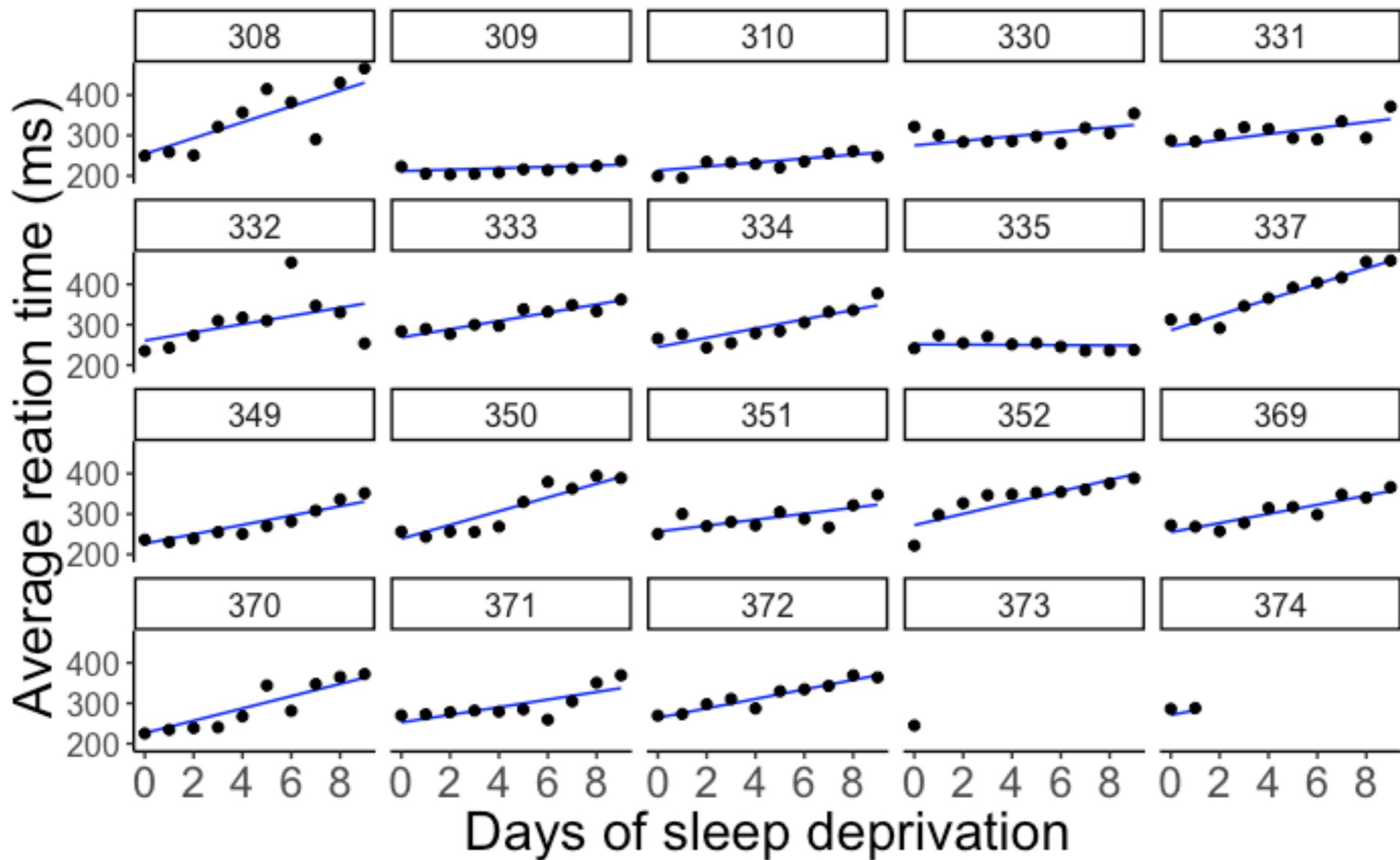
No pooling: Fit separate regressions



Partial pooling: Fit mixed effects model

intercepts and slopes differ
between participants

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

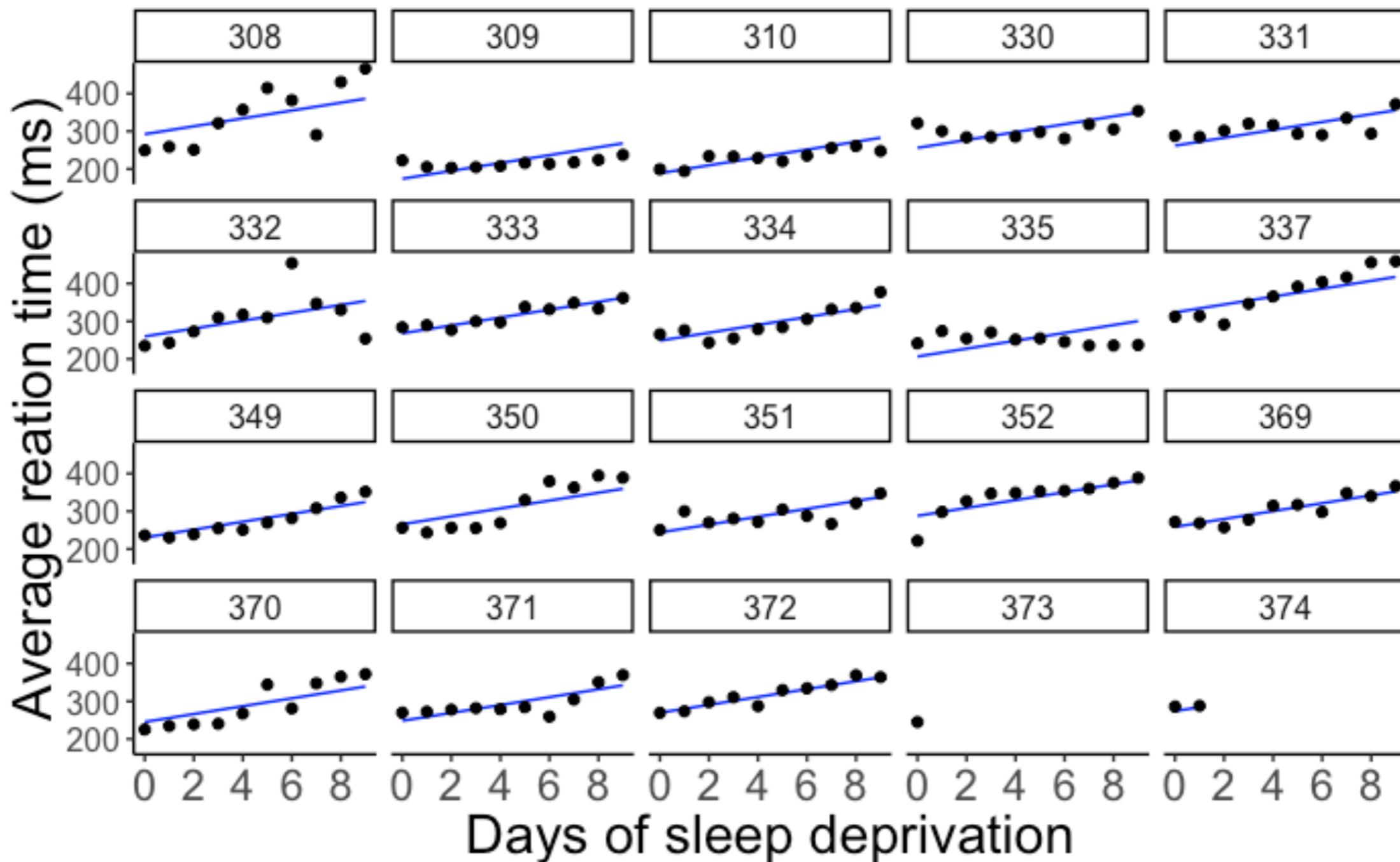


Partial pooling: Fit mixed effects model

only intercepts differ
between participants

random intercept

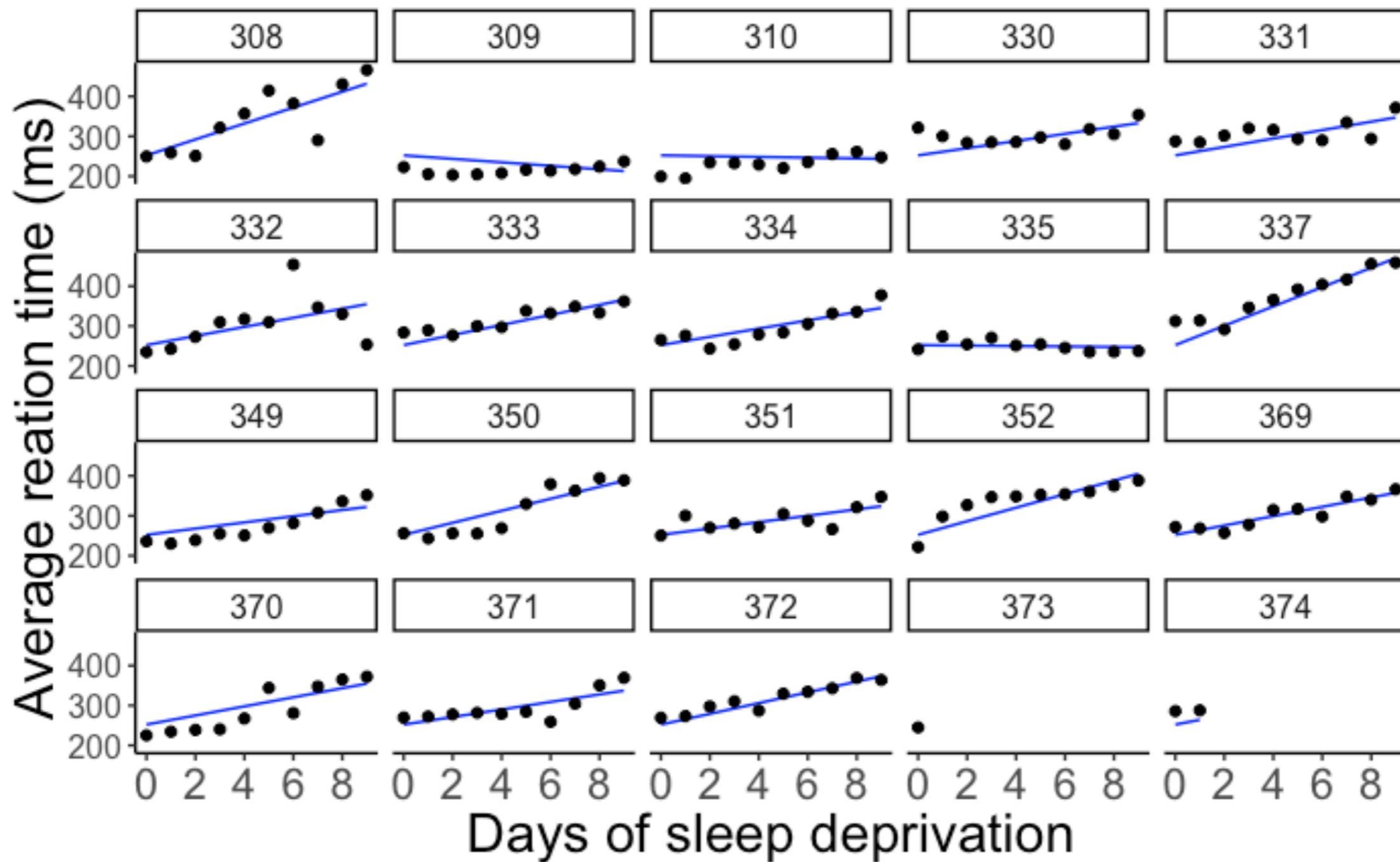
```
lmer (formula = reaction ~ 1 + days + (1 | subject),  
      data = df.sleep)
```



Partial pooling: Fit mixed effects model

only slopes differ between participants

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



Coefficients

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

$(1 \mid \text{subject})$

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

$(0 + \text{days} \mid \text{subject})$

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

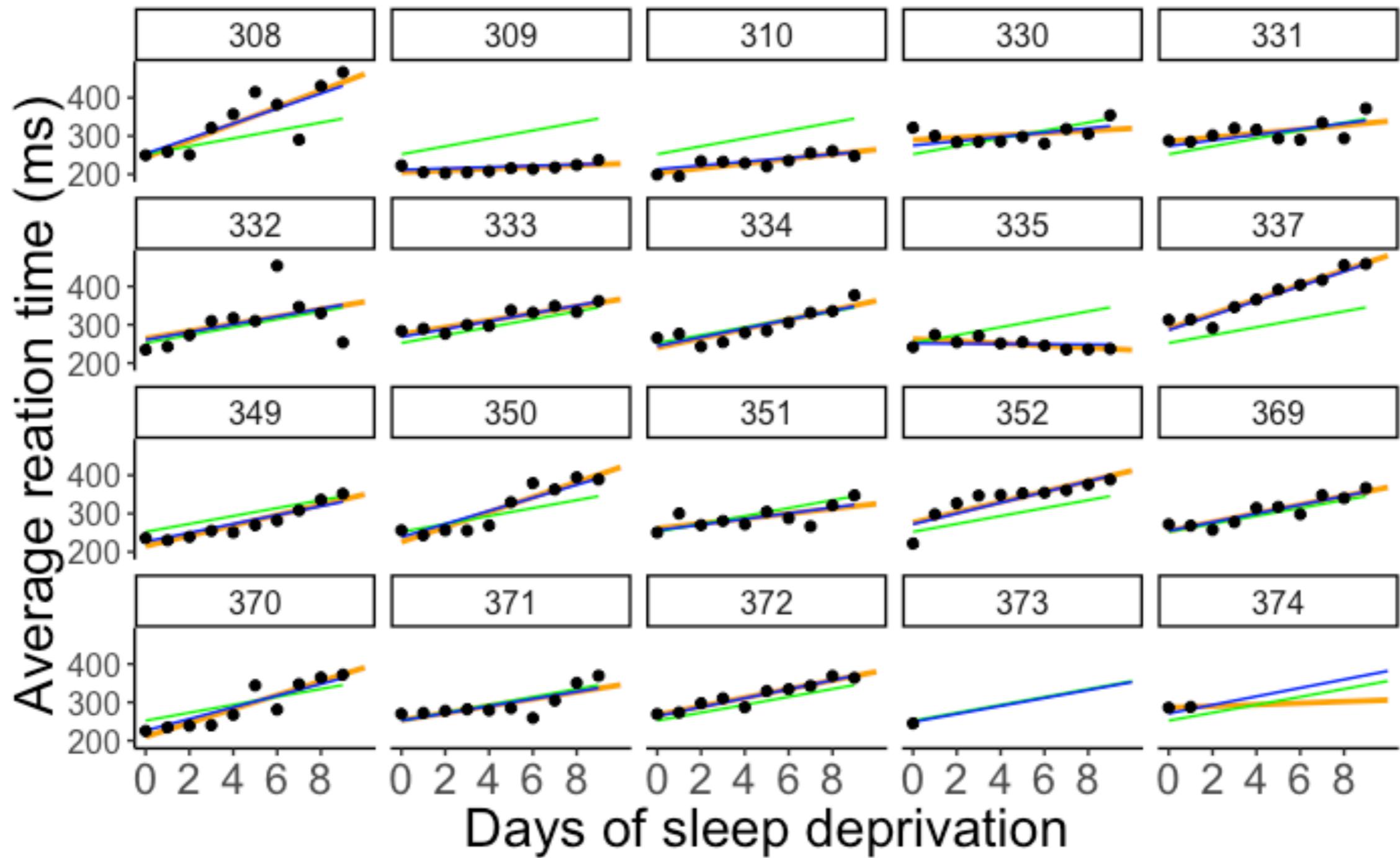
$\dots + (1 + \text{days} \mid \text{subject})$

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

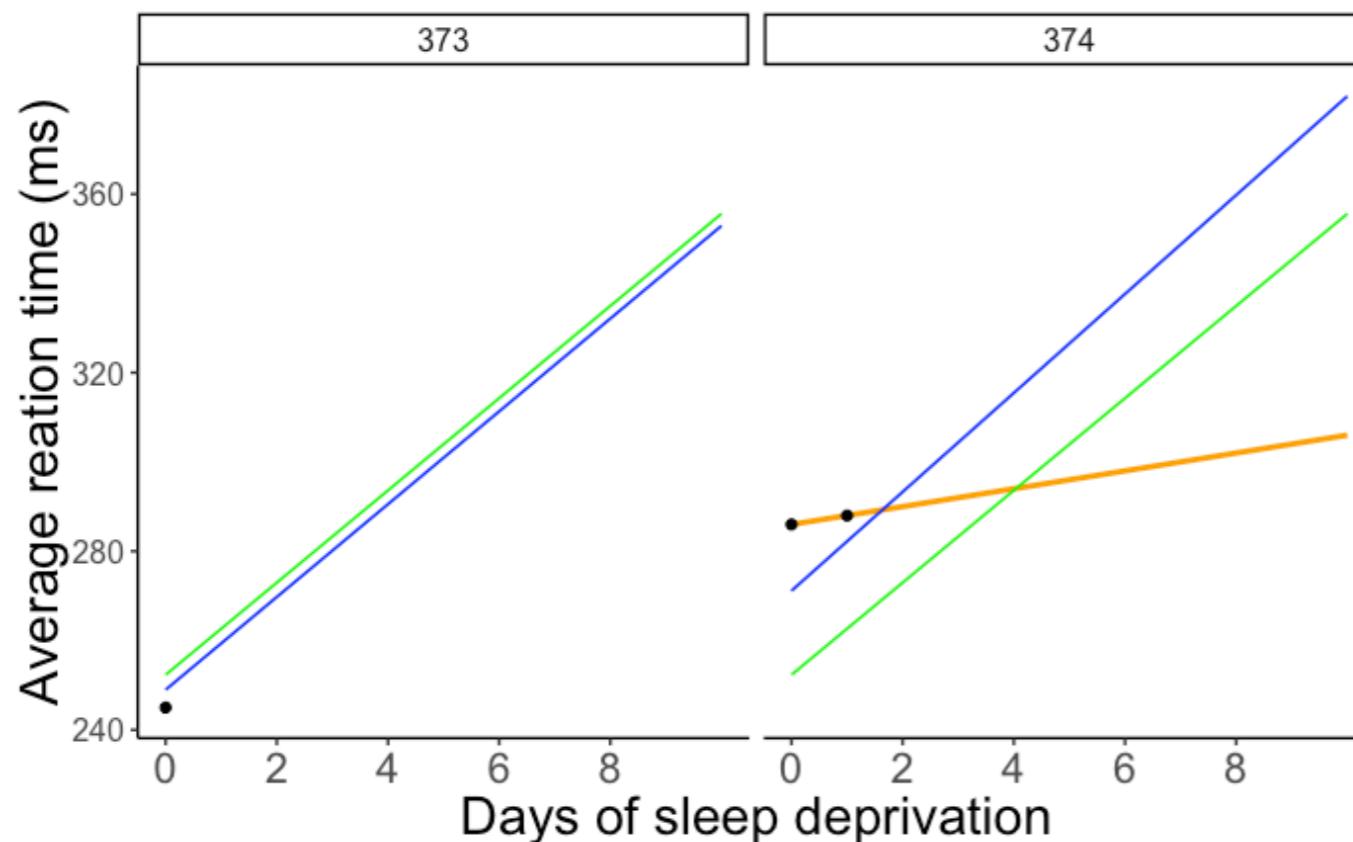
Comparison

complete pooling
no pooling
partial pooling



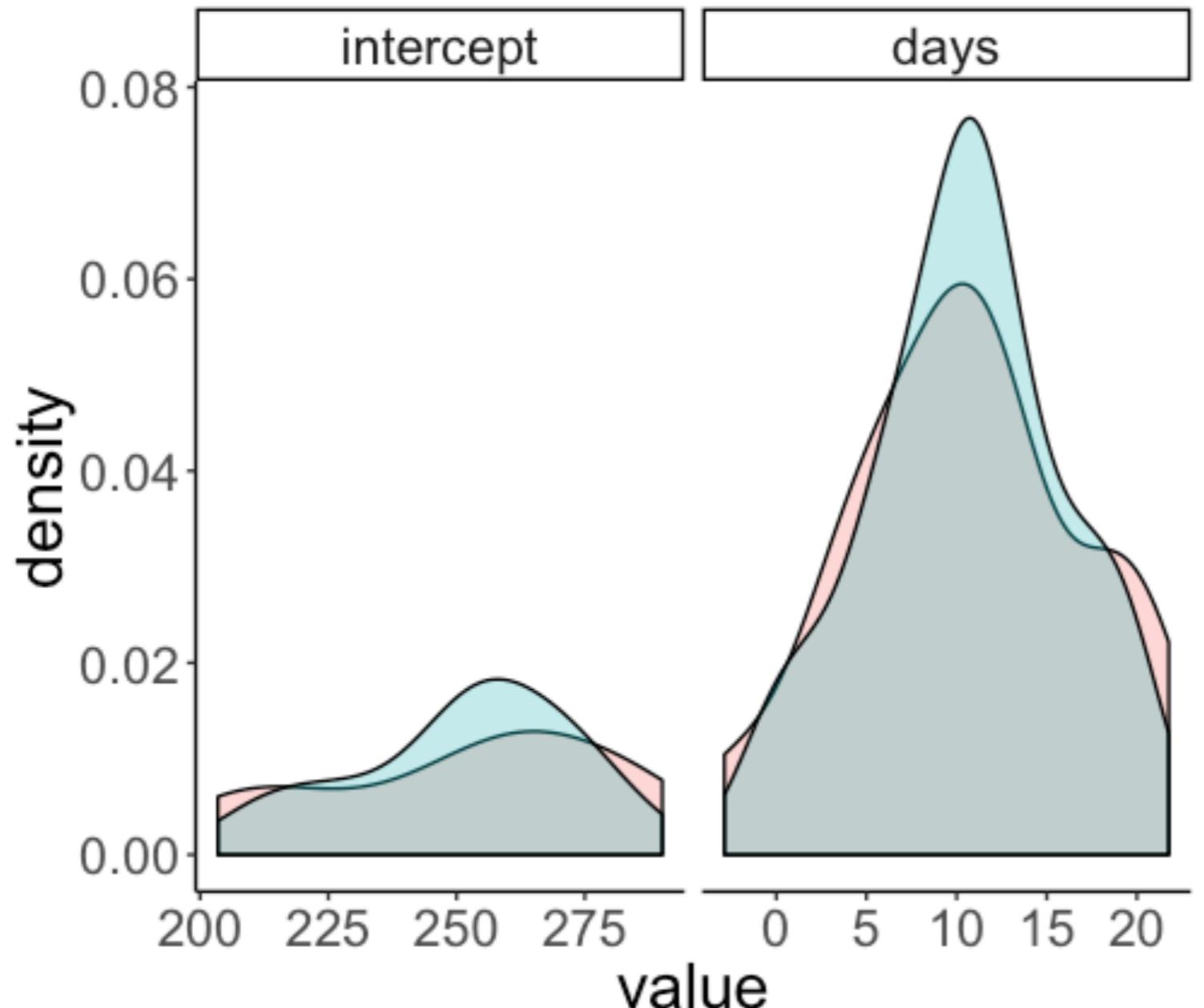
Comparison

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 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



method
no pooling
partial pooling

standard deviation

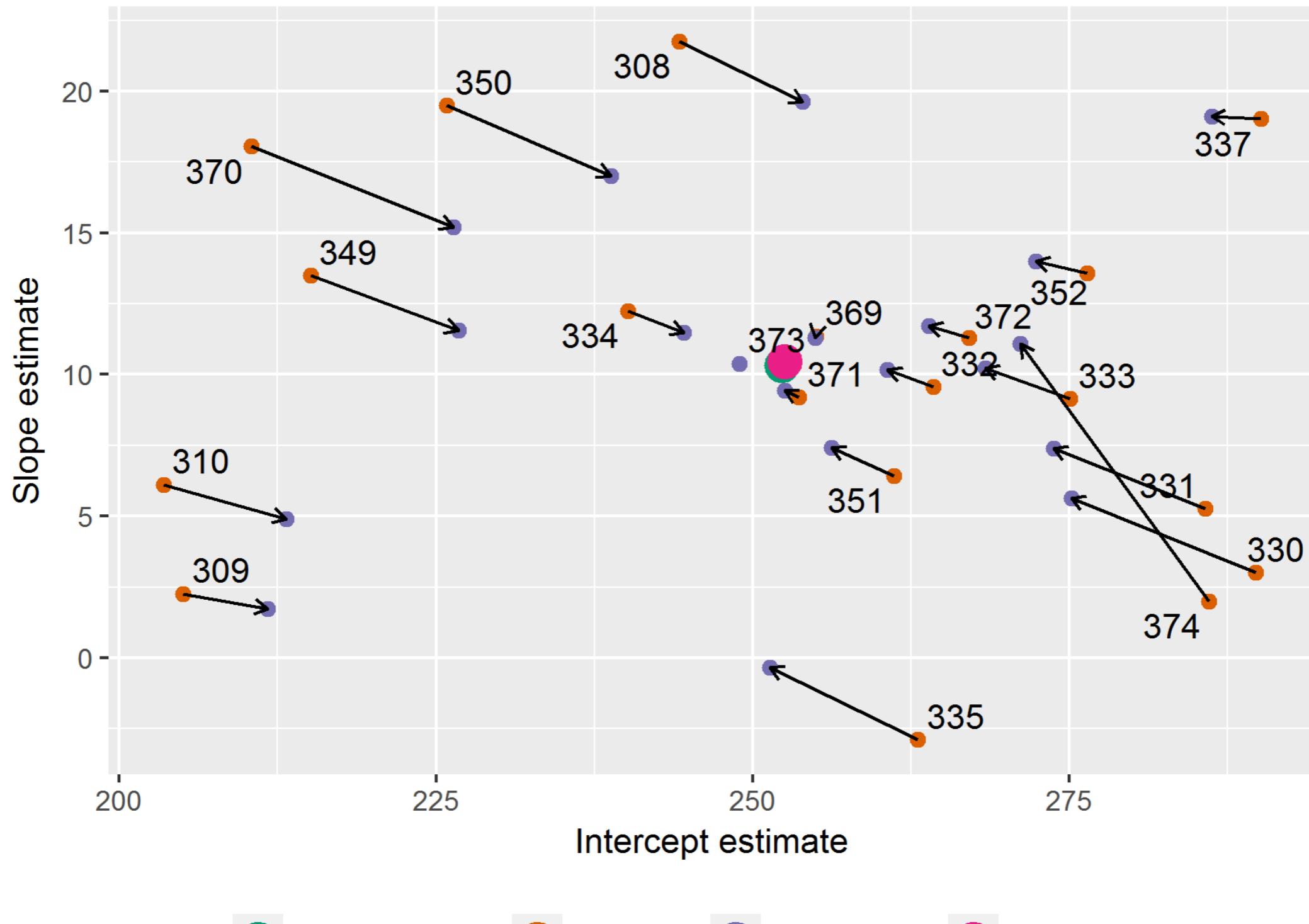
method	intercept	days
no pooling	28.95	6.56
partial pooling	21.59	5.46

variance "shrinks"



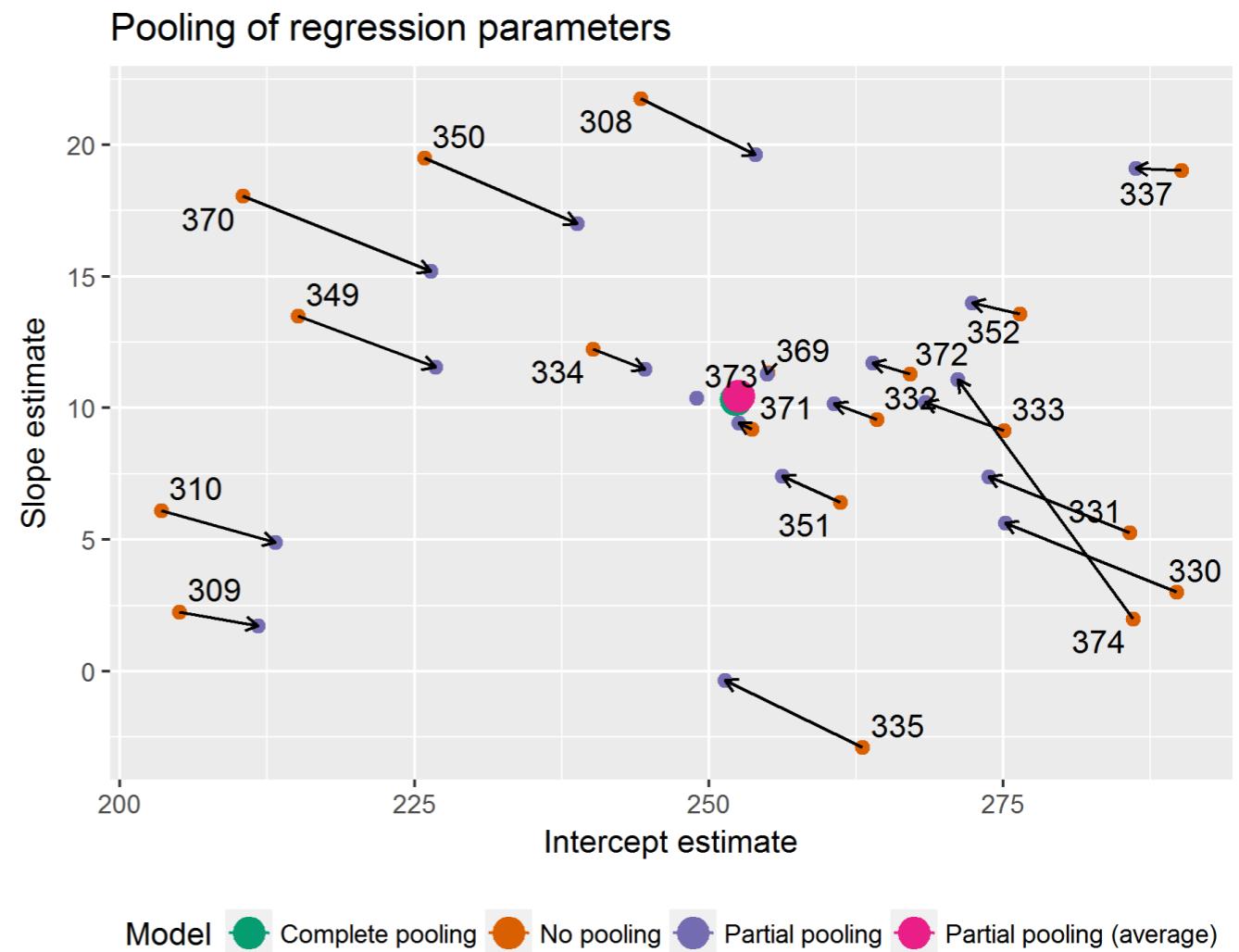
Shrinkage

Pooling of regression parameters



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



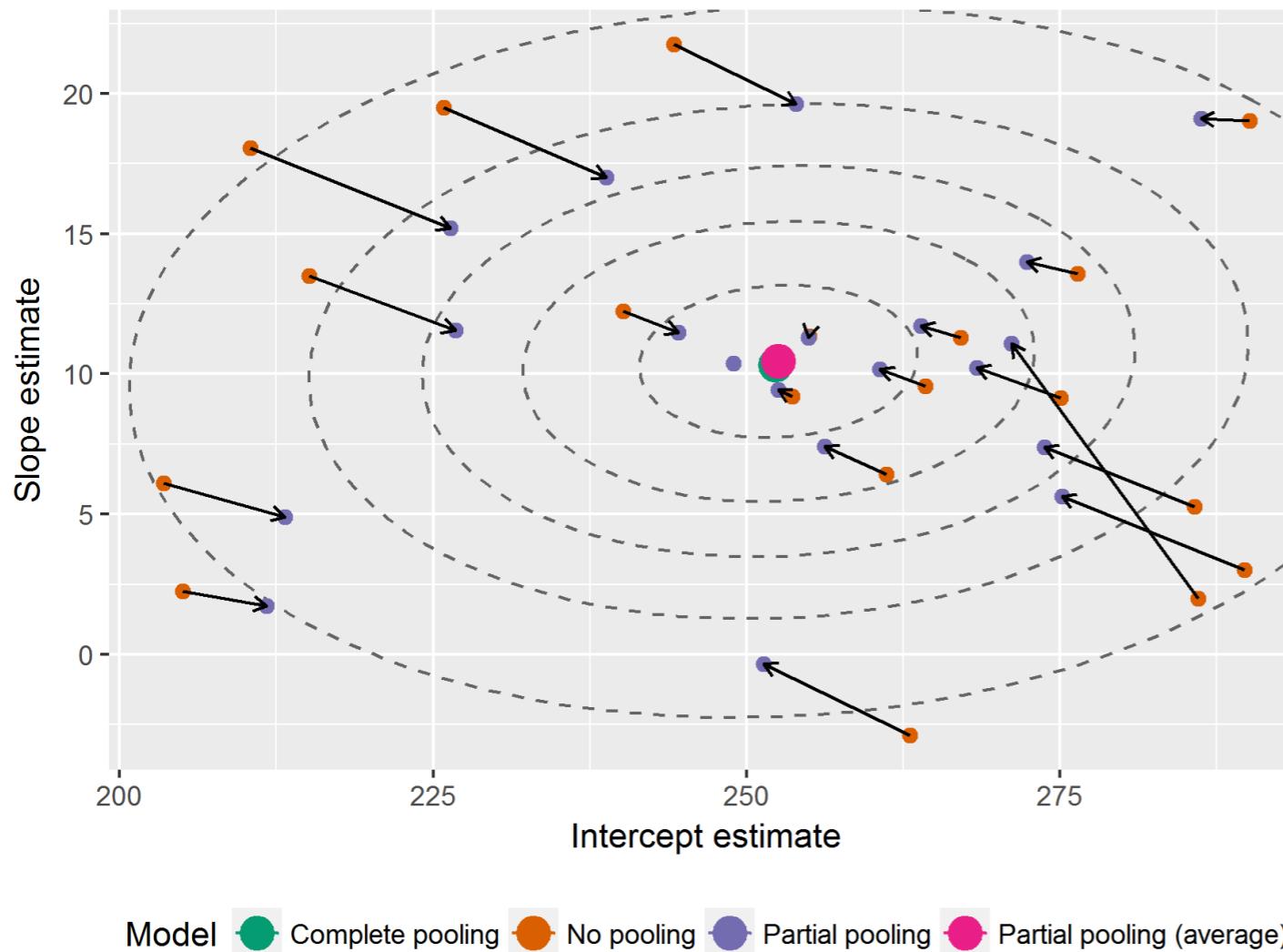
In [the lme4 book](#), Douglas Bates provides an alternative to *shrinkage*:

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

Topographic map of regression parameters

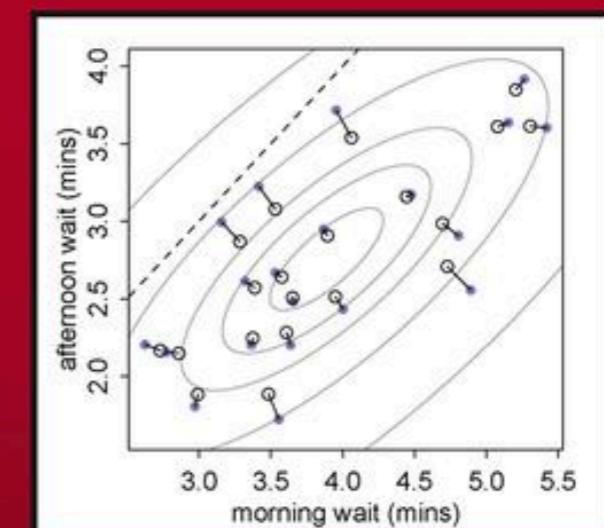


mixed effects model estimates a multi-variate Gaussian to account for (possible) correlations between intercepts and slopes

Texts in Statistical Science

Statistical Rethinking

A Bayesian Course with Examples in R and Stan



Richard McElreath

CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK

Bootstrapping linear mixed effects models

Bootstrapping

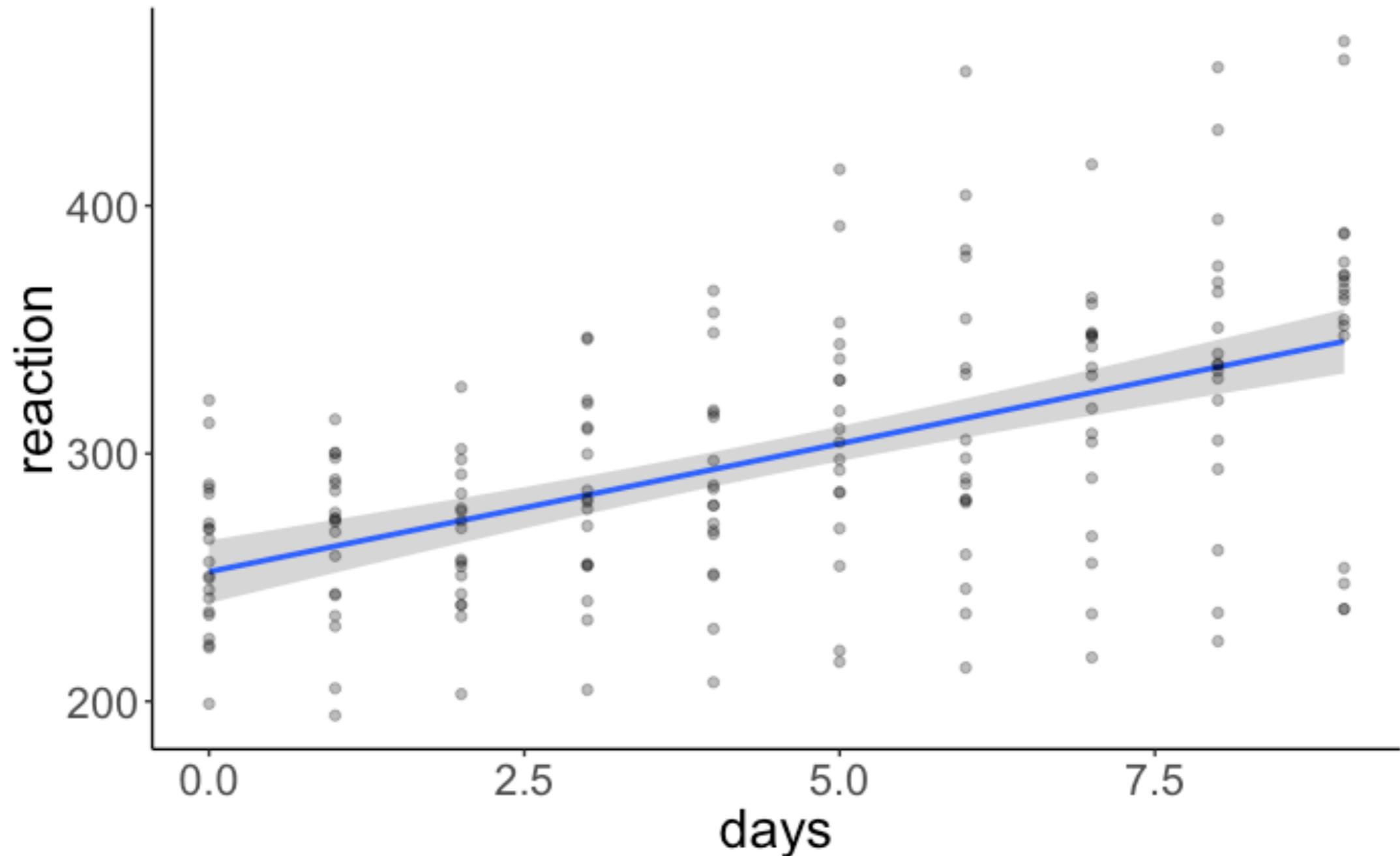
- bootstrapping allows us to determine uncertainty associated with parameter estimates in a model



Bootstrapping a linear model `lm()`

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

(Intercept)	days
252.32070	10.32766



Bootstrapping a linear model `lm()`

```
1 df.boot = df.sleep %>%
2   bootstrap(n = 100,
3             id = "id") %>%
4   mutate(fit = map(strap, ~ lm(formula = reaction ~ 1 + days, data = .)),
5         tidy = map(fit, tidy)) %>%
```

resampled data sets

model fit

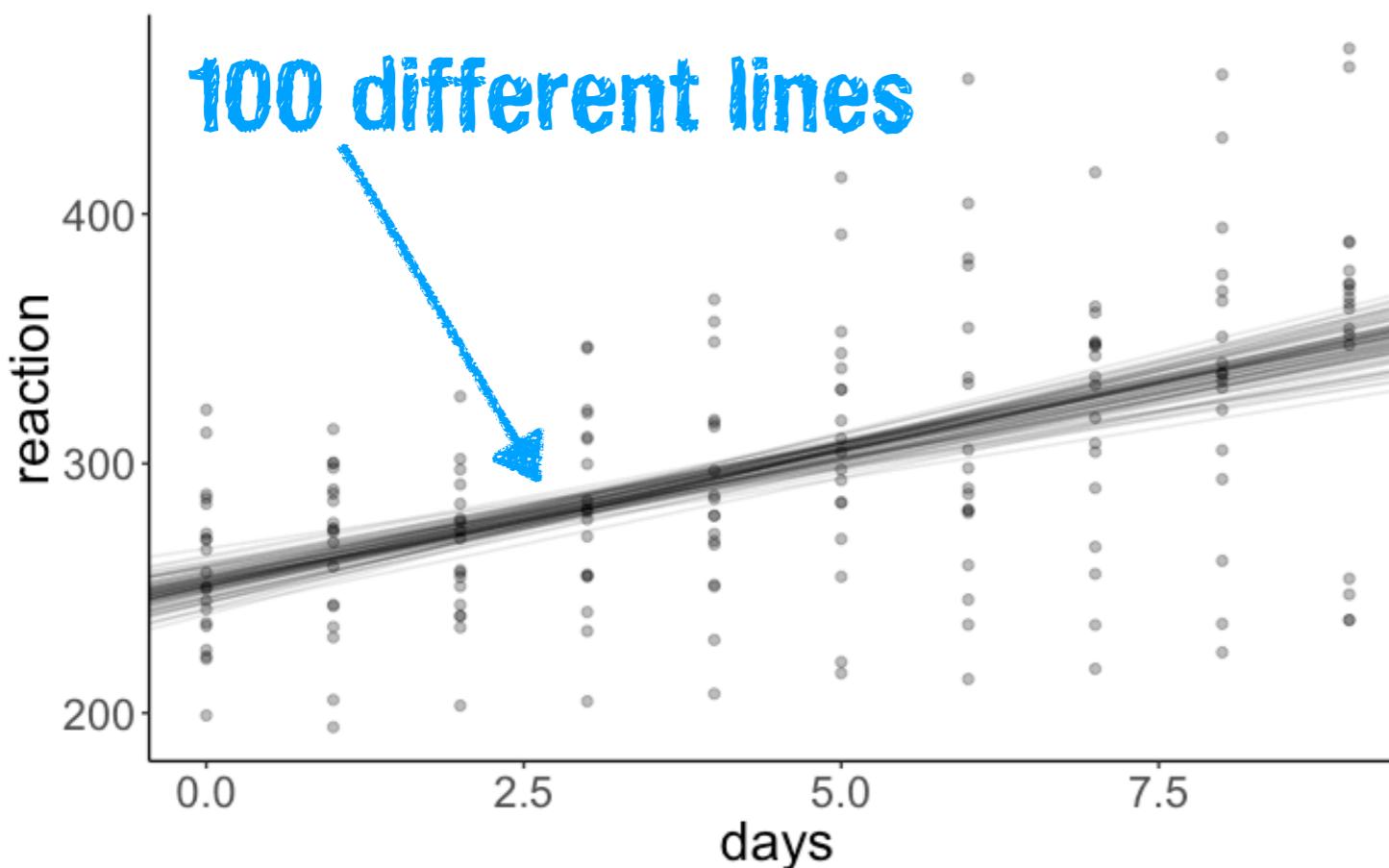
coefficients

strap	id	fit	tidy
1 list(data = list(subject = c("308", "308", "308", "308", ...	001	list(coefficients = c(`(Intercept)` = 257.3113887916...	list(term = c("(Intercept)", "days"), estimate = c(257.3...
2 list(data = list(subject = c("308", "308", "308", "308", ...	002	list(coefficients = c(`(Intercept)` = 249.5628035317...	list(term = c("(Intercept)", "days"), estimate = c(249.5...
3 list(data = list(subject = c("308", "308", "308", "308", ...	003	list(coefficients = c(`(Intercept)` = 249.9739858159...	list(term = c("(Intercept)", "days"), estimate = c(249.9...
4 list(data = list(subject = c("308", "308", "308", "308", ...	004	list(coefficients = c(`(Intercept)` = 256.7315066433...	list(term = c("(Intercept)", "days"), estimate = c(256.7...
5 list(data = list(subject = c("308", "308", "308", "308", ...	005	list(coefficients = c(`(Intercept)` = 252.2794990774...	list(term = c("(Intercept)", "days"), estimate = c(252.2...
6 list(data = list(subject = c("308", "308", "308", "308", ...	006	list(coefficients = c(`(Intercept)` = 243.6775486254...	list(term = c("(Intercept)", "days"), estimate = c(243.6...
7 list(data = list(subject = c("308", "308", "308", "308", ...	007	list(coefficients = c(`(Intercept)` = 253.1340278120...	list(term = c("(Intercept)", "days"), estimate = c(253.1...
8 list(data = list(subject = c("308", "308", "308", "308", ...	008	list(coefficients = c(`(Intercept)` = 243.5002402516...	list(term = c("(Intercept)", "days"), estimate = c(243.5...
9 list(data = list(subject = c("308", "308", "308", "308", ...	009	list(coefficients = c(`(Intercept)` = 246.6558663716...	list(term = c("(Intercept)", "days"), estimate = c(246.6...

Bootstrapping a linear model `lm()`

```
1 df.boot = df.sleep %>%
2   bootstrap(n = 100,
3             id = "id") %>%
4   mutate(fit = map(strap, ~ lm(formula = reaction ~ 1 + days, data = .)),
5         tidy = map(fit, tidy)) %>%
6   unnest(tidy) %>%
7   select(id, term, estimate) %>%
8   spread(term, estimate) %>%
9   clean_names()
```

intercept and slope for
each fitted model



	id	intercept	days
1	001	243.2329	11.610940
2	002	241.7720	11.985003
3	003	257.6160	9.306734
4	004	253.9766	8.481162
5	005	262.3882	7.256231
6	006	259.1082	9.377909
7	007	247.7098	9.589859
8	008	250.6057	9.822907
9	009	247.5809	9.883666
10	010	255.6068	10.351093

Bootstrapping an `lmer()`

- bootstrapping is more complex here because of the random mixed effects structure

```
1 # bootstrapping
2 df.boot = df.sleep %>%
3   bootstrap(n = 100,
4             id = "id") %>%
5   mutate(fit = map(strap, ~ lmer(formula = reaction ~ 1 + days + (1 + days | subject), data = .)),
6         tidy = map(fit, tidy)) %>%
7   unnest(tidy) %>%
8   select(id, term, estimate) %>%
9   spread(term, estimate) %>%
10  clean_names()
```

leads to overconfident results



```
Model failed to converge with max|grad| = 0.00262588 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00484377 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00307712 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00371263 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00288646 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00534272 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00211508 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00679552 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00362321 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00234684 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00322159 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.0107026 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00545384 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00203879 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.0028553 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00512284 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.00279116 (tol = 0.002, component 1)
Model failed to converge with max|grad| = 0.0036683 (tol = 0.002, component 1)
binding factor and character vector, coercing into character vector
binding character and factor vector, coercing into character
vectorbinding character and factor vector, coercing into character vector
binding character and factor vector, coercing into character vector
binding character and factor vector, coercing into character vector
binding character and factor vector, coercing into character vector
factor and c
```

Bootstrapping an `lmer()`

```
1 # fit the model
2 fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days | subject),
3                      data = df.sleep)
4
5 # bootstrap parameter estimates
6 boot.lmer = bootMer(fit.lmer,
7                      FUN = fixef,
8                      nsim = 100)
9
10 # compute confidence interval
11 boot.ci(boot.lmer, index = 2, type = "perc")
```

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 100 bootstrap replicates

CALL :

`boot.ci(boot.out = boot.lmer, type = "perc", index = 2)`

Intervals :

Level	Percentile
95%	(6.45, 14.26)

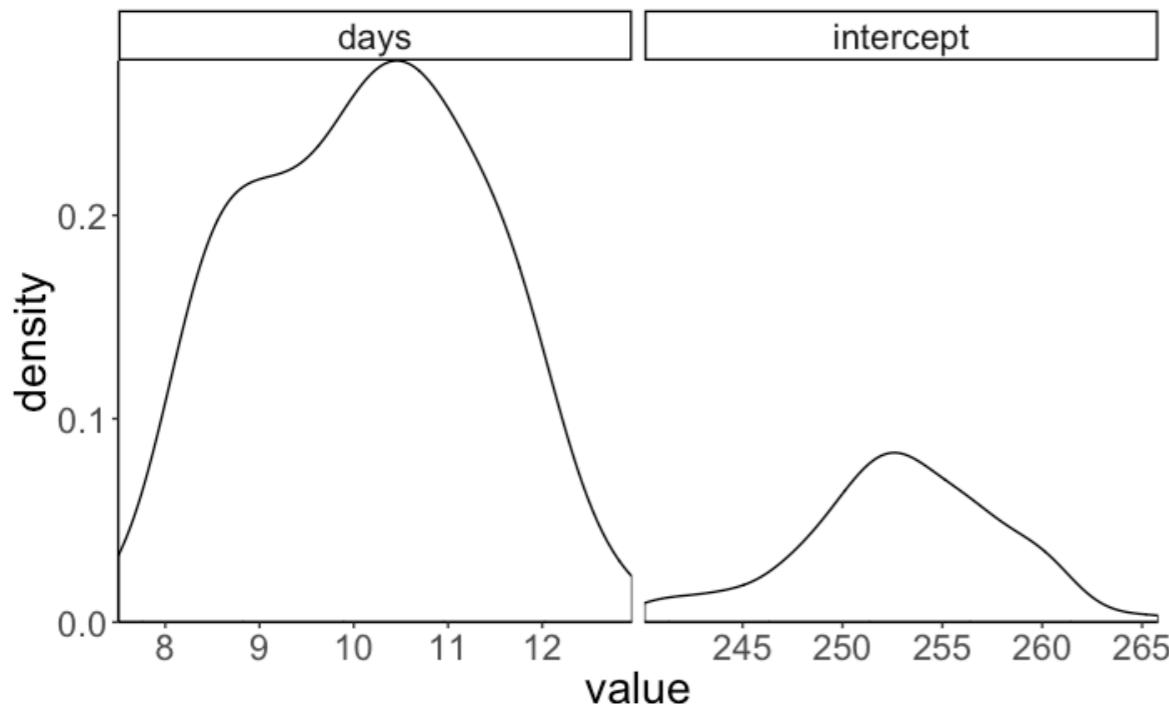
Calculations and Intervals on Original Scale

Some percentile intervals may be unstable

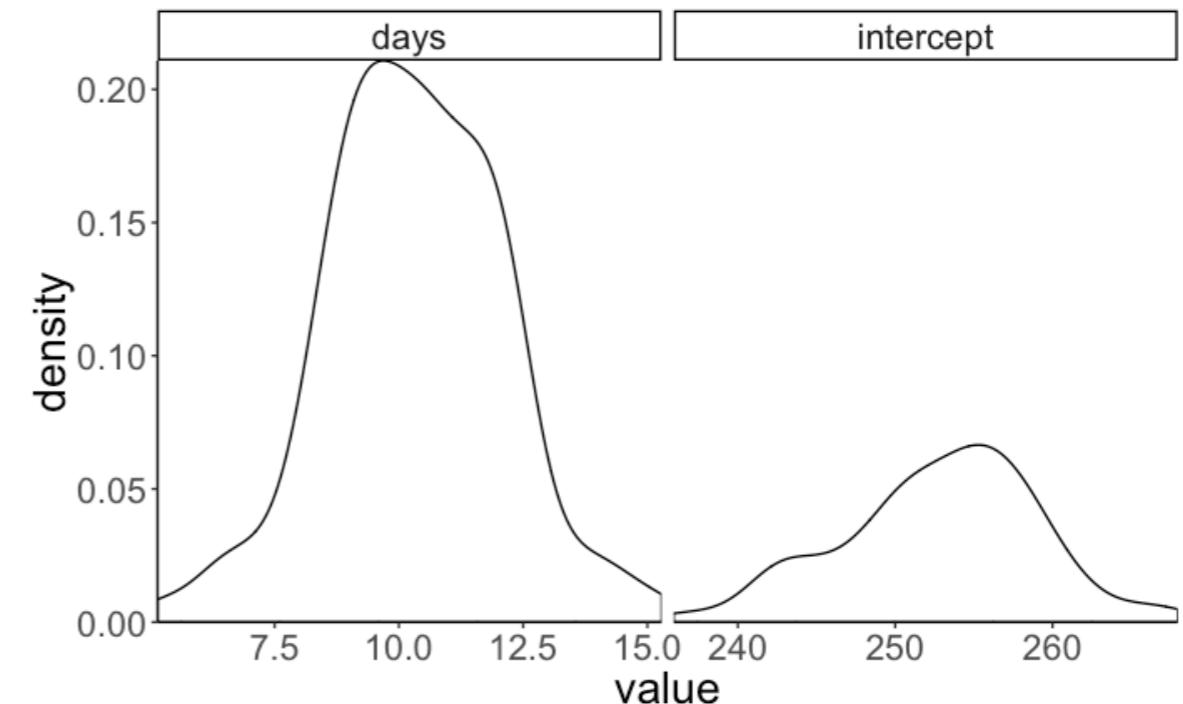
check whether the confidence
interval excludes 0

Bootstrapping an `lmer()`

lm()



lmer()

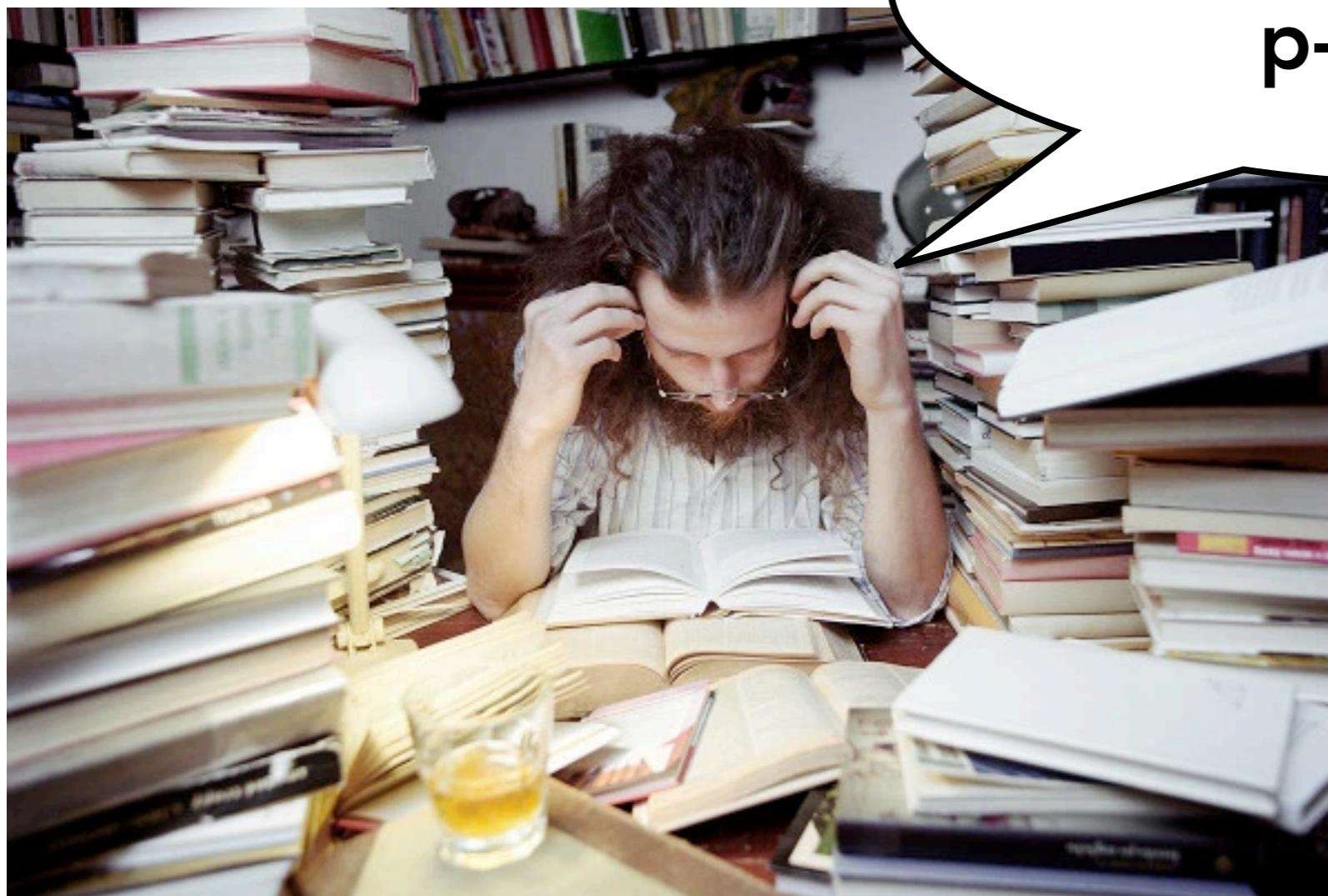


more confident in the
parameter estimates

bootstrapped parameter estimates

Getting p-values

Reviewer #2



I can't seem to find any
p-values ...

Model comparison

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
```

"lmerTest" package

- I recommend **not** to load the package as it doesn't play nicely with the "broom" package at the moment (this will presumably be fixed soon though ...)

"lmerTest" package

```
1 lmerTest::lmer(formula = reaction ~ 1 + days + (1 + days | subject),  
2 data = df.sleep) %>%  
3 summary()
```

```
Linear mixed model fit by REML. t-tests use Satterthwaite's method ['lmerModLmerTest']  
Formula: reaction ~ 1 + days + (1 + days | subject)  
Data: df.sleep  
  
REML criterion at convergence: 1771.4  
  
Scaled residuals:  
    Min     1Q Median     3Q    Max  
-3.9707 -0.4703  0.0276  0.4594  5.2009  
  
Random effects:  
 Groups   Name        Variance Std.Dev. Corr  
 subject (Intercept) 582.73   24.140  
           days       35.03   5.919   0.07  
 Residual            649.36   25.483  
Number of obs: 183, groups: subject, 20  
  
Fixed effects:  
             Estimate Std. Error      df t value Pr(>|t|)  
(Intercept) 252.543    6.433  19.294 39.256 < 2e-16 ***  
days         10.452    1.542  17.163   6.778 3.06e-06 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Correlation of Fixed Effects:

(Intr)	
days	-0.137

Pitfalls in fitting 1mer ()s

My model does not converge ...

- **lmer()**s are solved through a (complicated) process of iterative optimization
- only interpret the results of models that actually converged!
- here are some tricks that might help:
 - *continuous predictors*: center and scale
 - *categorical predictors*: choose a factor that has more data as your reference level
 - remove the correlation component from your model

My model does not converge ...

```
1 # fit the model
2 fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days | subject),
3                  data = df.sleep)
4
5 # explore different optimization algorithms
6 fit.all = allFit(fit.lmer)
7
8 # summarize result
9 fit.all %>% summary()
```

comparison of the different optimization algorithms



\$fixef	(Intercept)	days
bobyqa	252.5426	10.45212
Nelder_Mead	252.5426	10.45212
nlminbwrap	252.5426	10.45212
nloptwrap.NLOPT_LN_NELDERMEAD	252.5426	10.45212
nloptwrap.NLOPT_LN_BOBYQA	252.5426	10.45212

\$llik	bobyqa	Nelder_Mead	nlminbwrap
	-885.7239	-885.7239	-885.7239
	nloptwrap.NLOPT_LN_NELDERMEAD	nloptwrap.NLOPT_LN_BOBYQA	

\$sdcor	subject.(Intercept)	subject.days.(Intercept)	subject.days	sigma
bobyqa	24.13911		5.918866	0.06927657 25.48261
Nelder_Mead	24.13900		5.918891	0.06928125 25.48261
nlminbwrap	24.13911		5.918867	0.06927628 25.48261
nloptwrap.NLOPT_LN_NELDERMEAD	24.13979		5.918851	0.06927975 25.48255
nloptwrap.NLOPT_LN_BOBYQA	24.13979		5.918851	0.06927975 25.48255

<https://rdrr.io/cran/lme4/man/convergence.html>

Understanding lmer() syntax

Remove the correlation component from your model

```
1 # fit the model
2 fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days | subject),
3                   data = df.sleep)
4 # model summary
5 fit.lmer %>%
6   summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: reaction ~ 1 + days + (1 + days | subject)
Data: df.sleep

REML criterion at convergence: 1771.4

Scaled residuals:
    Min      1Q  Median      3Q     Max 
-3.9707 -0.4703  0.0276  0.4594  5.2009 

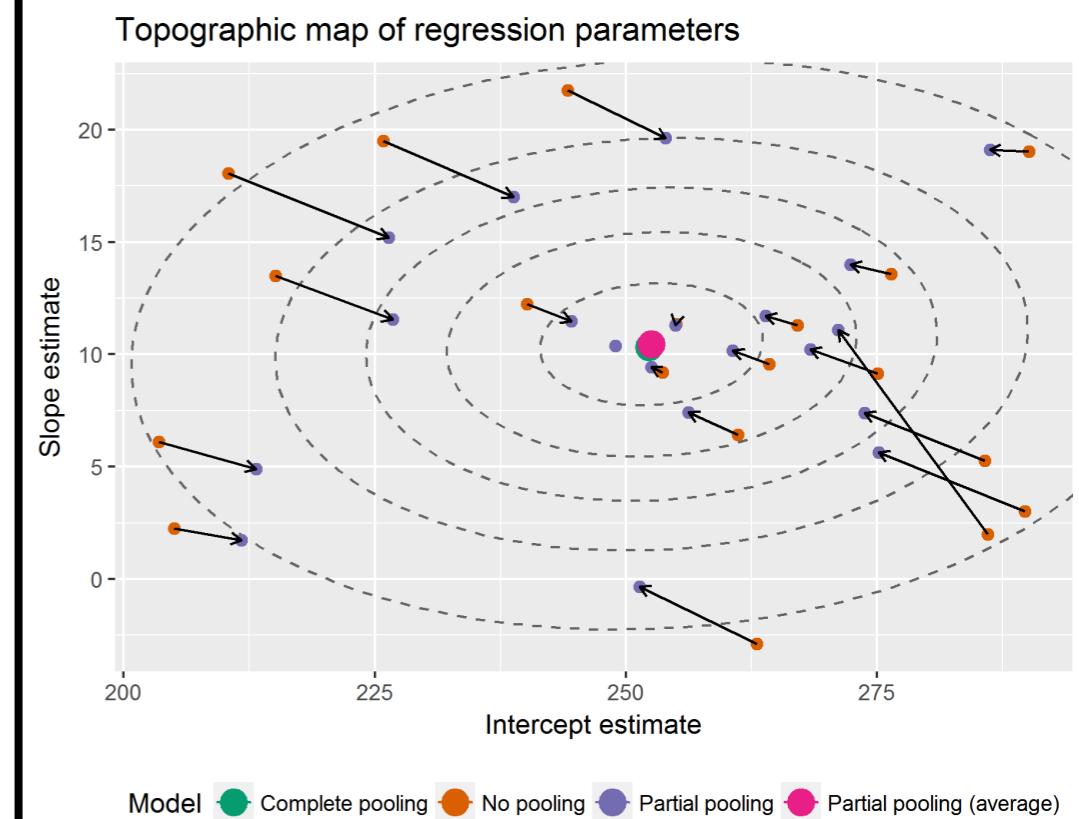
Random effects:
Groups      Name        Variance Std.Dev. Corr
subject (Intercept) 582.73   24.140
          days       35.03   5.919   0.07
Residual           649.36   25.483

Number of obs: 183, groups: subject, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept) 252.543    6.433  39.256
days         10.452    1.542   6.778

Correlation of Fixed Effects:
  (Intr) days  
days -0.137
```

multivariate Gaussian



Remove the correlation component from your model

```
1 # fit the model
2 fit.lmer = lmer(formula = reaction ~ 1 + days + (0 + days | subject) + (1 | subject),
3                  data = df.sleep)
4 # model summary
5 fit.lmer %>%
6   summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: reaction ~ 1 + days + (0 + days | subject) + (1 | subject)
Data: df.sleep

REML criterion at convergence: 1771.5

Scaled residuals:
    Min      1Q  Median      3Q     Max 
-3.9805 -0.4673  0.0250  0.4589  5.2083 

Random effects:
 Groups   Name        Variance Std.Dev.    
subject  days       35.88    5.99      
subject.1 (Intercept) 598.11   24.46    
Residual           647.90   25.45    
Number of obs: 183, groups: subject, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept) 252.550     6.491 38.907
days         10.439     1.556  6.708

Correlation of Fixed Effects:
  (Intr) days  
days -0.184
```

↑
random slopes
↑
random intercepts

independent Gaussians

Remove the correlation component from your model

```
1 # fit the model
2 fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days || subject),
3                  data = df.sleep)
4 # model summary
5 fit.lmer %>%
6   summary()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: reaction ~ 1 + days + (0 + days | subject) + (1 | subject)
Data: df.sleep

REML criterion at convergence: 1771.5

Scaled residuals:
    Min      1Q  Median      3Q     Max 
-3.9805 -0.4673  0.0250  0.4589  5.2083 

Random effects:
 Groups   Name        Variance Std.Dev.    
subject  days       35.88    5.99      
subject.1 (Intercept) 598.11   24.46    
Residual           647.90   25.45    
Number of obs: 183, groups: subject, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept) 252.550    6.491  38.907
days         10.439    1.556   6.708

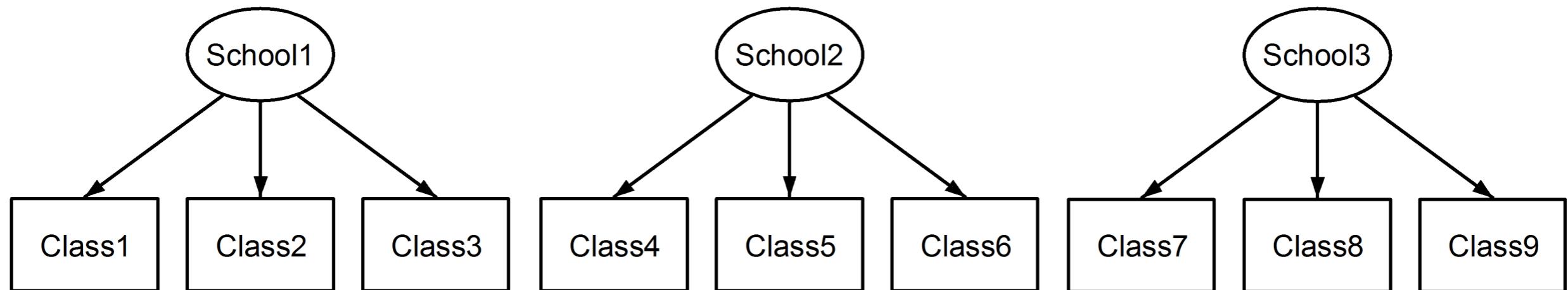
Correlation of Fixed Effects:
  (Intr) days  
days -0.184
```

alternative syntax (doesn't model correlation between random effects)

independent Gaussians

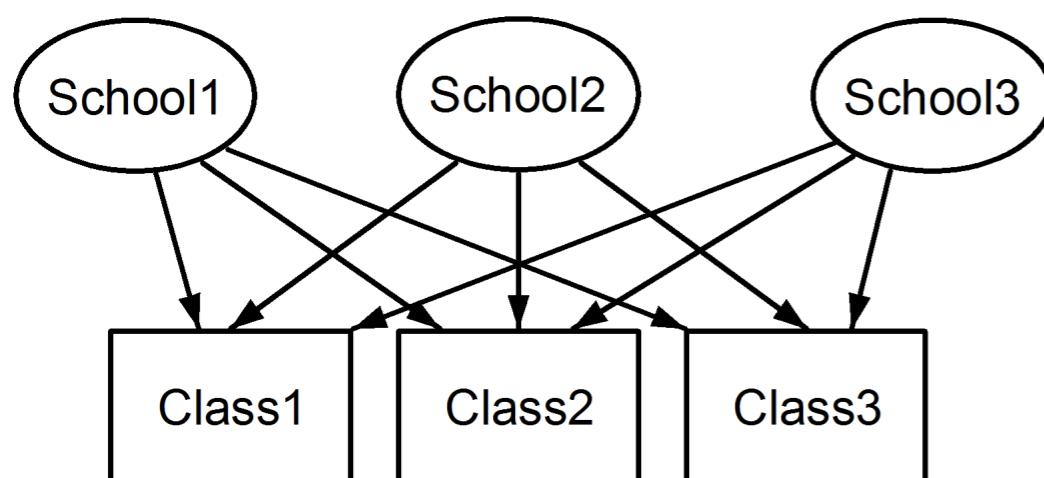
Multi-level models

nested $(1 | \text{School}/\text{Class})$



each class only appears within one school

crossed $(1 | \text{School}) + (1 | \text{Class})$

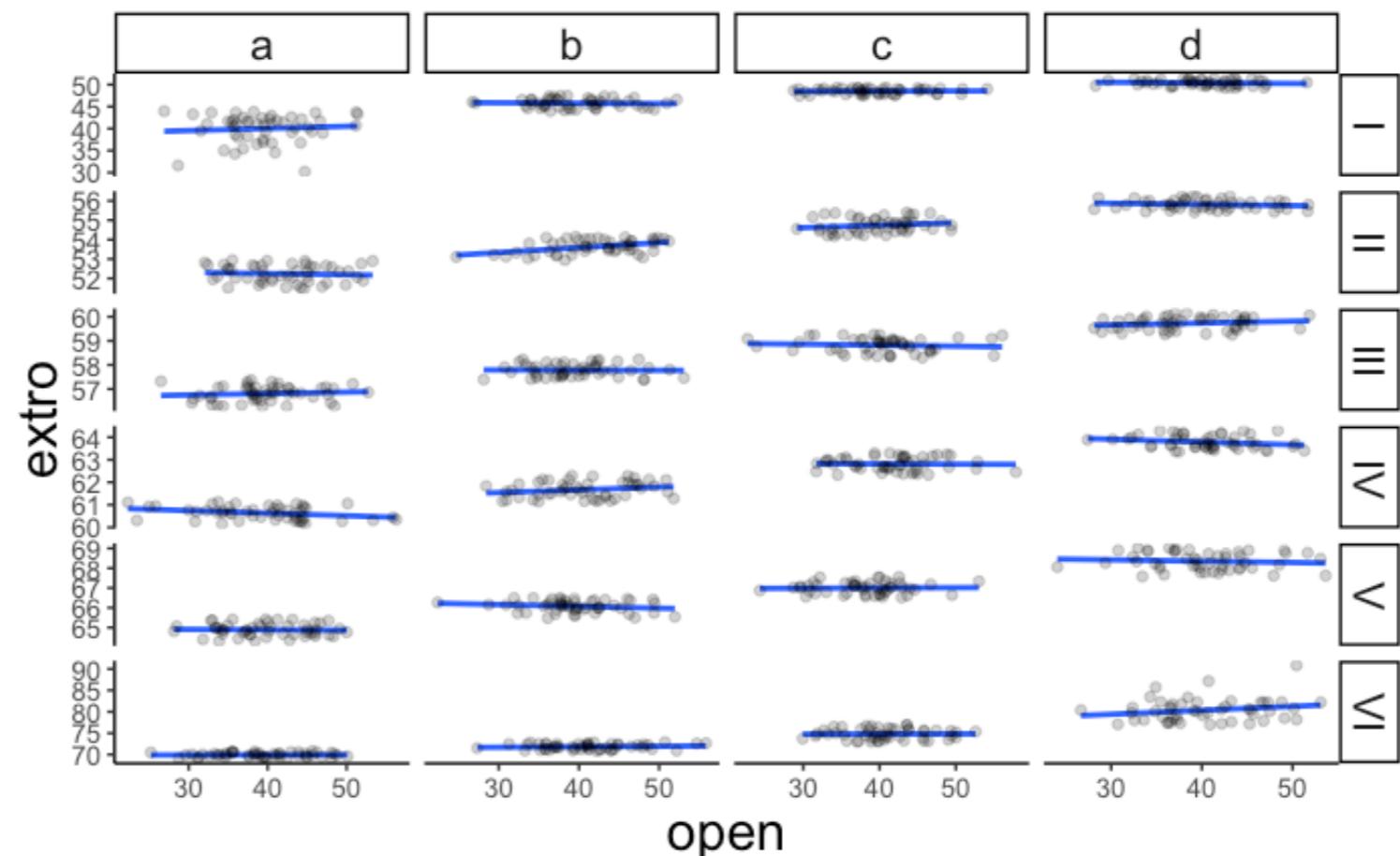


**each class
appears in each of
the schools**

Multi-level models

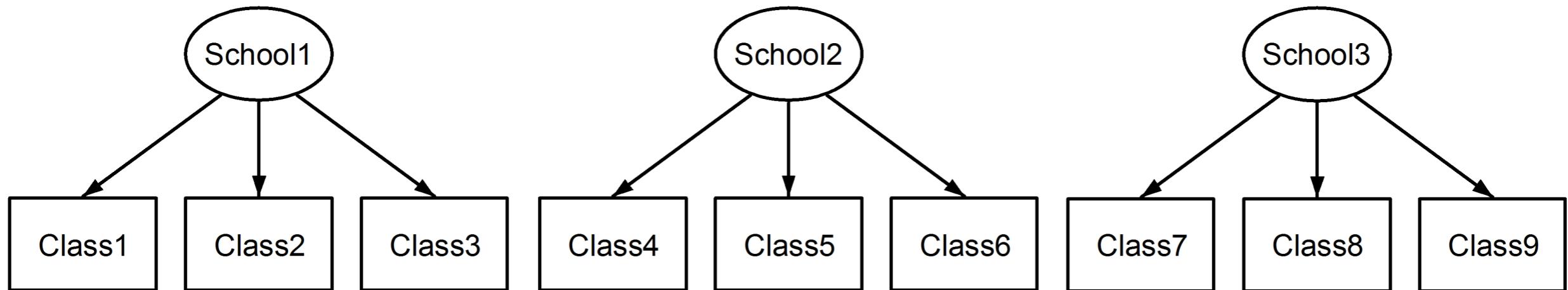
	id	extro	open	agree	social	class	school
1	1	63.69356	43.43306	38.02668	75.05811	d	IV
2	2	69.48244	46.86979	31.48957	98.12560	a	VI
3	3	79.74006	32.27013	40.20866	116.33897	d	VI
4	4	62.96674	44.40790	30.50866	90.46888	c	IV
5	5	64.24582	36.86337	37.43949	98.51873	d	IV
6	6	50.97107	46.25627	38.83196	75.21992	d	I
7	7	60.14740	37.04243	38.55959	95.91299	d	III
8	8	64.17886	42.16530	34.88235	91.45257	d	IV
9	9	56.67670	32.84933	31.68027	115.25167	a	III
10	10	47.23914	44.25764	24.99970	122.70848	b	I

relationship between
openness and extraversion



Multi-level models

nested (1 | School/Class)



```
1 # fit nested model
2 fit.nested = lmer(extro ~ open + agree + social + (1 | school/class), data = df.school)
3
```

```
4 # print model summary
5 fit.nested %>% summary()
6
7 # model coefficients
8 fit.nested %>% coef()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: extro ~ open + agree + social + (1 | school/class)
Data: df.school

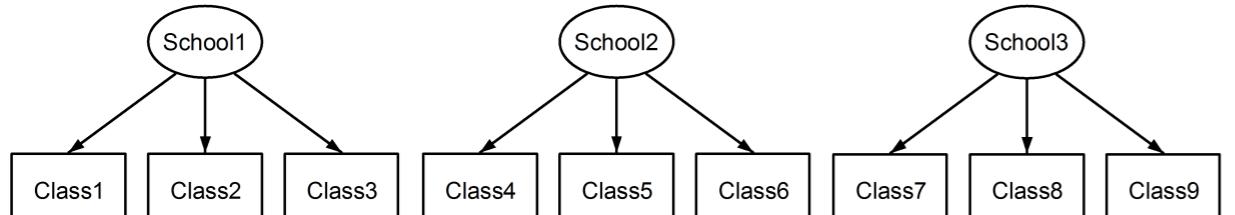
REML criterion at convergence: 3554.6

Scaled residuals:
    Min      1Q  Median      3Q     Max 
-9.9949 -0.3348  0.0057  0.3394 10.6476 

Random effects:
Groups          Name        Variance Std.Dev. 
class:school   (Intercept) 8.2046  2.8644 
school         (Intercept) 93.8433  9.6873 
Residual                    0.9684  0.9841 
Number of obs: 1200, groups: class:school, 24; school, 6
```

Multi-level models

nested (1 | School/Class)



random intercepts
of class within
each school

random intercepts of
schools

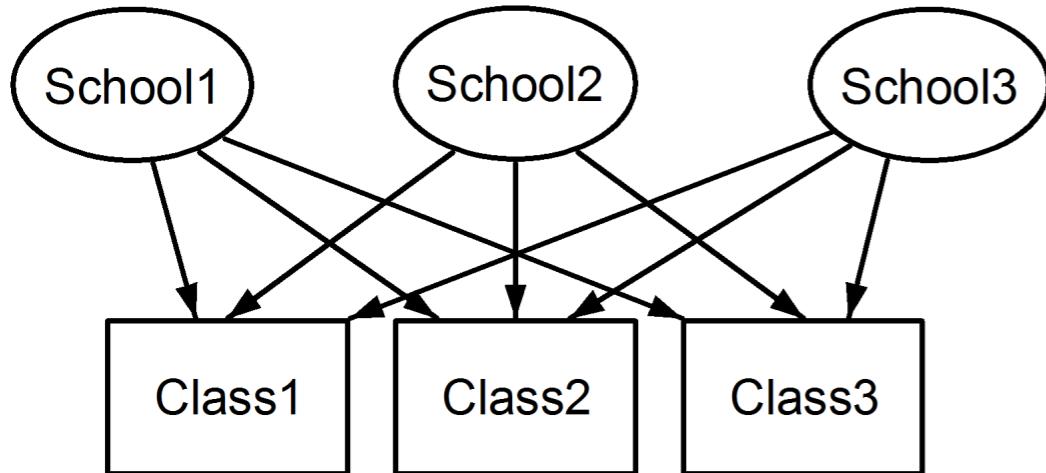
random intercepts

	\$`class:school`	(Intercept)	open	agree	social
a:I	53.77106	0.006106514	-0.007665927	0.0005404069	
a:II	58.23842	0.006106514	-0.007665927	0.0005404069	
a:III	58.71819	0.006106514	-0.007665927	0.0005404069	
a:IV	58.68813	0.006106514	-0.007665927	0.0005404069	
a:V	58.68268	0.006106514	-0.007665927	0.0005404069	
a:VI	56.23088	0.006106514	-0.007665927	0.0005404069	
b:I	59.54852	0.006106514	-0.007665927	0.0005404069	
b:II	59.62643	0.006106514	-0.007665927	0.0005404069	
b:III	59.70219	0.006106514	-0.007665927	0.0005404069	
b:IV	59.73276	0.006106514	-0.007665927	0.0005404069	
b:V	59.87036	0.006106514	-0.007665927	0.0005404069	
b:VI	58.14865	0.006106514	-0.007665927	0.0005404069	
c:I	62.28460	0.006106514	-0.007665927	0.0005404069	
c:II	60.74743	0.006106514	-0.007665927	0.0005404069	
c:III	60.70970	0.006106514	-0.007665927	0.0005404069	
c:IV	60.86062	0.006106514	-0.007665927	0.0005404069	
c:V	60.80225	0.006106514	-0.007665927	0.0005404069	
c:VI	61.10164	0.006106514	-0.007665927	0.0005404069	
d:I	64.14113	0.006106514	-0.007665927	0.0005404069	
d:II	61.81189	0.006106514	-0.007665927	0.0005404069	
d:III	61.65165	0.006106514	-0.007665927	0.0005404069	
d:IV	61.83703	0.006106514	-0.007665927	0.0005404069	
d:V	62.13593	0.006106514	-0.007665927	0.0005404069	
d:VI	66.66561	0.006106514	-0.007665927	0.0005404069	

	\$school	(Intercept)	open	agree	social
I	46.44407	0.006106514	-0.007665927	0.0005404069	
II	54.20862	0.006106514	-0.007665927	0.0005404069	
III	58.29847	0.006106514	-0.007665927	0.0005404069	
IV	62.15074	0.006106514	-0.007665927	0.0005404069	
V	66.41348	0.006106514	-0.007665927	0.0005404069	
VI	73.91156	0.006106514	-0.007665927	0.0005404069	

Multi-level models

crossed $(1 | \text{School}) + (1 | \text{Class})$



each class
appears in each of
the schools

```
1 # fit crossed model
2 fit.crossed = lmer(extro ~ open + agree + social + (1 | school) + (1 | class), data = df.school)
3
```

```
4 # print model summary
5 fit.crossed %>% summary()
6
7 # model coefficients
8 fit.crossed %>% coef()
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: extro ~ open + agree + social + (1 | school) + (1 | class)
Data: df.school

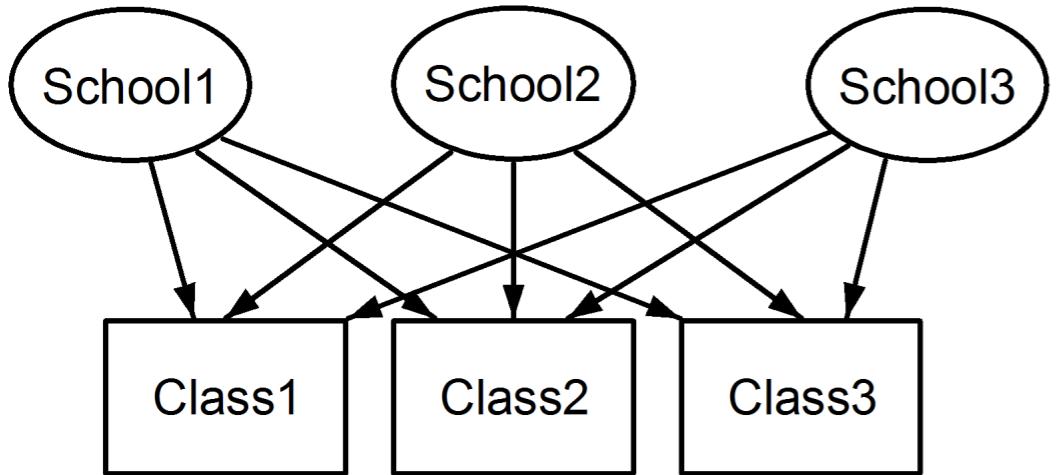
REML criterion at convergence: 4723.9

Scaled residuals:
    Min     1Q   Median     3Q    Max 
-7.8677 -0.5421  0.0101  0.5218  8.2282

Random effects:
Groups      Name        Variance Std.Dev.
school      (Intercept) 95.914   9.794
class       (Intercept)  5.787   2.406
Residual                2.787   1.669
Number of obs: 1200, groups: school, 6; class, 4
```

Multi-level models

crossed $(1 | \text{School}) + (1 | \text{Class})$



each class is in
each of the schools

random intercepts

**random intercepts
of school**

**random intercepts of
class**

	\$school	(Intercept)	open	agree	social
I		46.10663	0.01083374	-0.005420032	-0.001761963
II		54.02956	0.01083374	-0.005420032	-0.001761963
III		58.22277	0.01083374	-0.005420032	-0.001761963
IV		62.15508	0.01083374	-0.005420032	-0.001761963
V		66.51062	0.01083374	-0.005420032	-0.001761963
VI		74.16838	0.01083374	-0.005420032	-0.001761963

	\$class	(Intercept)	open	agree	social
a		57.35175	0.01083374	-0.005420032	-0.001761963
b		59.39261	0.01083374	-0.005420032	-0.001761963
c		61.04758	0.01083374	-0.005420032	-0.001761963
d		63.00342	0.01083374	-0.005420032	-0.001761963

`lmer()` syntax summary

formula	description
<code>dv ~ x1 + (1 g)</code>	Random intercept for each level of `g`
<code>dv ~ x1 + (0 + x1 g)</code>	Random slope for each level of `g`
<code>dv ~ x1 + (x1 g)</code>	Correlated random slope and intercept for each level of `g`
<code>dv ~ x1 + (x1 g)</code>	Uncorrelated random slope and intercept for each level of `g`
<code>dv ~ x1 + (1 sch) + (1 tch)</code>	Random intercept for each level of `sch` and for each level of `tch` (crossed)
<code>dv ~ x1 + (1 sch/tch)</code>	Random intercept for each level of `sch` and for each level of `tch` in `sch` (nested)

Summary

- Linear mixed effects model: A worked example
 - pooling:
 - complete pooling
 - no pooling
 - partial pooling
 - shrinkage
- Bootstrapping linear mixed effects models
- Getting p-values
- Pitfalls in fitting `lmer()`s (and what to do about it)
- Understanding `lmer()` syntax

Feedback

How was the pace of today's class?

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

How happy were you with today's class overall?



What did you like about today's class? What could be improved next time?

Thank you!