Linear regression for data analysis

workshop day

PhonLab; 19 April 2024

Basics

Linear models

- Model the change in an observed **dependent variable** (y) as a function of one or more **independent variables** (x)
- Independent variables are also called **predictors.** Dependents are also called **responses.**
- “Linear” refers to the fact that effects of predictors are **summed together**

learned **coefficients**

slope

intercept / bias

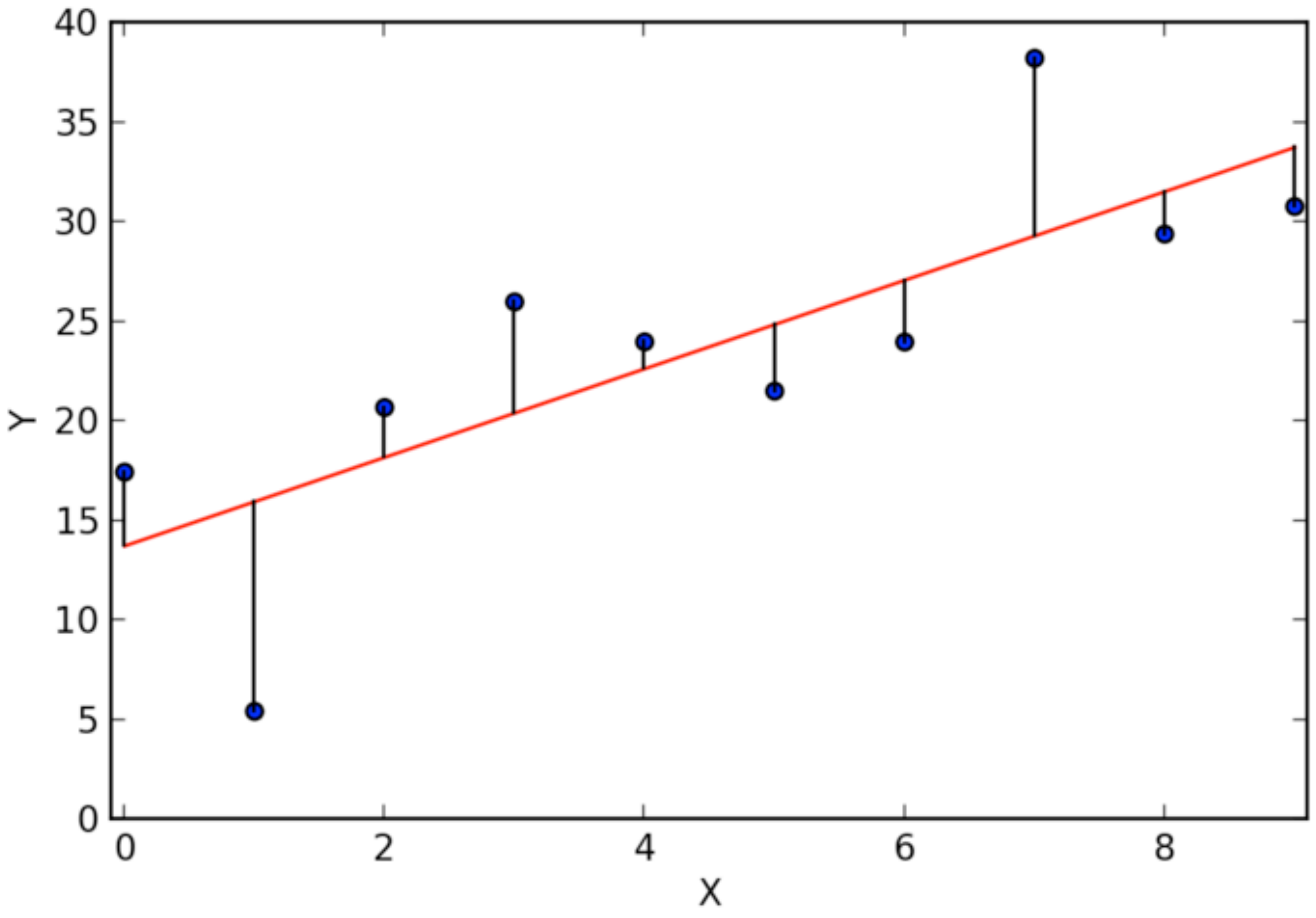
$$y = \beta x + \alpha \qquad (y = mx + b)$$

response

predictor

Error minimization

- Model coefficients are selected to **minimize the error** between the predicted line and observed datapoints
  - This is the **residual error** or **R^2**
  - Yields the **"best fit line"**
  - Sometimes explicitly modeled:
    - $y = \beta x + \alpha + \epsilon$
    - where ϵ is residual error



Multivariable regression

- Formula straightforwardly generalizes to **multiple predictors**
  - $y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots + \epsilon$
  - Matrix notation: $Y = X\beta + \epsilon$
  - Can be **solved in the same way**
- R code
  - `model = lm(y ~ variable)`
  - `model = lm(y ~ variable_1 + variable_2)`

Categorical variables

- A predictor x can be **categorical**, also known as a **factor**
  - e.g. which pond is a fish sampled from out of {pond_1, pond_2}
- Regression software usually converts this to a **binary “dummy” variable**
  - Pond 1 : $x = 0$
  - Pond 2: $x = 1$
- What is the result of this “dummy” encoding?

Categorical variables

with 2 values

- **Pond 1** case
  - $y = \beta * x + \alpha = \beta * \mathbf{0} + \alpha = \alpha$
- **Pond 2** case
  - $y = \beta * x + \alpha = \beta * \mathbf{1} + \alpha = \beta + \alpha$
- With the dummy encoding, α is the **mean of pond 1**
  - Implicitly assumes **pond 1** is the **"baseline/control" group**
  - β is the **difference** between pond 1 and pond 2

Categorical variables

with more than 2 values

- **Pond 1** case
  - $y = \beta_2 * 0 + \beta_3 * 0 + \alpha = \alpha$
- **Pond 2** case
  - $y = \beta_2 * \mathbf{1} + \beta_3 * 0 + \alpha = \alpha + \beta_2$
- **Pond 3** case
  - $y = \beta_2 * 0 + \beta_3 * \mathbf{1} + \alpha = \alpha + \beta_3$
- ($n - 1$ dummy variables used to represent n values)

My data

Experimental setup

- Need to assess the **performance of a multilingual language model** across several languages and tasks
- Want to know the **effect of parameters** used during training
    - Number of **training steps** (how long the model trains)
    - Size of **vocabulary**
    - Language **sampling rates** during training
- Different languages have **very different performance**

Variables

- **Training steps**: {100k, 200k, 400k}
- **Vocab size**: {16k, 32k, 64k}
- **Alpha** (sampling parameter): {0.1, 0.2, 0.3, 0.4}
  - **Lower** alpha → low-resource langs **upsampled**, high-resource **downsampled**. Closer to **uniform distribution**
  - **Higher** alpha → closer to **actual distribution** languages
- **Task**: {POS, UAS}
  - Part-of-Speech tagging and Unlabeled Attachment Score (syntax)
- **Language**: {Hungarian, Finnish, Estonian, Russian, Erzya, Sami}

Spreadsheet view

task	vocabulary size	steps	alpha	few-shot (512)						
				Erzya	Estonian	Finnish	Hungarian	North Sami	Russian	Avg
UAS	32k	100k	0.4	44.88	52.17	68.10	61.89	30.30	69.13	54.41
UAS	32k	100k	0.3	46.56	52.06	69.23	64.25	33.64	70.80	56.09
UAS	32k	100k	0.2	47.90	51.73	69.54	65.07	34.74	70.26	56.54
UAS	32k	100k	0.1	49.72	51.57	68.85	64.38	37.75	69.20	56.91
UAS	32k	100k	0.05	50.19	51.01	69.01	64.40	40.42	67.86	57.15
UAS	32k	200k	0.2	52.52	55.09	71.32	67.04	43.63	72.85	60.41
UAS	32k	200k	0.1	53.82	54.18	70.17	67.58	45.97	71.54	60.54
UAS	32k	400k	0.2	56.83	57.22	72.38	69.25	51.25	73.40	63.39
UAS	32k	400k	0.1	56.60	56.74	72.01	70.14	51.97	71.90	63.23
UAS	64k	100k	0.4	43.53	53.74	70.62	69.33	33.11	73.77	57.35
UAS	64k	100k	0.3	46.27	53.28	70.68	70.73	35.05	73.52	58.26
UAS	64k	100k	0.2	48.83	54.90	69.55	70.58	37.85	73.41	59.19
UAS	64k	100k	0.1	50.58	53.69	69.47	70.12	41.18	70.97	59.34
UAS	64k	200k	0.2	55.32	59.23	71.89	72.56	46.81	73.16	63.16
UAS	64k	200k	0.1	56.99	57.07	71.55	73.35	48.38	70.93	63.05
UAS	64k	400k	0.2	60.49	60.93	72.84	75.37	53.41	71.56	65.77
UAS	64k	400k	0.1	61.51	60.72	73.00	75.24	53.83	74.18	66.41

Complication

Artificial variable correlation

- We only **partially exhaust** all combinations of input variables
- Because long training is **expensive**, we only tested **two alpha values** for longer-running experiments
  - {100k, 200k, 400k} x {0.1, 0.2}
- For **shortest experiments**, we test **more alpha values**
  - {100k} x {0.1, 0.2, 0.3, 0.4}
- Problem: this introduces **artificial correlation** between the two variables
  - **Low alpha** is correlated with **longer training**

Solution?

- Our solution to correlation so far is to **break into two regressions**
  - Regression A: {100k, 200k, 400k} x {0.1, 0.2} x {16k, 32k, 64k}
  - Regression B: {100k} x {0.1, 0.2, 0.3, 0.4} x {16k, 32k, 64k}
- This gets rid of **artificial correlation** betwen training steps and alpha
- However…
  - It **complicates the analysis/interpretation**
  - **Reduces sample size**
  - Alpha can **appear non-significant** in case A (narrower range)

Other choices

- We handle each task as a **separate regression**
  - Results mostly mirror each other, so POS might be relegated to appendix
- Input variables are usually **normalized**
  - Makes the coefficients more **interpretable**
  - We divide each variable by its **minimum value**
    - e.g. {16k, 32k, 64k} → {1, 2, 4}
  - Normally variables are normalized around 0, but I think this way works better for our scale?

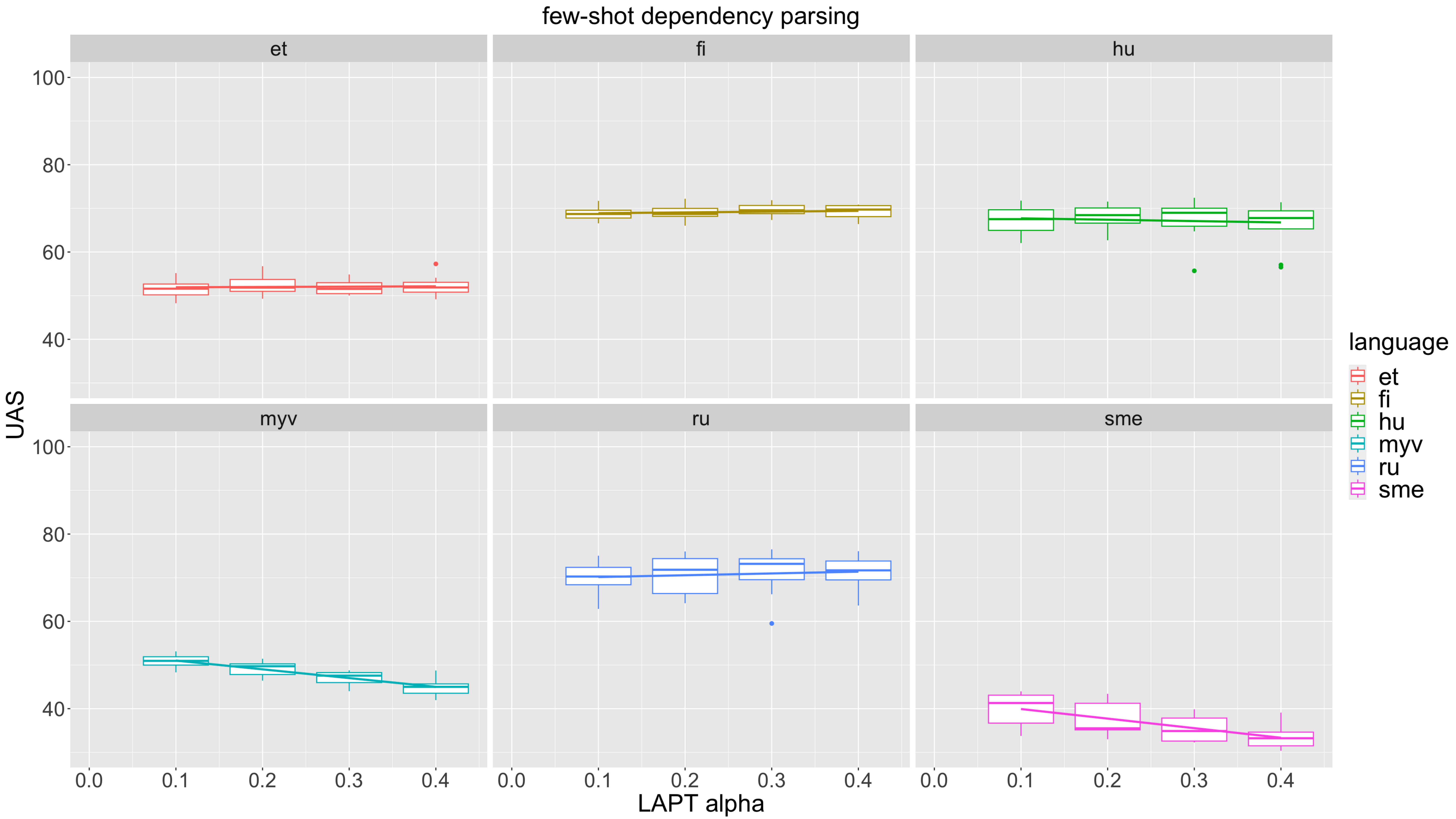
Setup so far

steps ↓ alpha ↓ vocab ↓

- POS accuracy ← {100k, 200k, 400k} x {0.1, 0.2} x {16k, 32k, 64k} x language
    - R: `lm(pos_acc ~ steps + alpha + vocab_size + language)`
- POS accuracy ← {100k} x {0.1, 0.2, 0.3, 0.4} x {16k, 32k, 64k} x language
    - R: `lm(pos_acc ~ alpha + vocab_size + language)`
- UAS accuracy ← {100k, 200k, 400k} x {0.1, 0.2} x {16k, 32k, 64k} x language
- UAS accuracy ← {100k} x {0.1, 0.2, 0.3, 0.4} x {16k, 32k, 64k} x language

Problem

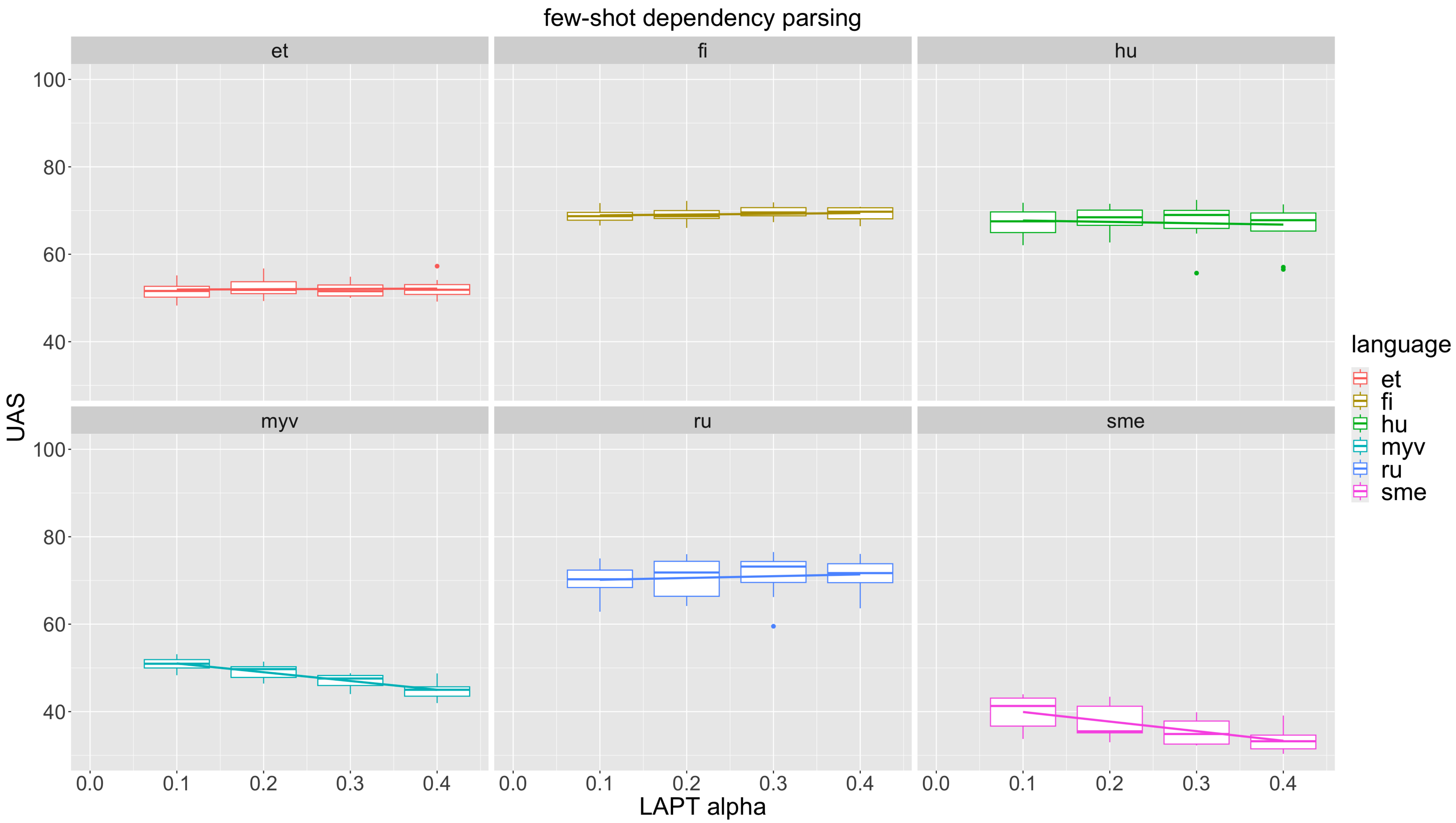
Different languages have different alpha slopes!



Problem

Different languages have different alpha slopes!

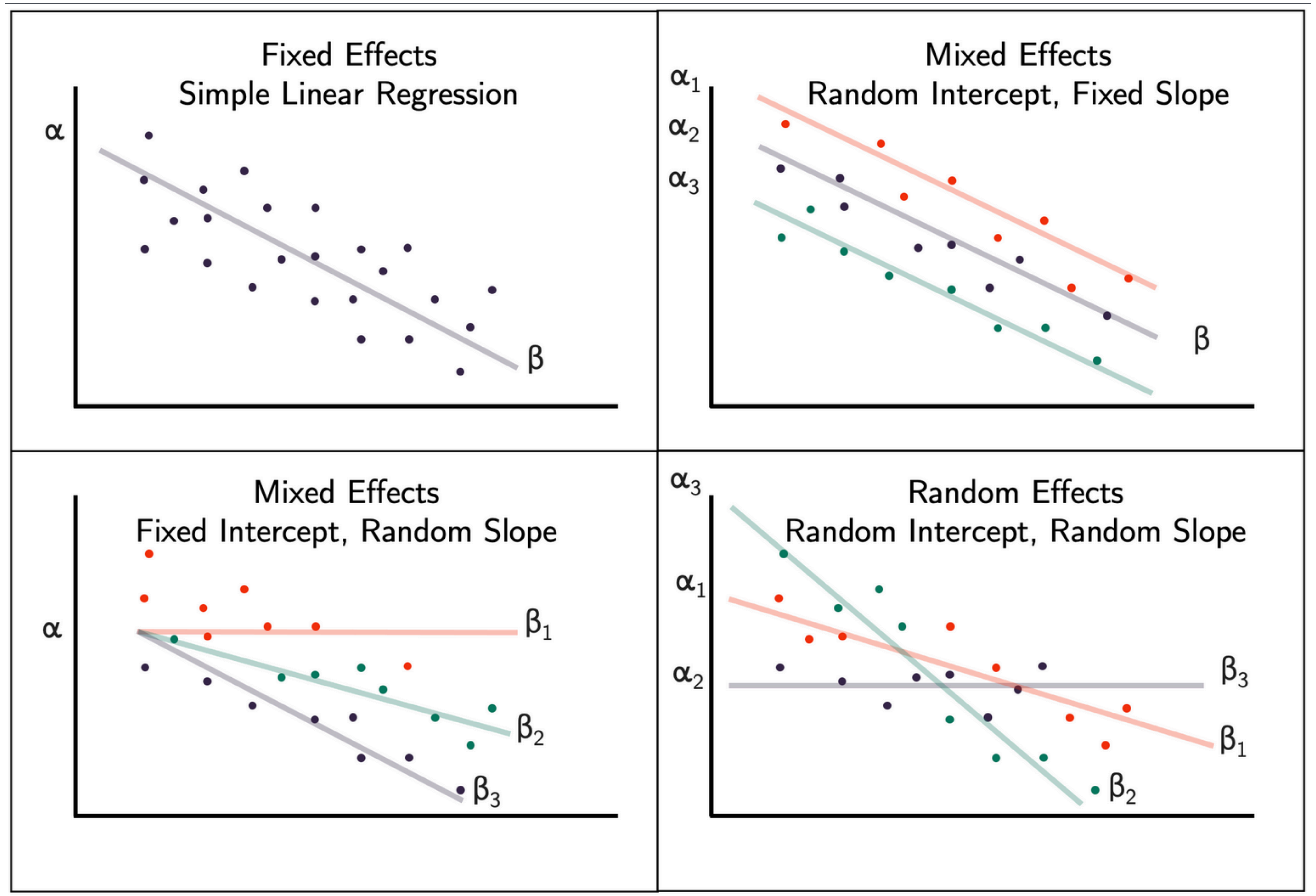
- Alpha affects languages differently **by design**
    - Meant to **increase sampling** of low-resource languages during training
- If used as a normal regression term (“main effect”), **only one slope** will be estimated
- How do we capture this language-wise variation?
    - **Random effects** (we think)



Random effects

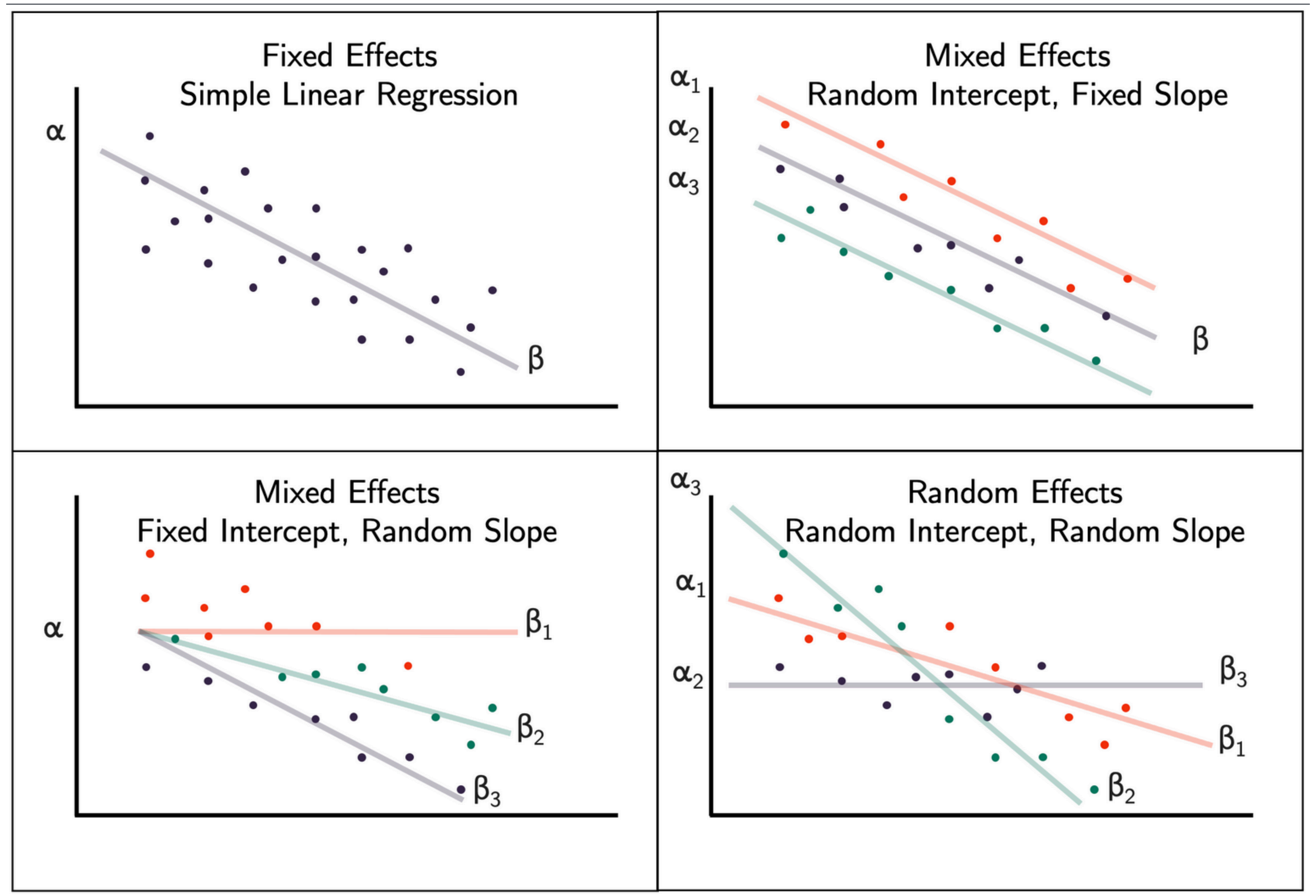
Random effects

- Simplest terms: get a **separate intercept** and/or **slope** for each value of a **categorical variable**
- **Warning:** my knowledge gets **hazier** from now on
- Check-in: which case is appropriate for **language-wise effect of alpha**?
  - We think **random intercept + random slope**



R syntax

- Fixed effects:
  - `lm(y ~ x)`
- Fixed slope, random intercept:
  - `lmer(y ~ x + (1 | cat_var))`
- Random slope, fixed intercept:
  - `lmer(y ~ x + (0 + x | cat_var))`
- Random slope and intercept:
  - `lmer(y ~ x + (x | cat_var))`



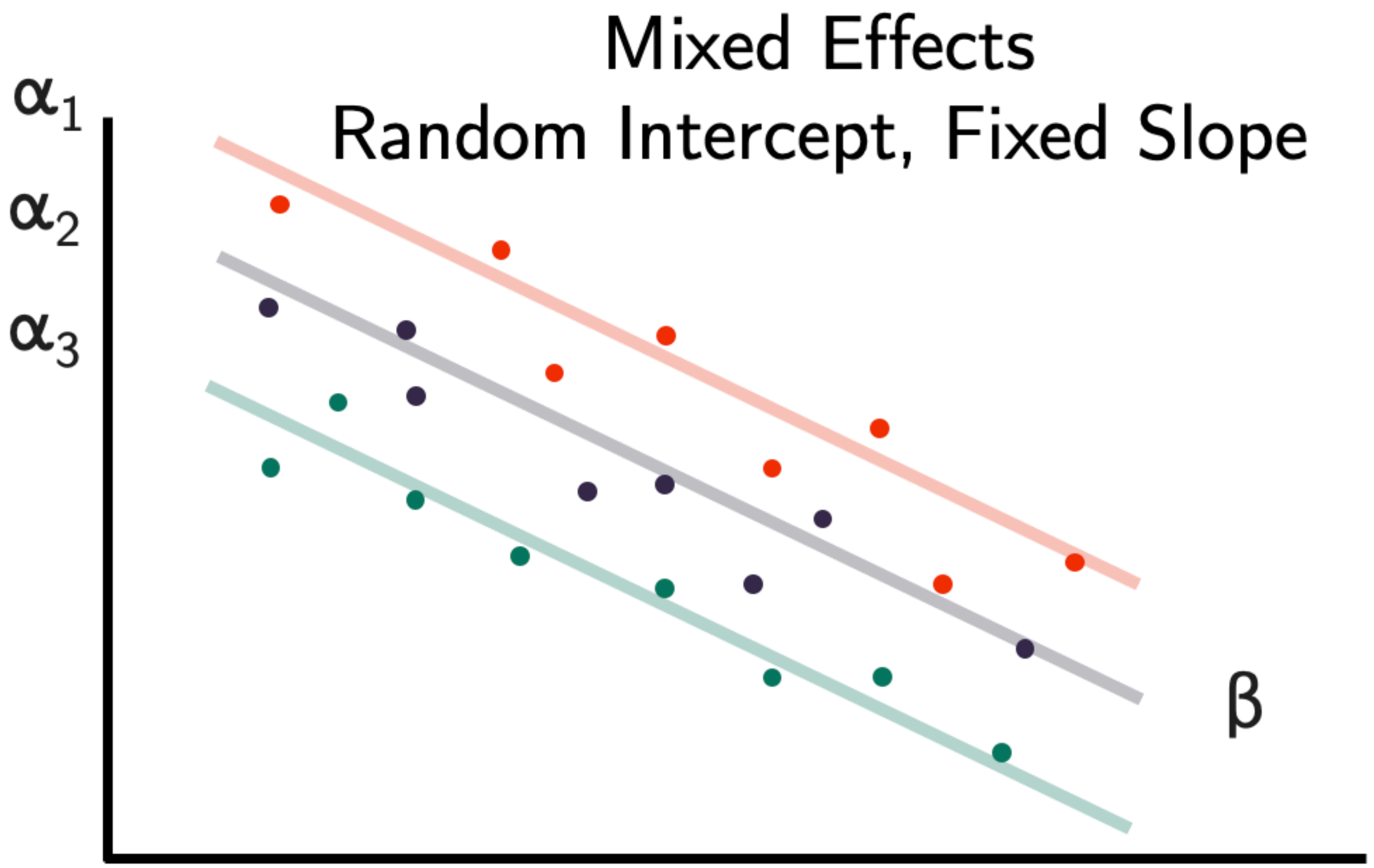
When to use random effects

(according to others)

- When values of a categorical variable are **non-exhaustive**
  - e.g. fixed effect if assuming binary sex, vs. random effect for sampling ponds
  - "Given levels in a random effect are **not separate and independent** but really representative levels from a larger collection" [source]
- "The built-in safety is that **if you have no real group-level information** or random effects at play, the random effects estimates will **essentially revert back** to fixed effects estimates"
- "Random effect estimates are a function of the **group level information as well as the overall (grand) mean** of the random effect. Group levels with low sample size and/or poor information (i.e., no strong relationship) are more strongly influenced by the grand mean"

A confusion of mine

- I don't understand how a **random intercept** conditioned on a categorical variable is different from a **fixed effect of a categorical variable**
- E.g. the difference between
  - lm(x_continuous + x_categorical)
  - lmer(x_continuous (1 | x_categorical))
- Both essentially define **category-wise offsets** from the global intercept (remember the pond example)



Bringing it back

Random effect for language

- We decide to handle language with **random effects**
  - **Random intercept** conditioned on language
  - **Random slope for alpha**, conditioned on language
- Fits the logic, since we **don’t** have an exhaustive set of languages
- R formulas
  - A: `lmer(accuracy ~ steps + vocab + alpha + (alpha | language))`
  - B: `lmer(accuracy ~ vocab + alpha + (alpha | language))`

Model results

- As expected, this gets us a **language-wise intercept** and **alpha slope**
  - (Steps and vocab size are **fixed across languages**)
- **Significance** value only given for **fixed effects**…
  - What if the alpha slope is significant for some languages but not others?

	(Intercept)	lapt_steps	vocab_size	lapt_alpha
et	49.54912	1.980228	0.8342288	-0.5968208
fi	63.30138	1.980228	0.8342288	0.4990893
hu	62.49371	1.980228	0.8342288	0.3978044
myv	48.91372	1.980228	0.8342288	-0.7947539
ru	64.18537	1.980228	0.8342288	0.5969614
sme	41.27124	1.980228	0.8342288	-1.4486693

Fixed effects:

	Estimate	Std. Error	t value
(Intercept)	54.9524	3.9783	13.813
lapt_steps	1.9802	0.1086	18.233
vocab_size	0.8342	0.1086	7.680
lapt_alpha	-0.2244	0.4436	-0.506

↑ significance

Points for feedback

- Should we have **separate regressions** for **each task**? Or should it be a categorical variable?
- Are we taking the right approach with "regression A" and "regression B"?
  - Right now, this is the only way I see around our **correlated input variables**
- Is our **normalization** adequate? {16k, 32k, 64k} →{1, 2, 4}
- Is our use of **random effects** appropriate? How do we tell if alpha is **significant for only some languages?** What significance test to use in general?
- Any other feedback? Things we should be doing differently?