LONGITUDINAL DATA ANALYSIS

Psych 252
March 13, 2020
Andrew Nam

AGENDA

- Practice Effect
- Cross-Validation: Part II
  - Avoiding data leakage
  - Logistic regression
- Segmented Regression

AGENDA

- **Practice Effect**
- Cross-Validation: Part II
  - Avoiding data leakage
  - Logistic regression
- Segmented Regression

PRACTICE EFFECT

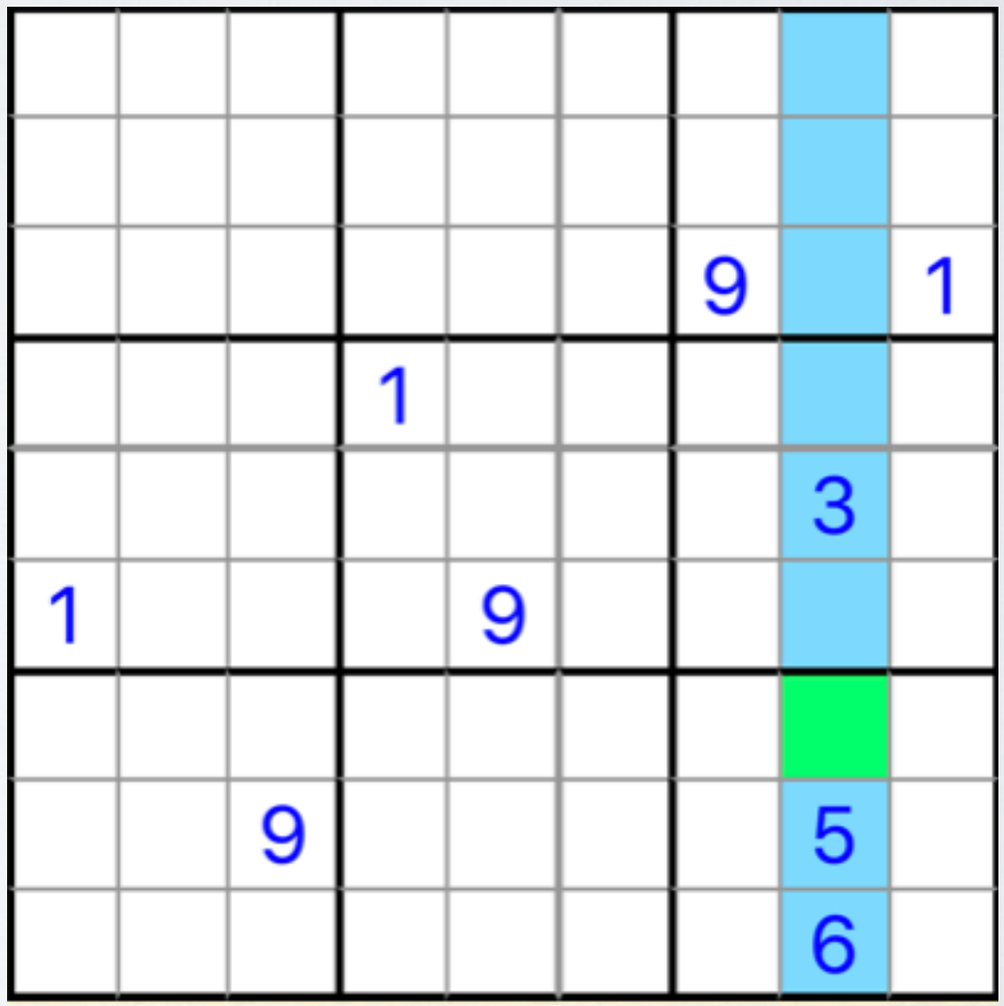
practice effect

any change or improvement that results from practice or repetition of task items or activities. The practice effect is of particular concern in experimentation involving within-subjects designs, as participants' performance on the variable of interest may improve simply from repeating the activity rather than from any study manipulation imposed by the researcher.

APA Dictionary of Psychology

PRACTICE EFFECT: CASE STUDY

- Puzzle solving task
- 71 participants, 64 puzzles each

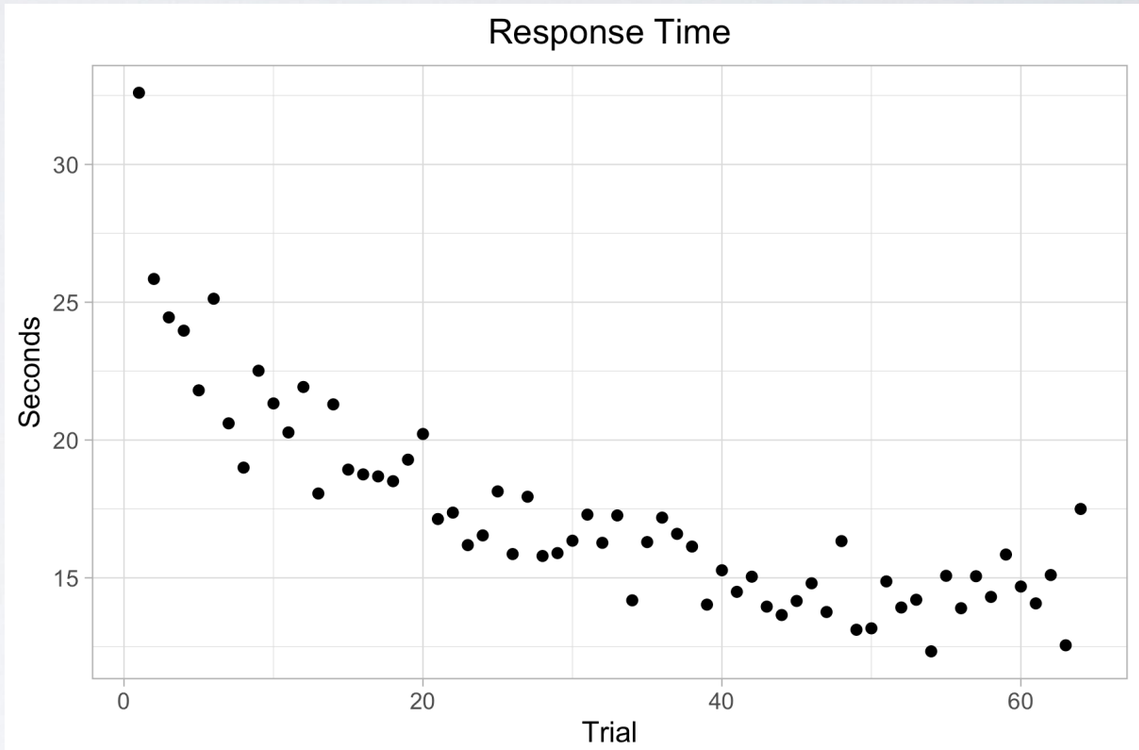


PRACTICE EFFECT: CASE STUDY

	Accuracy	RT
Overall	93.9%	18.2s
First 8	85.2%	27.6s
Last 8	94.9%	15.4s

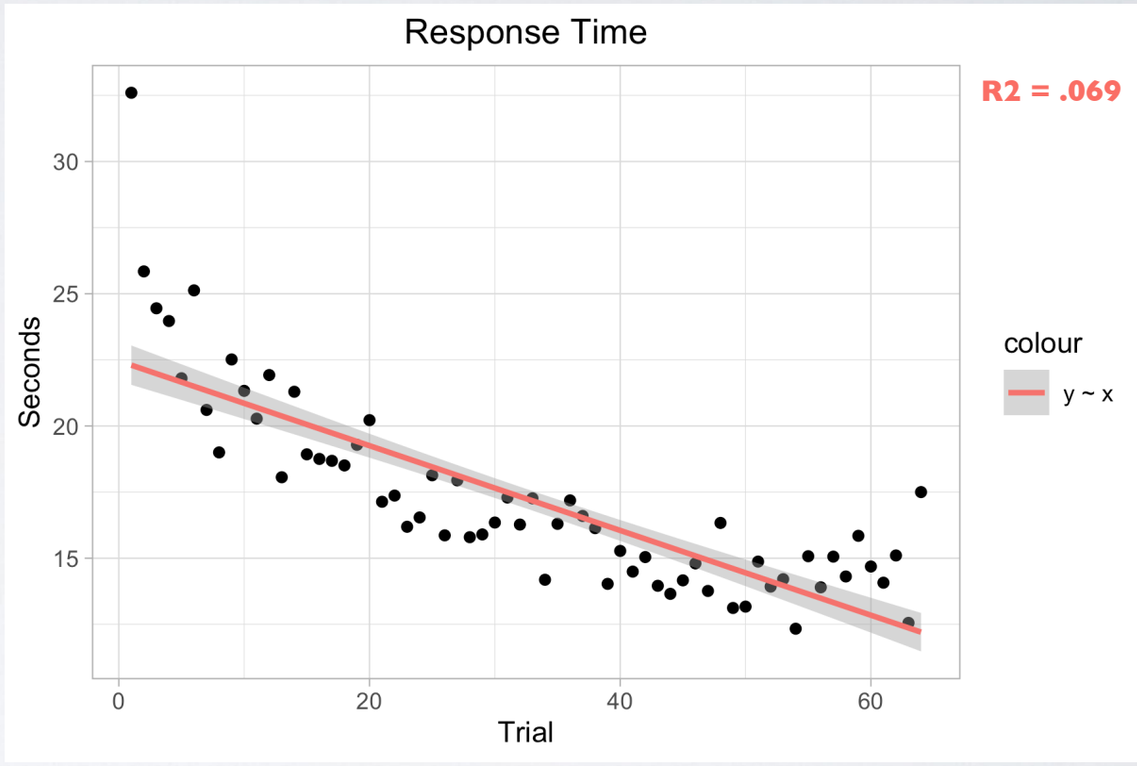
PRACTICE EFFECT: CASE STUDY

	Accuracy	RT
Overall	93.9%	18.2s
First 8	85.2%	27.6s
Last 8	94.9%	15.4s



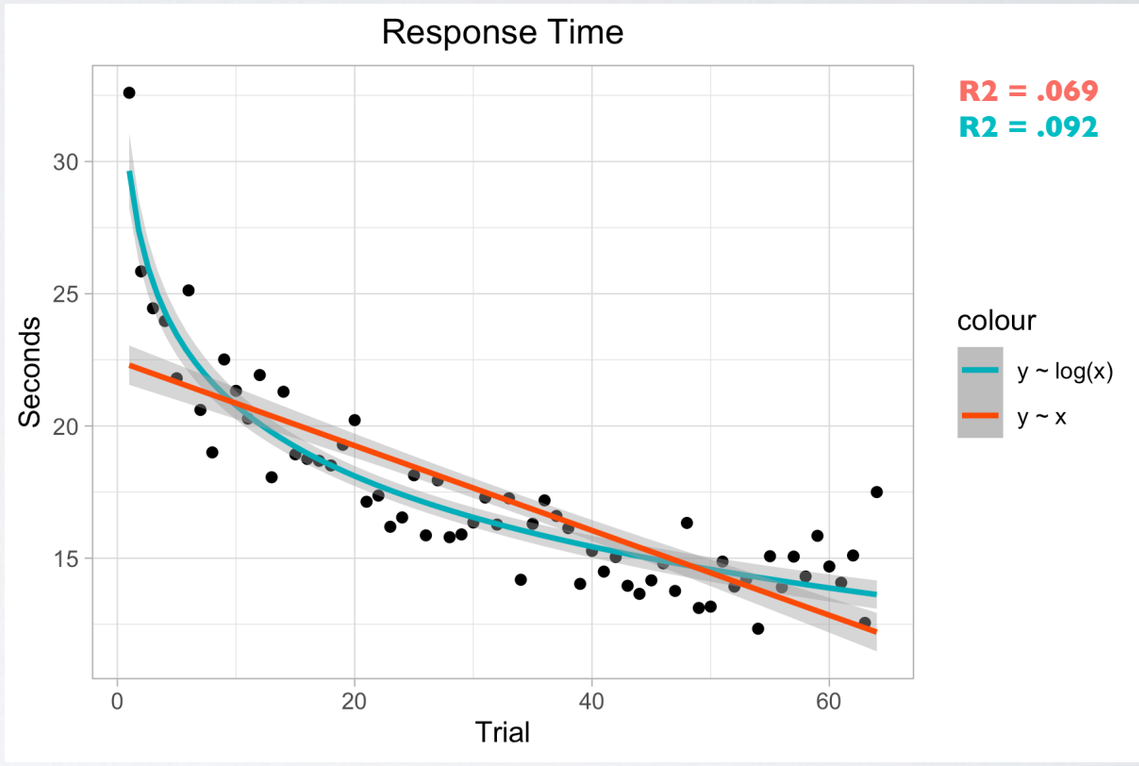
PRACTICE EFFECT: CASE STUDY

	Accuracy	RT
Overall	93.9%	18.2s
First 8	85.2%	27.6s
Last 8	94.9%	15.4s

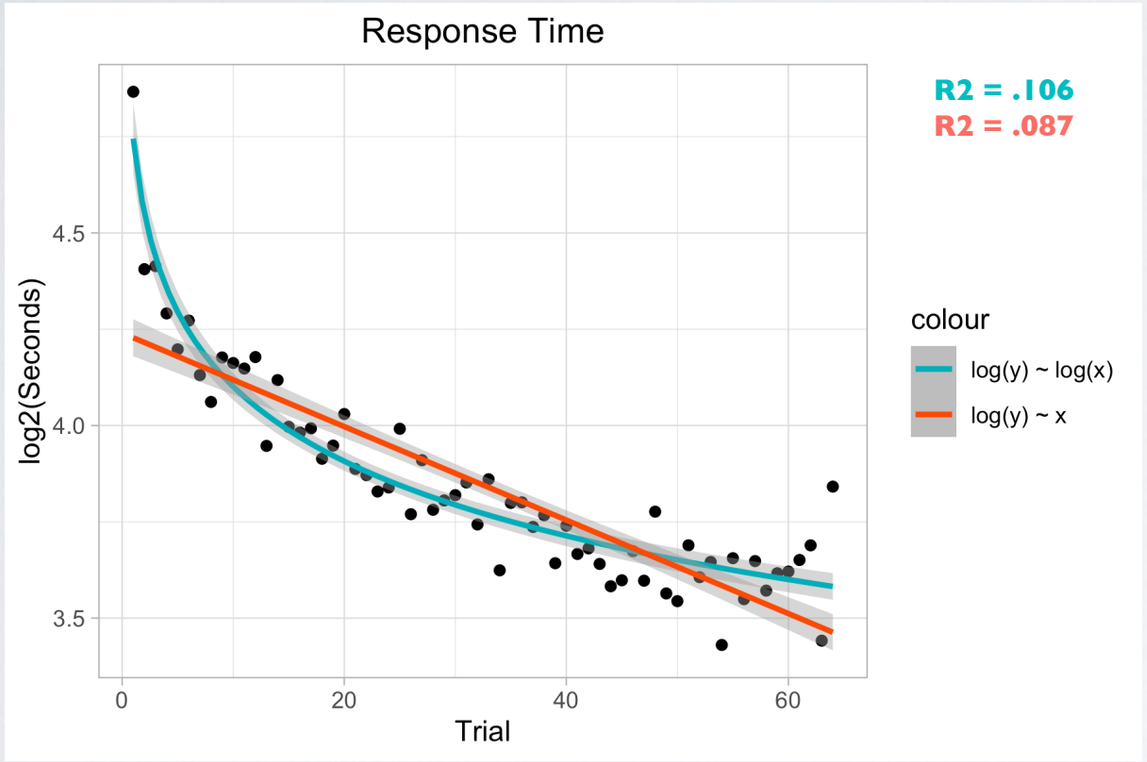


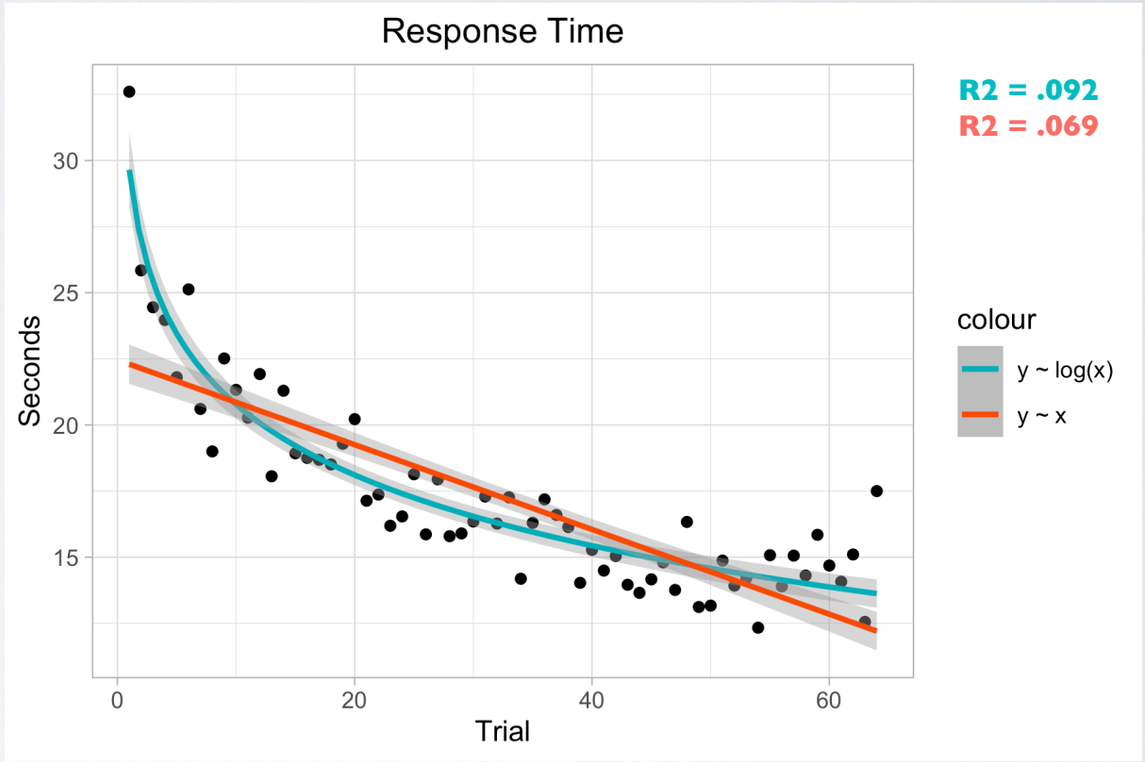
PRACTICE EFFECT: CASE STUDY

	Accuracy	RT
Overall	93.9%	18.2s
First 8	85.2%	27.6s
Last 8	94.9%	15.4s



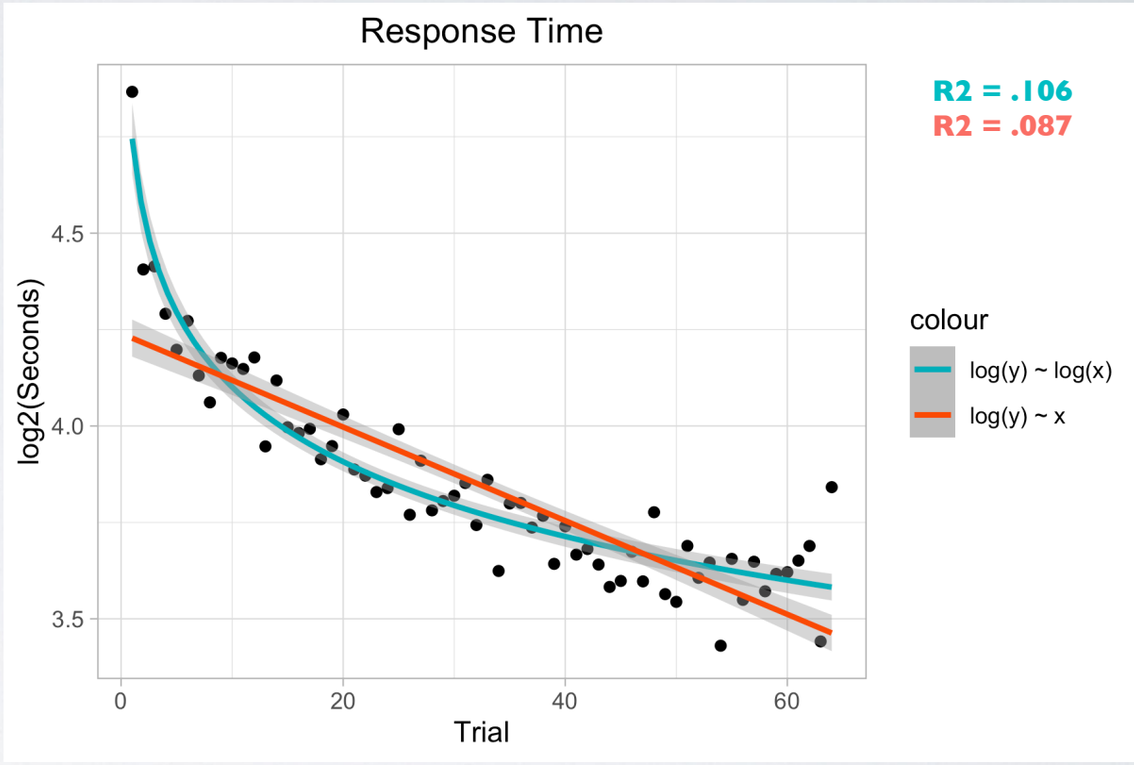
PRACTICE EFFECT: COMPARISON





PRACTICE EFFECT: VERDICT

- Power law: log(RT) ~ log(trial)
- Exponential law: log(RT) ~ trial
- I personally find power law to fit better more often
- There are arguments for preferring exponential law [1]

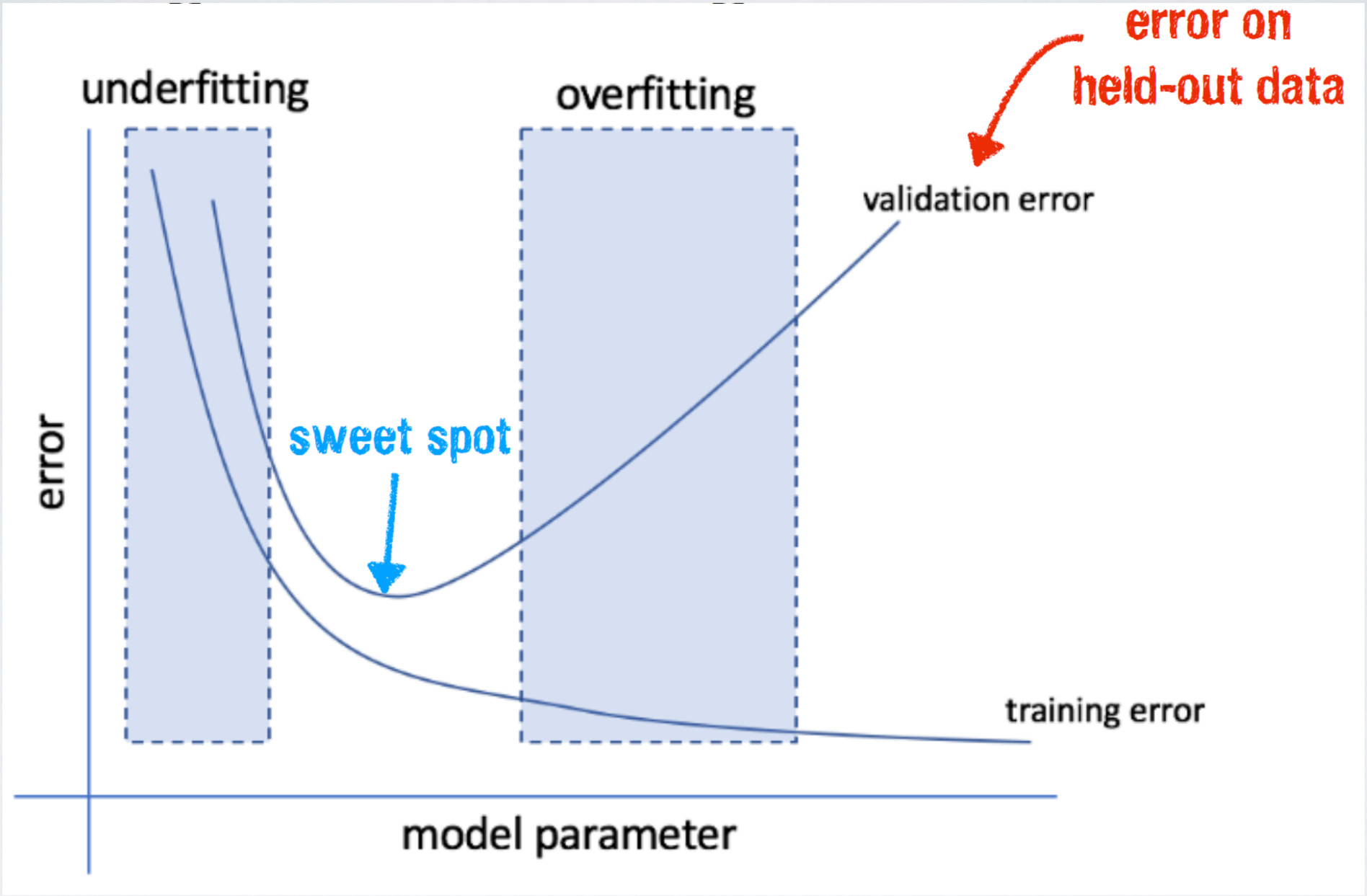


[1] Heathcote et al. (2000)

AGENDA

- Practice Effect
- **Cross-Validation: Part II**
  - **Avoiding data leakage**
  - Logistic regression
- Segmented Regression

CROSS-VALIDATION: RECALL



CROSS-VALIDATION: RECALL

```
df.cv = df.cities %>%
  crossv_mc(n = 25, test = .2) %>%
  mutate(model_med_age = map(train, ~ lm(mean_income ~ median_age, data = .)),
         model_med_age10 = map(train, ~ lm(mean_income ~ poly(median_age, 10), data = .)),
         model_edu = map(train, ~ lm(mean_income ~ less9thgrade + grade9to12 +
                                       highschool + somecollege +
                                       assoc + bachelors + grad,
                                     data = .)),
         model_race = map(train, ~ lm(mean_income ~ percent_white + percent_black +
                                        percent_amindian_alaskan + percent_asian +
                                        percent_nativeandother +
                                        percent_other_nativeandother +
                                        percent_hispanicorlatino + percent_race_other,
                                      data = .))) %>%
pivot_longer(cols = contains("model"),
             names_to = "model_name",
             values_to = "fit") %>%
mutate(training_rsquare = map2_dbl(.x = fit, .y = train, ~ rsquare(.x, .y)),
       training_rmse = map2_dbl(.x = fit, .y = train, ~ rmse(.x, .y)),
       validation_rsquare = map2_dbl(.x = fit, .y = test, ~ rsquare(.x, .y)),
       validation_rmse = map2_dbl(.x = fit, .y = test, ~ rmse(.x, .y))) %>%
select(model_name, training_rsquare,
       training_rmse, validation_rsquare, validation_rmse) %>%
group_by(model_name) %>%
summarize(training_rsquare = mean(training_rsquare),
          training_rmse = mean(training_rmse),
          validation_rsquare = mean(validation_rsquare),
          validation_rmse = mean(validation_rmse))
```

Homework 5: Model Comparison

CROSS-VALIDATION: RECALL

model_name	training_rsquare	training_rmse	validation_rsquare	validation_rmse
model_edu	0.71	11355	0.68	11425
model_med_age	0.03	20705	0.01	20195
model_med_age10	0.07	20256	0.01	20223
model_race	0.35	16966	0.31	16792

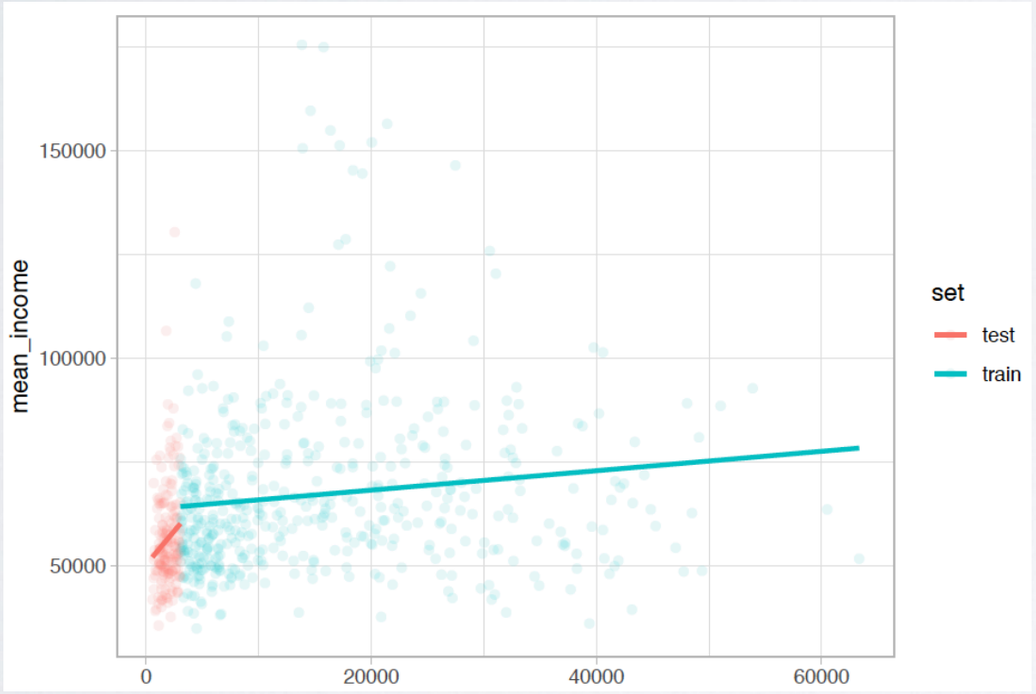
Homework 5: Model Comparison

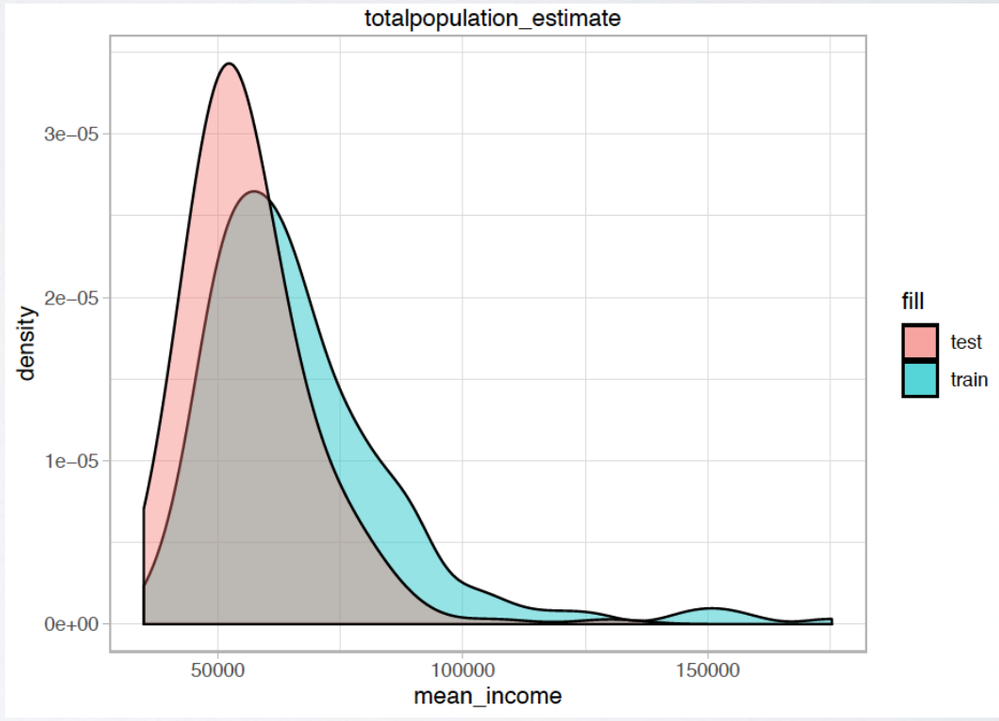
CROSS-VALIDATION: RECALL

model_name	test_rsquare	test_rmse	training_rsquare	training_rmse	validation_rsquare	validation_rmse
model_med_age	-0.15	18922	0.03	20705	0.01	20195
model_med_age10	-147.00	150285	0.07	20256	0.01	20223
model_edu	0.47	8974	0.71	11355	0.68	11425
model_race	-0.54	16355	0.35	16966	0.31	16792

CROSS-VALIDATION: RECALL

model_name	test_rsquare	test_rmse	training_rsquare	training_rmse	validation_rsquare	validation_rmse
model_med_age	-0.15	18922	0.03	20705	0.01	20195
model_med_age10	-147.00	150285	0.07	20256	0.01	20223
model_edu	0.47	8974	0.71	11355	0.68	11425
model_race	-0.54	16355	0.35	16966	0.31	16792





CROSS-VALIDATION: RECALL

model_name	test_rsquare	test_rmse	training_rsquare	training_rmse	validation_rsquare	validation_rmse
model_med_age	-0.15	18922	0.03	20705	0.01	20195
model_med_age10	-147.00	150285	0.07	20256	0.01	20223
model_edu	0.47	8974	0.71	11355	0.68	11425
model_race	-0.54	16355	0.35	16966	0.31	16792

Test

Train/Valid

	Carmel	La Palma	EPA	Gilroy	PA	SMateo	Berk	Oak	SF	Chic	LA	NYC
Wage												
Pop	4k	16k	30k	58k	67k	105k	122k	425k	884k	2.7M	4M	8.6M
Age												
Race												
Edu												

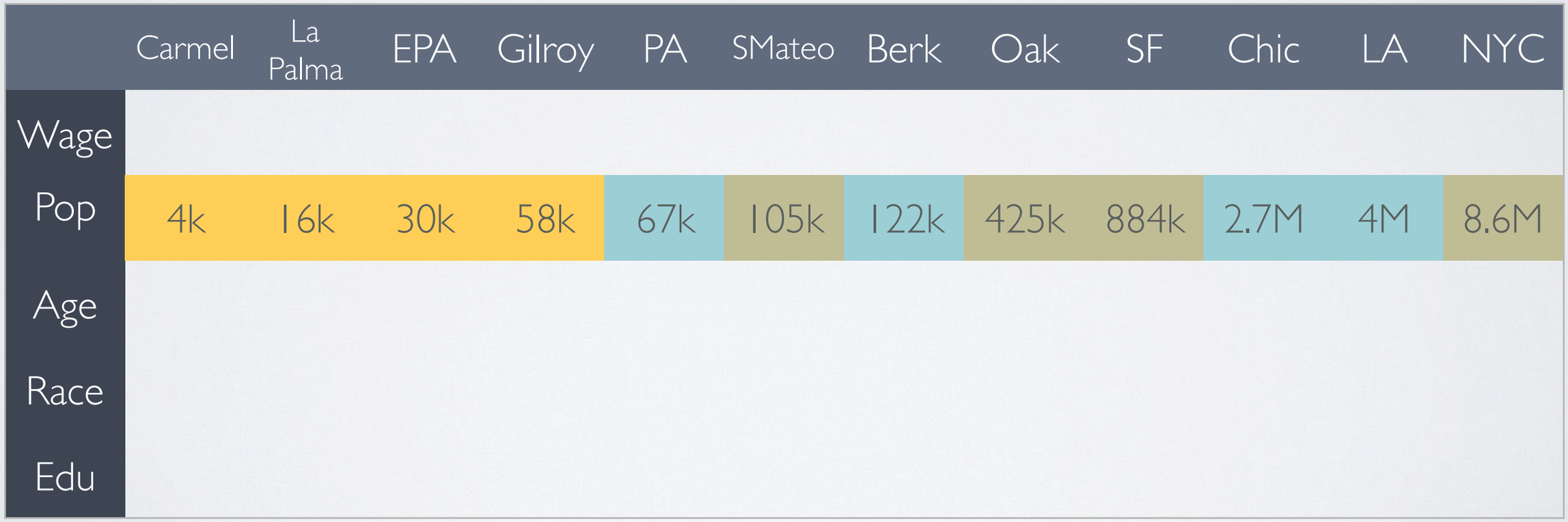
Homework 5: Model Comparison

CROSS-VALIDATION: RECALL

model_name	test_rsquare	test_rmse	training_rsquare	training_rmse	validation_rsquare	validation_rmse
model_med_age	-0.15	18922	0.03	20705	0.01	20195
model_med_age10	-147.00	150285	0.07	20256	0.01	20223
model_edu	0.47	8974	0.71	11355	0.68	11425
model_race	-0.54	16355	0.35	16966	0.31	16792

Test

Train/Valid



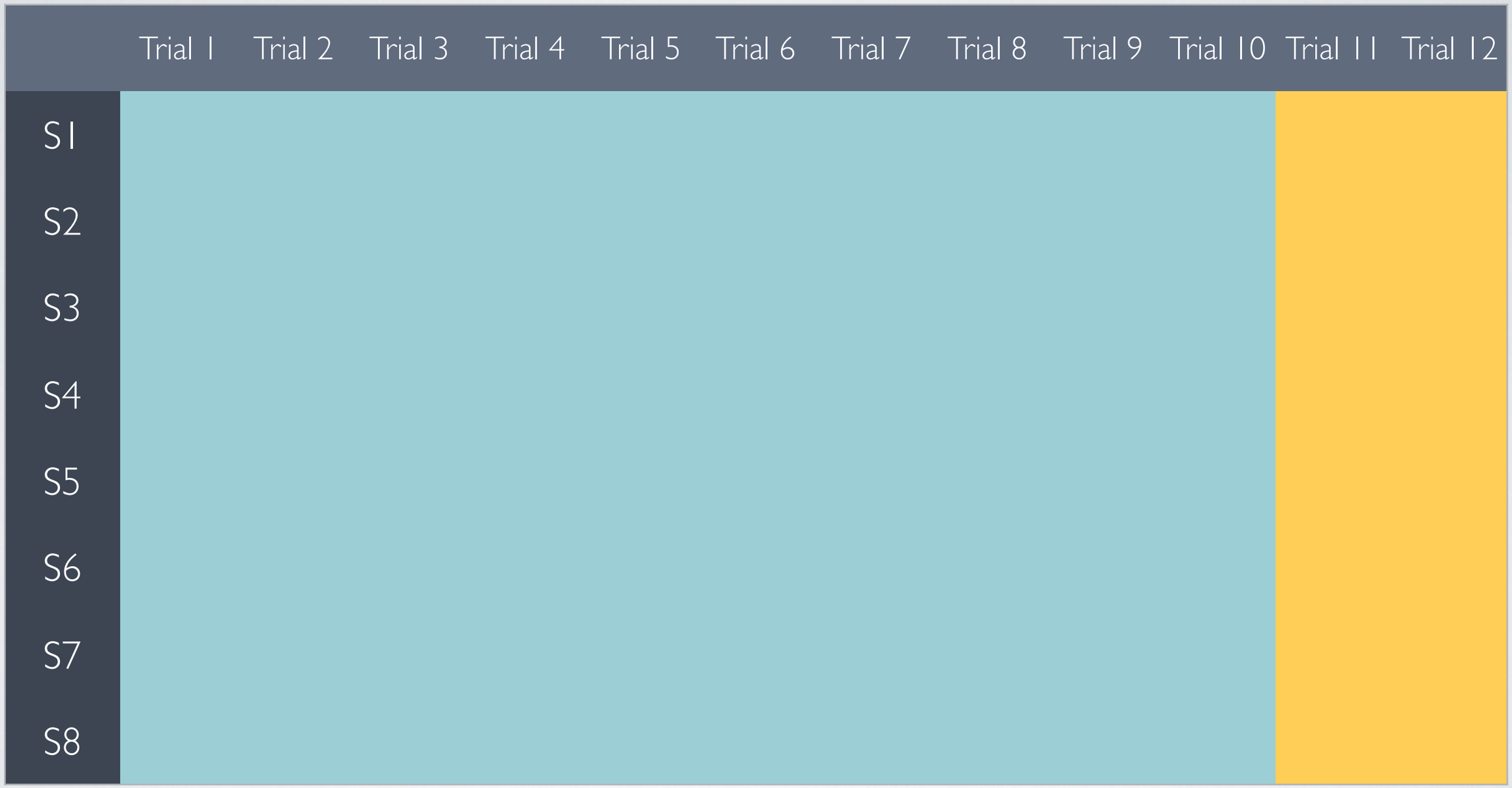
Homework 5: Model Comparison

CROSSV_MC/KFOLD

	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Trial 6	Trial 7	Trial 8	Trial 9	Trial 10	Trial 11	Trial 12
S1												
S2												
S3												
S4												
S5												
S6												
S7												
S8												

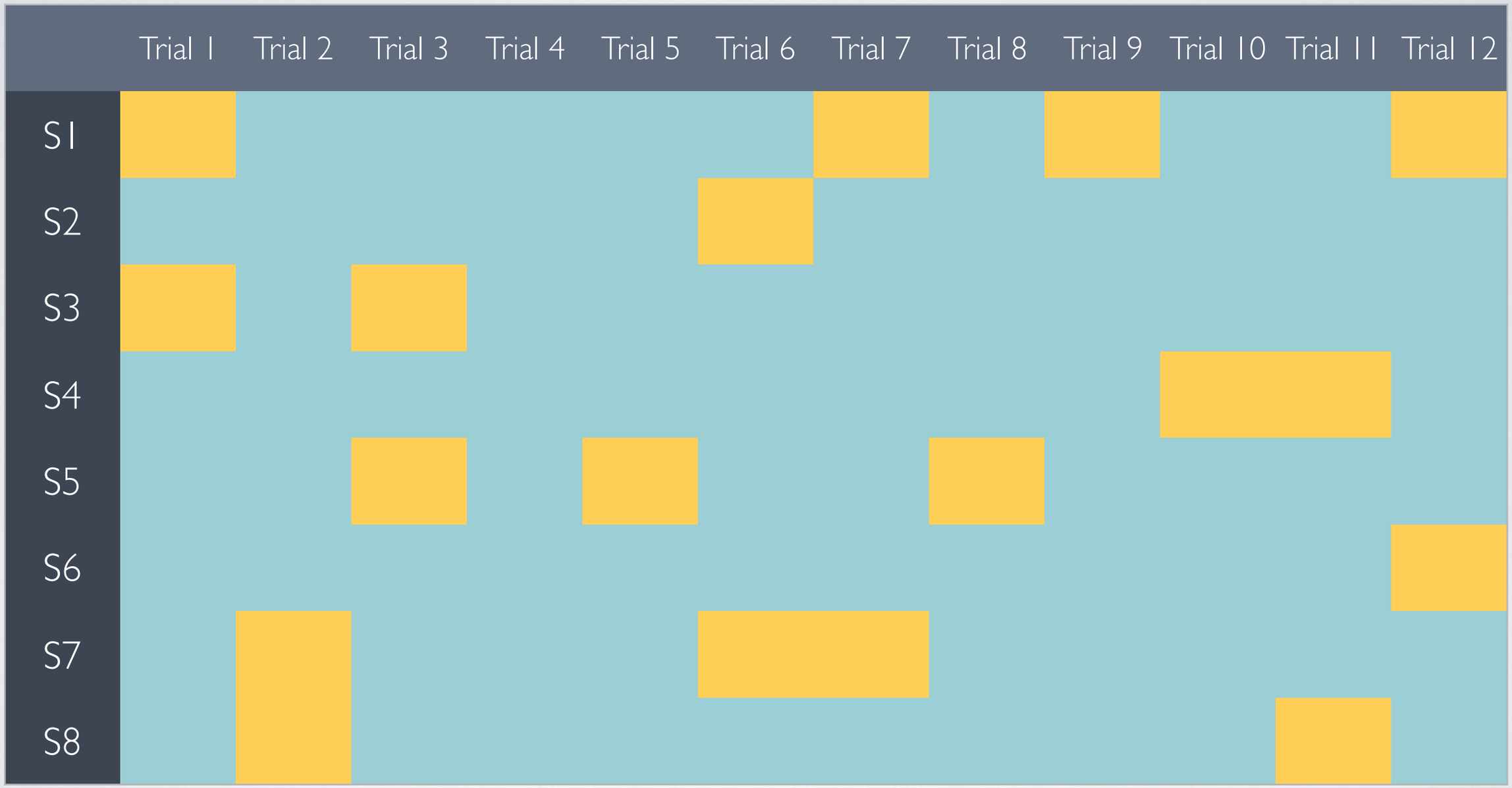
CROSSV_MC/KFOLD

CROSSV_MC/KFOLD



*This wouldn't be CV, but you can bootstrap

CROSSV_MC/KFOLD



CROSSV_MC/KFOLD

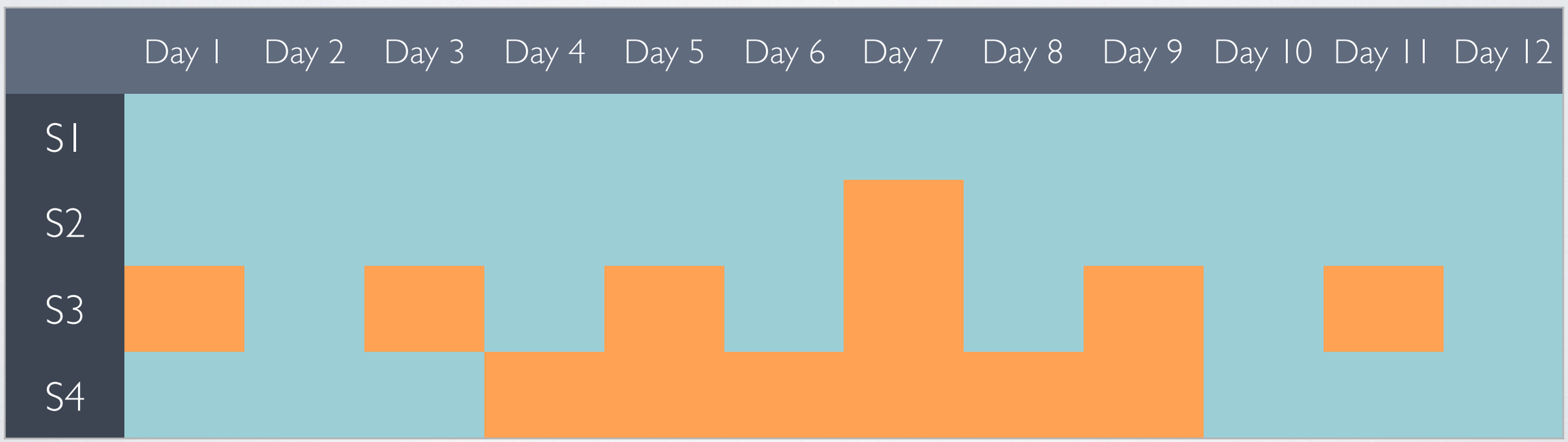
- Cross-validation function assumes samples are IID
- Data leakage: training set contaminated with test set
- Solution: randomly assign subjects instead of individual subject-trials

```
df %>%
  crossv_mc(100, 2)
```

```
df %>%
  group_by(subject) %>%
  nest() %>%
  ungroup() %>%
  crossv_mc(100, .2) %>%
  mutate(train = map(train, ~ as.data.frame(.) %>%
                             unnest(c("data"))),
         test = map(test, ~ as.data.frame(.) %>%
                            unnest(c("data"))))
```

CV WITH RANDOM EFFECTS

- If comparing fits of different functional forms, fixed-effects should suffice
- But sometimes we want to check our model fit including random effects
  - e.g. different number of samples per participant



CV WITH RANDOM EFFECTS

- If comparing fits of different functional forms, fixed-effects should suffice
- But sometimes we want to check our model fit including random effects
  - e.g. different number of samples per participant

CV WITH RANDOM EFFECTS

- If comparing fits of different functional forms, fixed-effects should suffice
- But sometimes we want to check our model fit including random effects
  - e.g. different number of samples per participant
- Problem: test-set subjects don't have assigned z-scores for RE

CV WITH RANDOM EFFECTS

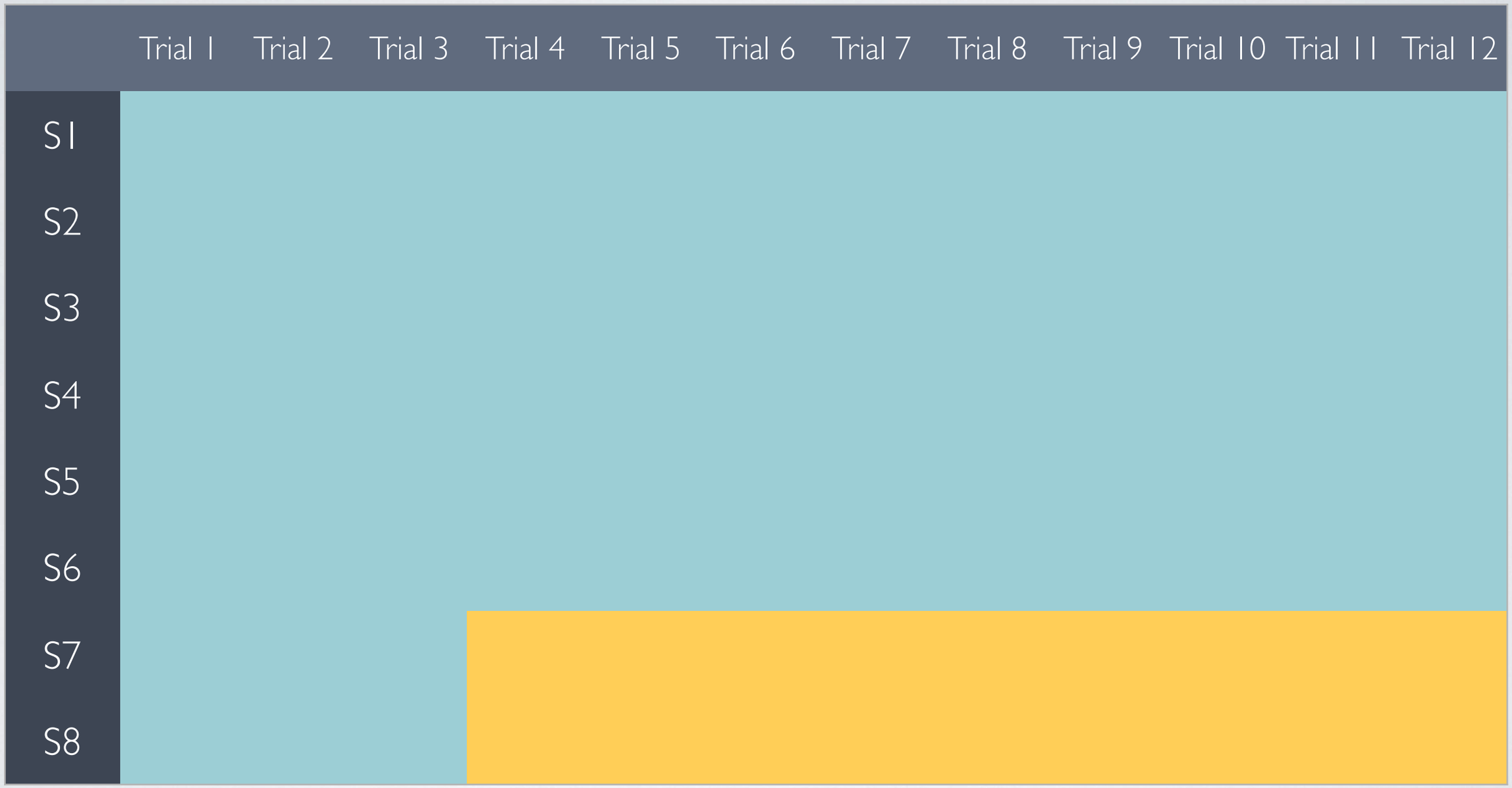
- If comparing fits of different functional forms, fixed-effects should suffice
- But sometimes we want to check our model fit including random effects
  - e.g. different number of samples per participant
- Problem: test-set subjects don't have assigned z-scores for RE

```
Error in levelfun(r, n, allow.new.levels =
 allow.new.levels) : new levels detected in newdata
```

CV WITH RANDOM EFFECTS

- If comparing fits of different functional forms, fixed-effects should suffice
- But sometimes we want to check our model fit including random effects
  - e.g. different number of samples per participant
- Problem: test-set subjects don't have assigned z-scores for RE
  - Can use population means (fit using RE but ignore RE)
  - Can train with subsample of test-set

CV WITH RANDOM EFFECTS



FIXED ONLY

```r
vector_rmse = function(y, y_hat) {
  mse = mean((y - y_hat)^2)
  return (sqrt(mse))
}

vector_rsquare = function(y, y_hat) {
  rss = sum((y - y_hat)^2)
  tss = sum((y - mean(y))^2)
  return (1 - rss/tss)
}
```

```r
mixed_effects_loss = function(model, data, loss.f, y_var, fixed_only) {
  target_col <- deparse(substitute(y_var))

  if (fixed_only) {
    y_hat = predict(model,
                    newdata = data,
                    allow.new.levels = T,
                    re.form = NA)
  } else {
    y_hat = predict(model, newdata = data)
  }

  loss = data %>%
    mutate(y_hat = y_hat,
           loss = loss.f(.[[target_col]], y_hat)) %>%
    summarize(loss = mean(loss)) %>%
    pull(loss)
  return (loss)
}
```

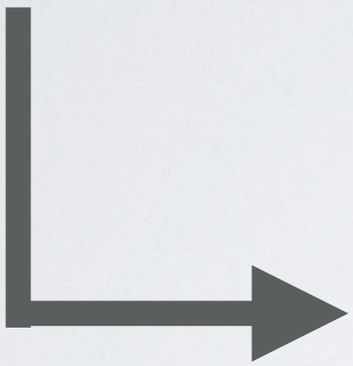
```r
mutate(tr_r2 = map2_dbl(.x = fit,
                        .y = train,
                        .f = ~ mixed_effects_loss(.x, .y, vector_rsquare, lduration, F)),
       tr_rmse = map2_dbl(.x = fit,
                          .y = train,
                          .f = ~ mixed_effects_loss(.x, .y, vector_rmse, lduration, F)),
       v_r2 = map2_dbl(.x = fit,
                       .y = test,
                       .f = ~ mixed_effects_loss(.x, .y, vector_rsquare, lduration, T)),
       v_rmse = map2_dbl(.x = fit,
                         .y = test,
                         .f = ~ mixed_effects_loss(.x, .y, vector_rmse, lduration, T))) %>%
```

CV WITH RANDOM EFFECTS

model_log_log	Train R^2	Train RMSE	Test R^2	Test RMSE
Fixed Only	0.573	0.538	0.010	0.796
Train w/ first 4				
Train w/ first 16				

TRAIN W/ FIRST 4

```
df %>%
  group_by(subject) %>%
  nest() %>%
  ungroup() %>%
  crossv_mc(100, .2) %>%
  mutate(train = map(train, ~ as.data.frame(.) %>%
                       unnest(c("data"))),
         test = map(test, ~ as.data.frame(.) %>%
                      unnest(c("data"))))
```



```
mutate(test = map(test, ~ as.data.frame(.) %>%
                      unnest(c("data"))),
       heads = map(test, ~ group_by(., worker_id) %>%
                   top_n(n = 4, wt = -trial) %>%
                   ungroup()),
       test = map(test, ~ group_by(., worker_id) %>%
                   top_n(n = nrow(.) - 4, wt = trial) %>%
                   ungroup()),
       train = map2(.x = train,
                    .y = heads,
                    ~ as.data.frame(.x) %>%
                     unnest(c("data")) %>%
                     bind_rows(.y)))  %>%
```

CV WITH RANDOM EFFECTS

model_log_log	Train R^2	Train RMSE	Test R^2	Test RMSE
Fixed Only	0.573	0.538	0.010	0.796
Train w/ first 4	0.578	0.538	0.342	0.653
Train w/ first 16				

CV WITH RANDOM EFFECTS

model_log_log	Train R^2	Train RMSE	Test R^2	Test RMSE
Fixed Only	0.573	0.538	0.010	0.796
Train w/ first 4	0.578	0.538	0.342	0.653
Train w/ first 16	0.579	0.538	0.416	0.611

AGENDA

- Practice Effect
- **Cross-Validation: Part II**
  - Avoiding data leakage
  - **Logistic regression**
- Segmented Regression

CV: LOGISTIC REGRESSION

```
mutate(model_lin_lin = map(train, ~glm(correct ~ trial, data = ., family = "binomial")),
       model_lin_log = map(train, ~glm(correct ~ ltrial, data = ., family = "binomial")),
       model_log_lin = map(train, ~glm(correct ~ trial, data = ., family = "binomial")),
       model_log_log = map(train, ~glm(correct ~ ltrial, data = ., family = "binomial"))
) %>%
```

	Train R^2	Train RMSE	Test R^2	Test RMSE
lin_lin	-2.99	1.93	0.225	-3.28
log_lin	-3.03	1.95	0.223	-3.32
lin_log	-2.99	1.93	0.225	-3.28
log_log	-3.03	1.95	0.223	-3.32

CV: LOGISTIC REGRESSION

- Our metrics make no sense
- Default rsquare and rmse functions assume similar outputs
- Our y is {0, 1} but our y-hat is logits (-inf, inf)

$$\pi = P(Y = 1)$$

just a placeholder

$$\ln\left(\frac{\pi}{1-\pi}\right) = V$$

logit transformation

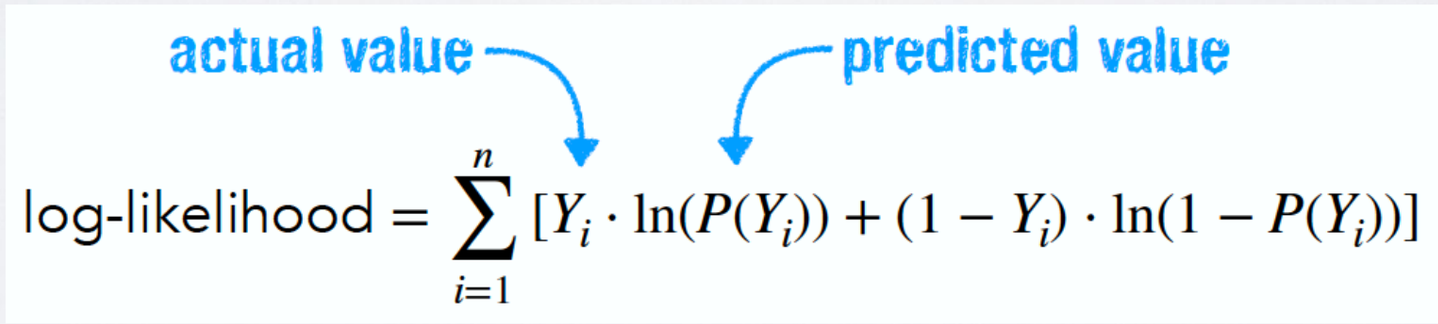
$$\pi = \frac{e^V}{1+e^V}$$

inverse logit

gives us back the probability
(which is much easier to interpret)

CV: LOGISTIC REGRESSION

- Our metrics make no sense
- Default rsquare and rmse functions assume similar outputs
- Our y is {0, 1} but our y-hat is logits (-inf, inf)
- SSE is not the right measure
- Use log-likelihood (cross-entropy)



CV: LOGISTIC REGRESSION

```
sigmoid = function(x) {
  return (1 / (1 + exp(-x)))
}

cross_entropy = function(y, logit) {
  y_hat = sigmoid(logit)
  ce = -(y * log(y_hat) + (1-y) * log(1 - y_hat))
  return (mean(ce))
}
```

```
tr_ce = map2_dbl(.x = fit,
                 .y = train,
                 .f = ~ cross_entropy(.y$correct,
                                      as.vector(predict(.x, .y)))),
v_ce = map2_dbl(.x = fit,
                .y = test,
                .f = ~ cross_entropy(.y$correct,
                                     as.vector(predict(.x, .y))))
```

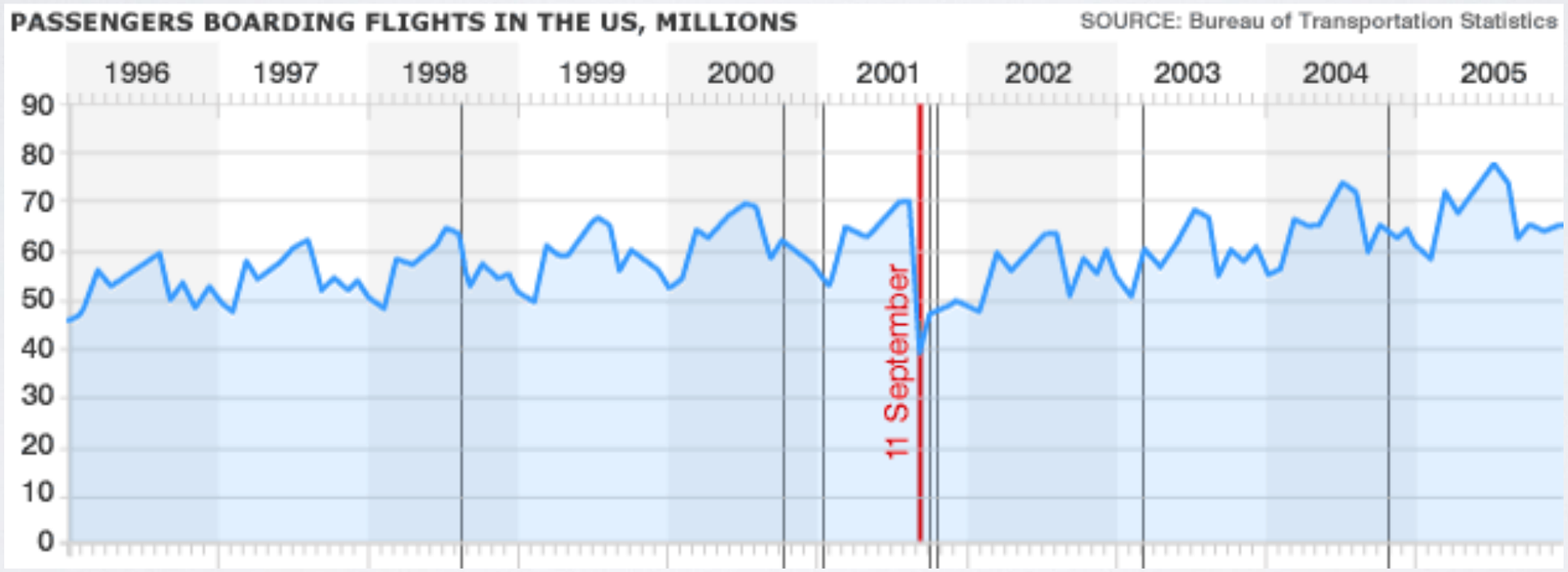
	Tr R^2	Tr RMSE	Tr CE	Te R^2	Te RMSE	Te CE
lin_lin	-2.99	1.93	0.225	0.225	-3.28	0.222
log_lin	-3.03	1.95	0.223	0.223	-3.32	0.219
lin_log	-2.99	1.93	0.225	0.225	-3.28	0.222
log_log	-3.03	1.95	0.223	0.223	-3.32	0.219

AGENDA

- Practice Effect
- Cross-Validation: Part II
  - Avoiding data leakage
  - Logistic regression
- **Segmented Regression**

SEGMENTED REGRESSION

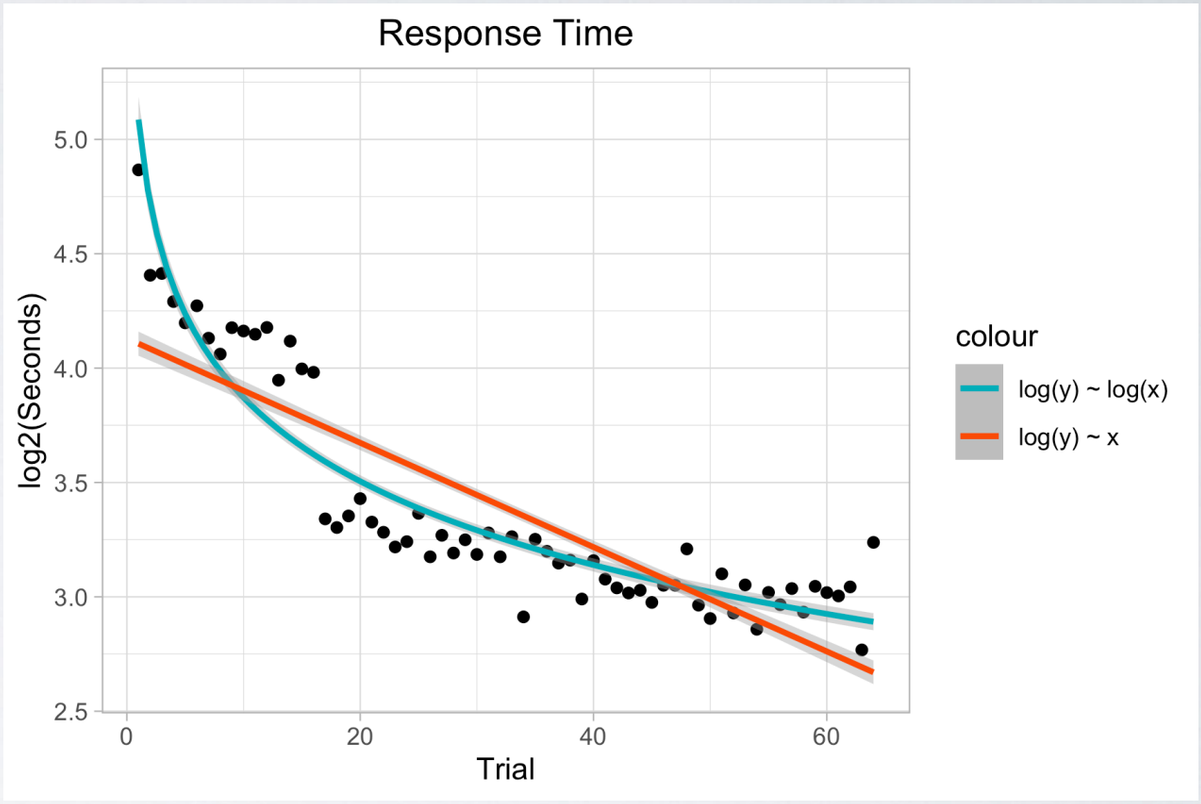
- At some point in the experiment, intervene
- How to distinguish between intervention effect and natural progression?



BBC

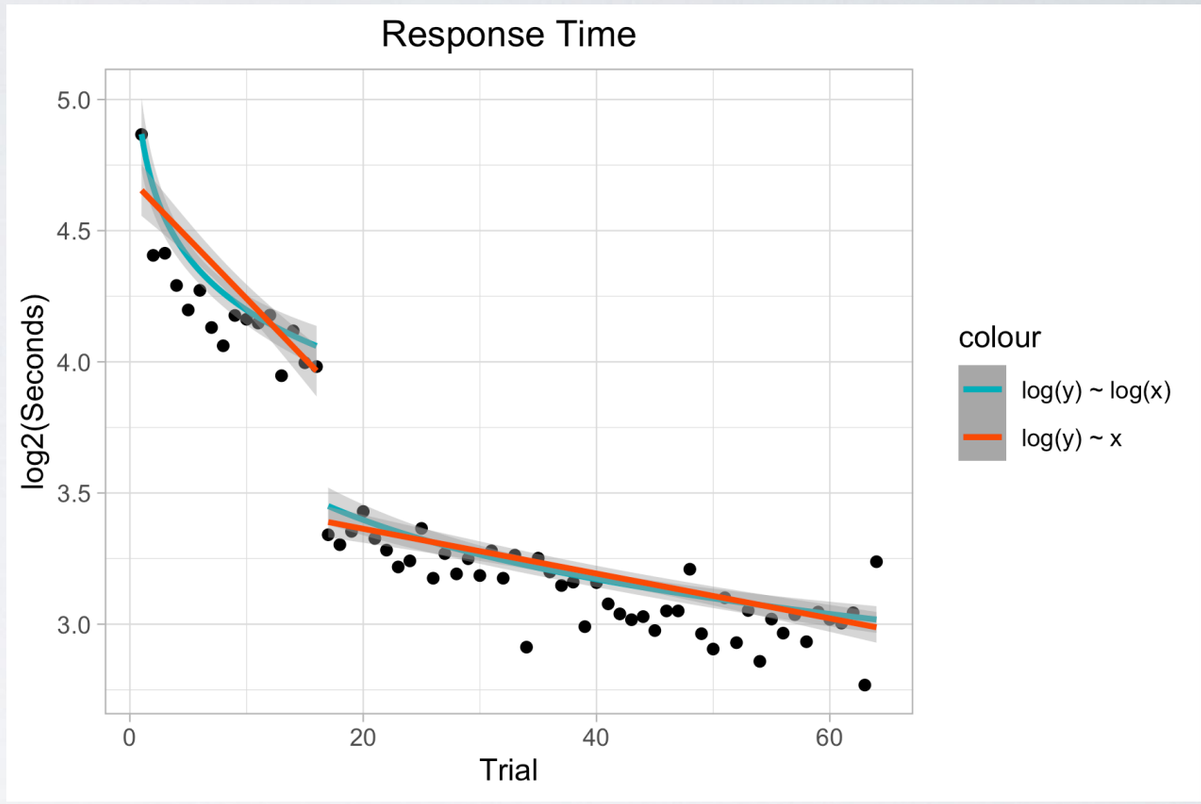
SEGMENTED REGRESSION

- After 16 trials, perform an intervention
- Visible drop, so how to statistically test?



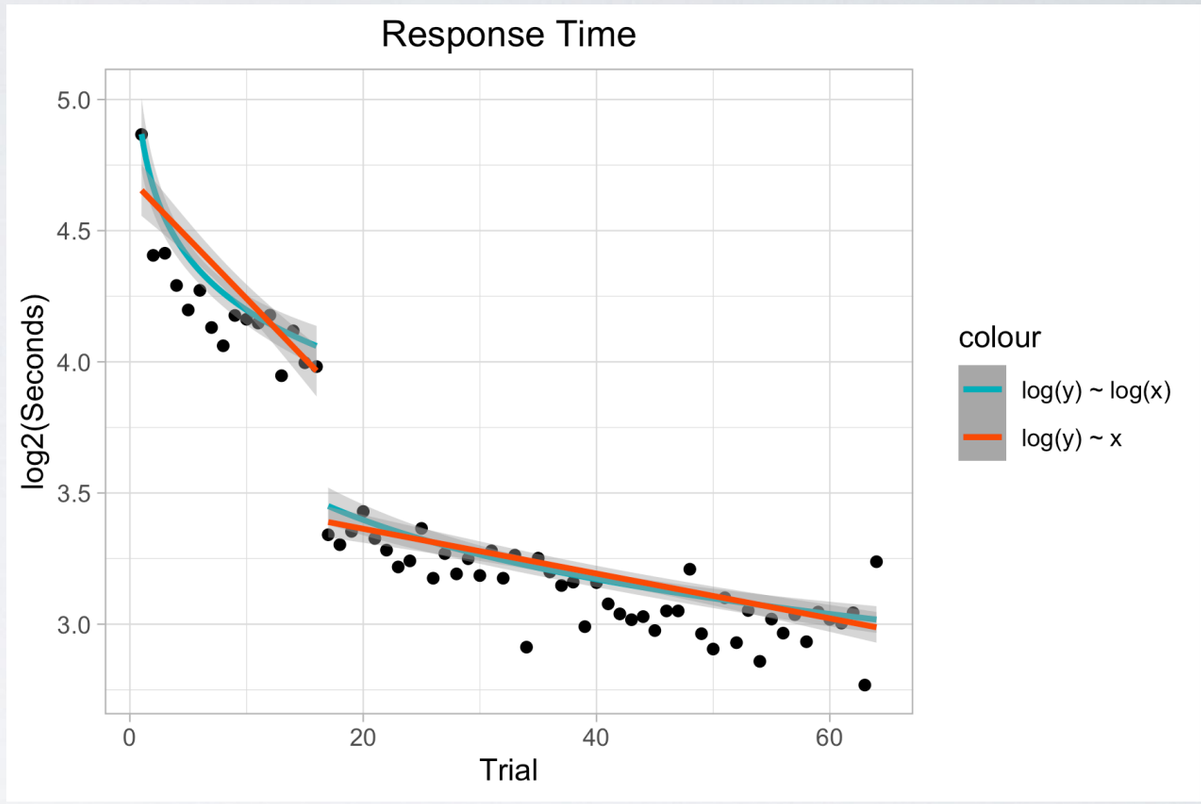
SEGMENTED REGRESSION

- After 16 trials, perform an intervention
- Visible drop, so how to statistically test?
- Fit the regression piece-wise
  - Create dummy variable for post-intervention
  - Cross it with every variable



SEGMENTED REGRESSION

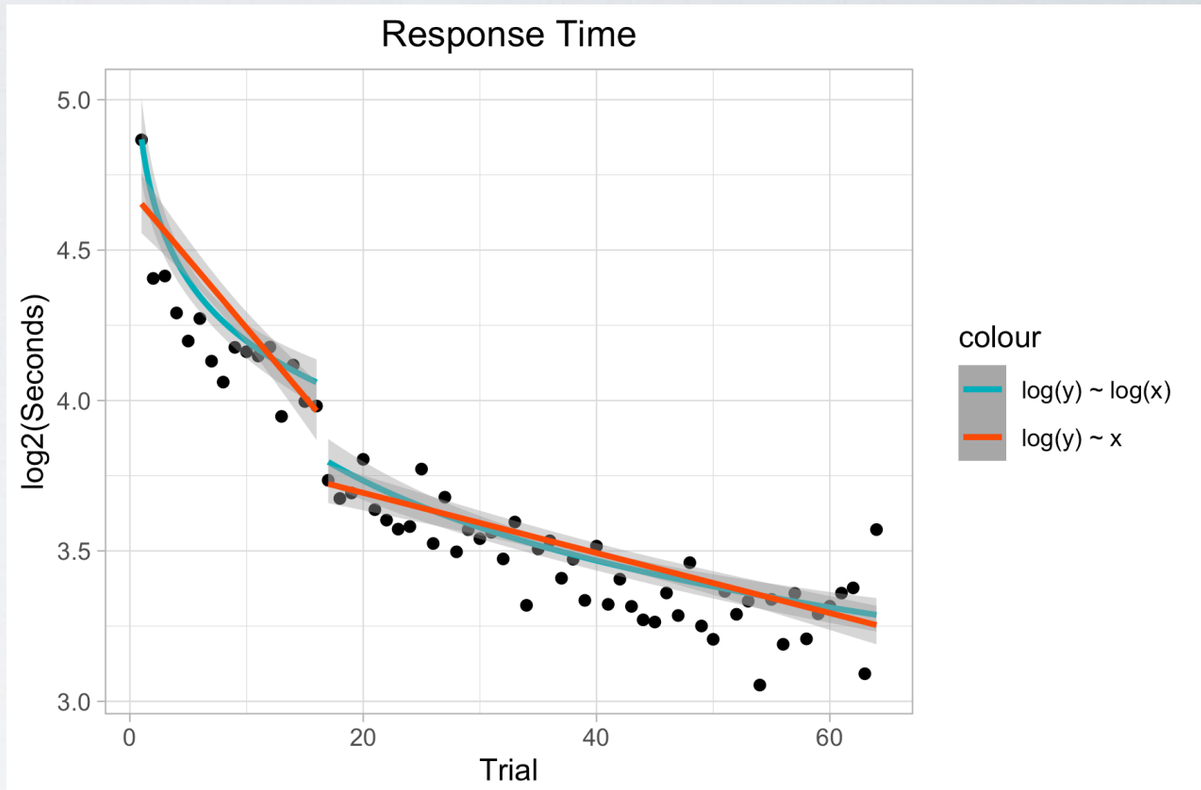
- After 16 trials, perform an intervention
- Visible drop, so how to statistically test?
- Fit the regression piece-wise
  - Create dummy variable for post-intervention
  - Cross it with every variable



	Estimate	Std. Error	df	t value	Pr(>|t|)	
(Intercept)	4.7360	0.0883	68.6387	53.65	< 2e-16	***
ltrial	-0.1798	0.0222	57.4275	-8.10	4.5e-11	***
post_treatTRUE	-0.4028	0.1715	66.6022	-2.35	0.022	*
ltrial:post_treatTRUE	-0.0509	0.0413	64.2207	-1.23	0.222	

SEGMENTED REGRESSION

- Visible does not always mean significant
- Depends on how you choose your time window
  - May wash out in long run



	Estimate	Std. Error	df	t value	Pr(>|t|)	
(Intercept)	4.7344	0.0881	68.7935	53.76	< 2e-16	***
ltrial	-0.1797	0.0220	57.3723	-8.16	3.5e-11	***
post_treatTRUE	0.0624	0.1917	67.0229	0.33	0.746	
ltrial:post_treatTRUE	-0.0786	0.0448	64.9230	-1.75	0.084	.

QUESTIONS?