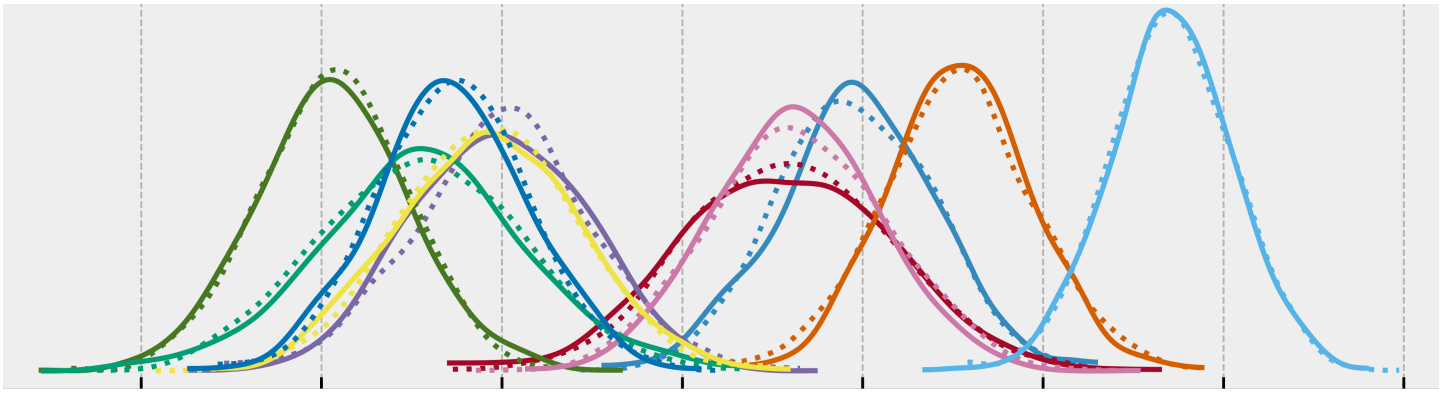
# Introduction to hierarchical modeling

Model naturally clustered data using Bayesian hierarchical models



Surya Krishnamurthy Aug 3, 2020 · 6 min read ★



Distribution of a coefficient across different groups from this analysis

## Introduction

It is not uncommon to find samples in our datasets that are not completely independent. Samples in datasets often form clusters or groups within which some properties are shared. This often requires some special attention while modeling to build reliable models for two major reasons. First, the independence of samples is an important assumption for several statistical analysis procedures like maximum likelihood estimation. Second, we want to capture the variation of the influence of the predictors across groups in our model called contextual effect.

A commonly used example, that I find sufficient to understand the scenario is that of the performance of students in a school. In a school with students in multiple classes, the academic performance of a student is influenced by their individual capabilities

(called fixed effects) and the class they're a part of (called random effects). Maybe the teacher assigned to a particular class teaches better than the others, or a higher proportion of intelligent students in a class creates a competitive environment for the students to perform better.

One approach to handle the case is to build multiple models for each class, referred to as **pooling**. But such an approach might not always produce reliable results. For example, the model corresponding to a class with very few students will be very misleading. A single **unpooled** model might not be able to fit sufficiently on the data. We want to find a middle ground that finds a compromise between these extremes — partial pooling. This brings us to **Bayesian hierarchical modeling,** also known as multilevel modeling.

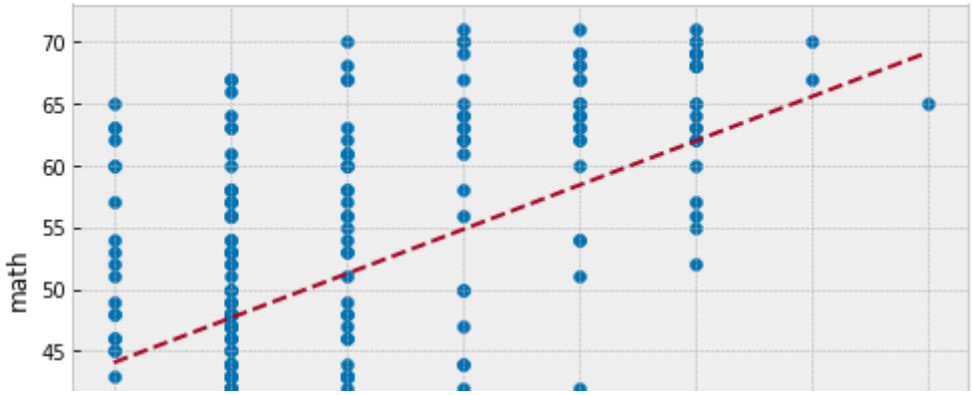
In this method, parameters are nested within one another at different levels of groups. Roughly, it gives us the weighted average of the unpooled and pooled model estimates. Hierarchical modeling is one of the most powerful, yet simple, techniques in Bayesian inference and possibly in statistical modeling. In this post, I will introduce the idea with a practical example. Note that this post does not cover the fundamentals of Bayesian analysis. The source code for the example is available as a notebook in GitHub.

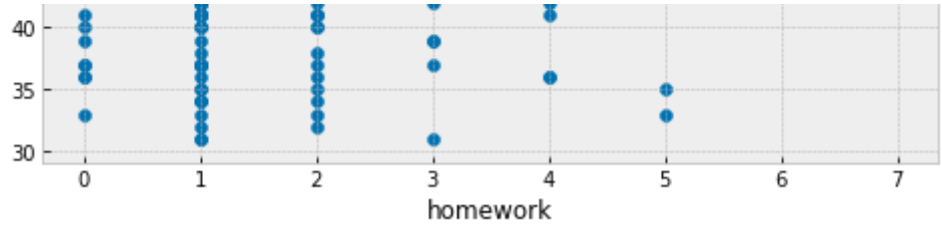
## Data

The dataset used for illustration is similar to the students example above except we're trying to find the math scores of students from different schools. We use the homework completion as a predictor for our example. You can find the original data here and it's csv version here.

## First look

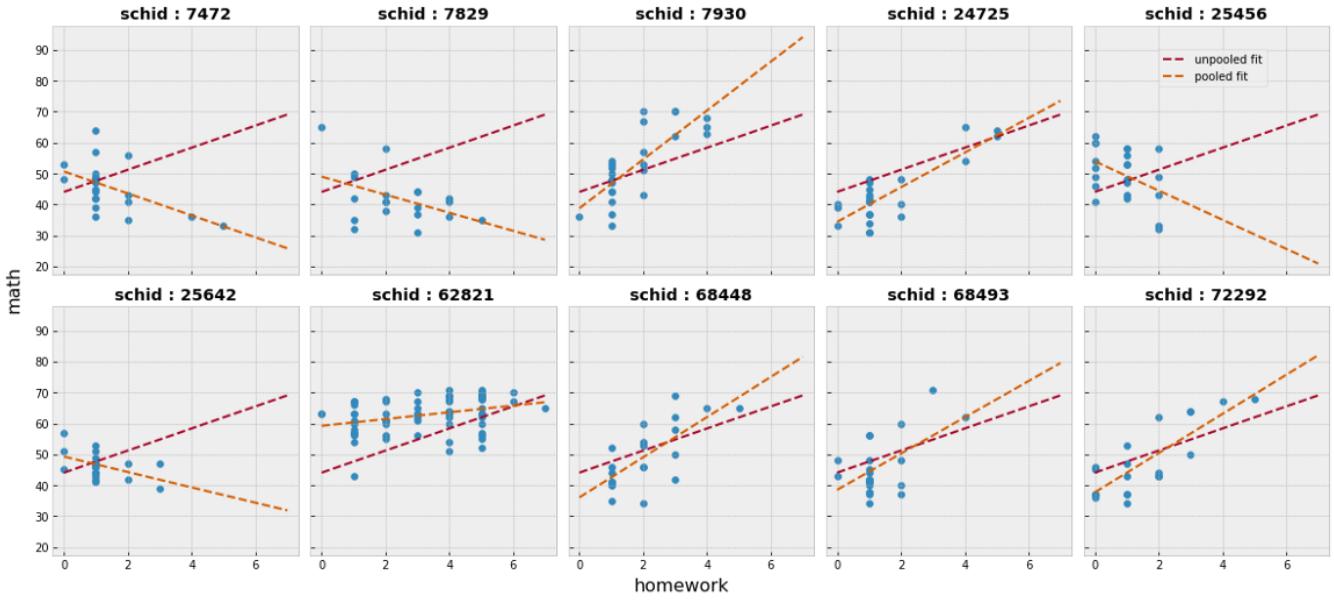
Plotting the math scores of the students against homework along with the **unpooled** OLS regression fit gives us this:






The data samples with unpooled regression fit

Visualizing the data at the school level reveals some interesting patterns. We also plot the **pooled** regression lines fit on each school and the unpooled regression fit for reference. For simplicity, we use OLS regression.




The data samples with pooled regression fit across groups

The plot shows the variation of the relationship across groups. We also notice that the estimates are highly influenced by the few data points (possible outliers) with high homework completions in some of the groups.

## Hierarchical model

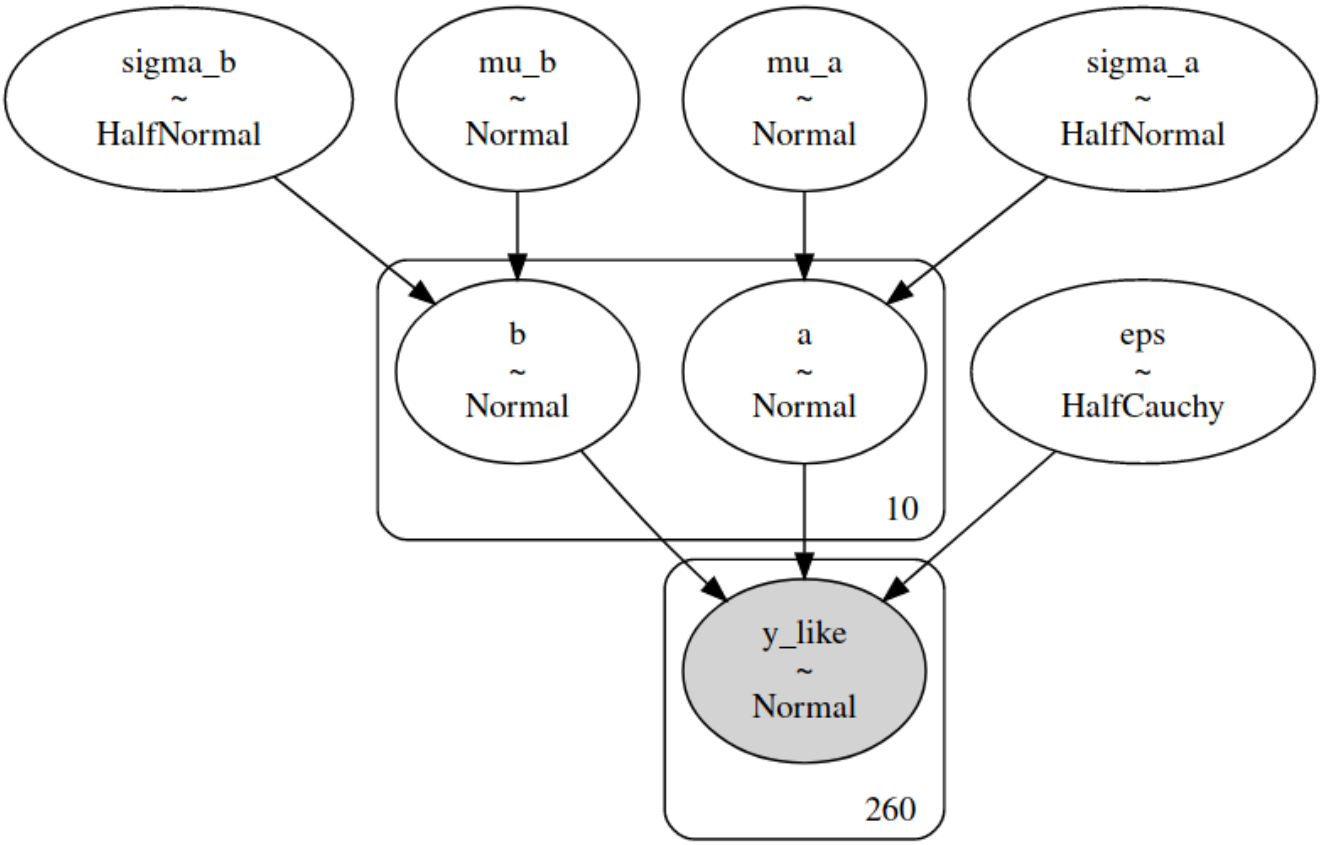
We will construct our Bayesian hierarchical model using PyMC3. We will construct **hyperpriors** on our group-level parameters to allow the model to share the individual properties of the student among the groups.

The model can be represented as $y_i = \alpha_{ji} + \beta_{ji} x_i + \varepsilon_i$ ,

or in probabilistic notation as $y \sim N(\alpha_j + \beta_j x, \varepsilon)$.

For this model, we will use a random slope $\beta$ and intercept $\alpha$. This means that they will vary with each group instead of a constant slope and intercept for the entire data. The

graphical representation of the probabilistic model is shown below.




Graph representation of the hierarchical model used in this example

While I choose my priors here by eyeballing the general distribution of the samples, using uninformative priors will lead to similar results. The code snippet below defines the PyMC3 model used.

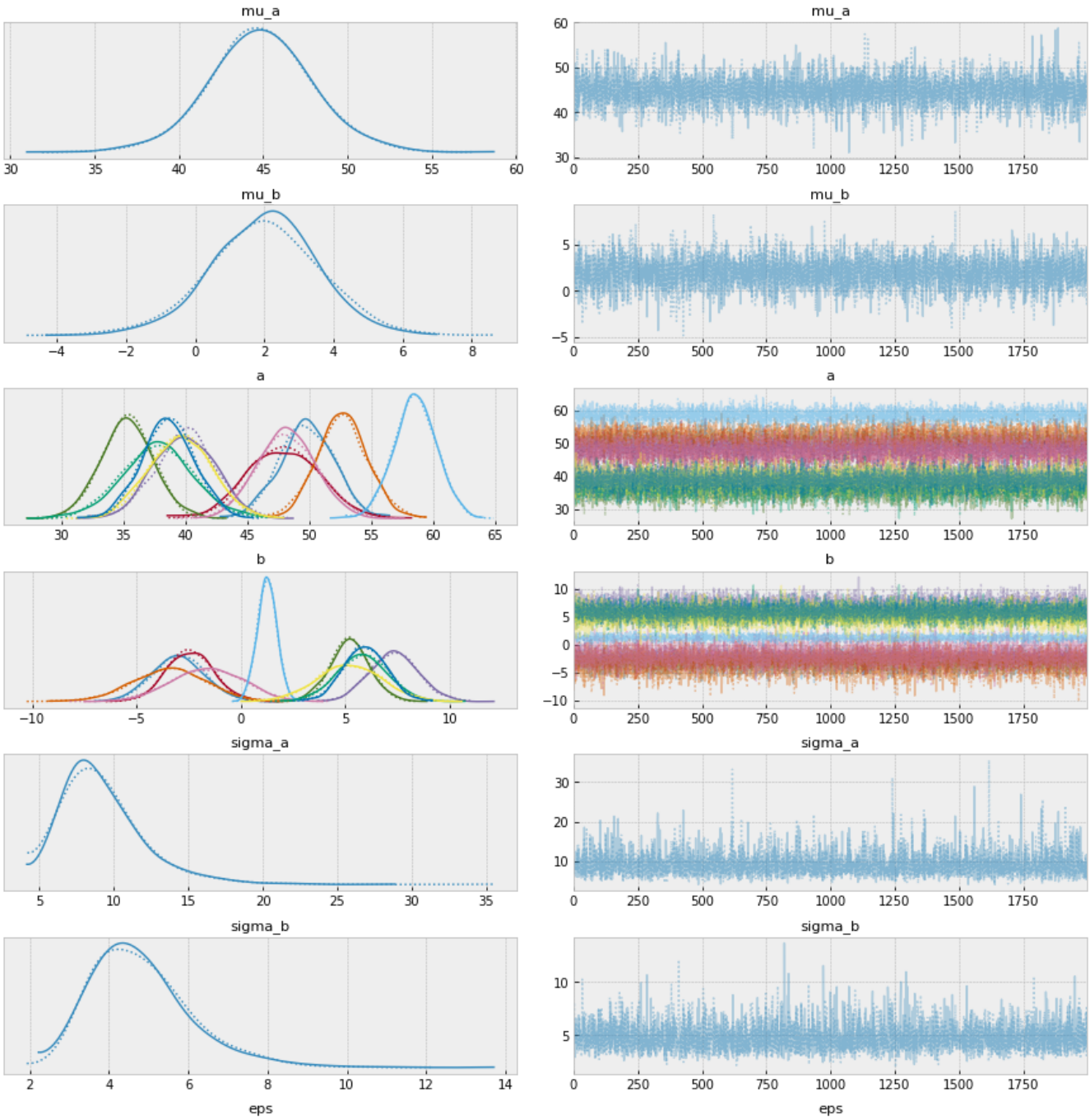
```
with pm.Model() as model:
    # Hyperpriors
    mu_a = pm.Normal('mu_a', mu=40, sigma=50)
    sigma_a = pm.HalfNormal('sigma_a', 50)

    mu_b = pm.Normal('mu_b', mu=0, sigma=10)
    sigma_b = pm.HalfNormal('sigma_b', 5)

    # Intercept
    a = pm.Normal('a', mu=mu_a, sigma=sigma_a, shape=n_schools)

    # Slope
    b = pm.Normal('b', mu=mu_b, sigma=sigma_b, shape=n_schools)

    # Model error
    eps = pm.HalfCauchy('eps', 5)

    # Model
    y_hat = a[school] + b[school] * homework
```

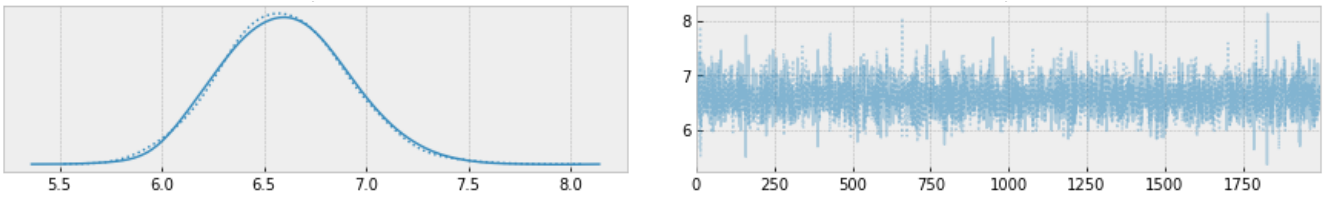
```
    # Likelihood
    y_like = pm.Normal('y_like', mu=y_hat, sigma=eps, observed=math)
```

We will use the NUTS sampler for drawing samples from the posterior distribution.

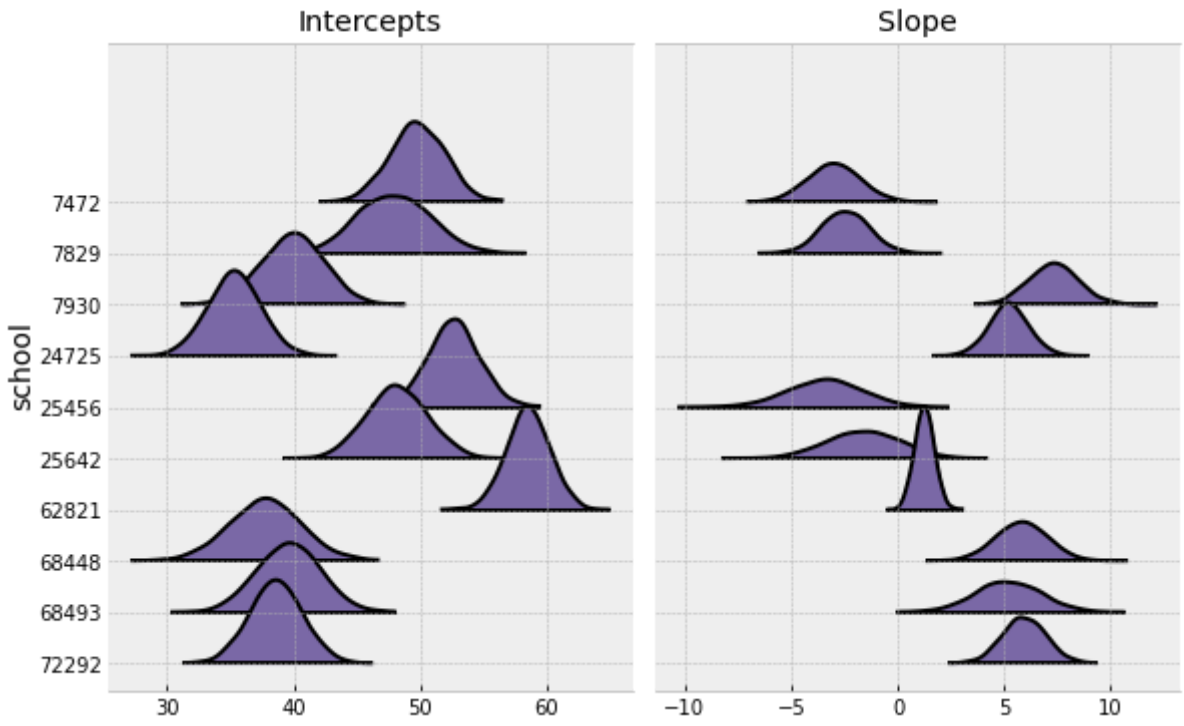
```
with model:
    step = pm.NUTS()
    trace = pm.sample(2000, tune=1000)
```

The trace plot and the posterior distribution of the slope and intercept corresponding to each school is visualized below.



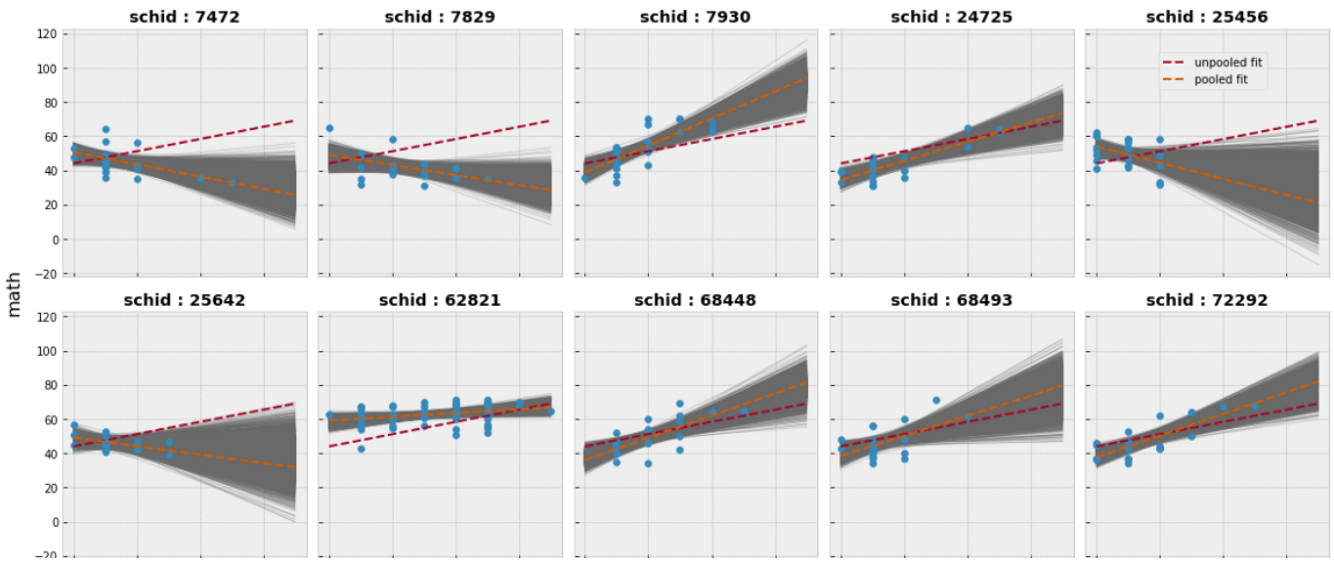


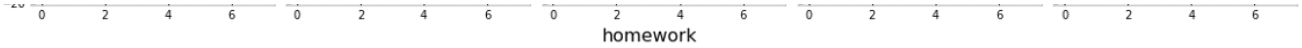
Trace plot for the hierarchical model




Posterior distribution of the intercept and slope for each group

We can see the variation in the estimates of our coefficients across different schools. We can also clearly interpret the uncertainty associated with our estimates from the distributions. The posterior predictive regression (gray) lines below, sampled from the posterior distribution of the estimates of each group, gives a better picture of the model with respect to the data.






Posterior predictive fits of the hierarchical model

Note the general higher uncertainty around groups that show a negative slope. The model finds a compromise between sensitivity to noise at the group level and the global estimates at the student level (apparent in IDs 7472, 7930, 25456, 25642). This implies that we must be a little warier of the decisions derived from the model on these groups. We also observe that with more data and lesser deviation, the Bayesian model converges to the OLS model of the group (ID 62821) as expected. We can also check the student level relationship by plotting the regression lines from `mu_a` and `mu_b` (which I omit here).

Cross-validation with the different models will show the superiority of the hierarchical modeling approach. Cross-validation can be performed at 2 levels:

1. Hold out students within a group and evaluate against its prediction.
2. Hold out an entire group and evaluate its prediction. Note that this is not possible with the pooling model.

I do not perform validation here as the frequentist and Bayesian models used here don't make for a fair (or easy) comparison. But the CV can be performed by replacing the OLS regression with Bayesian linear regression and comparing their Root Mean Squared Deviation (RMSD) of the models.

## Conclusion

Bayesian hierarchical modeling can produce robust models with naturally clustered data. They often allow us to build simple and interpretable models as opposed to the frequentist techniques like ensembling or neural networks that are commonly used for such complex data. They also prevent overfitting despite the increase in the number of parameters in the model. The post is merely an introduction to hierarchical modeling and its inherent simplicity allows us to implement different variations of the model specific to our data (eg: adding sub-groups, using more group-level predictors) and conduct different types of analysis (eg: find correlation among levels).

## Resources

**GLM: Hierarchical Linear Regression - PyMC3 3.8 documentation**

This tutorial is adapted from a blog post by Danne Elbers and