Linear mixed effects models 3

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

02/24/2020

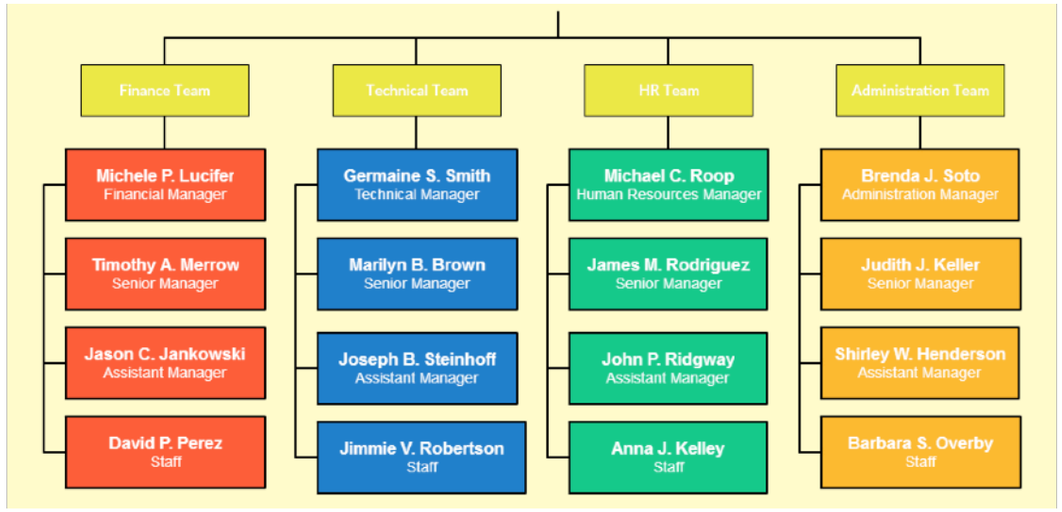
Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- Let's stimulate some `lmer()`s
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- Bootstrapping linear mixed effects models
- Getting p-values
- Understanding `lmer()` syntax
- Pitfalls in fitting `lmer()`s (and what to do about it)

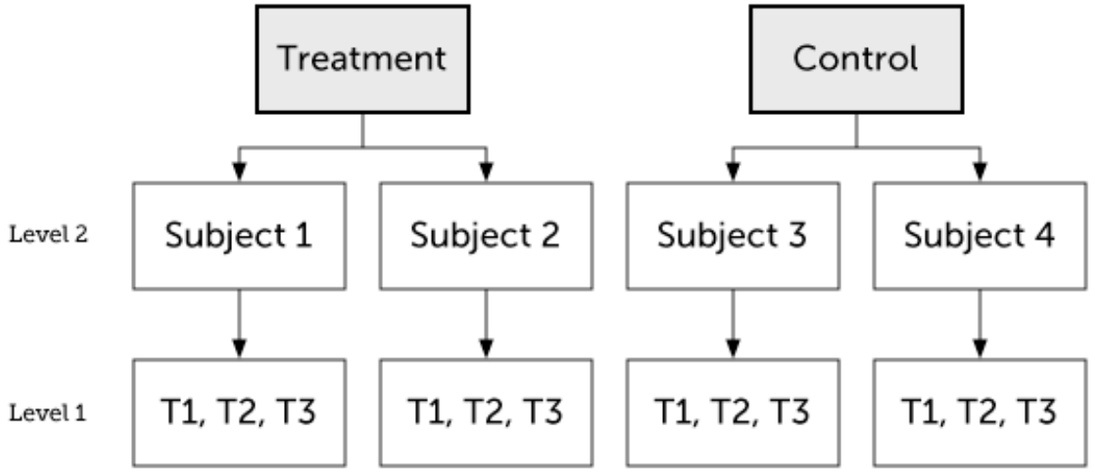
A worked example

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

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

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

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

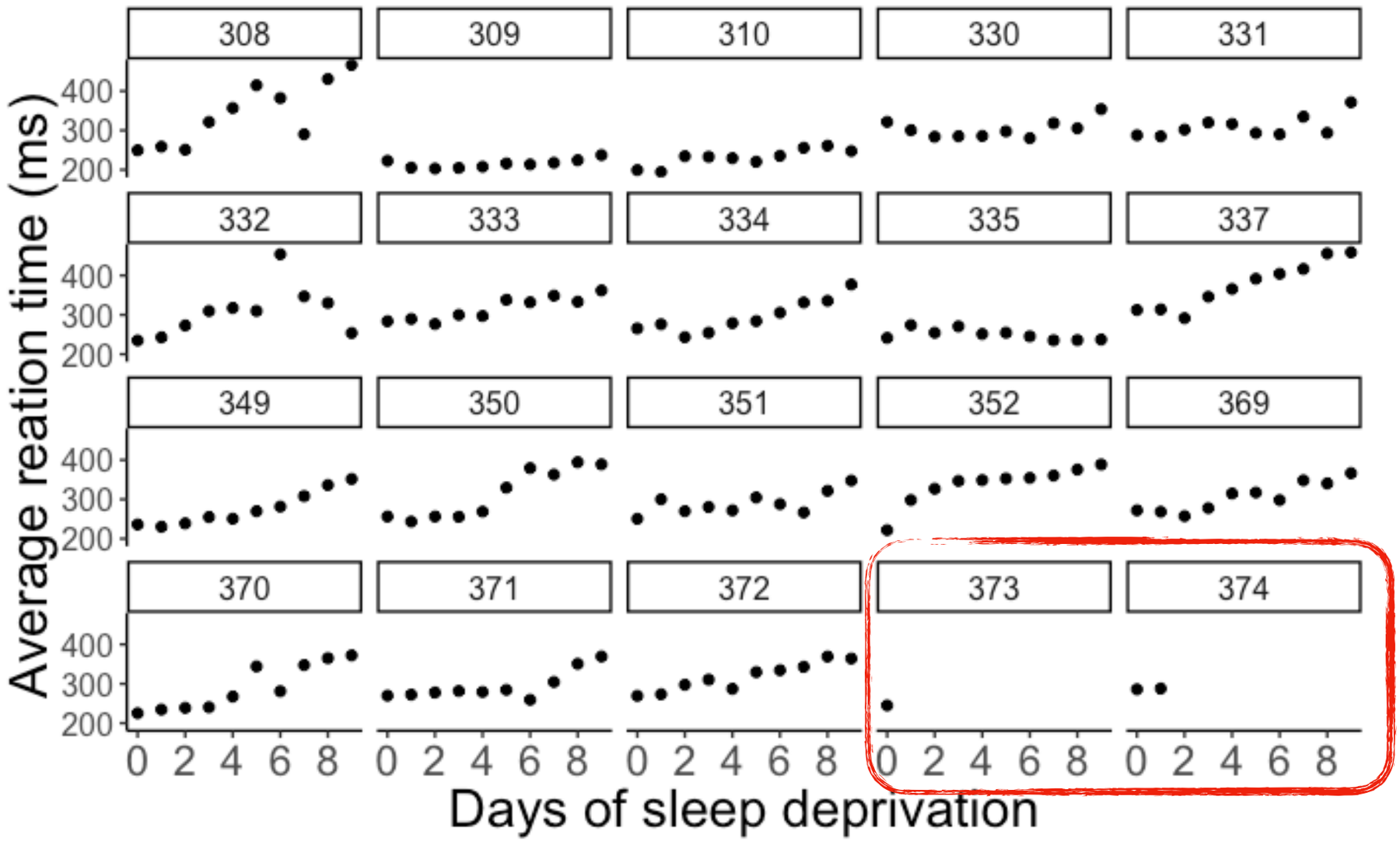
How does sleep deprivation affect reaction time?

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



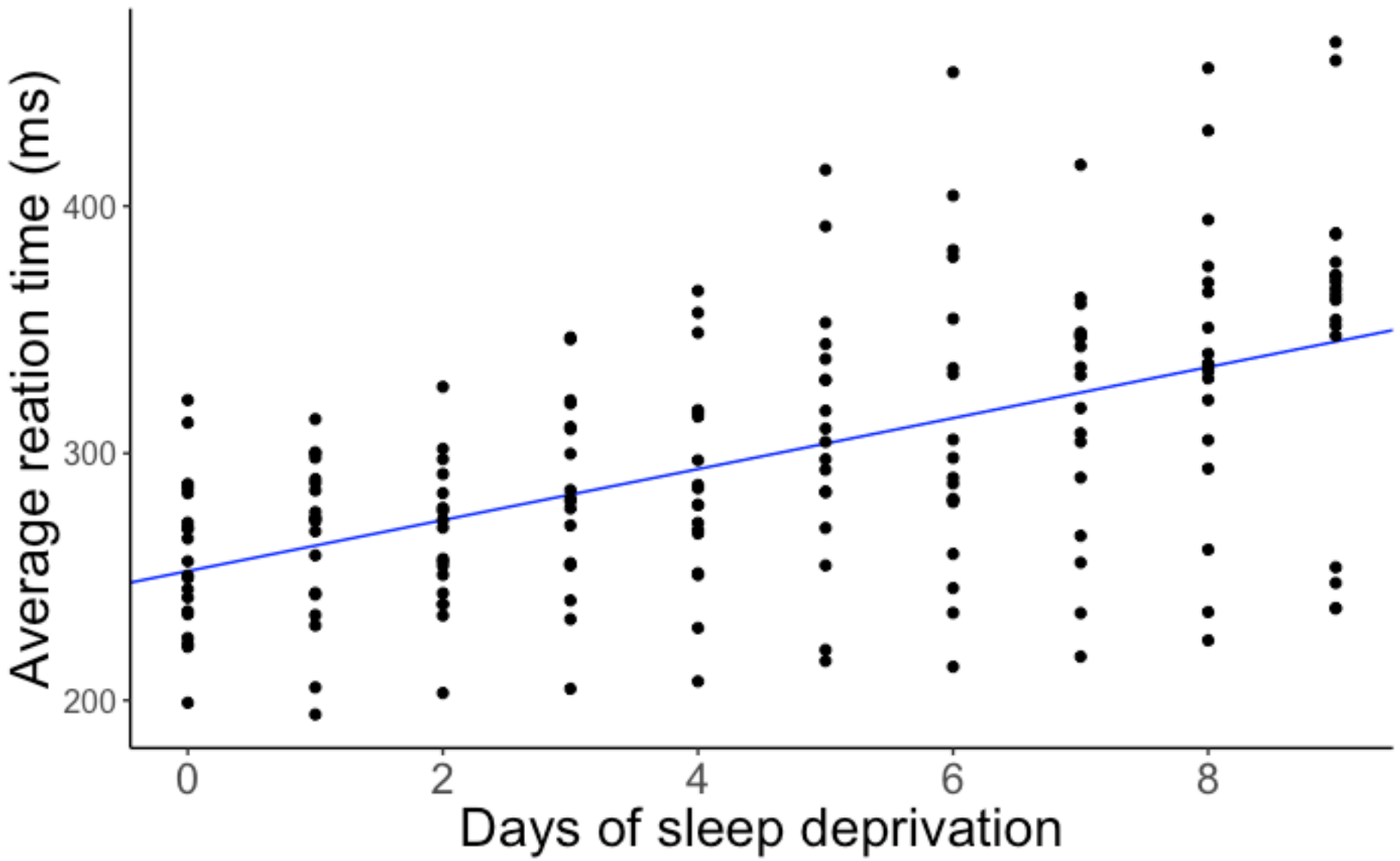
20 participants 2 with incomplete information

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

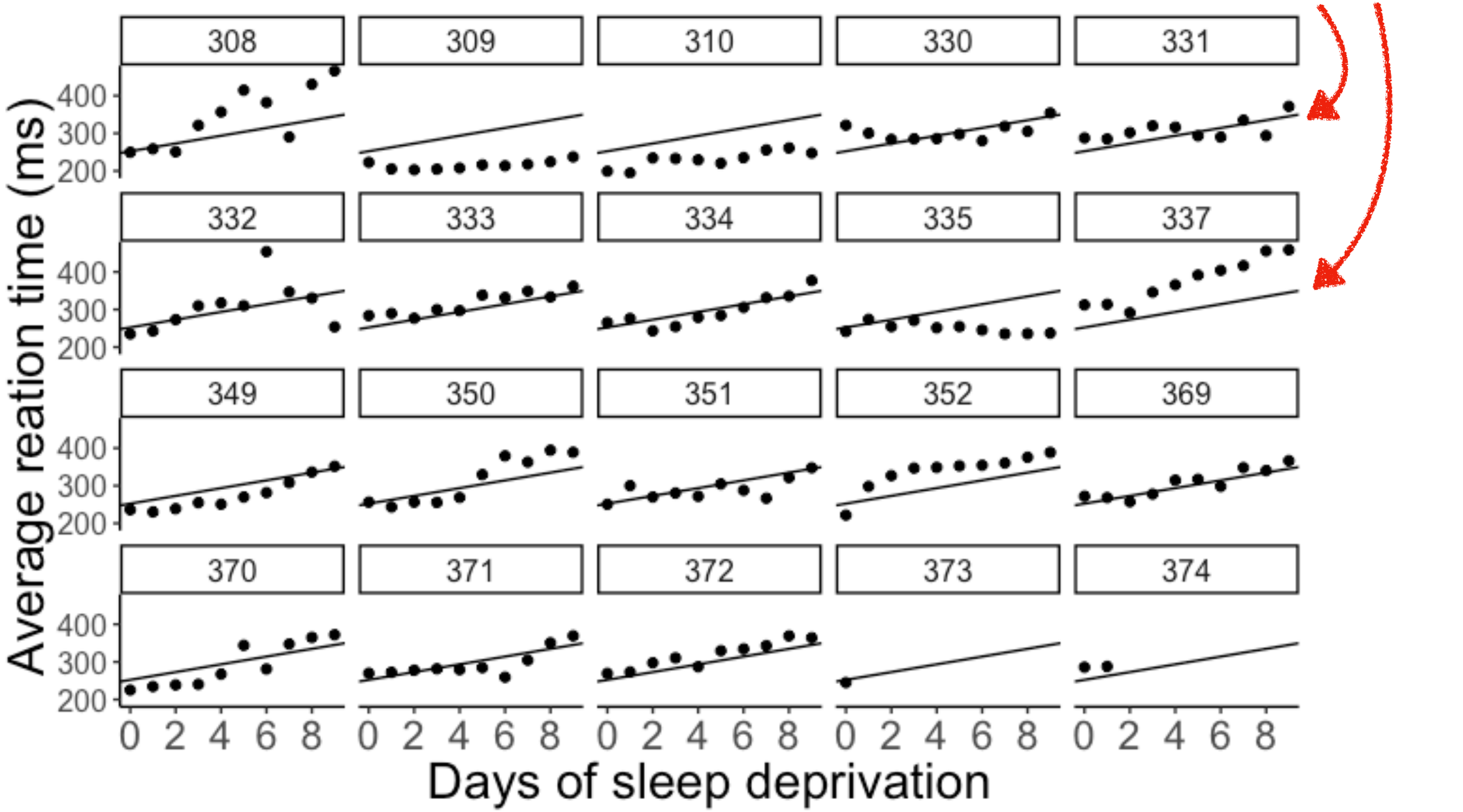
```
lm(formula = reaction ~ days,
    data = df.sleep)
```



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(data = c(days, reaction)) %>%
 4   mutate(fit = map(data, ~ lm(reaction ~ days, data = .)),
 5          params = map(fit, tidy)) %>%
 6   unnest(c(params)) %>%
 7   select(subject, term, estimate) %>%
 8   complete(subject, term, fill = list(estimate = 0)) %>%
 9   pivot_wider(names_from = term, values_from = estimate) %>%
10   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

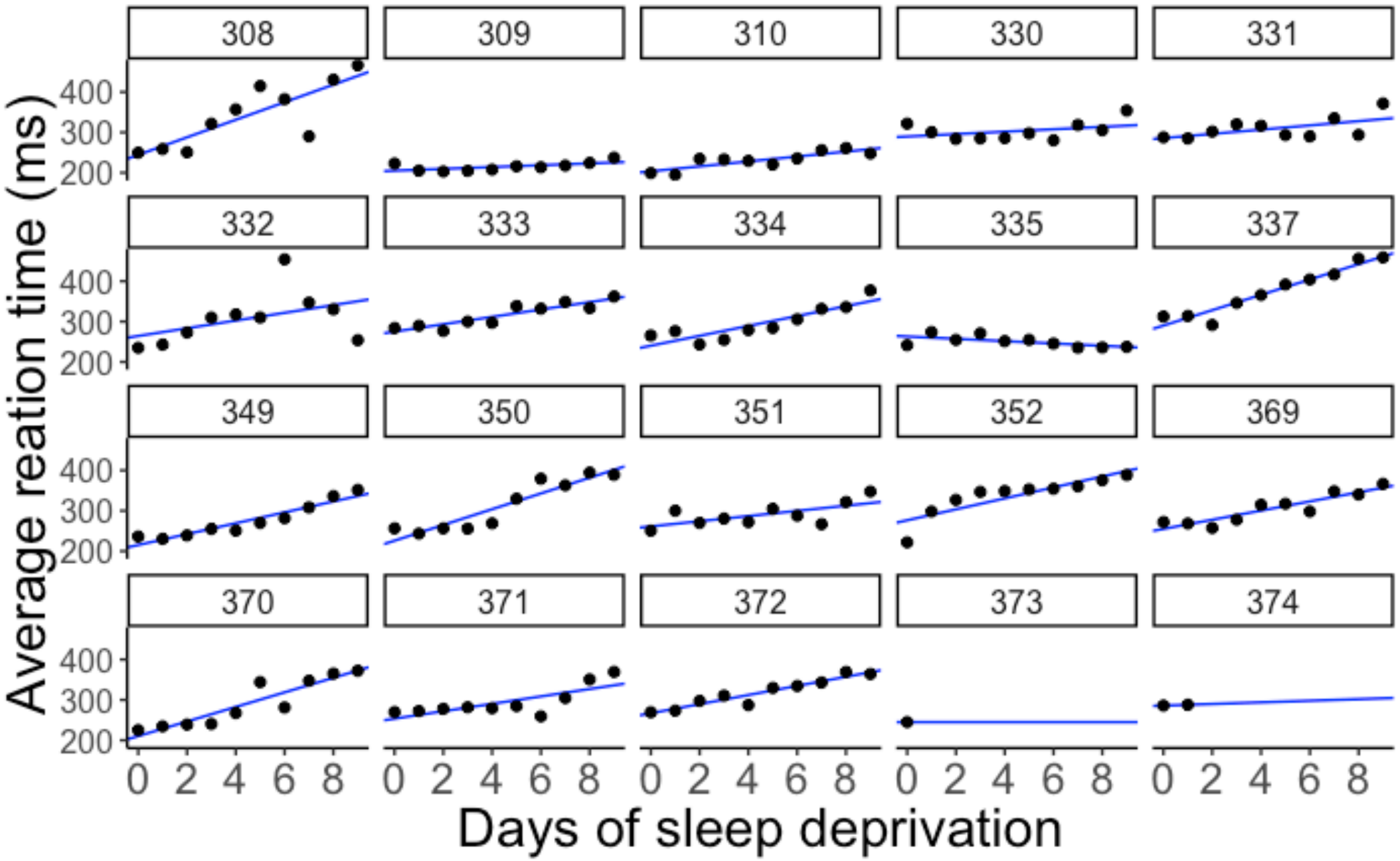
```
df.no_pooling = df.sleep %>%
  group_by(subject) %>%
  nest(data = c(days, reaction)) %>%
  mutate(fit = map(data, ~ lm(reaction ~ days, data = .)),
         params = map(fit, tidy)) %>%
  unnest(c(params)) %>%
  select(subject, term, estimate) %>%
  complete(subject, term, fill = list(estimate =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

different line for each participant



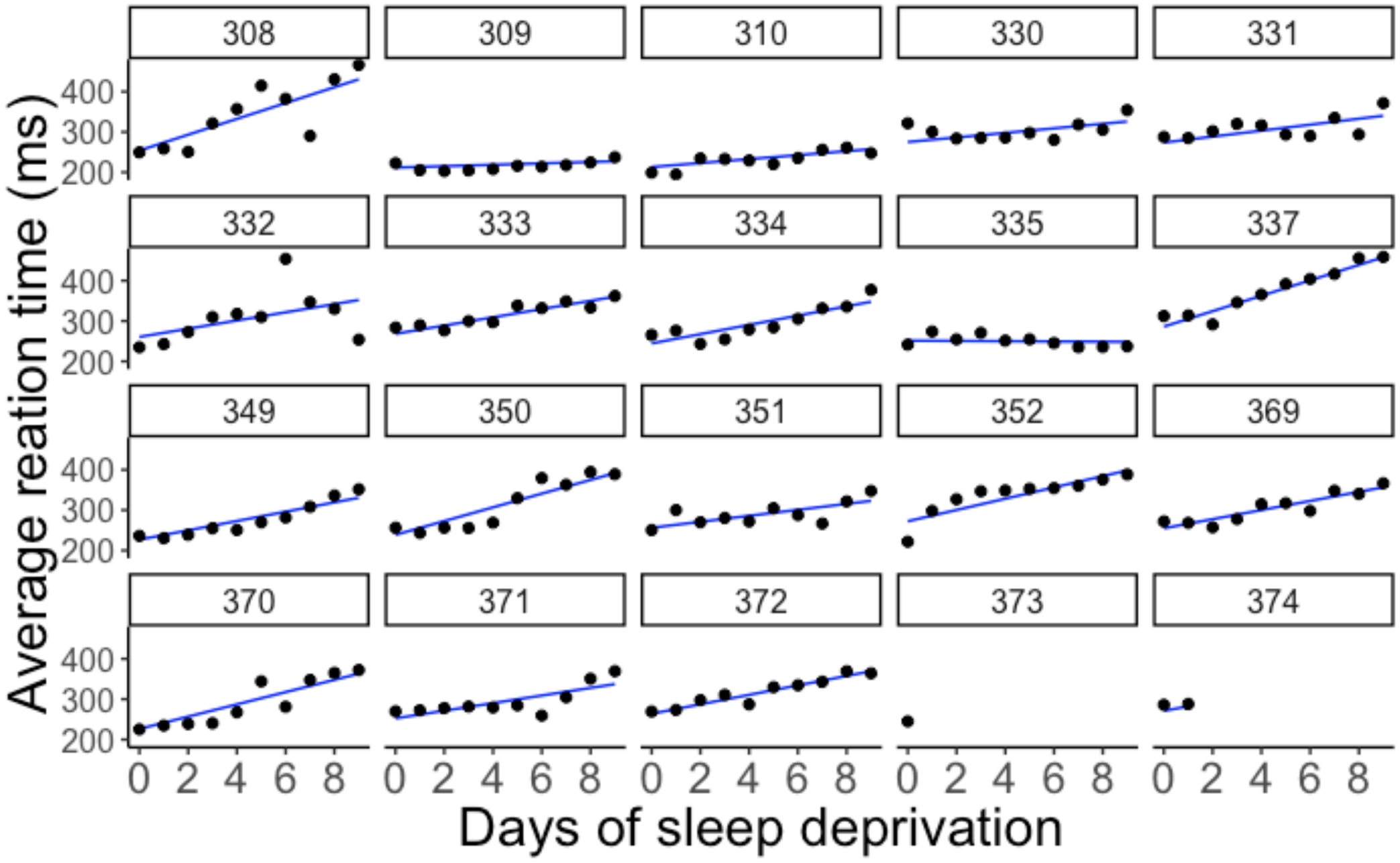
Partial pooling: Fit mixed effects model

intercepts and slopes differ between participants

random intercept

random slope

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

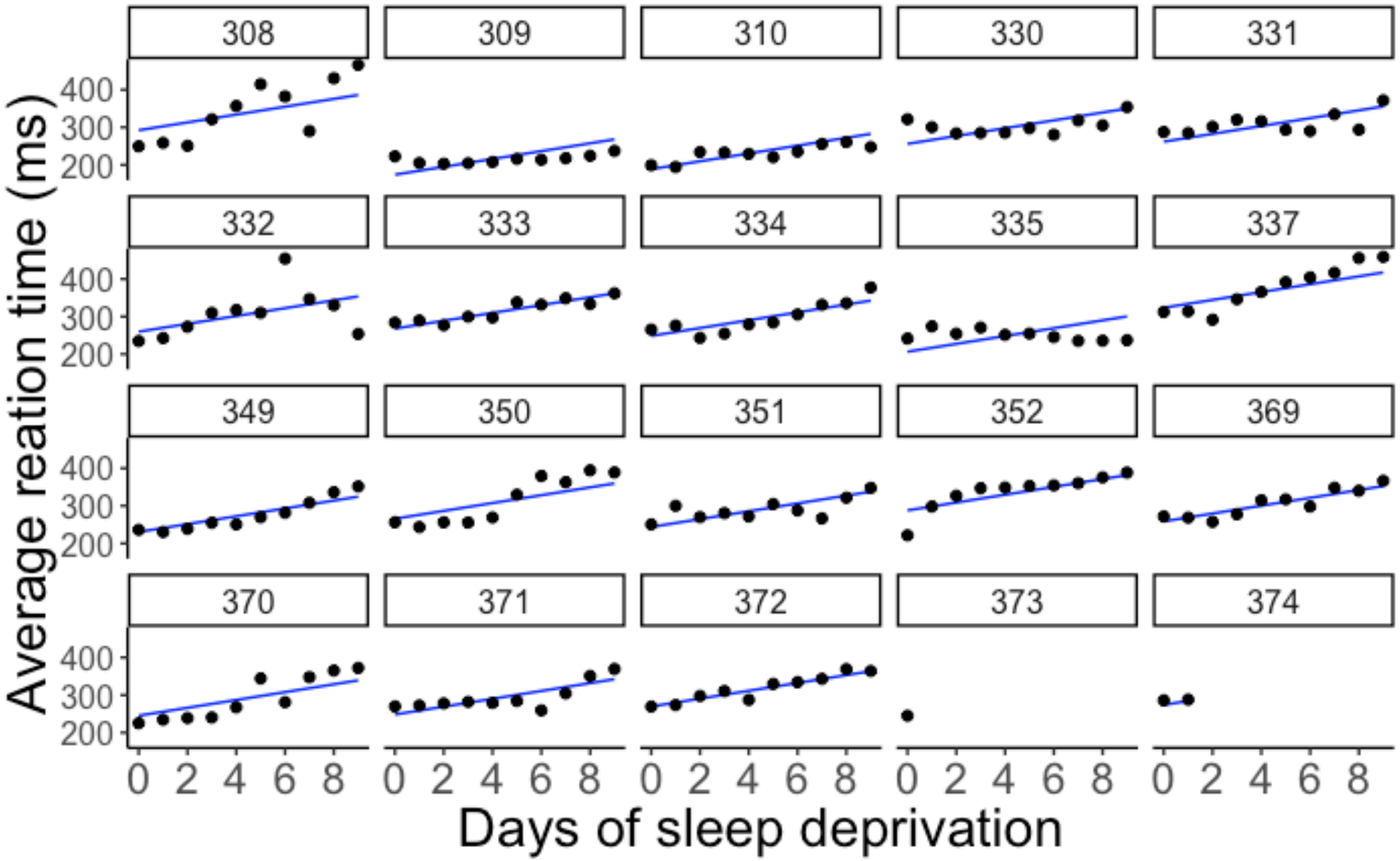


Partial pooling: Fit mixed effects model

only intercepts differ between participants

random intercept

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

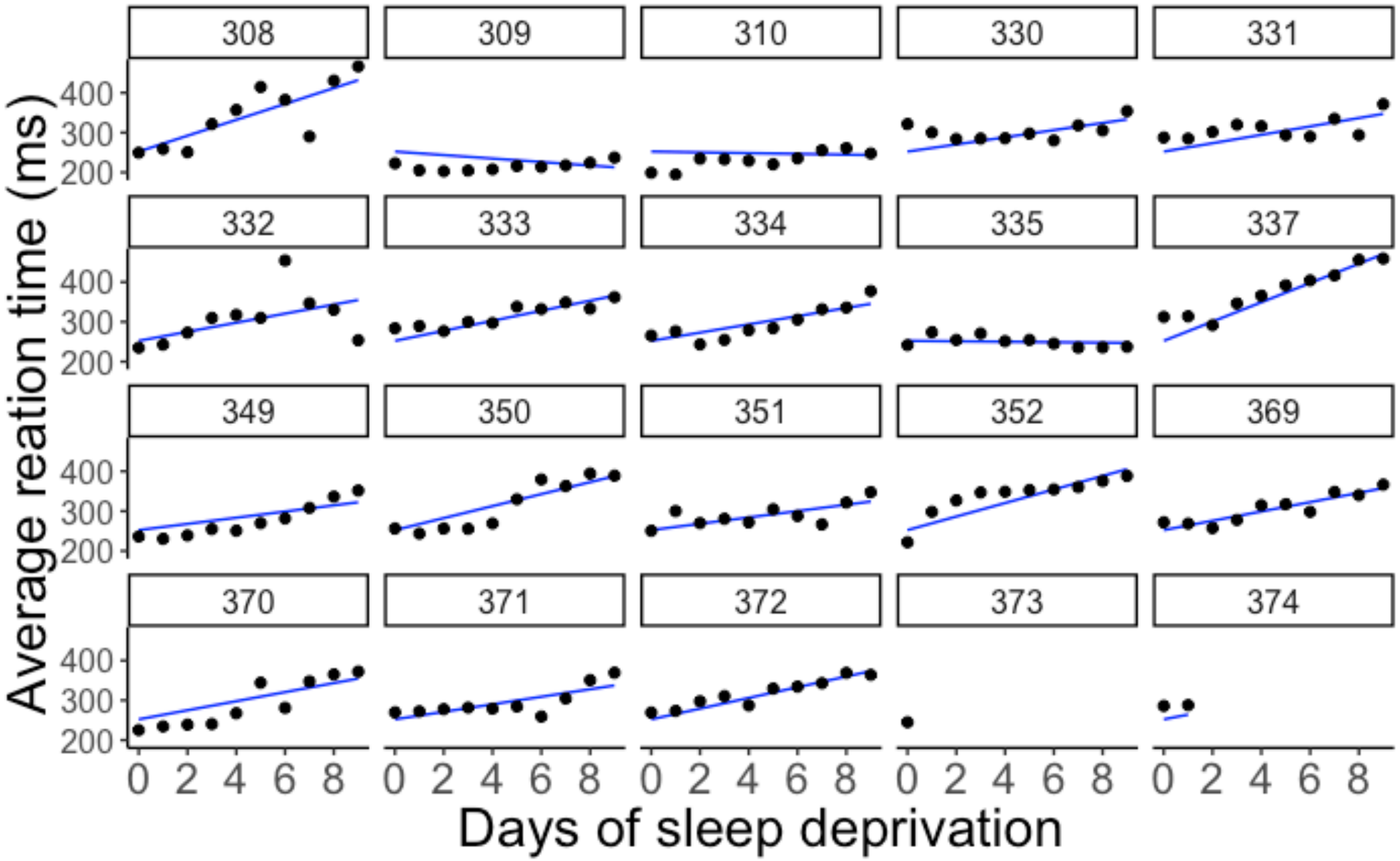


Partial pooling: Fit mixed effects model

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)

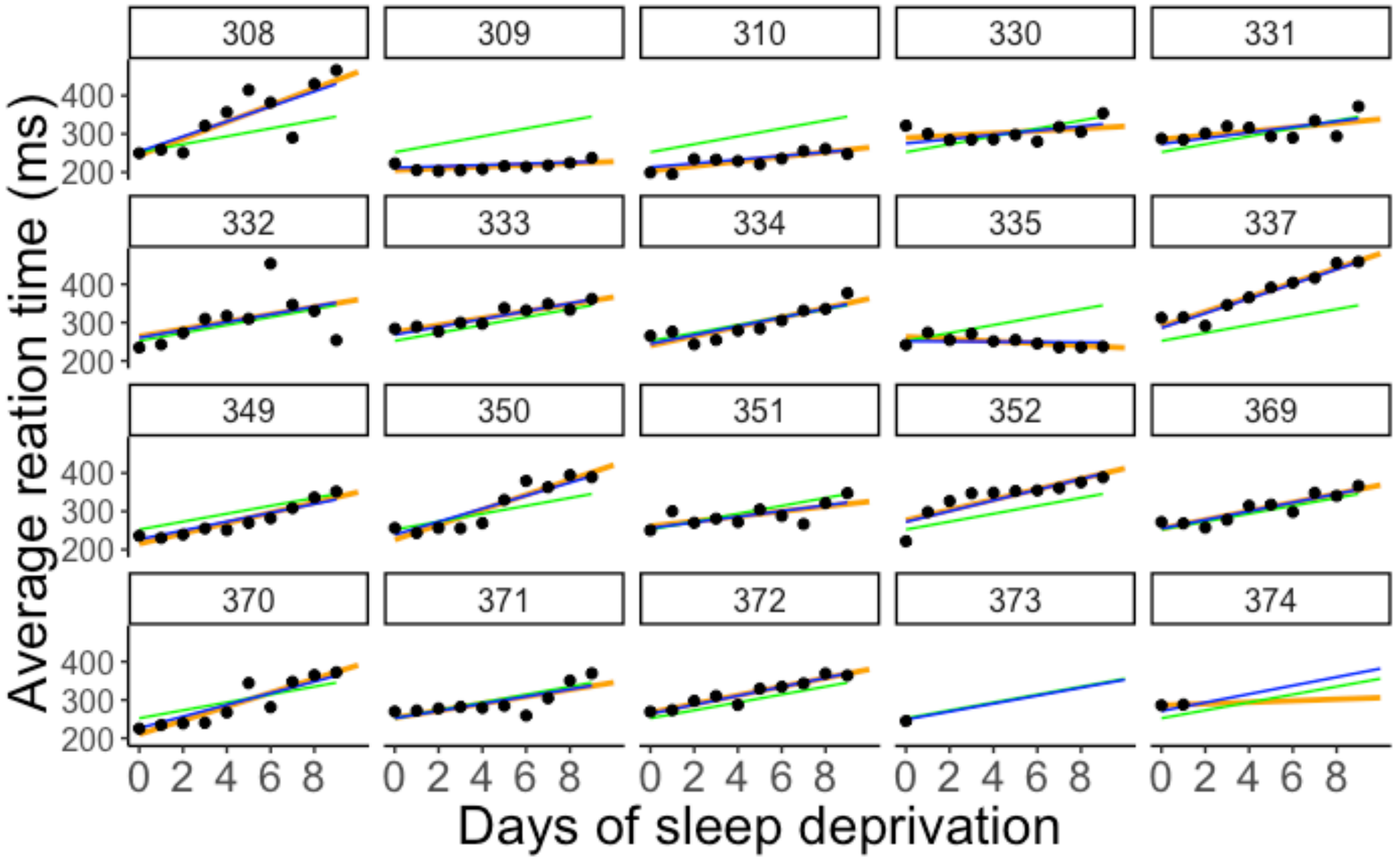
```
$subject
    (Intercept)       days
308    253.9479 19.6264139
309    211.7328  1.7319567
310    213.1579  4.9061843
330    275.1425  5.6435987
331    273.7286  7.3862680
332    260.6504 10.1632535
333    268.3684 10.2245979
334    244.5523 11.4837825
335    251.3700 -0.3355554
337    286.2321 19.1090061
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

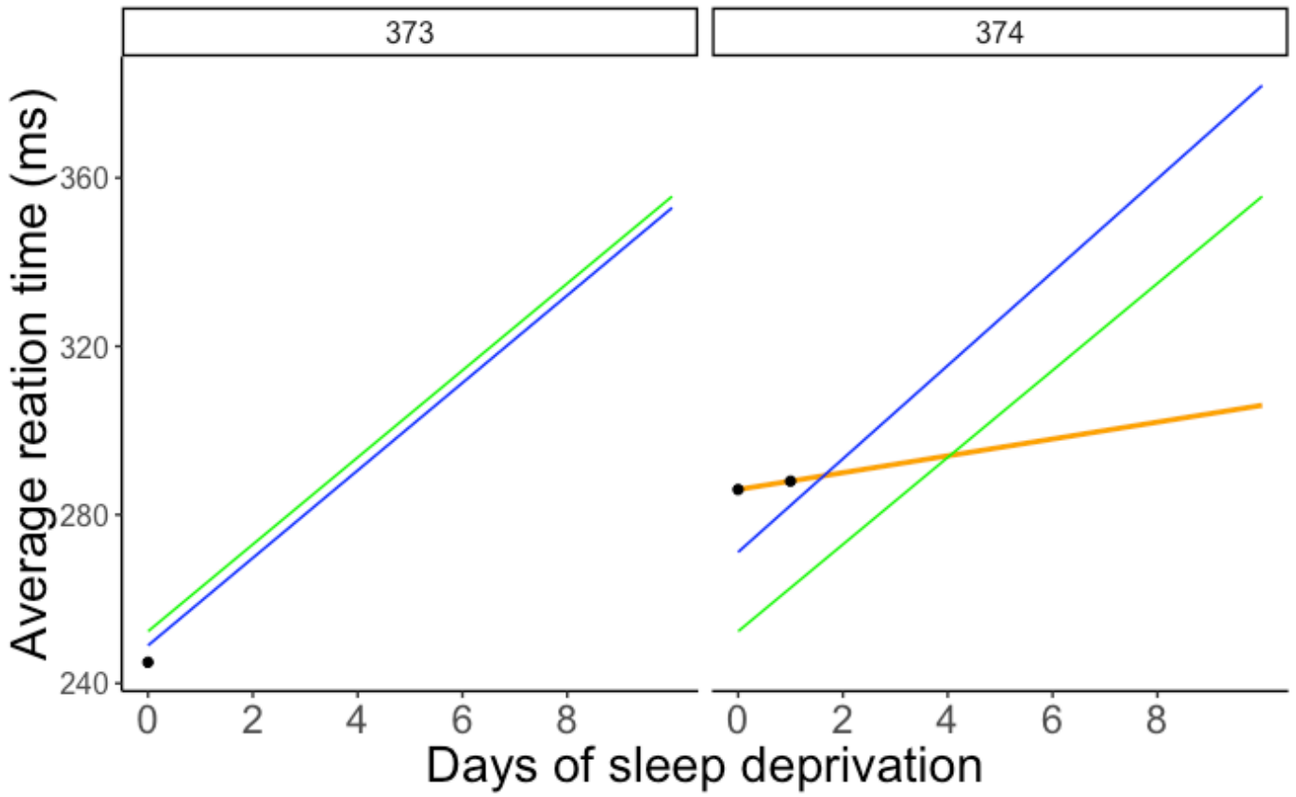


Comparison

complete pooling

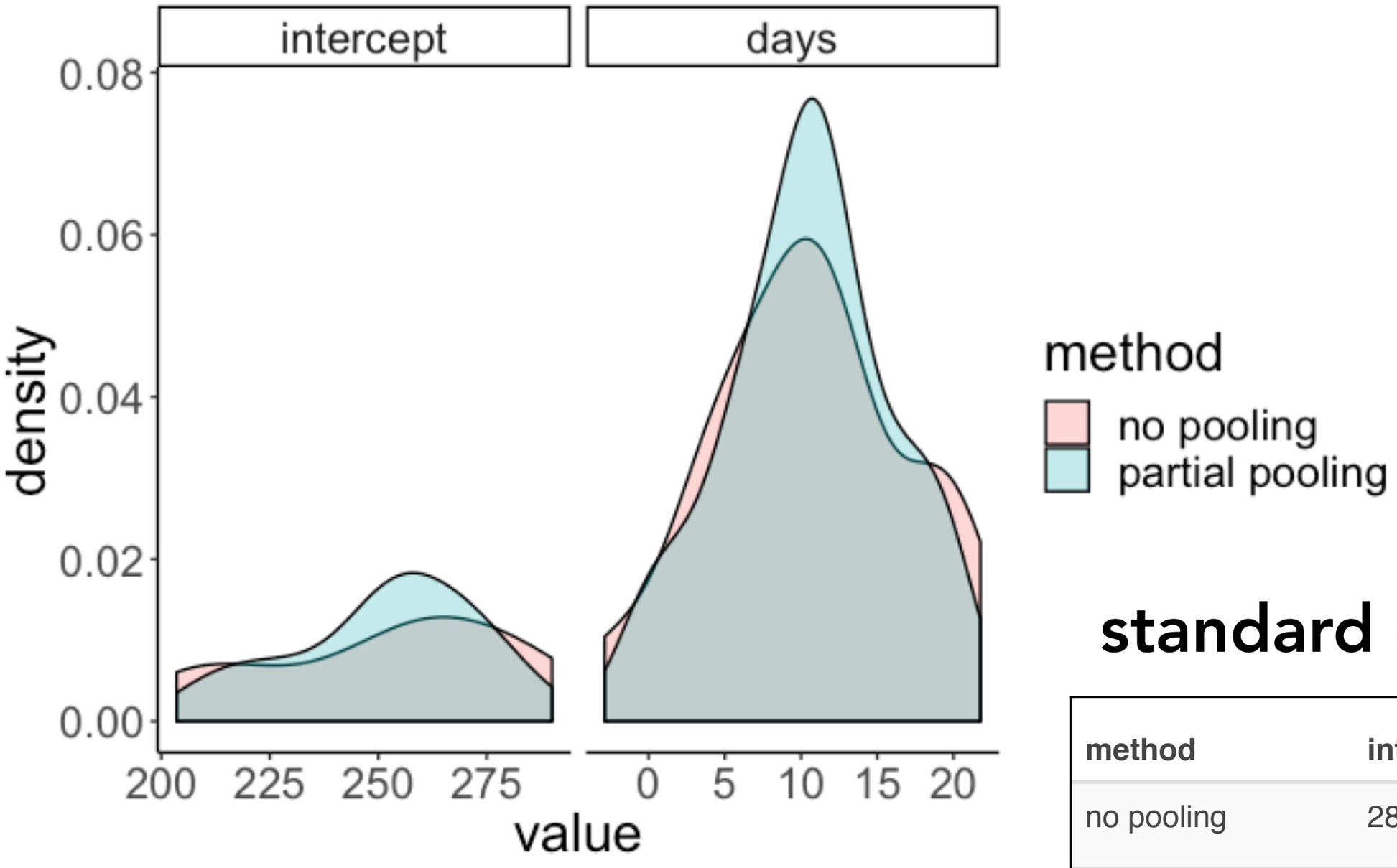
no pooling

partial pooling



- **complete pooling**:
  - doesn't account for any individual variation
- **no pooling**:
  - doesn't yield predictions when we only have 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

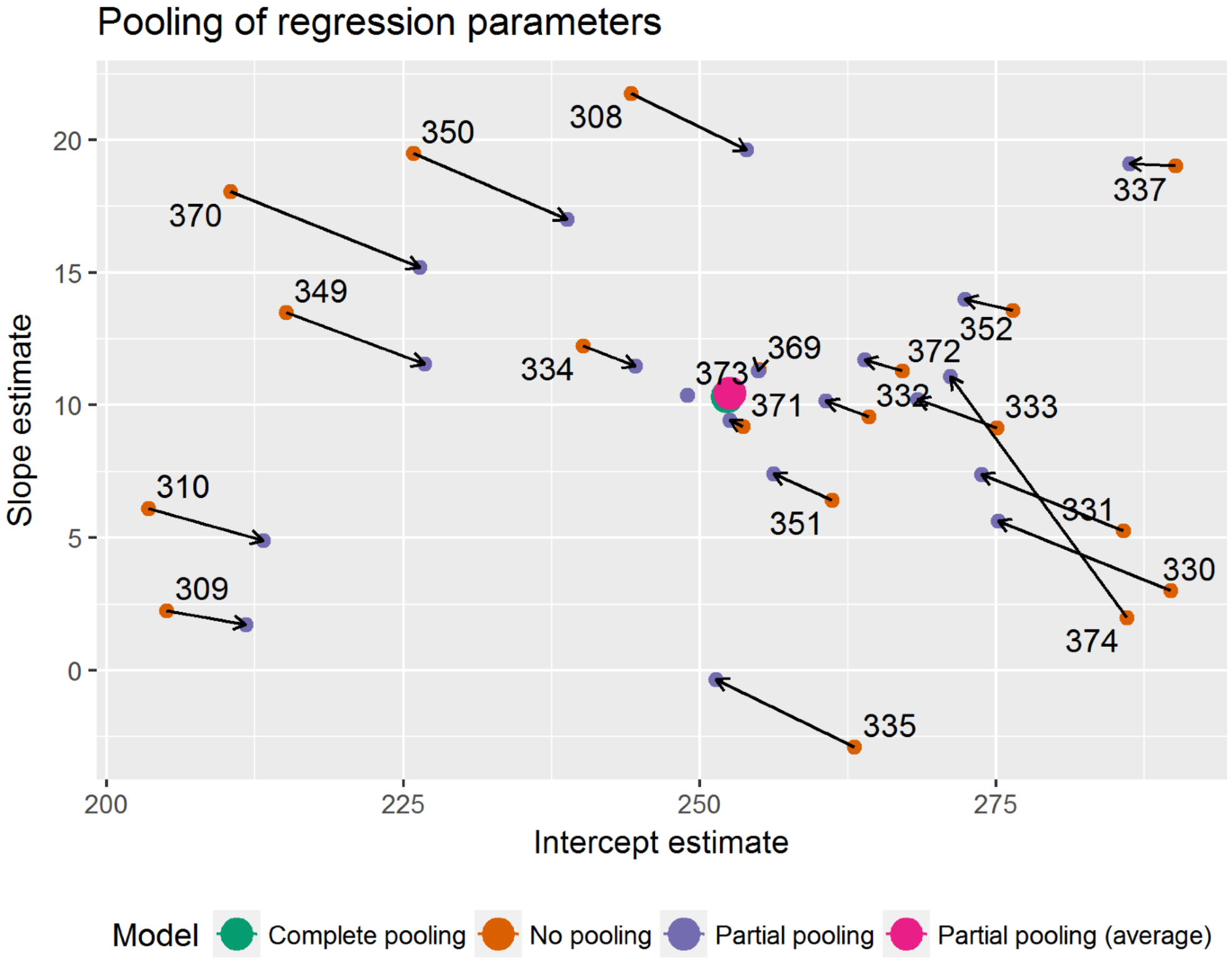


standard deviation

method	intercept	days
no pooling	28.95	6.56
partial pooling	21.59	5.46

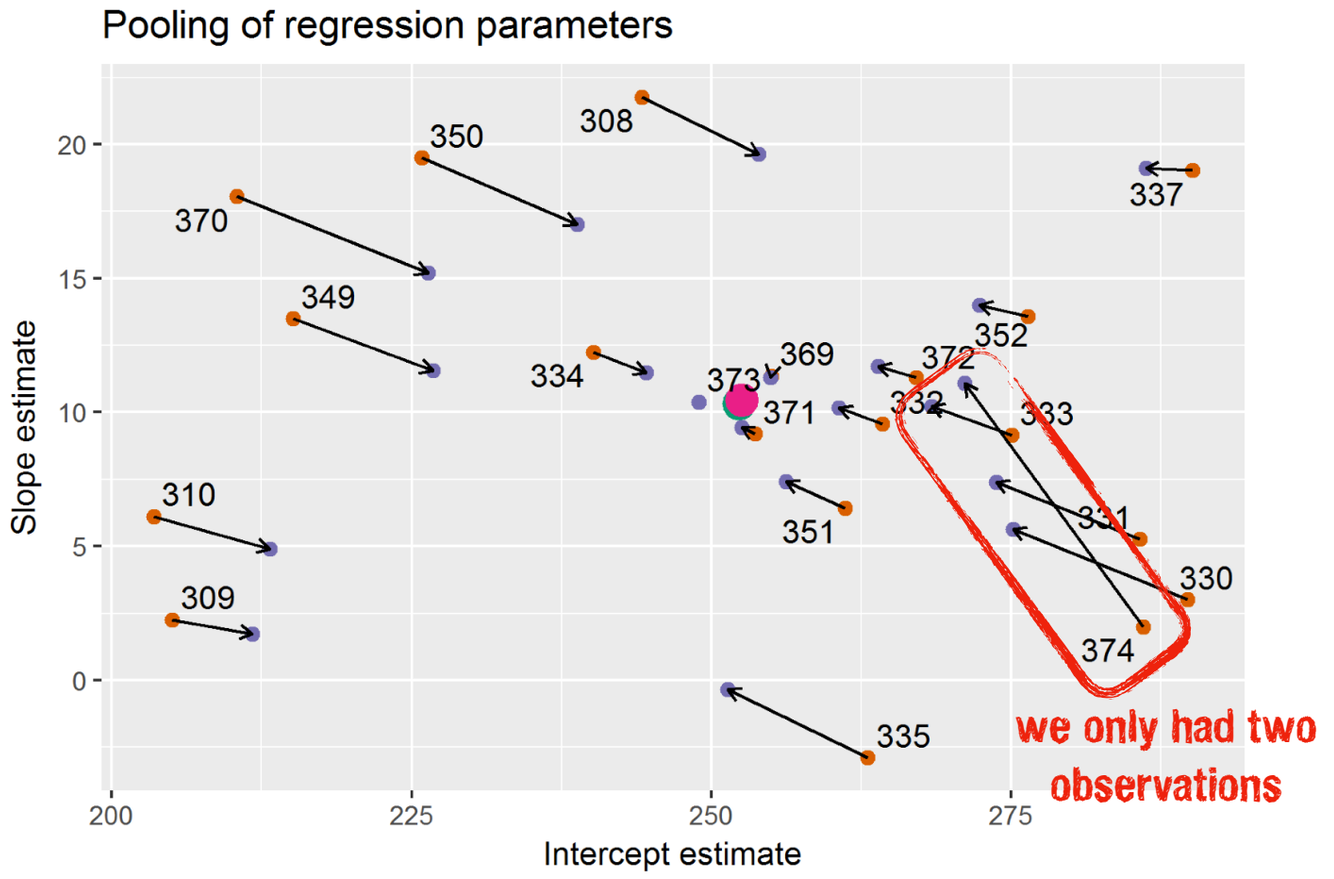
variance "shrinks"

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
371
332
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

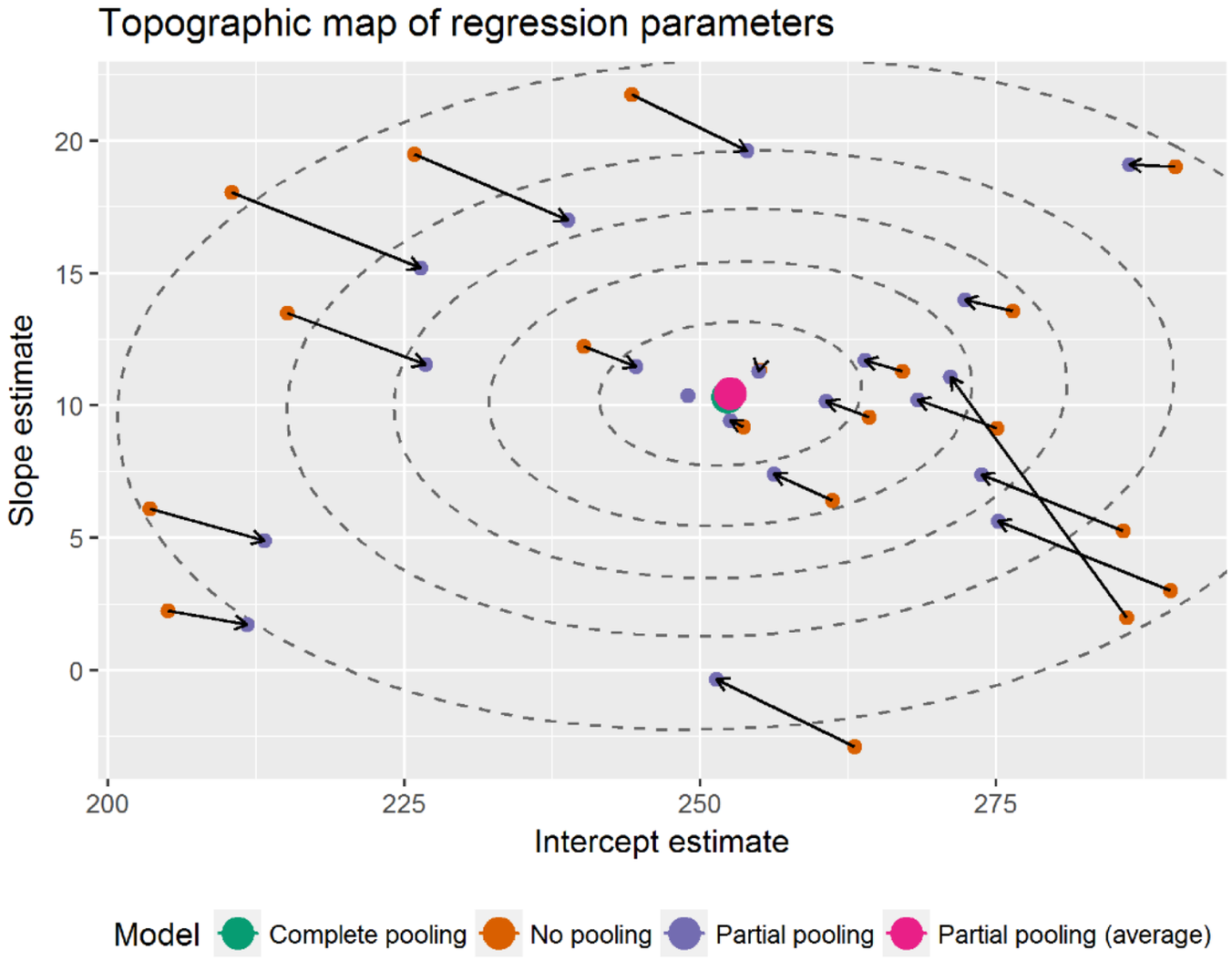


In the lme4 book, Douglas Bates provides an alternative to *shrinkage*:

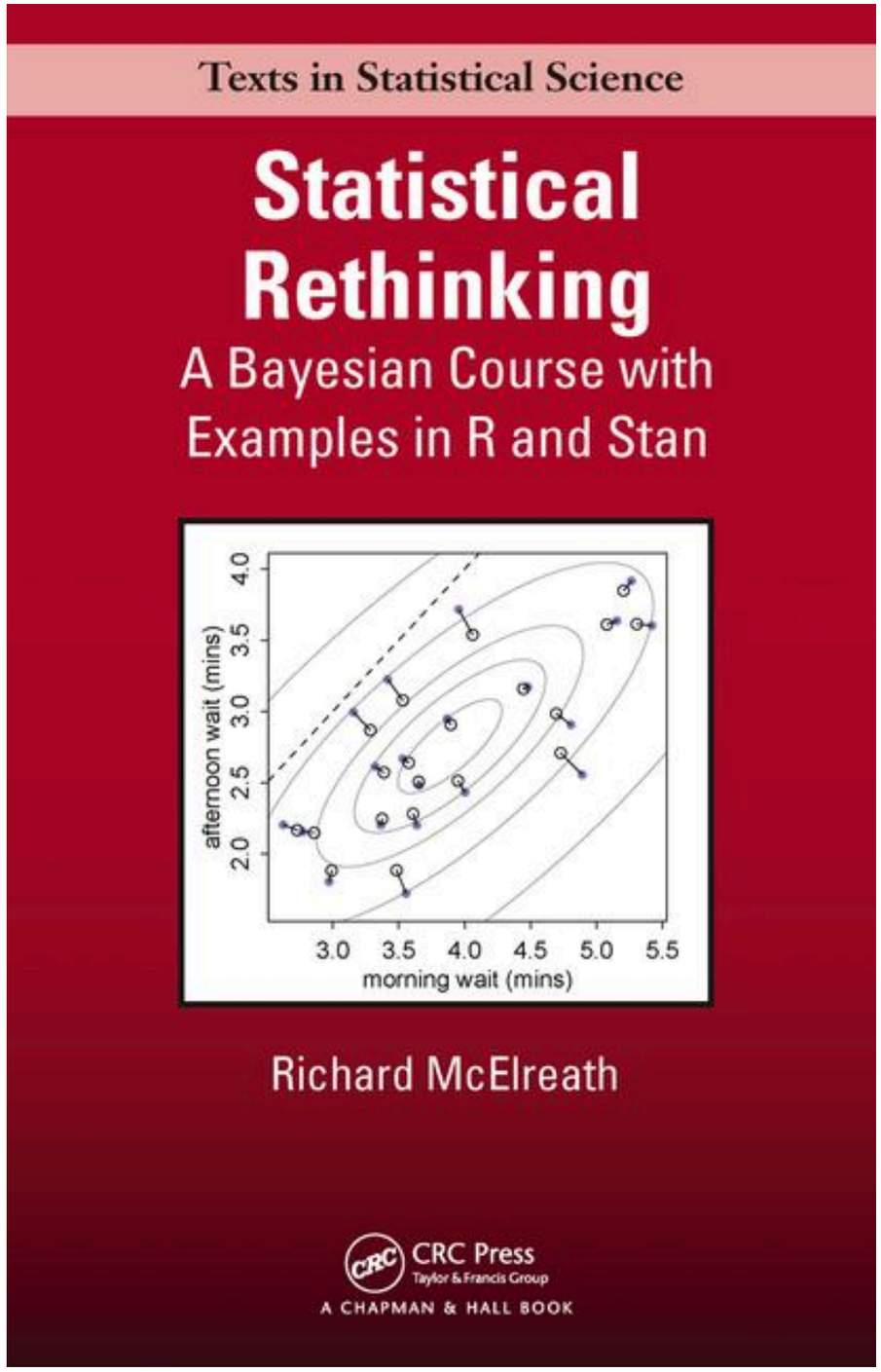
The term "shrinkage" may have negative connotations. John Tukey preferred to refer to the process as the estimates for individual subjects **"borrowing strength" from each other**.

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

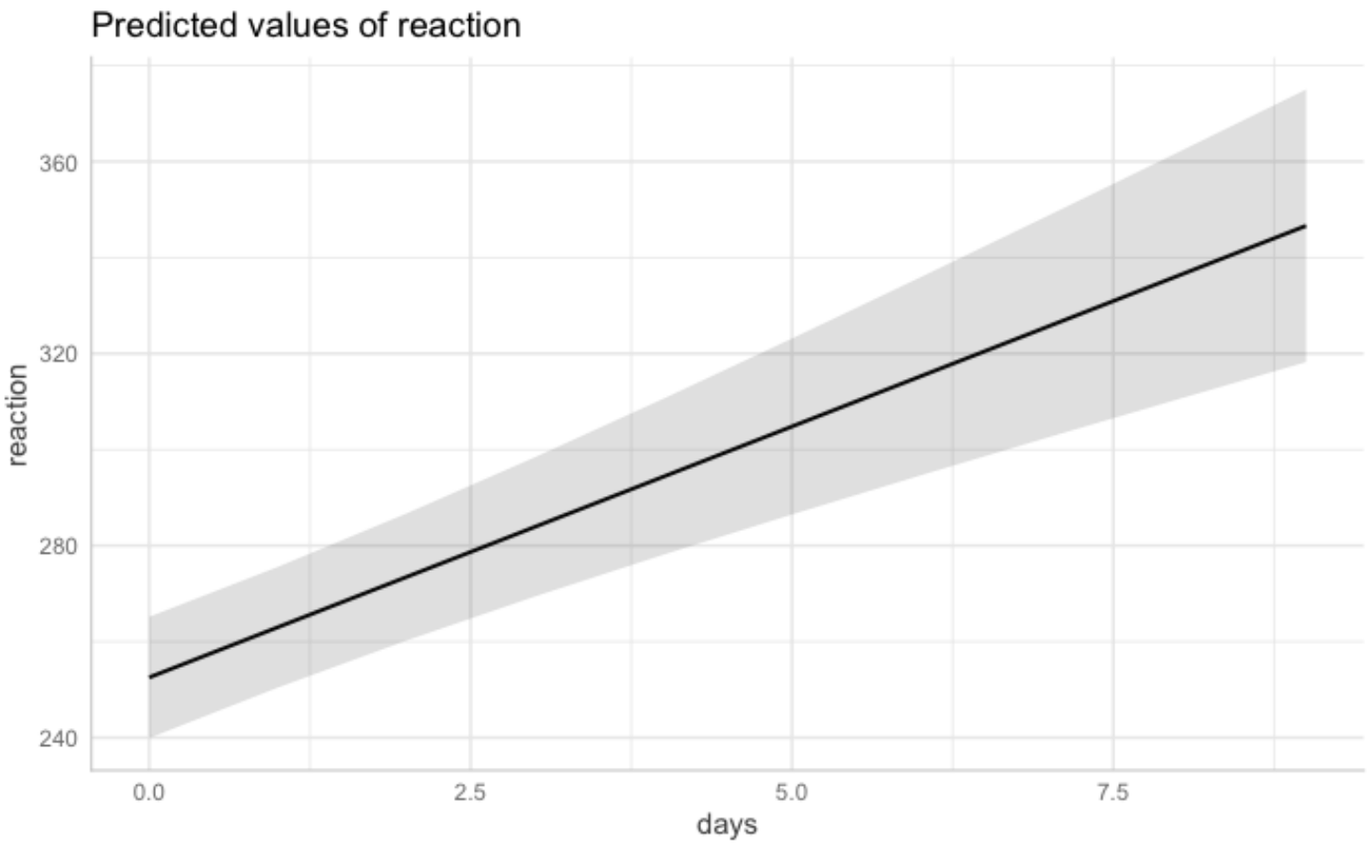


mixed effects model estimates a multi-variate Gaussian to account for (possible) correlations between intercepts and slopes



Marginal effects


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



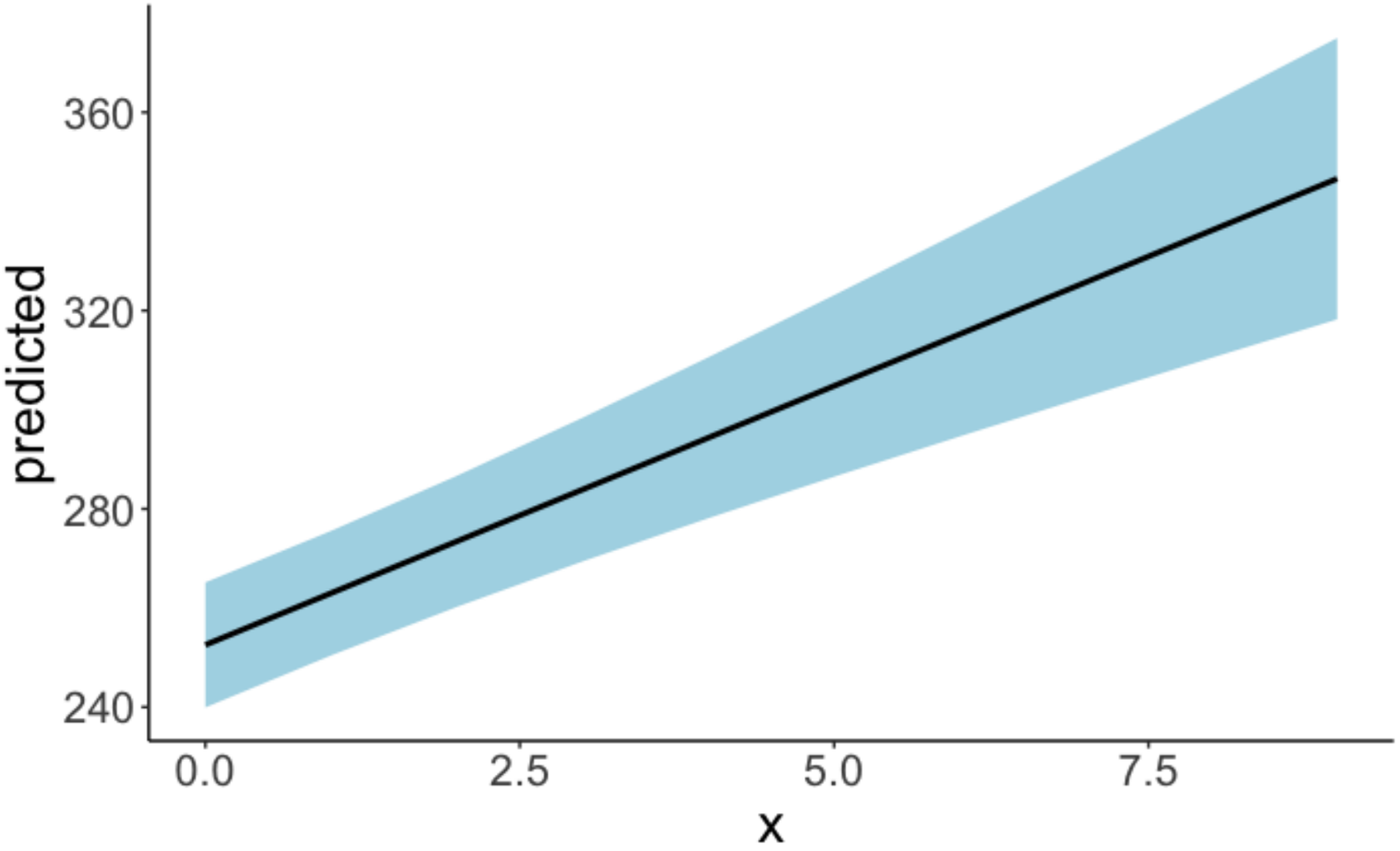
- the relationship between the variables of interest (marginalizing over other variables)

Marginal effects

returns a tidy data frame

```
df.plot = ggpredict(model = fit.random_intercept_slope,
                    terms = "days",
                    type = "fe")

ggplot(data = df.plot,
       mapping = aes(x = x,
                     y = predicted,
                     ymin = conf.low,
                     ymax = conf.high)) +
  geom_ribbon(fill = "lightblue") +
  geom_line(size = 1)
```

x	predicted	std.error	conf.low	conf.high
0	252.5426	6.433144	239.9338	265.1513
1	262.9947	6.407157	250.4369	275.5525
2	273.4468	6.743391	260.2300	286.6636
3	283.8989	7.392586	269.4097	298.3881
4	294.3510	8.281466	278.1197	310.5824
5	304.8032	9.341862	286.4935	323.1129
6	315.2553	10.522045	294.6325	335.8781
7	325.7074	11.786086	302.6071	348.8077
8	336.1595	13.109751	310.4649	361.8542
9	346.6117	14.476693	318.2379	374.9855

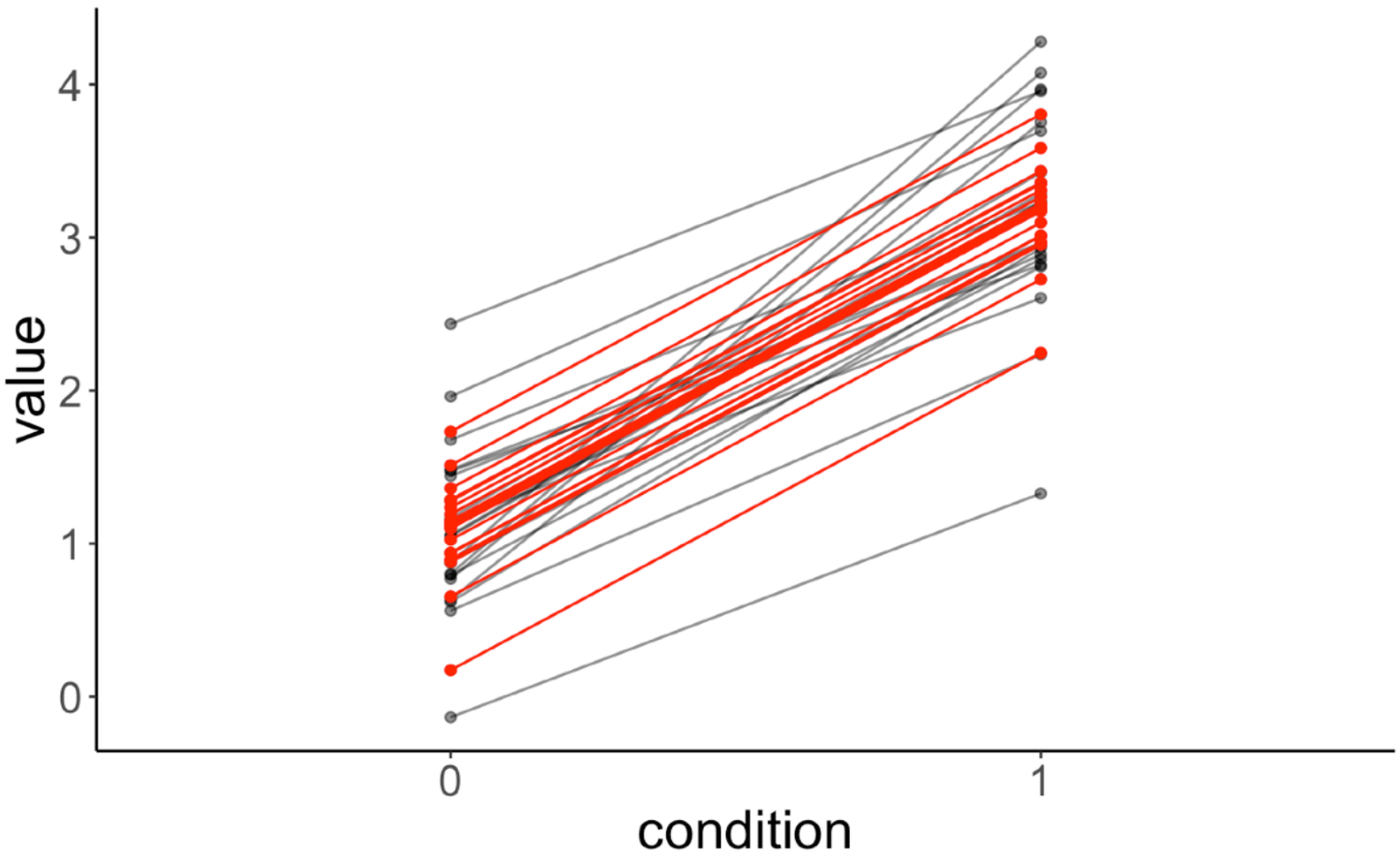


Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- **Let's stimulate some `lmer()`s**
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- Bootstrapping linear mixed effects models
- Getting p-values
- Understanding `lmer()` syntax
- Pitfalls in fitting `lmer()`s (and what to do about it)

Let's stimulate some `lmer()`s

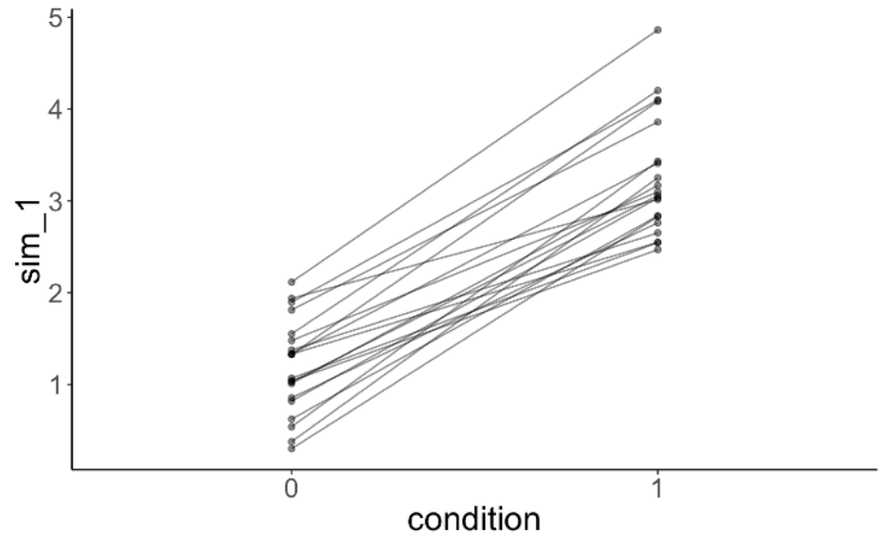
Outliers



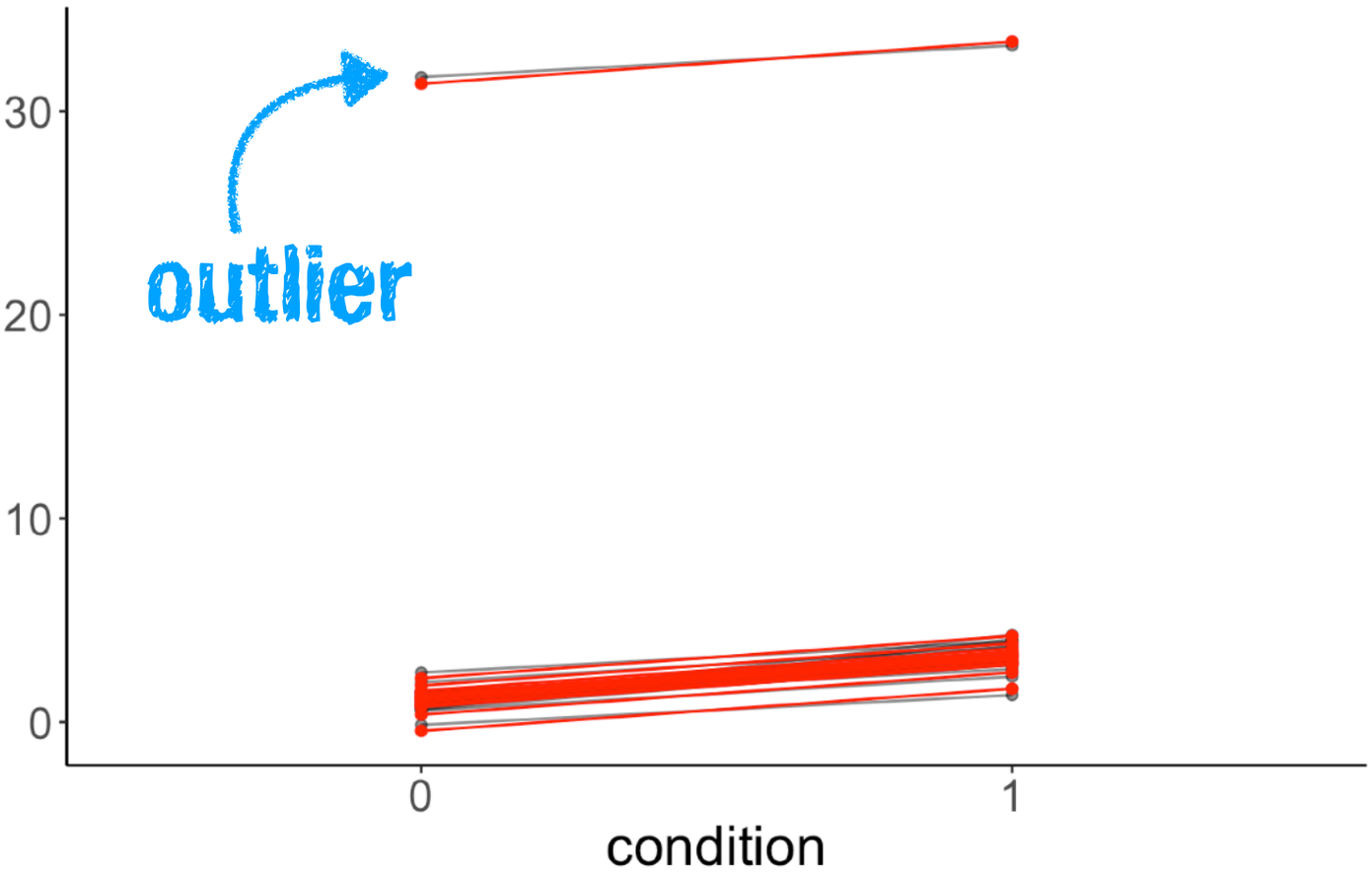

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + condition + (1 | participant)
   Data: df.test

REML criterion at convergence: 74.9

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

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

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

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

```
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



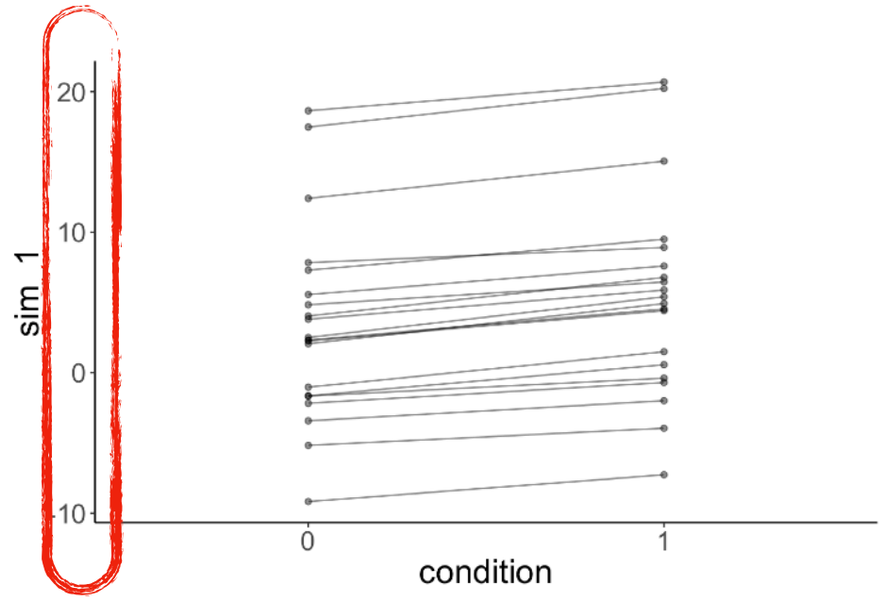
Outliers



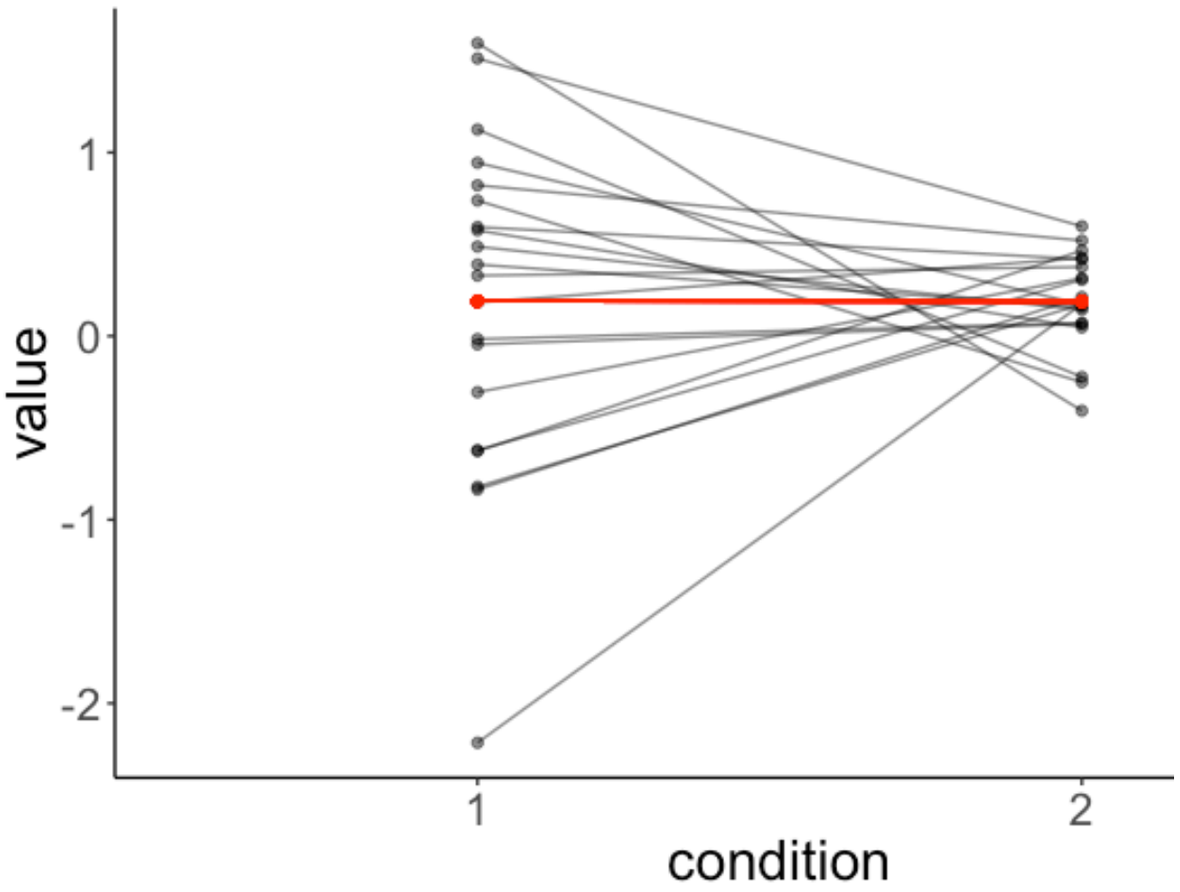

```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + condition + (1 | participant)
   Data: df.test

REML criterion at convergence: 171.7

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

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

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

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

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

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

simulated data



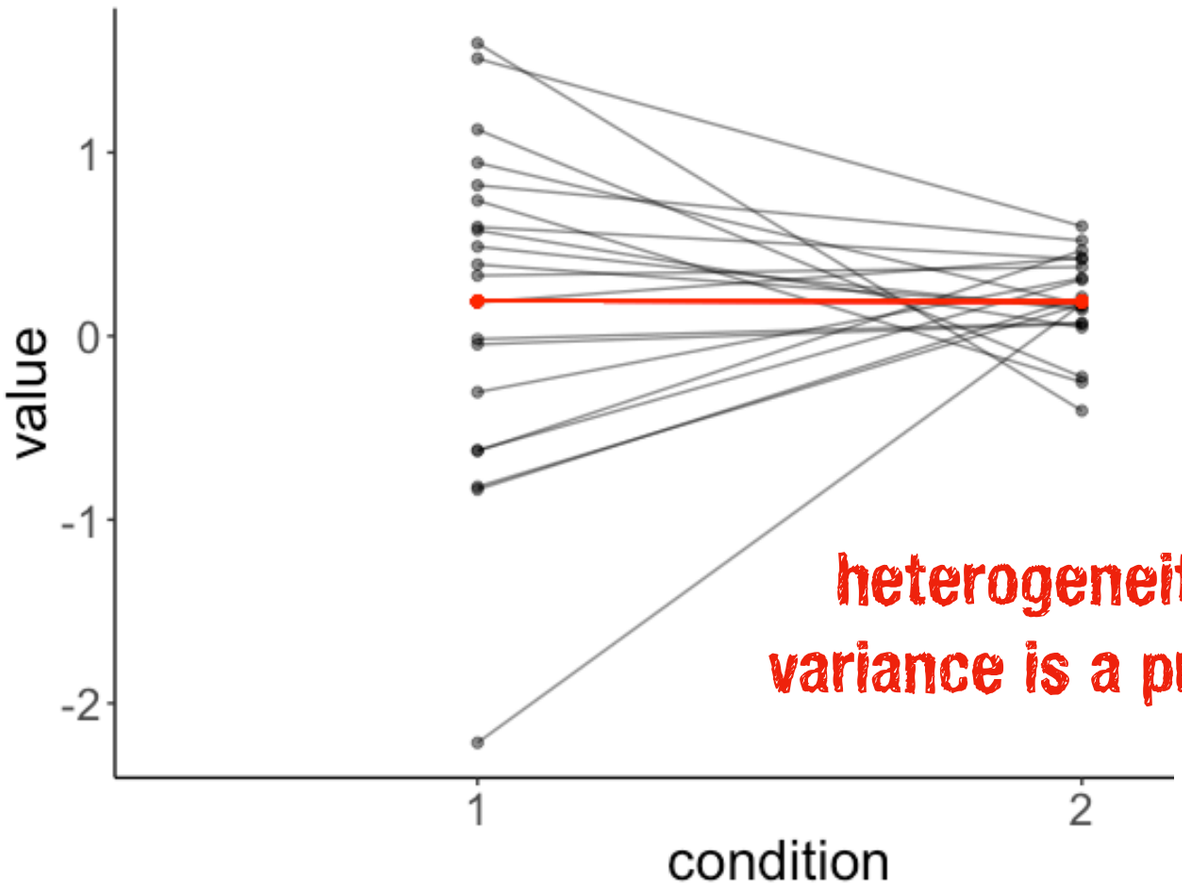
Different slopes




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

REML criterion at convergence: 83.6

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.5808 -0.3184  0.0130  0.4551  2.0913

Random effects:
 Groups      Name        Variance Std.Dev.
 participant (Intercept) 0.0000   0.0000
 Residual                0.4512   0.6717
Number of obs: 40, groups:  participant, 20

Fixed effects:
            Estimate Std. Error t value
(Intercept)  0.190524   0.150197   1.268
condition2  -0.001941   0.212411  -0.009

Correlation of Fixed Effects:
           (Intr)
condition2 -0.707
convergence code: 0
singular fit
```

clearly there are interindividual differences though?!

Different slopes



heterogeneity of variance is a problem

more parameters than data points

the "model" would just reproduce the data

random intercept

random slope

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

won't work

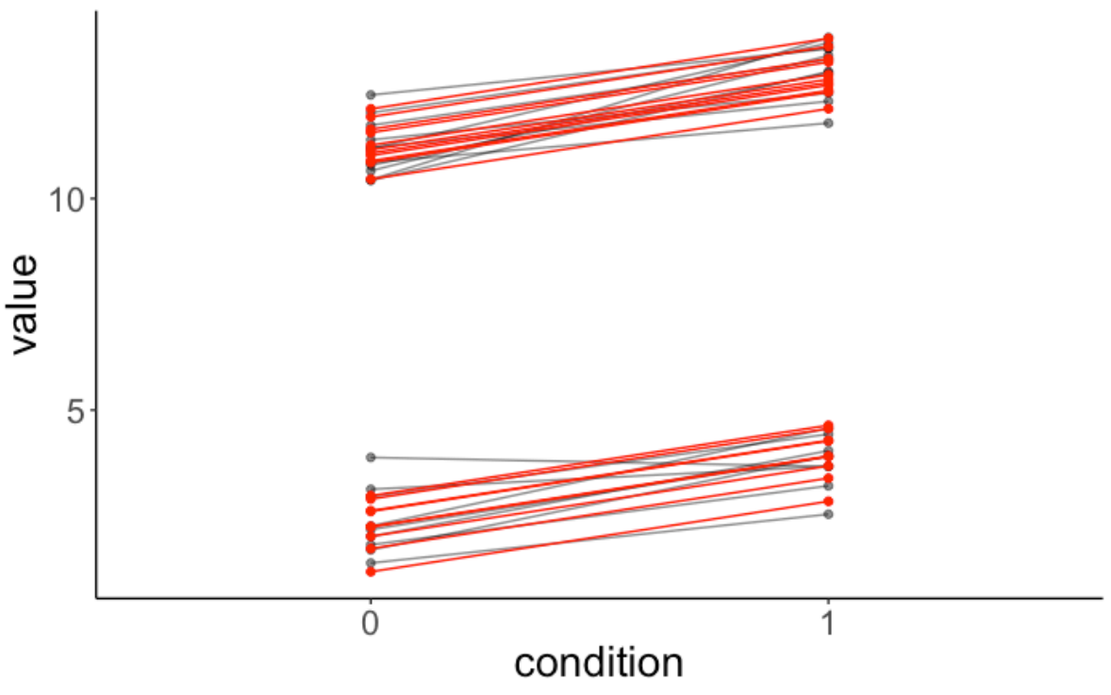
```
Error: number of observations (=40) <= number of random effects (=40) for term
(1 + condition | participant); the random-effects parameters and the residual
variance (or scale parameter) are probably unidentifiable
```

Mixture of participants

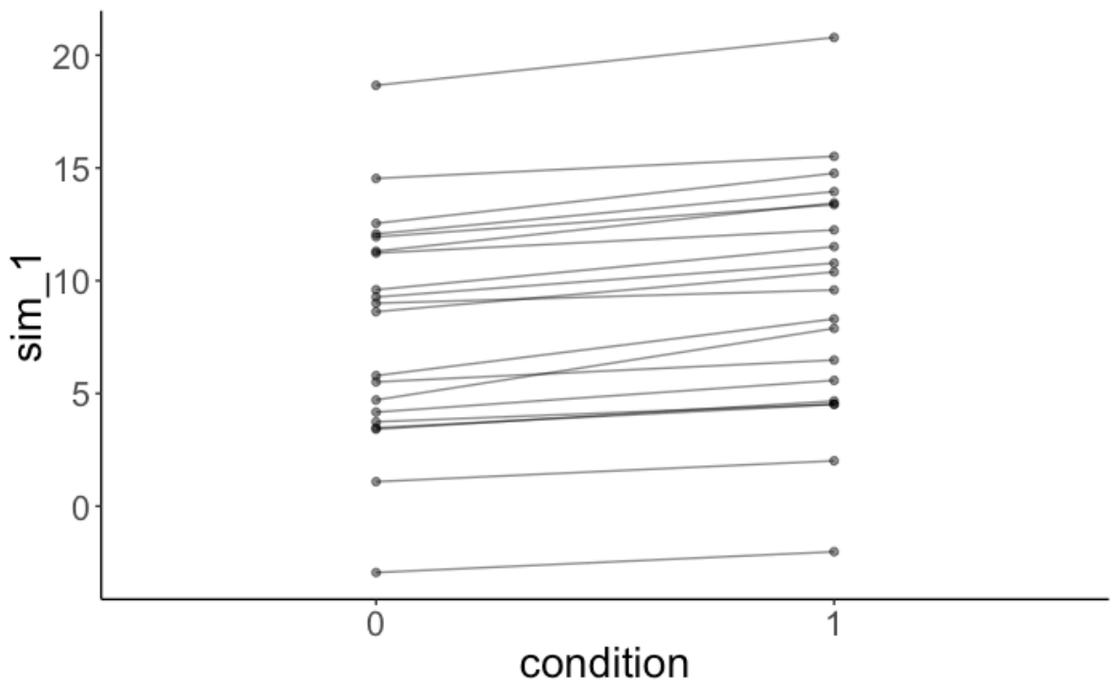
```
lmer(formula = value ~ 1 + condition +
                (1 | participant),
     data = df.mixed)
```

```
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
```

actual data



simulated data



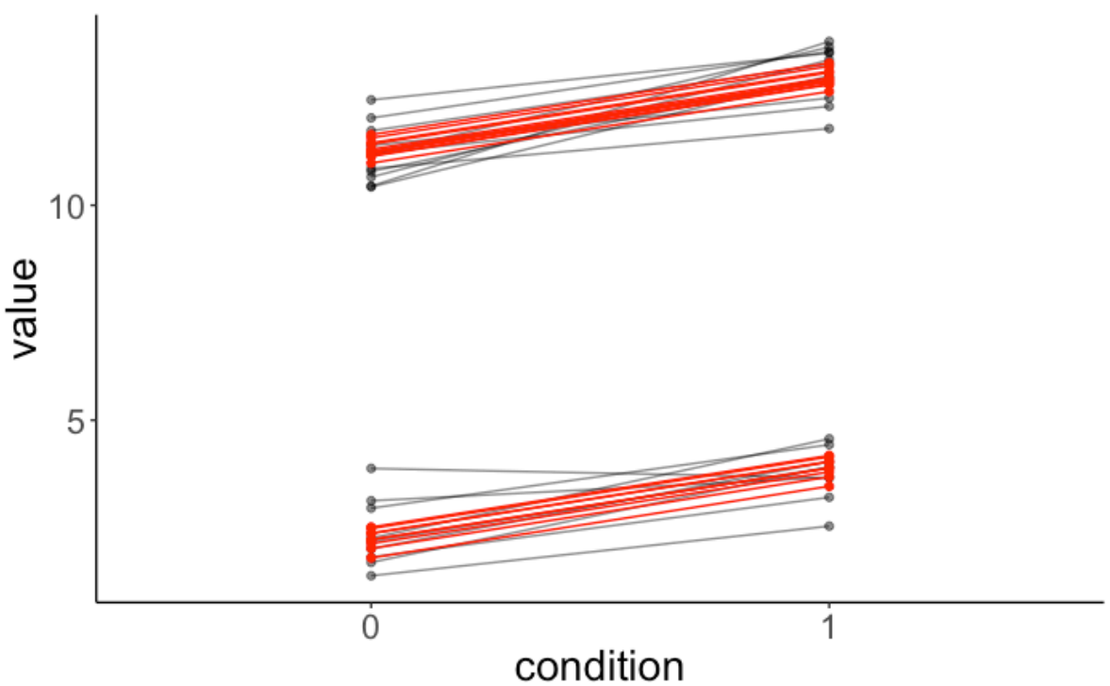
Mixture of participants

explicitly model the different groups

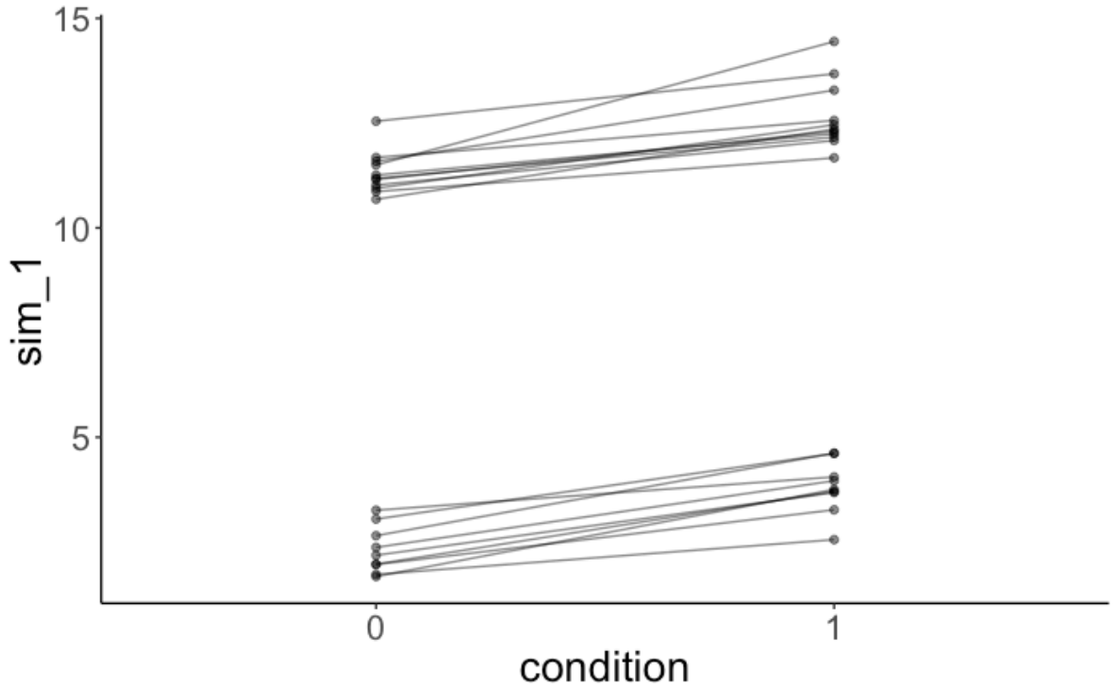
```
lmer(formula = value ~ 1 + group +
                condition +
                (1 | participant),
     data = df.mixed)
```

```
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   0.5934
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
```

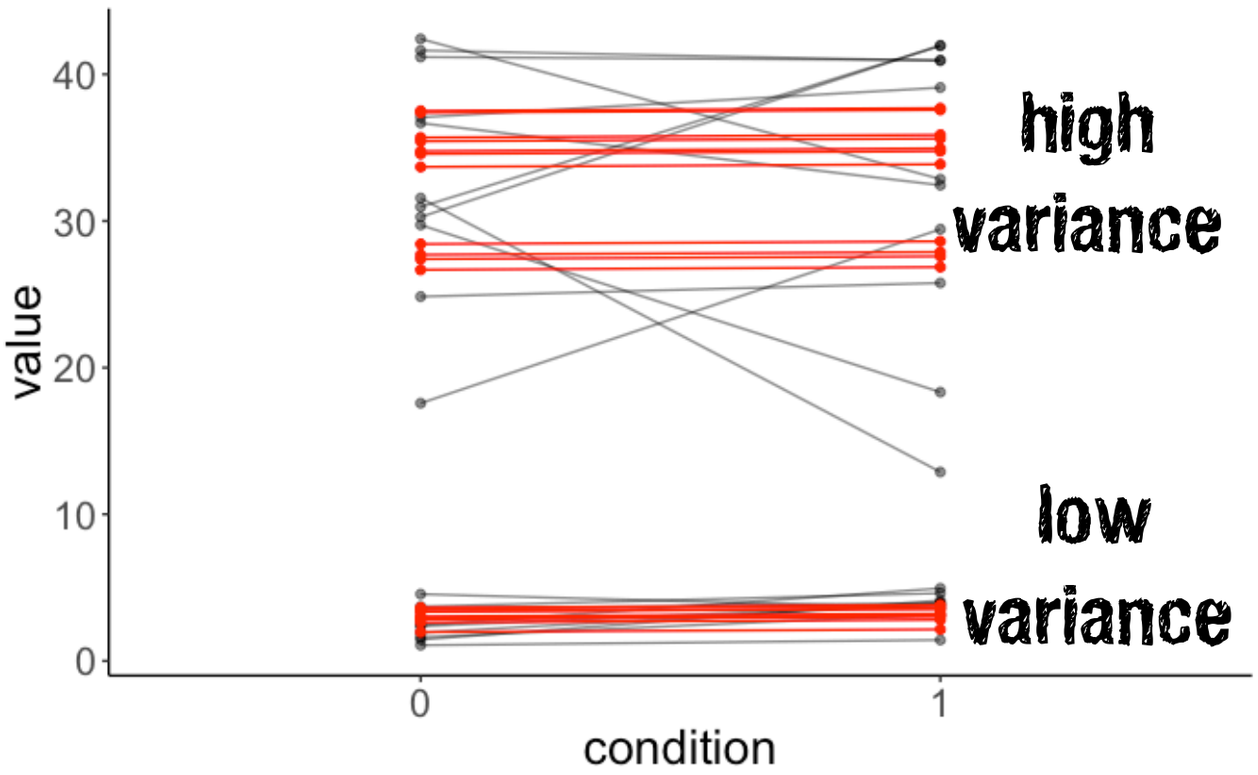
actual data

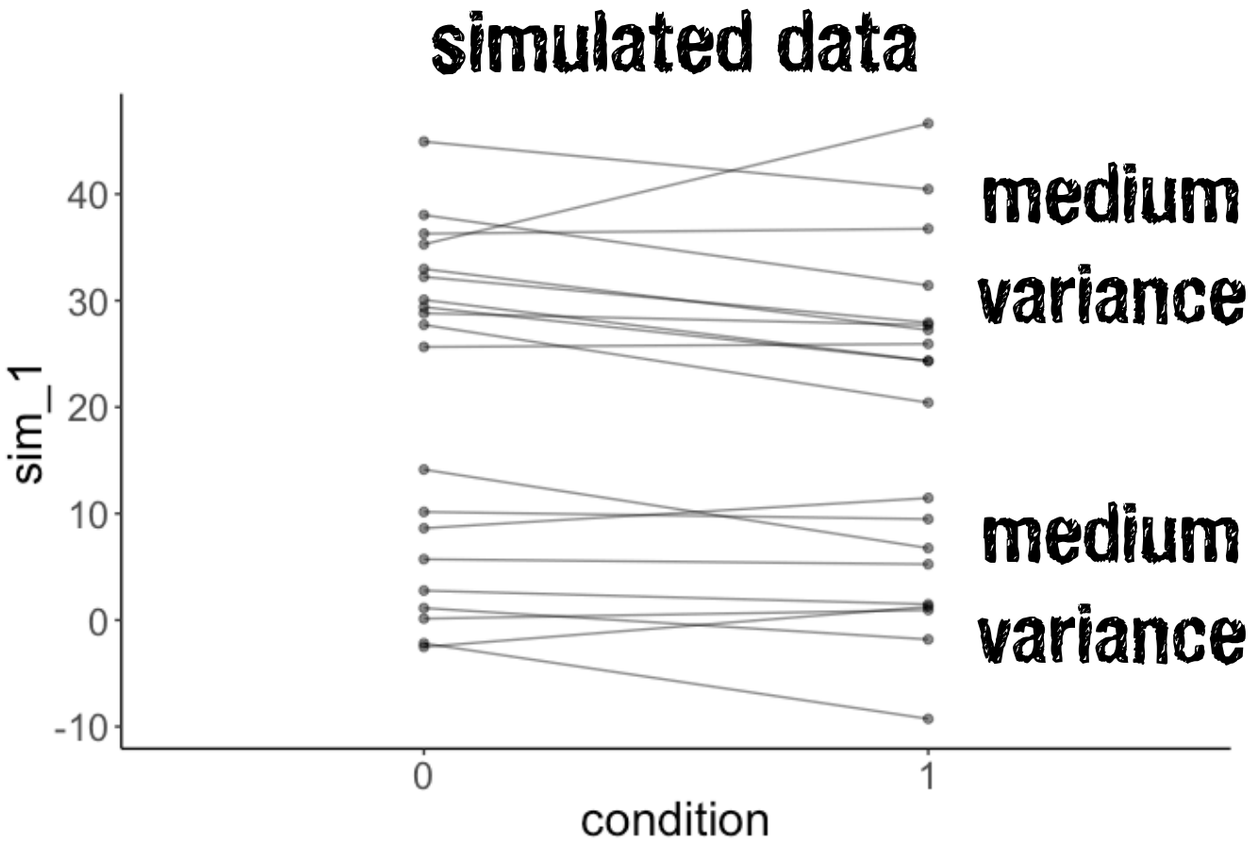


simulated data



Problem case: Heterogeneity in variance between groups



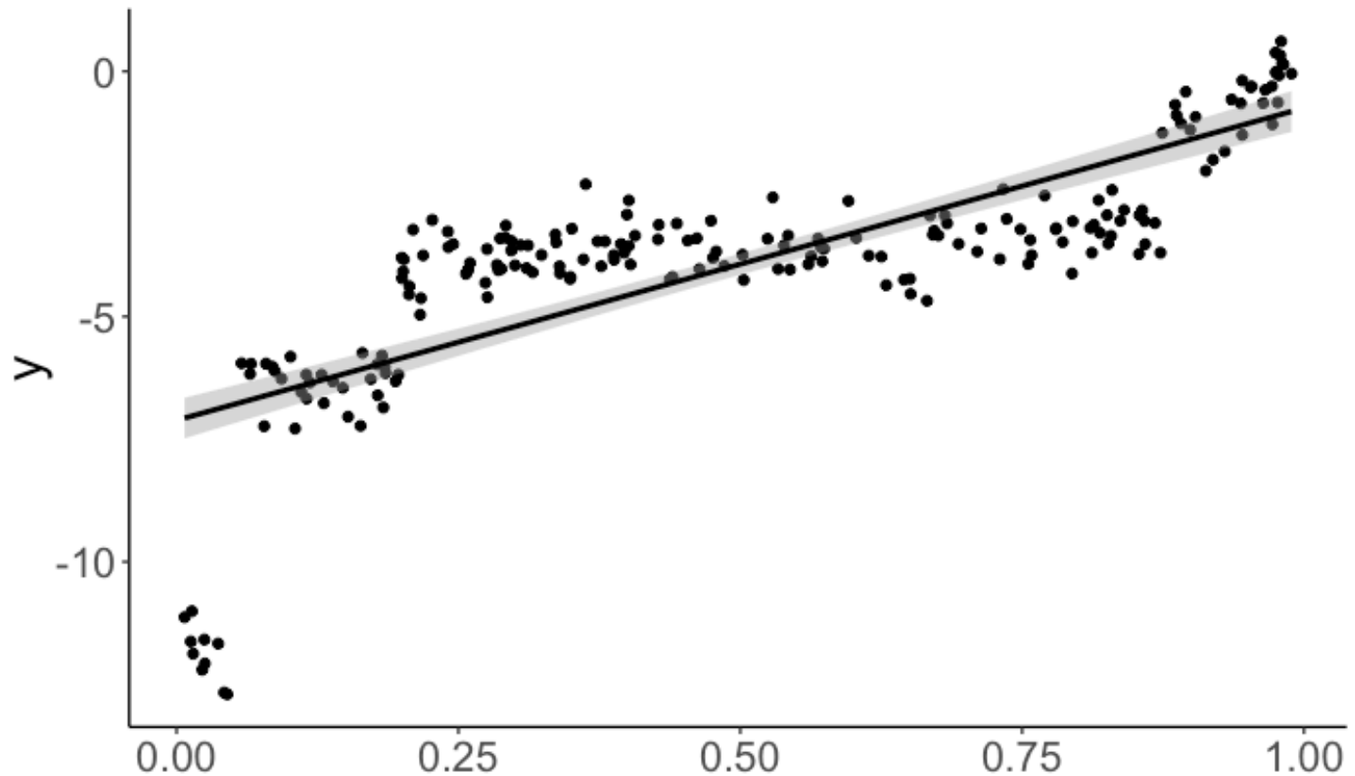



```
Linear mixed model fit by REML ['lmerMod']
Formula: value ~ 1 + group + condition + (1 | participant)
   Data: df.mixed

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

Simpson's paradox

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



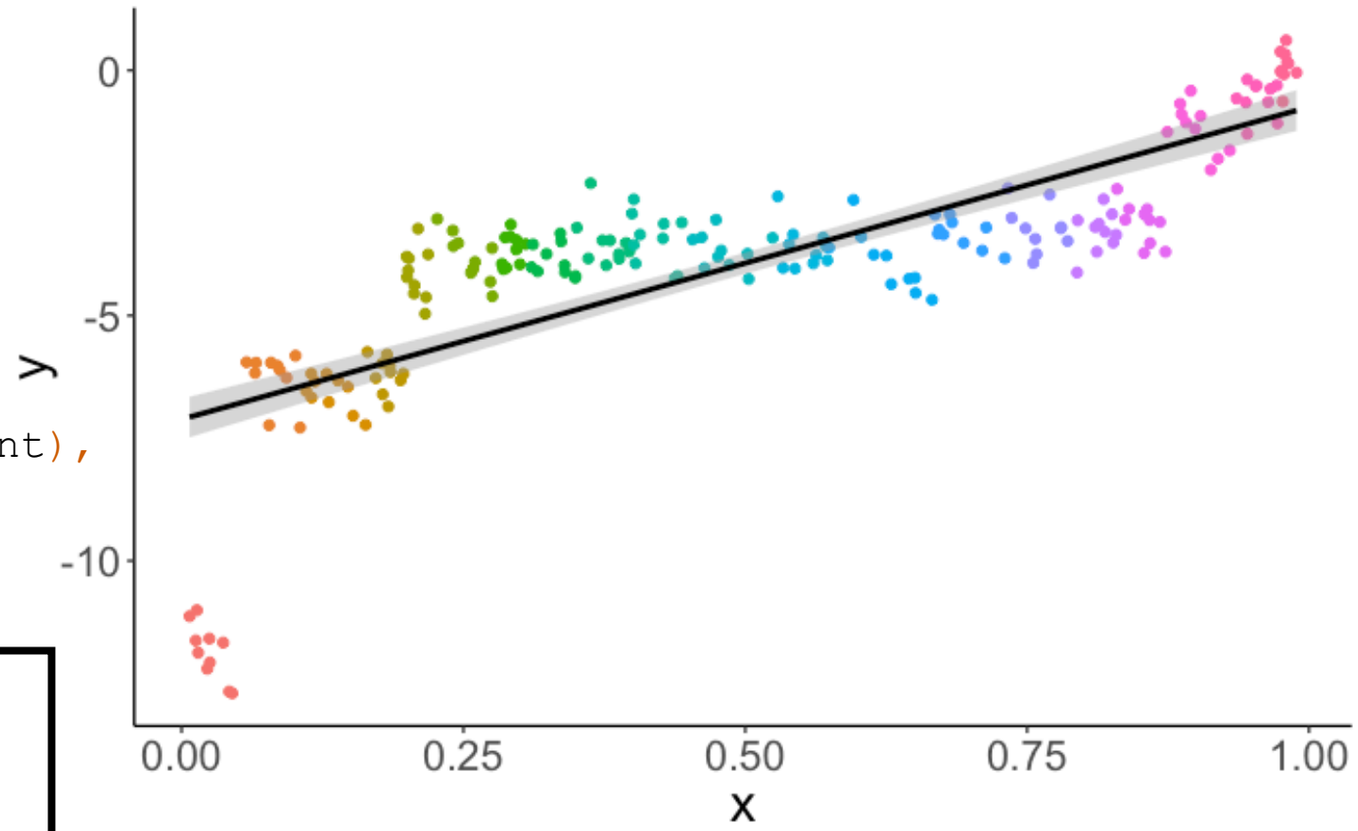

```
Call:
lm(formula = y ~ x, data = df.simpson)

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

```
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
```

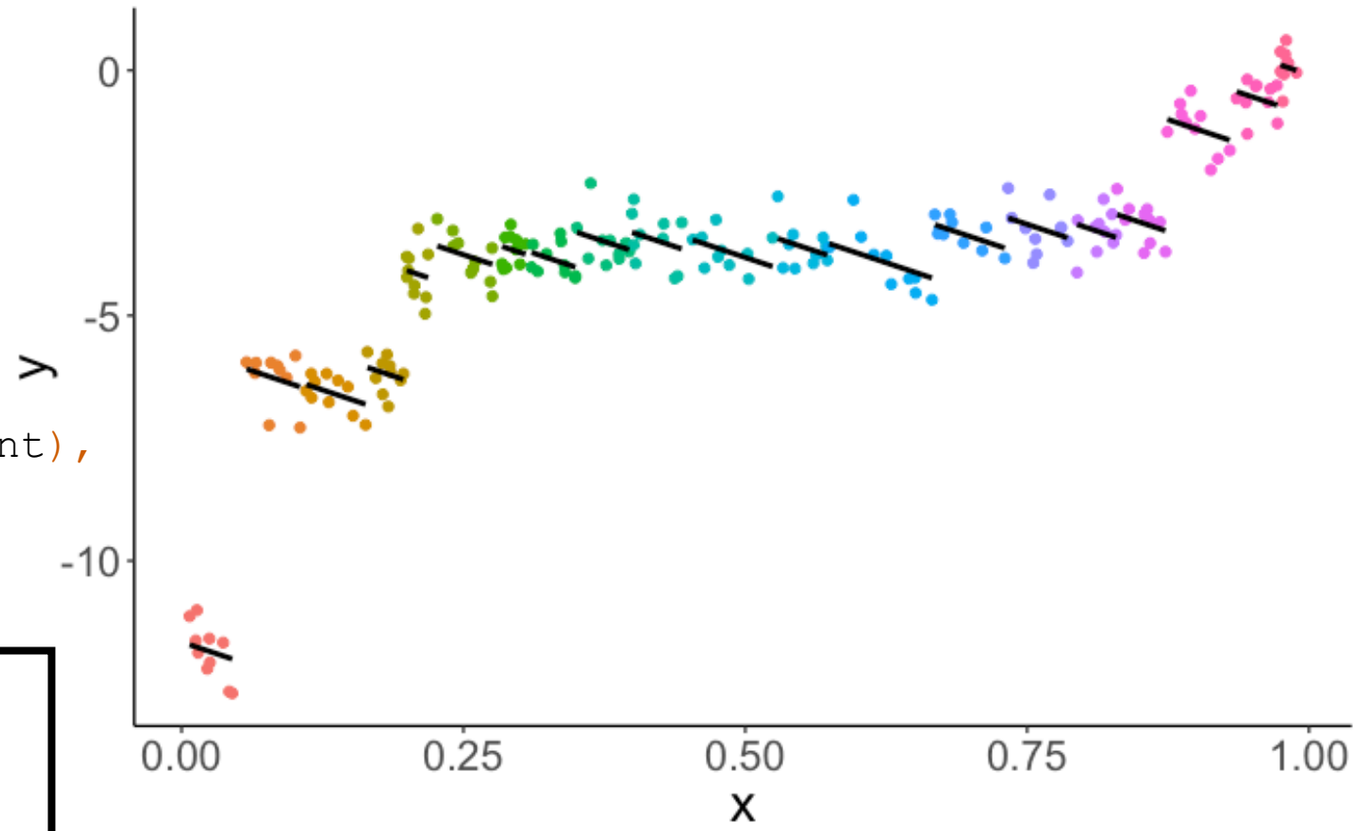


negative (!) relationship between x and y

Simpson's paradox

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

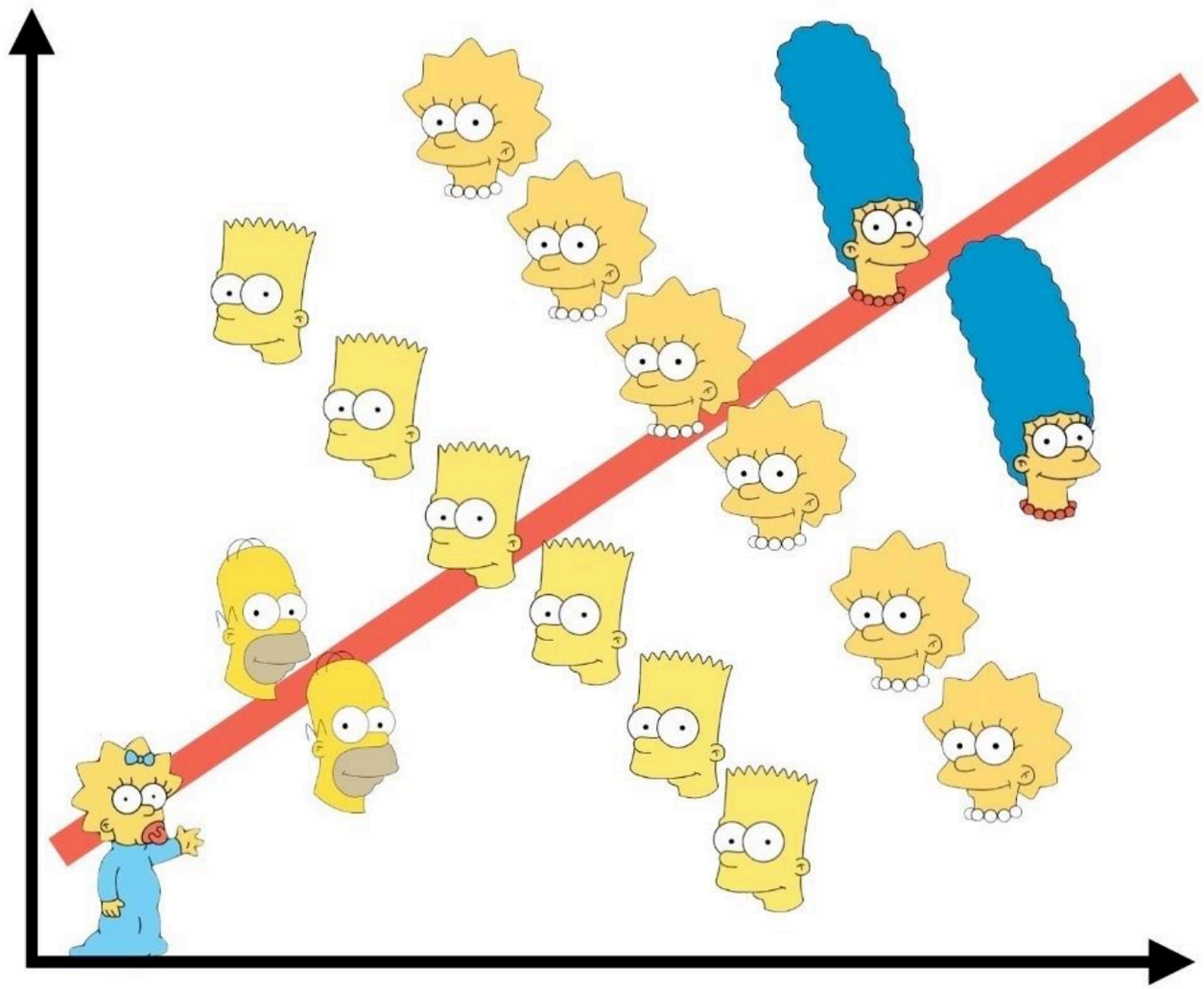
```
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ 1 + x + (1 | participant)
   Data: df.simpson

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)

Simpson's paradox



- when the relationship between two variables changes strongly after conditioning on a grouping variable (e.g. individual participants)
- interesting real world cases
- **google** it!

Simpson's paradox

UC Berkeley gender bias (1973)

	Men		Women	
	Applicants	Admitted	Applicants	Admitted
Total	8442	**44%**	4321	35%

Department	Men		Women	
	Applicants	Admitted	Applicants	Admitted
A	***825***	62%	108	**82%**
B	***560***	63%	25	**68%**
C	325	**37%**	***593***	34%
D	417	33%	375	**35%**
E	191	**28%**	***393***	24%
F	373	6%	341	**7%**

overall men are more likely to be admitted

men not more likely to be admitted when broken down by department

In fact, the pooled and corrected data showed a "small but statistically significant bias in favor of women."

women applied to more competitive departments

Bickel, P. J., Hammel, E. A., & O'Connell, J. W. (1975). Sex Bias in Graduate Admissions: Data from Berkeley. *Science*, 187(4175), 398-404.

Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- Let's stimulate some `lmer()`s
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- **Bootstrapping linear mixed effects models**
- Getting p-values
- Understanding `lmer()` syntax
- Pitfalls in fitting `lmer()`s (and what to do about it)

Bootstrapping linear mixed effects models

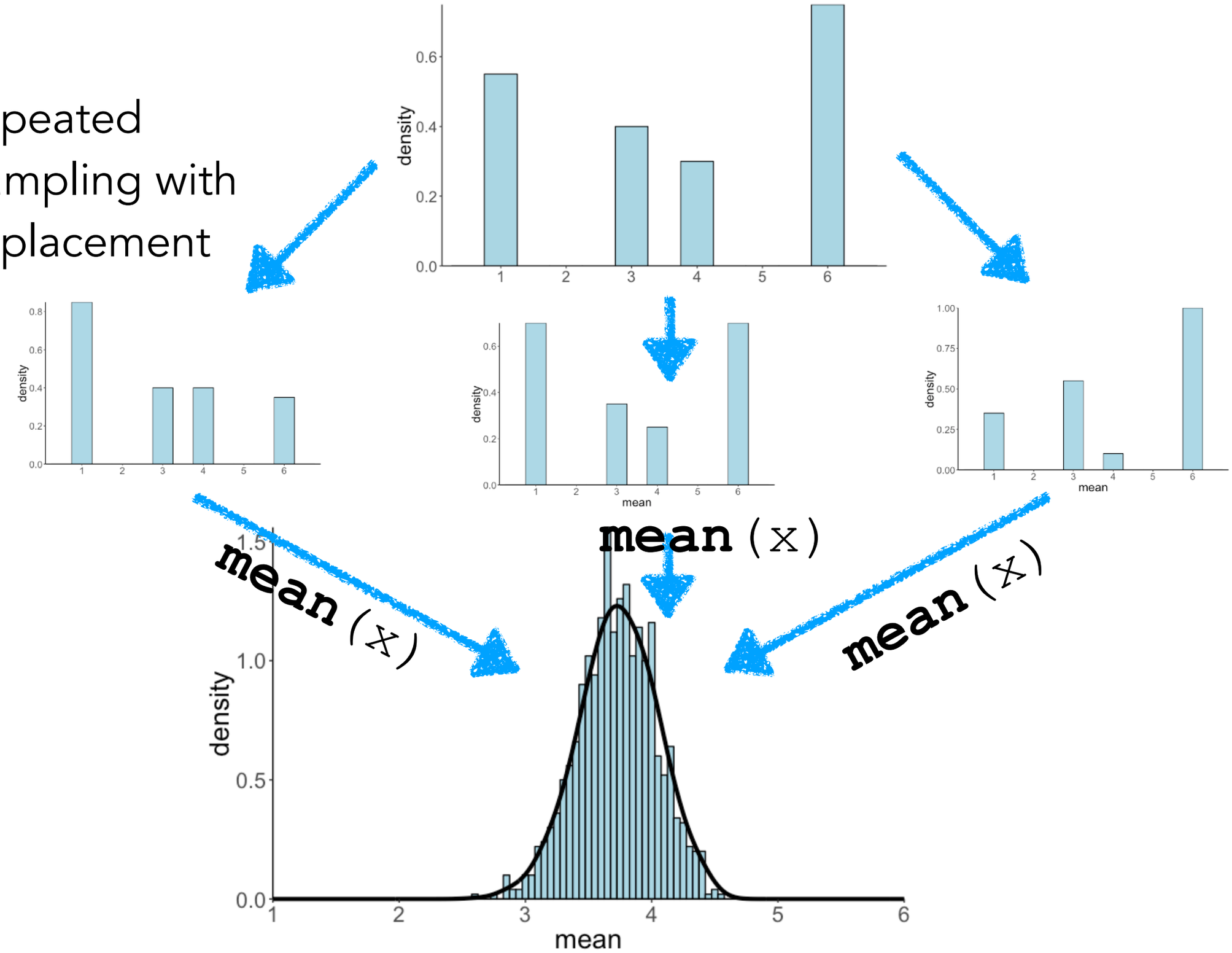
Bootstrapping

- bootstrapping allows us to determine uncertainty associated with parameter estimates in a model
- we can use it to get **confidence intervals** on parameter estimates
- the main idea is:
  - we treat our sample as if it was the population
  - we then re-sample from our sample **with replacement**
  - we look at the distribution over our parameter estimate across those re-samples

Bootstrap

all we have is our sample

repeated sampling with replacement



sampling distribution

← **Tweet**

Sean J. Taylor
@seanjtaylor

One time, I had to estimate a confidence interval for a sample estimate and there was no convenient formula for analytic standard errors. #IBootstrapped

> **Carol Roth** @caroljsroth · Feb 6
> Did you bootstrap your way to success? Reply to this w your story & #IBootstrapped
> Show this thread

2:30 PM · Feb 9, 2020 · Twitter Web App

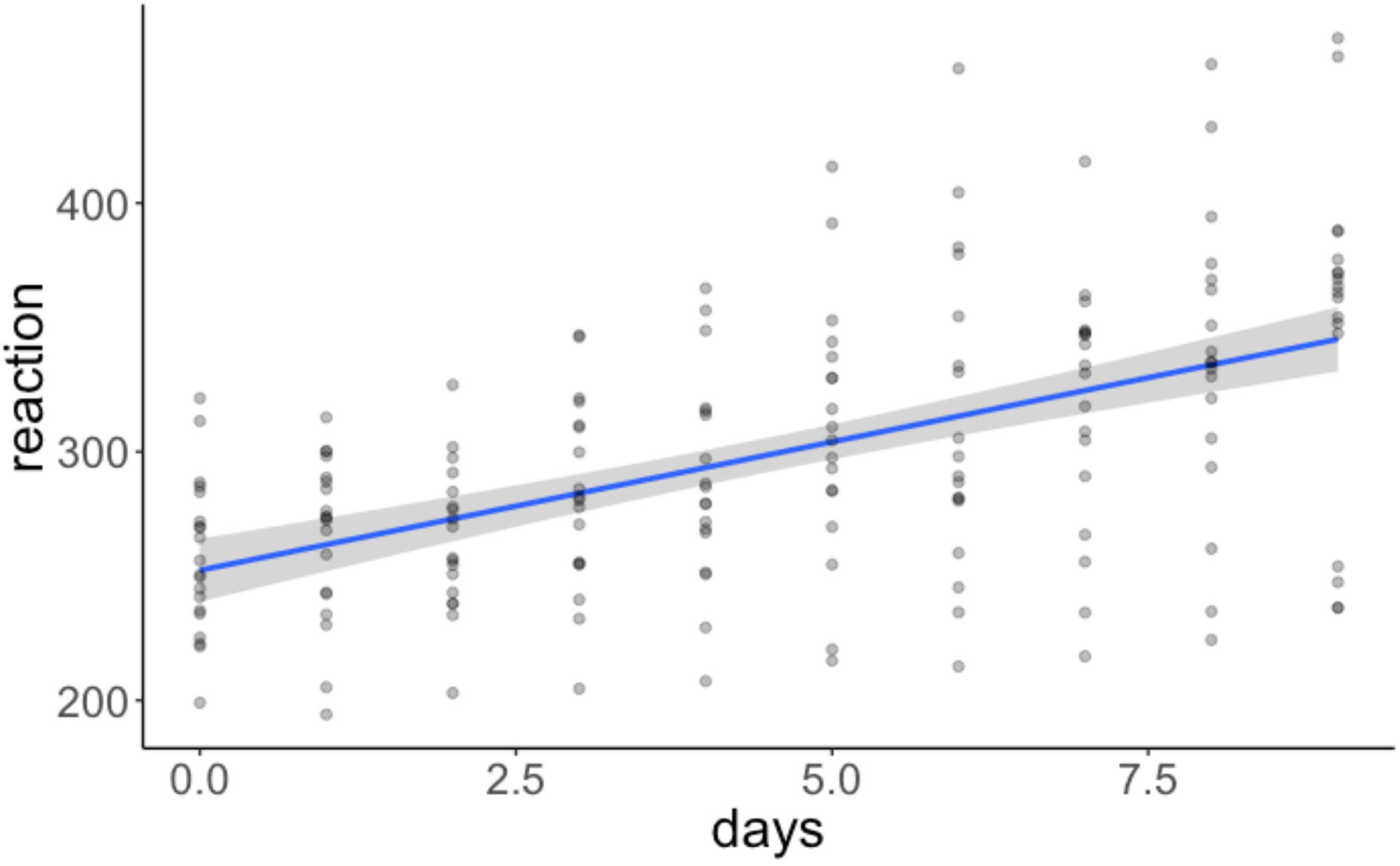
42 Retweets **600** Likes

Bootstrap your way to success!

Bootstrapping a linear model `lm()`

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

```
(Intercept)            days
  252.32070        10.32766
```



Bootstrapping a linear model `lm()`

function in the `modelr` library

```
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)) %>%
```

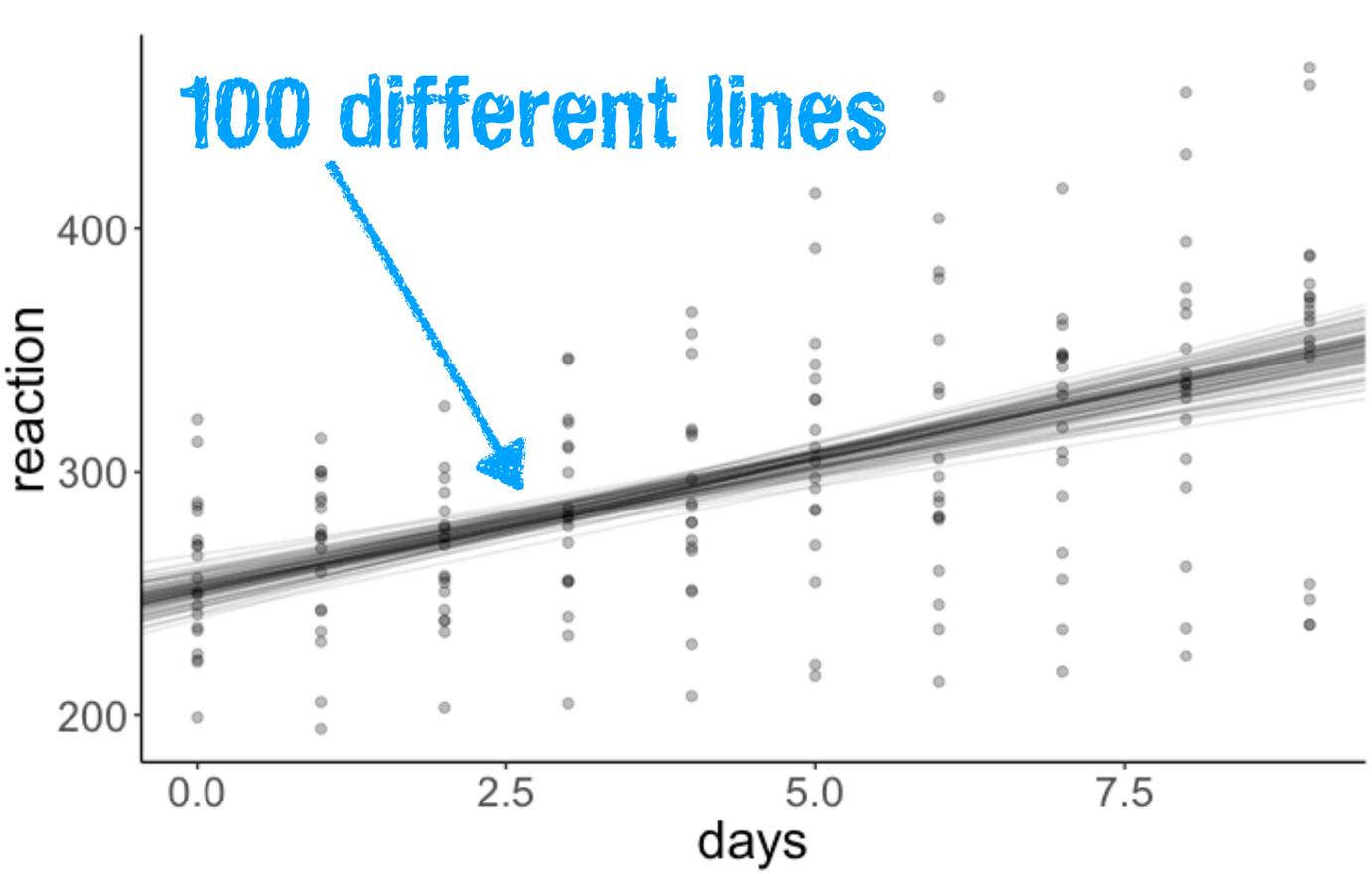
resampled data sets | model fit | coefficients

	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()`

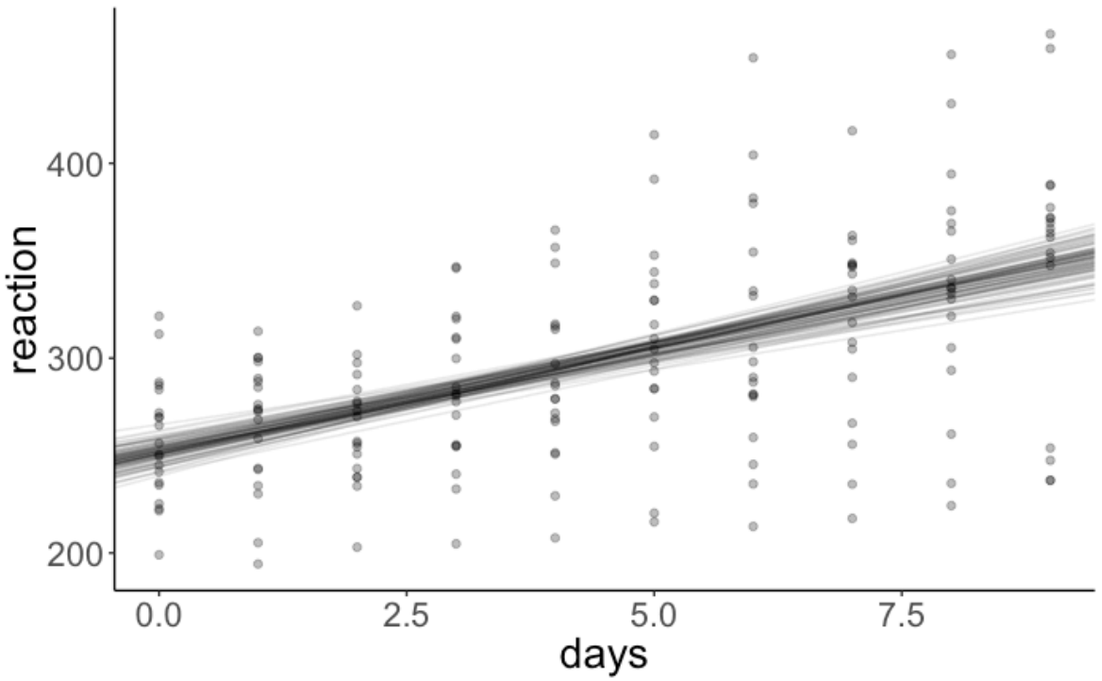
```
df.boot = df.sleep %>%
  bootstrap(n = 100,
            id = "id") %>%
  mutate(fit = map(strap, ~ lm(formula = reaction ~ 1 + days, data = .)),
         tidy = map(fit, tidy)) %>%
  unnest(tidy) %>%
  select(id, term, estimate) %>%
  pivot_wider(names_from = term,
              values_from = estimate) %>%
  clean_names()
```

intercept and slope for each fitted model

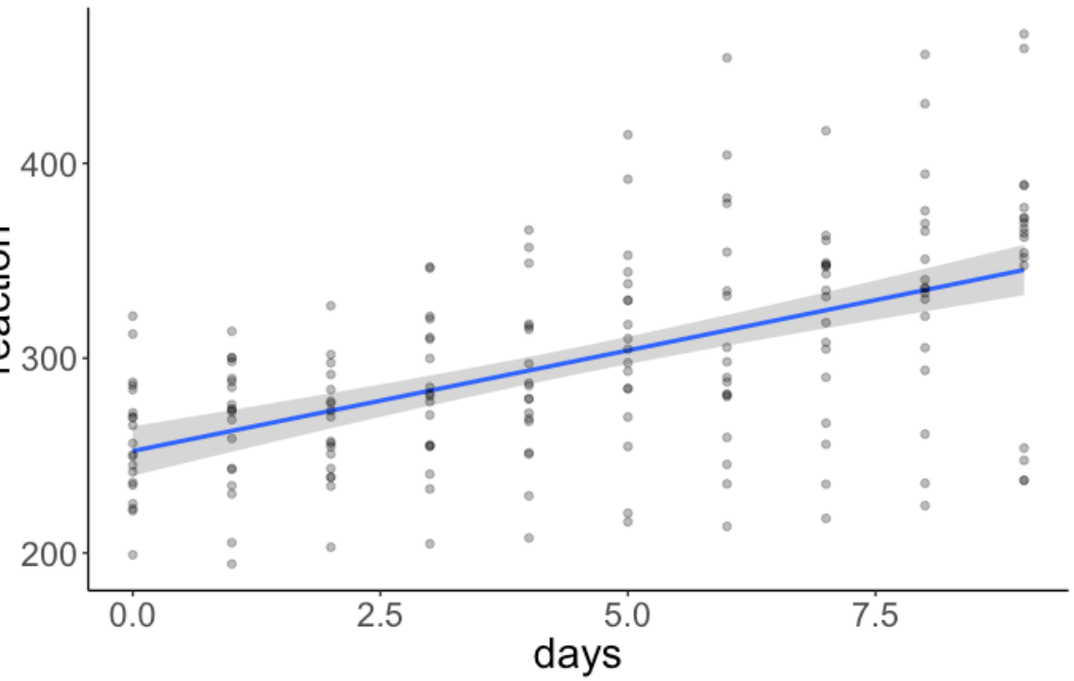


	id	intercept	days
1	001	243.2329	11.610940
2	002	241.7720	11.985003
3	003	257.6160	9.306734
4	004	253.9766	8.481162
5	005	262.3882	7.256231
6	006	259.1082	9.377909
7	007	247.7098	9.589859
8	008	250.6057	9.822907
9	009	247.5809	9.883666
10	010	255.6068	10.351093

Bootstrapping a linear model `lm()`



confidence interval via bootstrap



confidence interval with parametric assumptions

(using the standard error of the sampling distribution)

Bootstrapping an **lmer()**

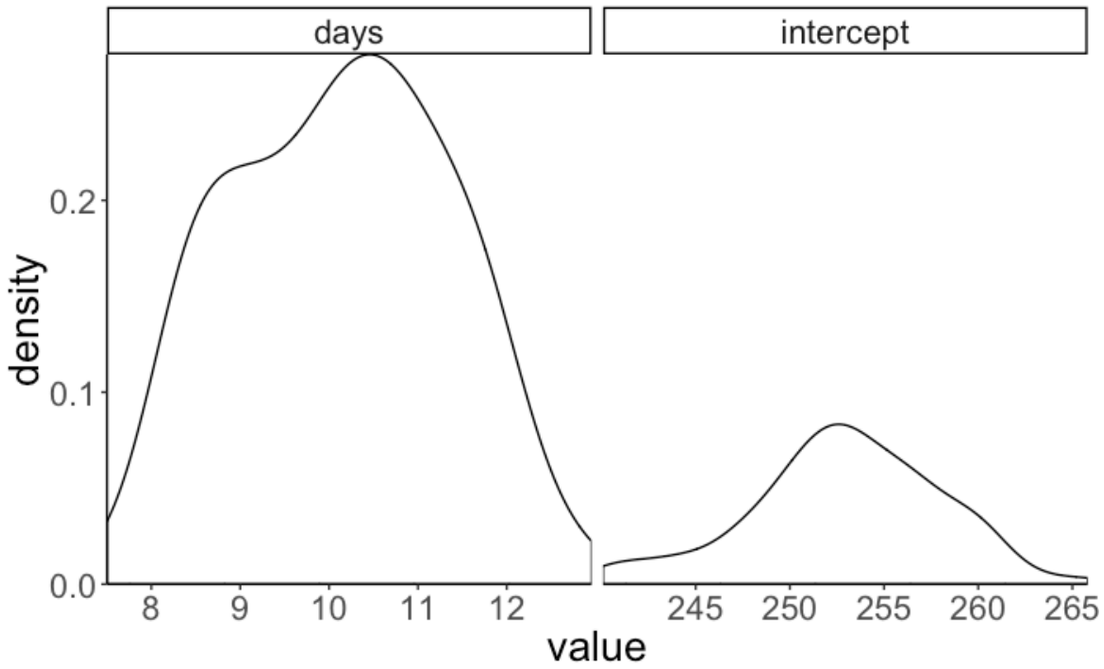
```
library("boot")

# fit the model
fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days | subject),
                data = df.sleep)

# bootstrap parameter estimates
boot.lmer = bootMer(fit.lmer,
                    FUN = fixef,
                    nsim = 100)

# compute confidence interval
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
```

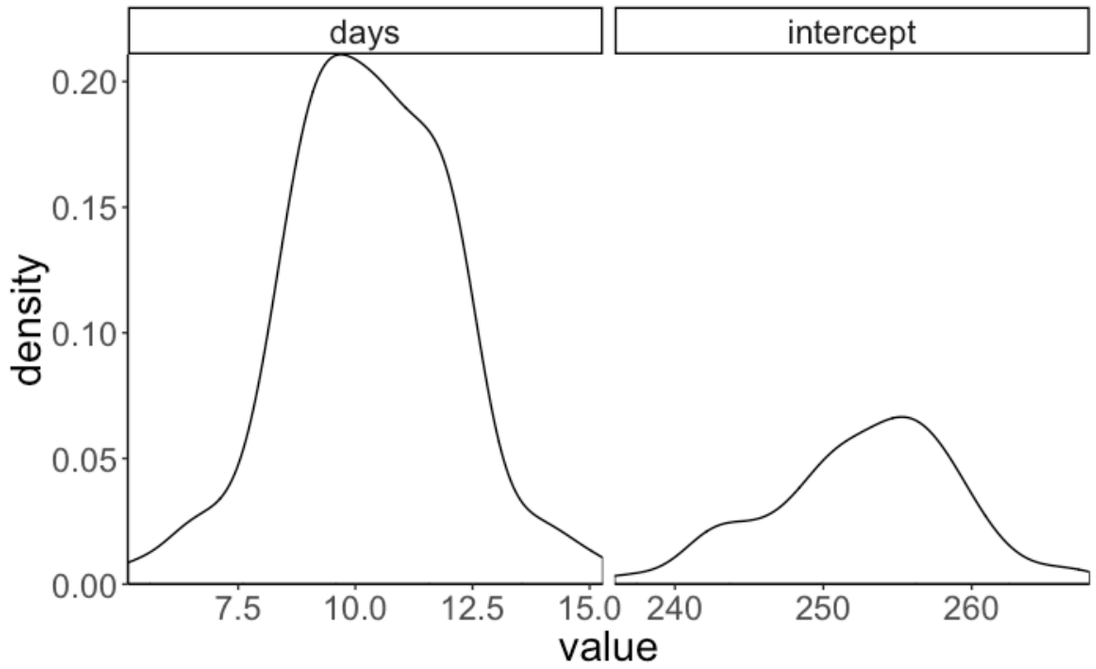
check whether the confidence interval excludes 0

Bootstrapping an `lmer()`

`lm()`



`lmer()`



more confident in the parameter estimates

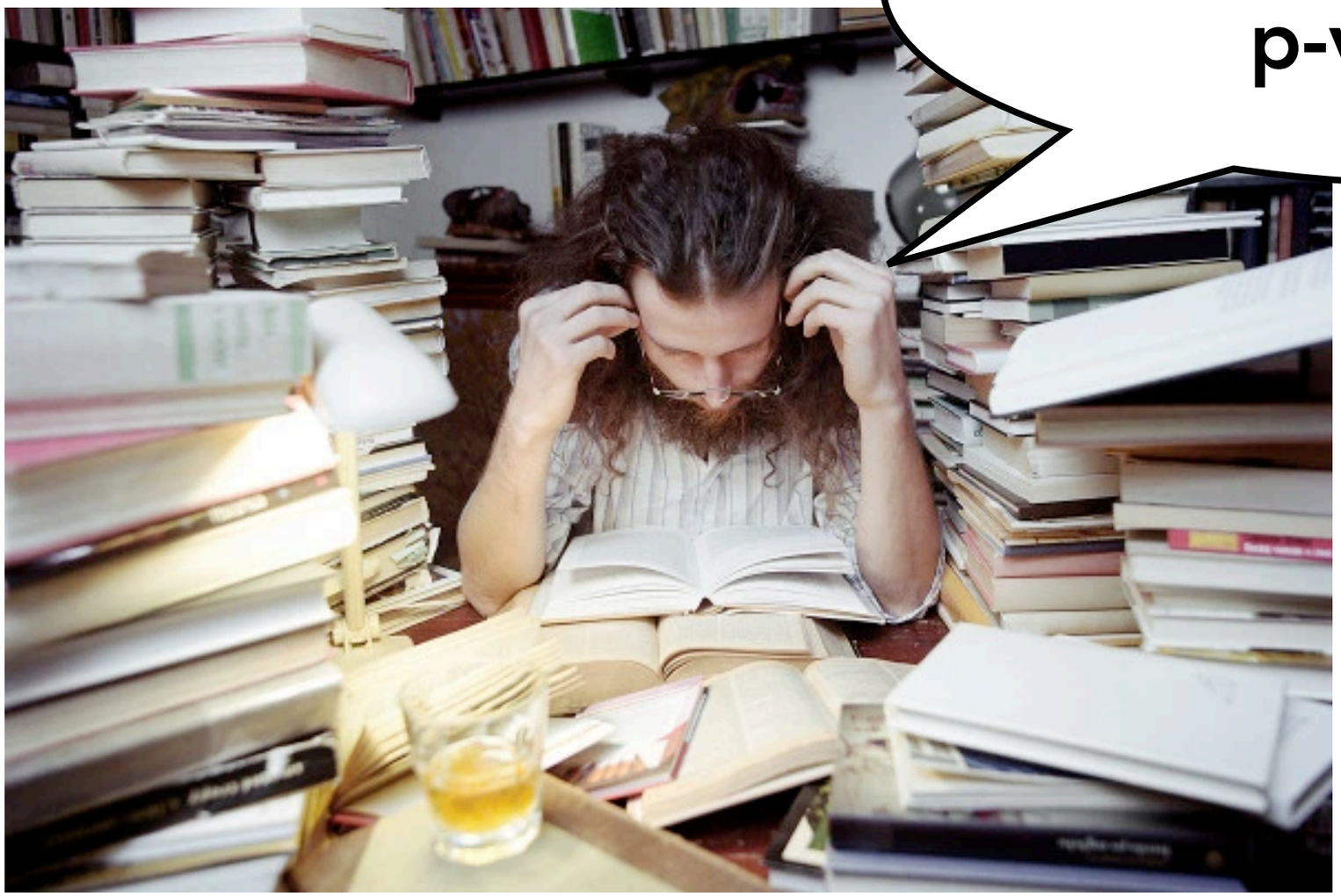
bootstrapped parameter estimates

Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- Let's stimulate some `lmer()`s
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- Bootstrapping linear mixed effects models
- **Getting p-values**
- Understanding `lmer()` syntax
- Pitfalls in fitting `lmer()`s (and what to do about it)

Getting p-values

Reviewer #2



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

```
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
```

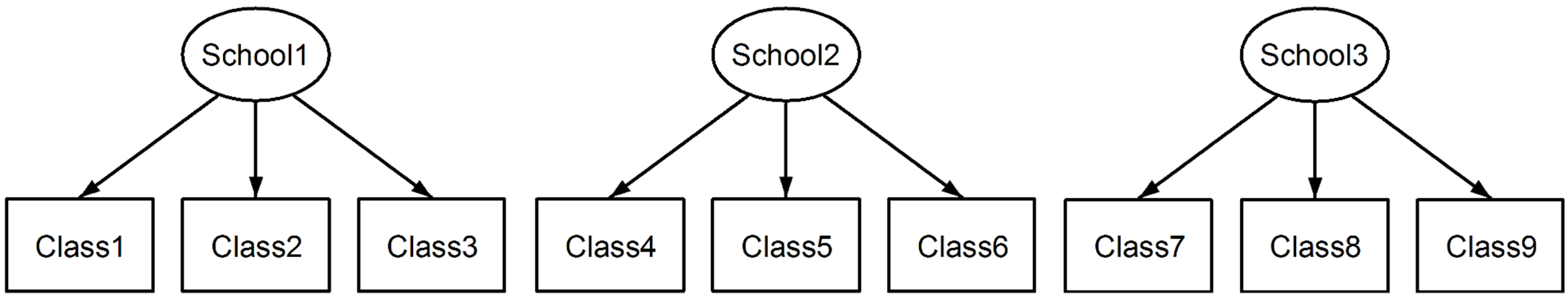
Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- Let's stimulate some `lmer()`s
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- Bootstrapping linear mixed effects models
- Getting p-values
- **Understanding `lmer()` syntax**
- Pitfalls in fitting `lmer()`s (and what to do about it)

Understanding `lmer()` syntax

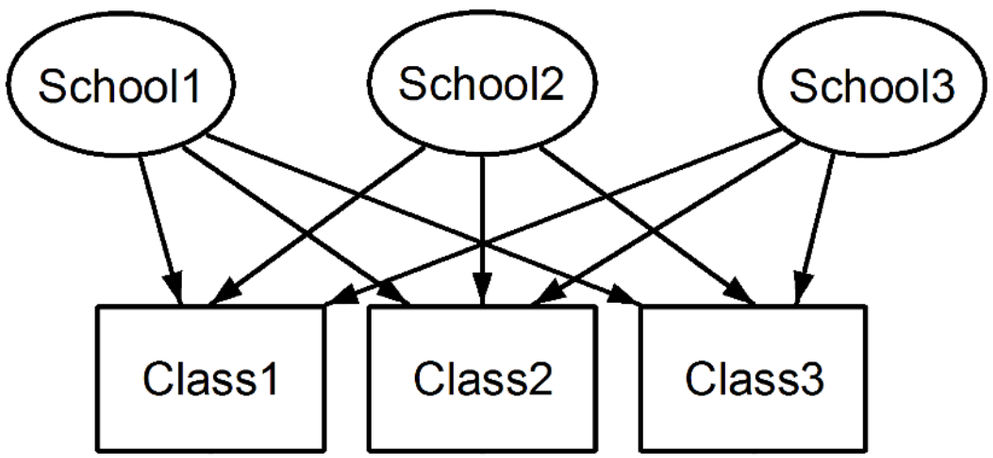
Multi-level models

nested `(1|School/Class)`



each class only appears within one school

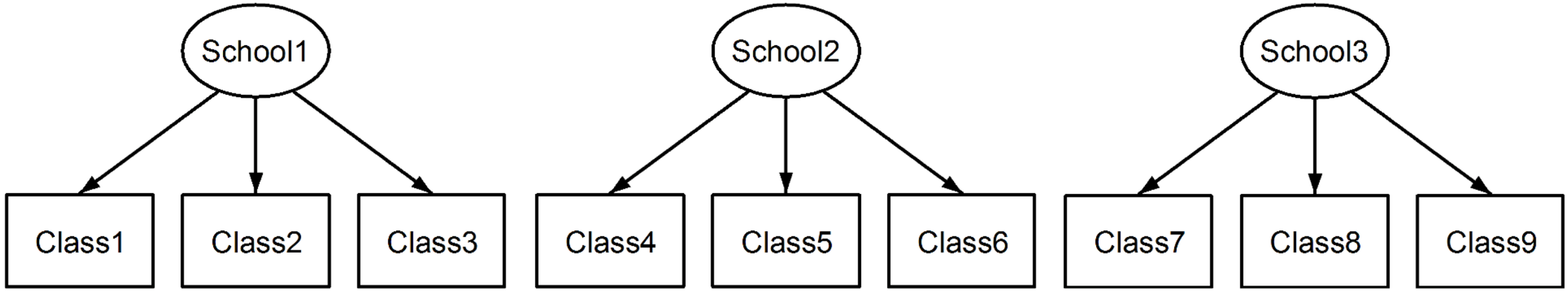
crossed `(1|School) + (1|Class)`



each class appears in each of the schools

Multi-level models

nested `(1|School/Class)`

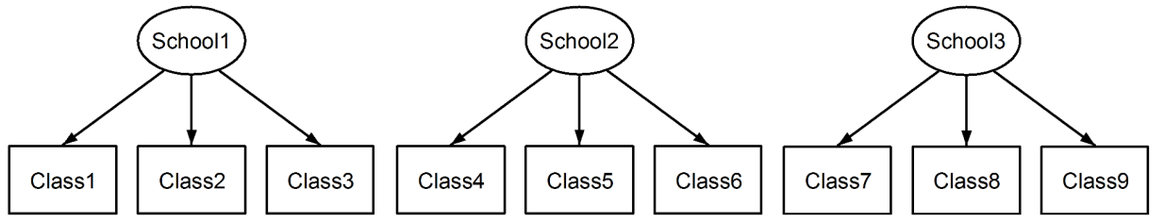



```
1 # fit nested model
2 fit.nested = lmer(extro ~ open + agree + social + (1 | school/class), data = df.school)
3
4 # print model summary
5 fit.nested %>% summary()
6
7 # model coefficients
8 fit.nested %>% coef()
```

```
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

random intercepts of class within each school

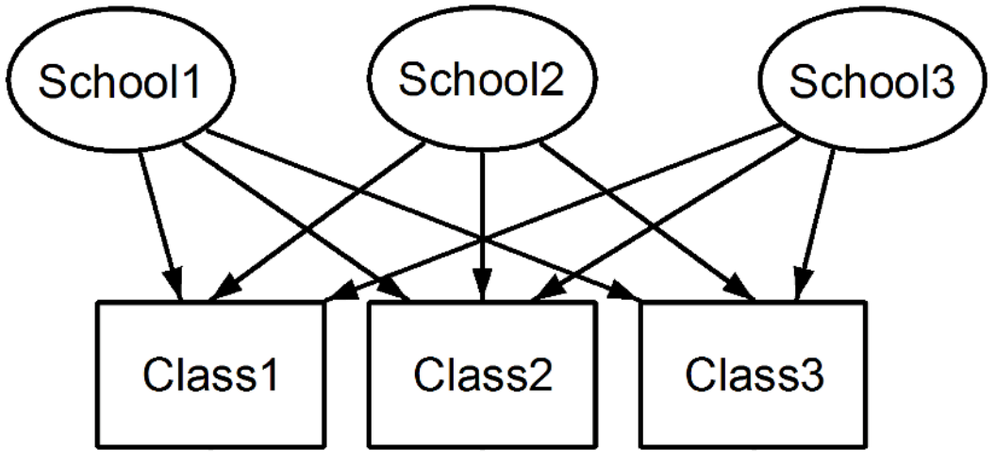
```
$`class:school`
      (Intercept)         open        agree       social
a:I      53.77106 0.006106514 -0.007665927 0.0005404069
a:II     58.23842 0.006106514 -0.007665927 0.0005404069
a:III    58.71819 0.006106514 -0.007665927 0.0005404069
a:IV     58.68813 0.006106514 -0.007665927 0.0005404069
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
```

random intercepts of schools

```
# model coefficients
fit.nested %>% coef()
```

Multi-level models

crossed `(1|School) + (1|Class)`



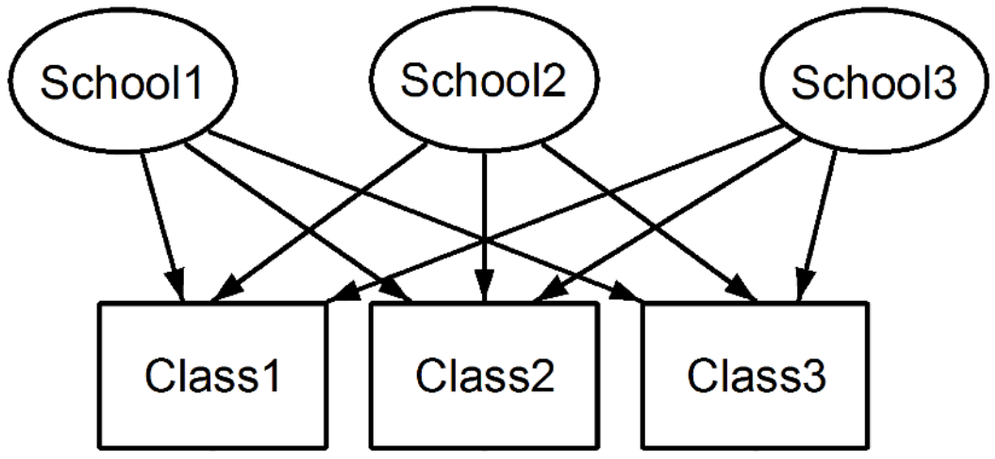
each class appears in each of the schools

```
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()
```

```
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|School) + (1|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
```

```
# model coefficients
fit.crossed %>% coef()
```

`lmer()` syntax summary

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

What shall I include as random effects?

- mixed opinions on the topic
- go maximal!

Journal of Memory and Language 68 (2013) 255–278

Contents lists available at SciVerse ScienceDirect

Journal of Memory and Language

journal homepage: www.elsevier.com/locate/jml

Random effects structure for confirmatory hypothesis testing: Keep it maximal

Dale J. Barr [a,*], Roger Levy [b], Christoph Scheepers [a], Harry J. Tily [c]

[a] Institute of Neuroscience and Psychology, University of Glasgow, 58 Hillhead St., Glasgow G12 8QB, United Kingdom
[b] Department of Linguistics, University of California at San Diego, La Jolla, CA 92093-0108, USA
[c] Department of Brain and Cognitive Sciences, Massachussetts Institute of Technology, Cambridge, MA 02139, USA

"Through theoretical arguments and Monte Carlo simulation, we show that LMEMs generalize best when they include the maximal random effects structure justified by the design. ...

Maximal LMEMs should be the 'gold standard' for confirmatory hypothesis testing in psycholinguistics and beyond."

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3), 255--278.

What shall I include as random effects?

- Failure to include maximal random-effect structures in LMEMs (when such random effects are present in the underlying populations) **inflates Type I error rates**.
- For designs including within-subjects (or within-items) manipulations, random-intercepts-only LMEMs **can have catastrophically high Type I error rates**, regardless of how p-values are computed from them.
- The performance of a data-driven approach to determining random effects (i.e., model selection) depends strongly on the specific algorithm, size of the sample, and criteria used; **moreover, the power advantage of this approach over maximal models is typically negligible**.
- In terms of power, maximal models perform surprisingly well even in a "worst case" scenario where they assume random slope variation that is actually not present in the population.

Barr, D. J., Levy, R., Scheepers, C., & Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3), 255--278.

What shall I include as random effects?

- general advice:
  - start maximal (as supported by the design)
    - random intercepts for different participants
    - random slopes when participants are tested multiple times
    - random intercepts for items
  - reduce complexity of the random effects structure step by step
    - remove the correlation between random effects first

Plan for today

- A worked example
  - pooling:
    - complete pooling
    - no pooling
    - partial pooling
  - shrinkage
- Let's stimulate some `lmer()`s
  - outliers
  - singular model fit
  - heterogeneity in variance
  - Simpson's paradox
- Bootstrapping linear mixed effects models
- Getting p-values
- Understanding `lmer()` syntax
- **Pitfalls in fitting `lmer()`s (and what to do about it)**

Pitfalls in fitting `lmer()`s

My model does not converge ...

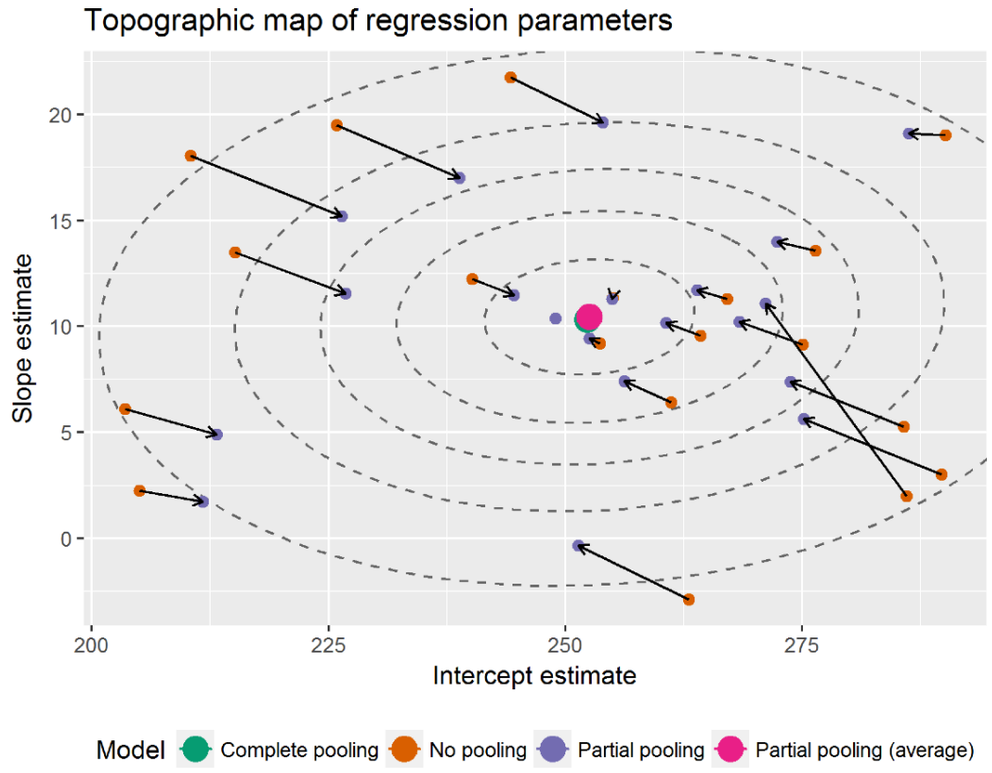
- **`lmer`**`()`s are solved through a (complicated) process of iterative optimization
- only interpret the results of models that actually converged!
- here are some tricks that might help:
  - *continuous predictors*: center and scale
  - *categorical predictors*: choose a factor that has more data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

Remove the correlation component from your model

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

random slopes

random intercepts

```
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 -0.184
```

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()
```

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

```
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 -0.184
```

independent Gaussians

My model does not converge ...

```r
# fit the model
fit.lmer = lmer(formula = reaction ~ 1 + days + (1 + days | subject),
                data = df.sleep)

# explore different optimization algorithms
fit.all = allFit(fit.lmer)

# summarize result
fit.all %>% summary()
```

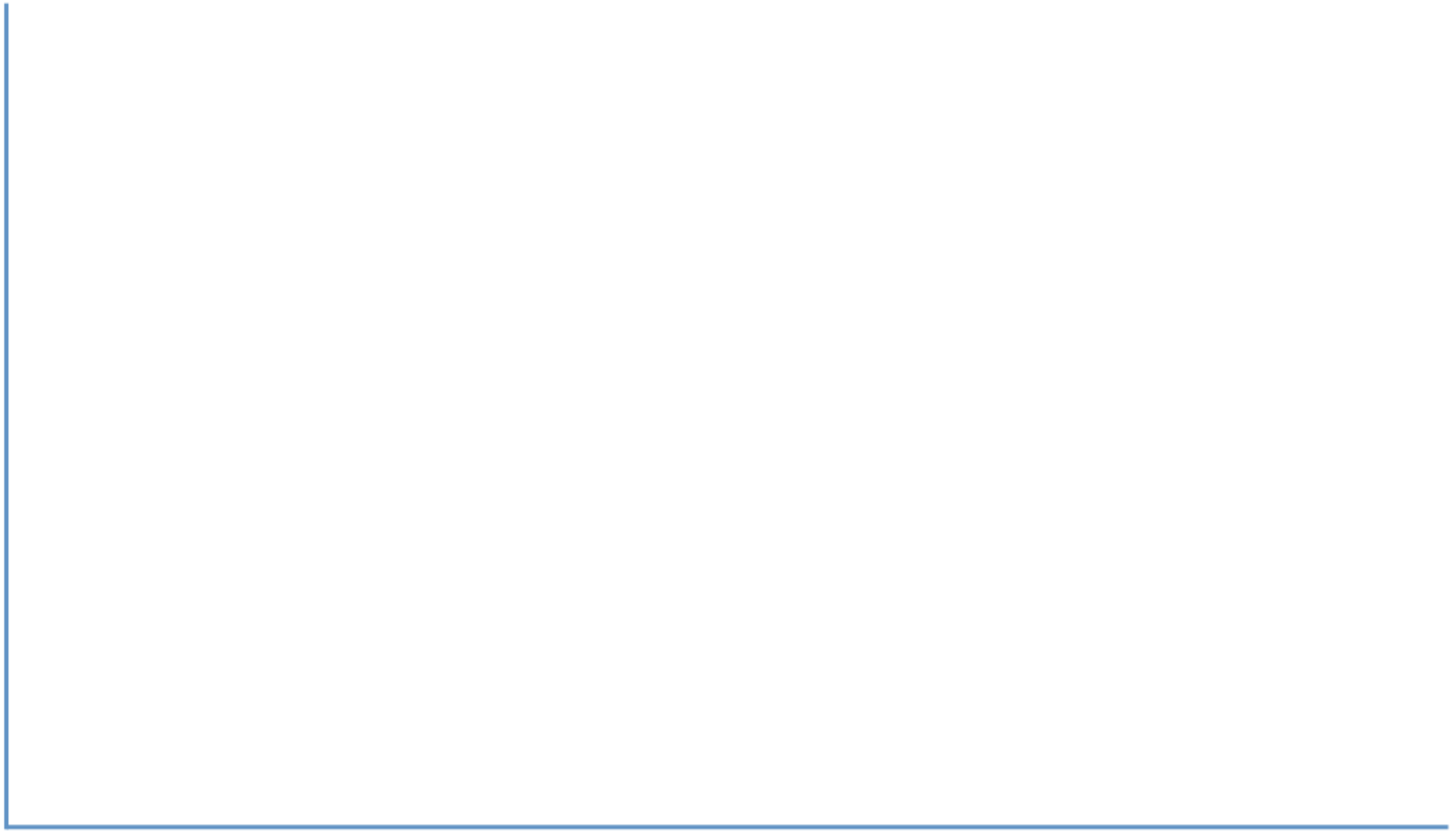
comparison of the different optimization algorithms

```
$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
                       -885.7239                      -885.7239

$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

Feedback

How was the pace of today's class?



How happy were you with today's class overall?

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

Thank you!