A Position-Sensitive Mixture Item Response Model

Klint Kanopka
Assistant Professor of Applied Statistics
New York University
klint.kanopka@nyu.edu

Paper forthcoming in the Journal of Educational and Behavioral Statistics

Overview

1. Position Effects
2. Position-Sensitive Mixture IRT Model
3. Software and Estimation
4. Empirical Application (ENEM)

Position Effects

Visualizing Position Effects

Booklet One

$$7 \times 8 =$$

$$9 - 3 \div \frac{1}{3} + 1 =$$

$$\int_0^{\pi} \sin x \; dx =$$

Booklet Two

$$\int_0^{\pi} \sin x \; dx =$$

$$9 - 3 \div \frac{1}{3} + 1 =$$

$$7 \times 8 =$$

Booklet Three

$$9 - 3 \div \frac{1}{3} + 1 =$$

$$\int_0^{\pi} \sin x \; dx =$$

$$7 \times 8 =$$

Motivation

Position effects are also not even remotely new

- Position effects are often thought of as negative (fatigue or disengagement)
- Can also be positive (practice effects) or more complex
- Earlier work includes linear logistic test model (Hohensinn etmal., 2008) and explanatory item response models (De Boeck, 2004)
    - Assume all respondents experience the same degree of parameter displacement
    - Individual differences in willingness to persist or develop practice effects complicate the counterfactual reasoning that underlies the equating procedure (Tucker-Drob, 2011)
- Some more recent work models individual differences(Albano, 2013; Trendtel & Robitzsch, 2018; Weirich et al., 2014)
    - Assumes item parameters are fixed across all respondents
    - Position effects can depend on item features (Kingston & Dorans, 1984)

Motivation

Position effects are only becoming more important

- Computer adaptive tests present items many different orders
- Previous work finds that the within-person relationship between time use and accuracy evolves over the course of a large computer adaptive test and there is population heterogeneity in the evolution (Domingue, et al., 2021)
- Implies response processes are evolving throughout a test making *when* an item is encountered potentially very important
- If position effects are bad enough, they can make using a test for between-person comparisons impossible

Position Effects

The order that items are encountered in can make a difference

Setup

"Position effects" is a catch-all term for any dependency of the probability of correct response on item location

Item-Side Effects

Item parameters (usually difficulty) change based on item position (Debeer and Janssen, 2013)

Person-Side Effects

Usually thought of as performance decline (Jin and Wang, 2014), but could also reflect practice effects

Complications

Test experiences that were designed to be equivalent may not be in practice

Can we develop an item response model in a way that controls for position effects while also providing more information about both persons and items?

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

$$P(X_{ij} = 1|\theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

Person *i's* ability

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

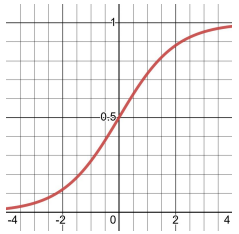
Person *i's* ability

Item *j's* difficulty

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Logistic function constrains output between 0 and 1



Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

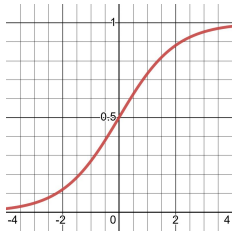
Person *i's* ability

Item *j's* difficulty

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Logistic function constrains output between 0 and 1



Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

Person *i's* ability

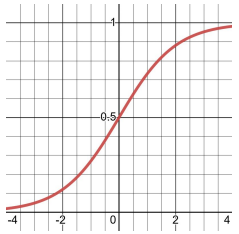
Item *j's* difficulty

Probability only depends on the *difference* between ability and difficulty

Item response function (IRF)

Probability of correct response depends on the difference between ability and difficulty

Logistic function constrains output between 0 and 1



Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_j) = \frac{1}{1 + e^{-(\theta_i - b_j)}}$$

For convenience, we'll replace the right-hand side of this equation with:

$$= \sigma(\theta_i - b_j)$$

Probability only depends on the *difference* between ability and difficulty

Incorporating Item Position into the Model

What changes?

- Item probability must depend on item position
- New item and person parameters to capture position effects

What stays?

- Single person ability assumed constant over test
- Based on IRT

Potential issues?

- Insufficient variation in item position
- Overfitting to non-existent position effects

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

$$P(X_{ij} = 1|\theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

$$P(X_{ij} = 1 | \theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

$$P(X_{ij} = 1|\theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

Logistic functions that depend on the *difference* between ability and difficulty

$$P(X_{ij} = 1 | \theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

Logistic functions that depend on the *difference* between ability and difficulty

$$P(X_{ij} = 1|\theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Item *j's* difficulty at the *beginning* of the test

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

Logistic functions that depend on the *difference* between ability and difficulty

$$P(X_{ij} = 1|\theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Item *j's* difficulty at the *beginning* of the test

Item *j's* difficulty at the *end* of the test

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

Logistic functions that depend on the *difference* between ability and difficulty

$$P(X_{ij} = 1 | \theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

Item *j's* difficulty at the *beginning* of the test

A weight that depends on where person *i* encounters item *j*

Item *j's* difficulty at the *end* of the test

Position Sensitive IRF

Probability of correct response now depends explicitly on item position

Person *i* responds to item *j* correctly

Person *i's* ability

Logistic functions that depend on the *difference* between ability and difficulty

$$P(X_{ij} = 1 | \theta_i, b_{j\alpha}, b_{j\omega}, \pi_{ij}) = \pi_{ij}\sigma(\theta_i - b_{j\alpha}) + (1 - \pi_{ij})\sigma(\theta_i - b_{j\omega})$$

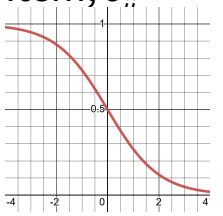
Item *j's* difficulty at the *beginning* of the test

A weight that depends on where person *i* encounters item *j*

Item *j's* difficulty at the *end* of the test

Key understanding:
The weight is another logistic function of the difference between a person-specific location parameter, k_i, and the position of the item, s_{ij}

$$\pi_{ij} = \sigma(k_i - s_{ij})$$



Related Models

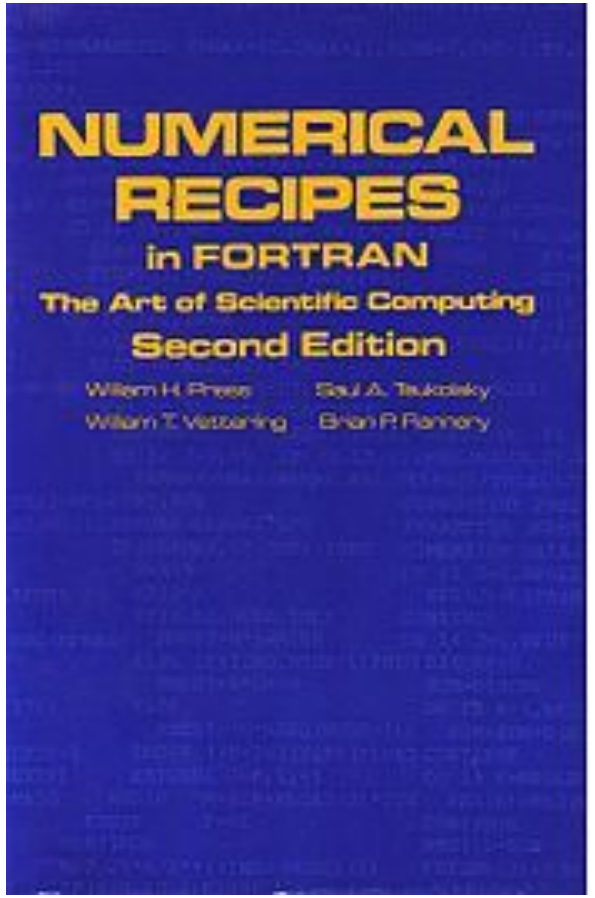
- This approach is similar to the HYBRID model proposed by Yamamoto (1995)
    - Modeled two different response processes as a discrete mixture
    - Combined effortful responding and random guessing
- Closer to the continuous HYBRID (or C-HYBRID) model of Nagy & Robitzsch (2021)
    - Modeled two different response processes as a discrete mixture
    - Combined effortful responding and random guessing
    - Had similar functional form for the mixing parameter

Estimation

New Models Often Require New Software

Software development can gatekeep model development

- Model fitting software is pretty well understood and built out of well-known blocks
  - You can look those blocks up and Ctrl+C / Ctrl+V
  - You can use built-in tools like `optim()` in `R`
- There are typically three bespoke components required
  - Data handling
  - Model specification
  - Model-specific utilities



Deep Learning Frameworks

Purpose built for efficiency and rapid iteration

- Code efficiency
  - Data loading, model specifications, loss functions, and optimization are treated as independent components
  - Auto-differentiation makes trying out different model specifications extremely easy
  - Most commonly used options are single lines of code
  - The bulk of the code you'll write is completely boilerplate
- Computational efficiency
  - Different built in optimization algorithms to balance convergence speed and precision
  - Built in data structures designed for minimal memory footprints
  - Easy access to multiprocessing on CPU or GPU computation
- Data efficiency
  - Data loader objects efficiently process and iterate over large amounts of data
  - Functions can accommodate full data, single observations, or minibatches

Torch or Tensorflow?

I clearly have strong opinions on this

- Both are good
  - Open source
  - Usable with Python and R
  - Tons of documentation and community support
- Tensorflow
  - More common in production environments
  - Faster
- PyTorch
  - More common in research communities
  - Easier to use
  - More readable code
  - Pyro if you’re Bayesian

Estimating the Position Sensitive Model

After lots of iterations, here's where we ended up

- Optimization done using a joint maximum likelihood Alternating-Maximization procedure
  1. (E-step) Item parameters are held fixed while person parameters are estimated
  2. (M-step) Person parameters are held fixed while item parameters are estimated
  3. Repeat Steps 1&2 until convergence
- E and M-step optimizers can be tuned and configured separately
  - Both use minibatch stochastic gradient ascent
  - Both use momentum and variable learning rates
  - During the E-step, person parameters are regularized (L_2)
  - Final person parameters are estimated using MLE
- Computation done on GPU

Summary of Simulation Results

More information about simulations available in preprint linked at the end

- Parameter recovery is good when the position sensitive model generates data
- Model does not overfit to non-existent position effects
- When position effects are not present, model does not separate the early and late test parameters and produces results similar to an IRT model without position effects
- Model fitting speed is somewhat dependent on amount of data, initial values of estimable parameters, and hyperparameter selection
- Reasoning about sample size becomes really difficult, as item-by-position coverage becomes incredibly important
- Adding more items can make estimation *worse* if they aren't observed enough in their possible positions

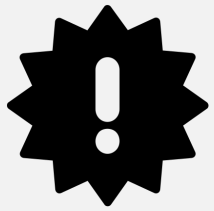
Applying the Model

Exame Nacional do Ensino Médio (ENEM)

ENEM is Brazil's national college entrance exam

Massive Exam

Second largest exam of its kind. 3.4 million students took ENEM in 2022



High Stakes

ENEM is given once yearly and used for high school certification, university admissions, and scholarships

Mostly Standardized

Everyone gets the same items at the same time, but one of four booklets with randomized item order

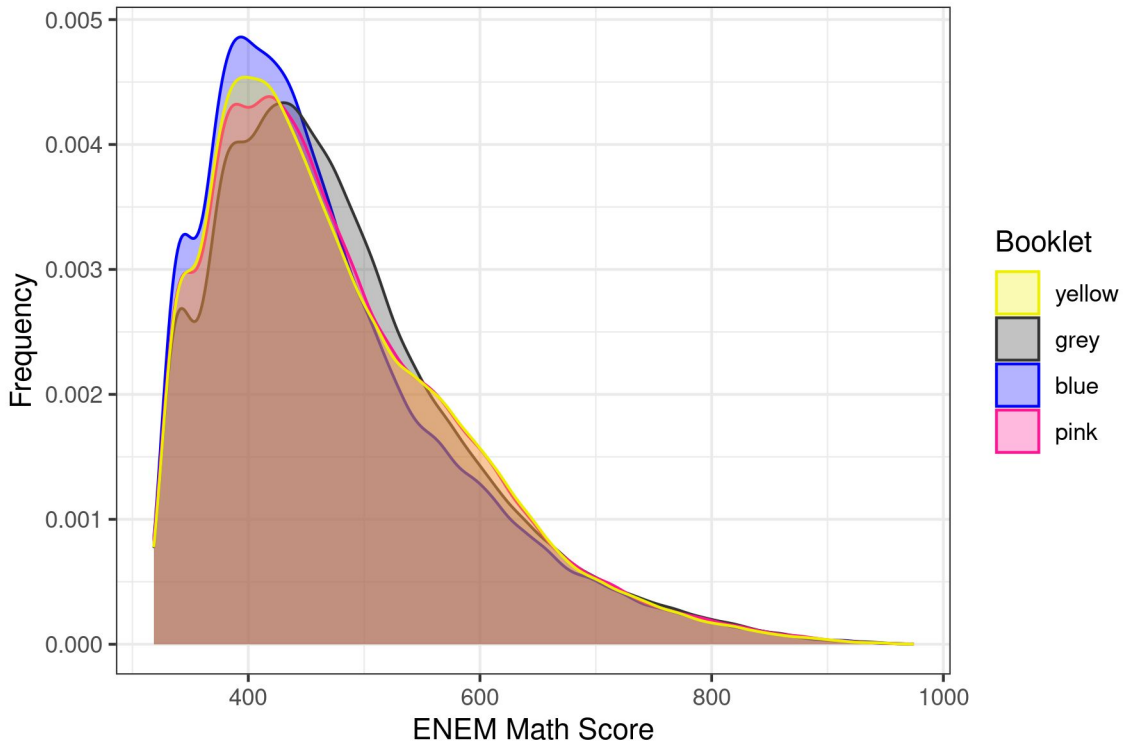
Unequal Results

Score distributions can vary depending on booklet assignment

Especially Bad in 2014

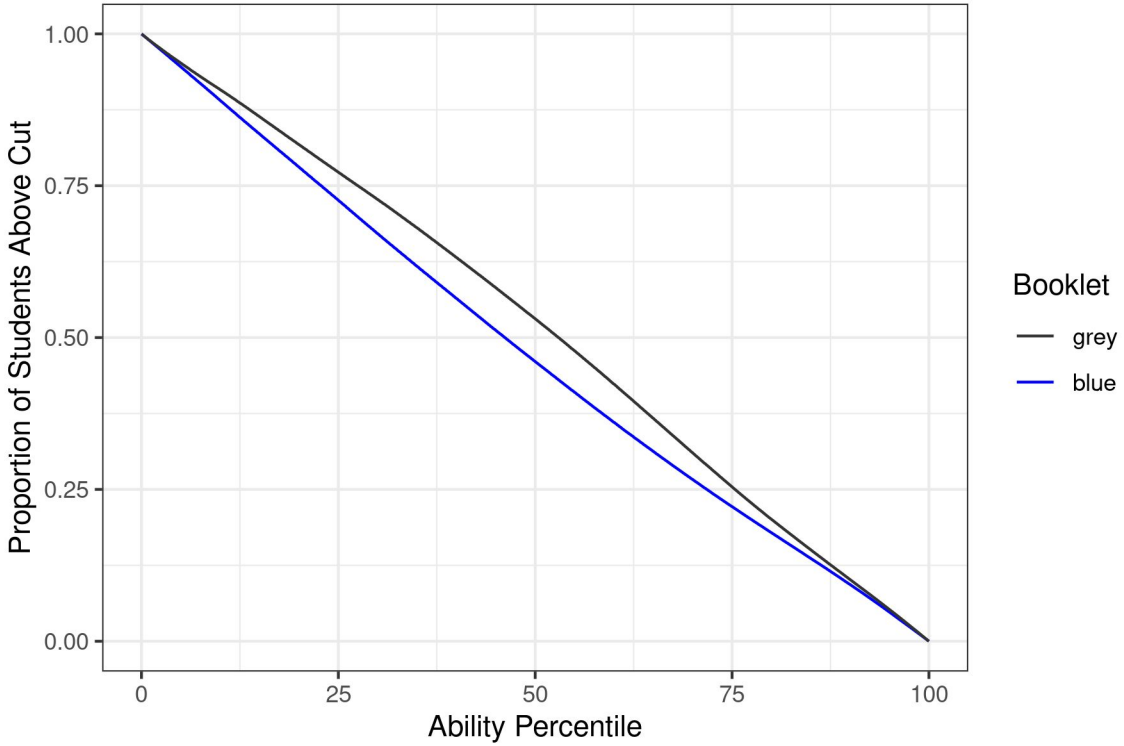
ENEM Math score distributions were different by booklet

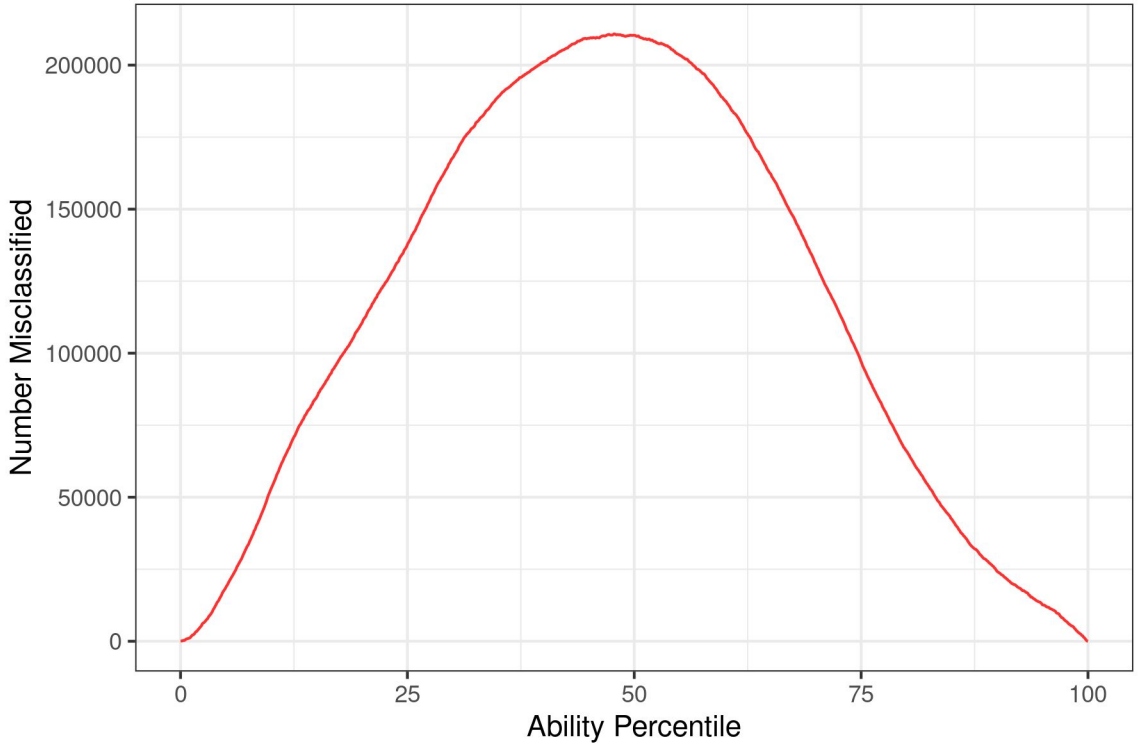
- Data from 6 million test takers in 2014
- Four colored booklets are assigned randomly within classrooms, so score distributions should be identical in expectation
- Difference in mean score between blue and grey booklet of 11.3 points



Mean Score Differences Undersell Problem

Position effects results in the misclassification of potentially thousands of students





ENEM is a Test Case

The real application is computerized and adaptive testing

Equating Solves ENEM's Problem

If all you care about is misclassification, this model is overkill

Position Effects are Identified

The administration and design of ENEM plausibly attributes score differences to position effects

Position Effects Have Impact

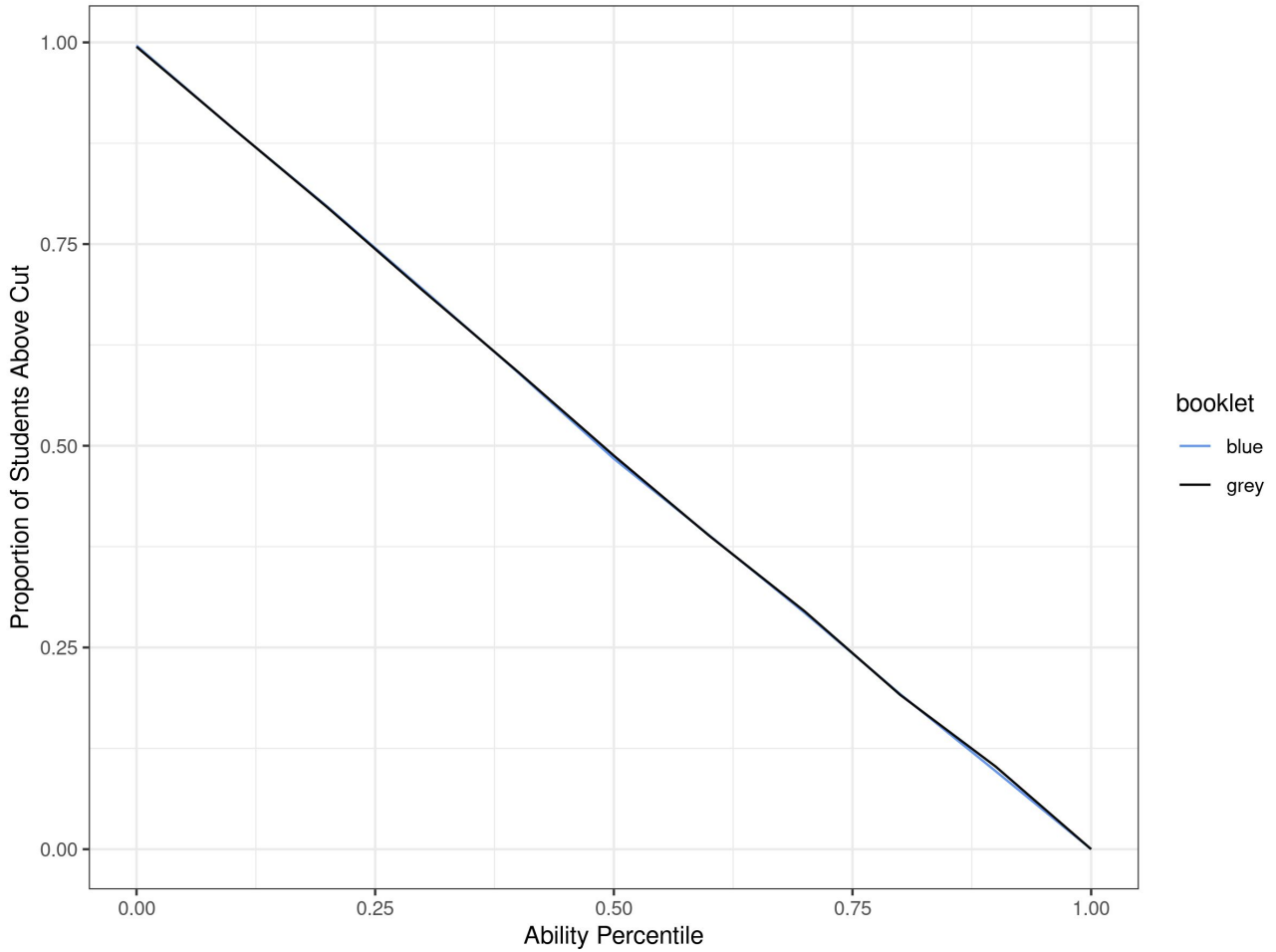
When position effects exist but are ignored, they can have material consequences for test takers

Data Generating Process Unknown

ENEM is real world data and the exact DGP by which position effects manifest here is unknowable

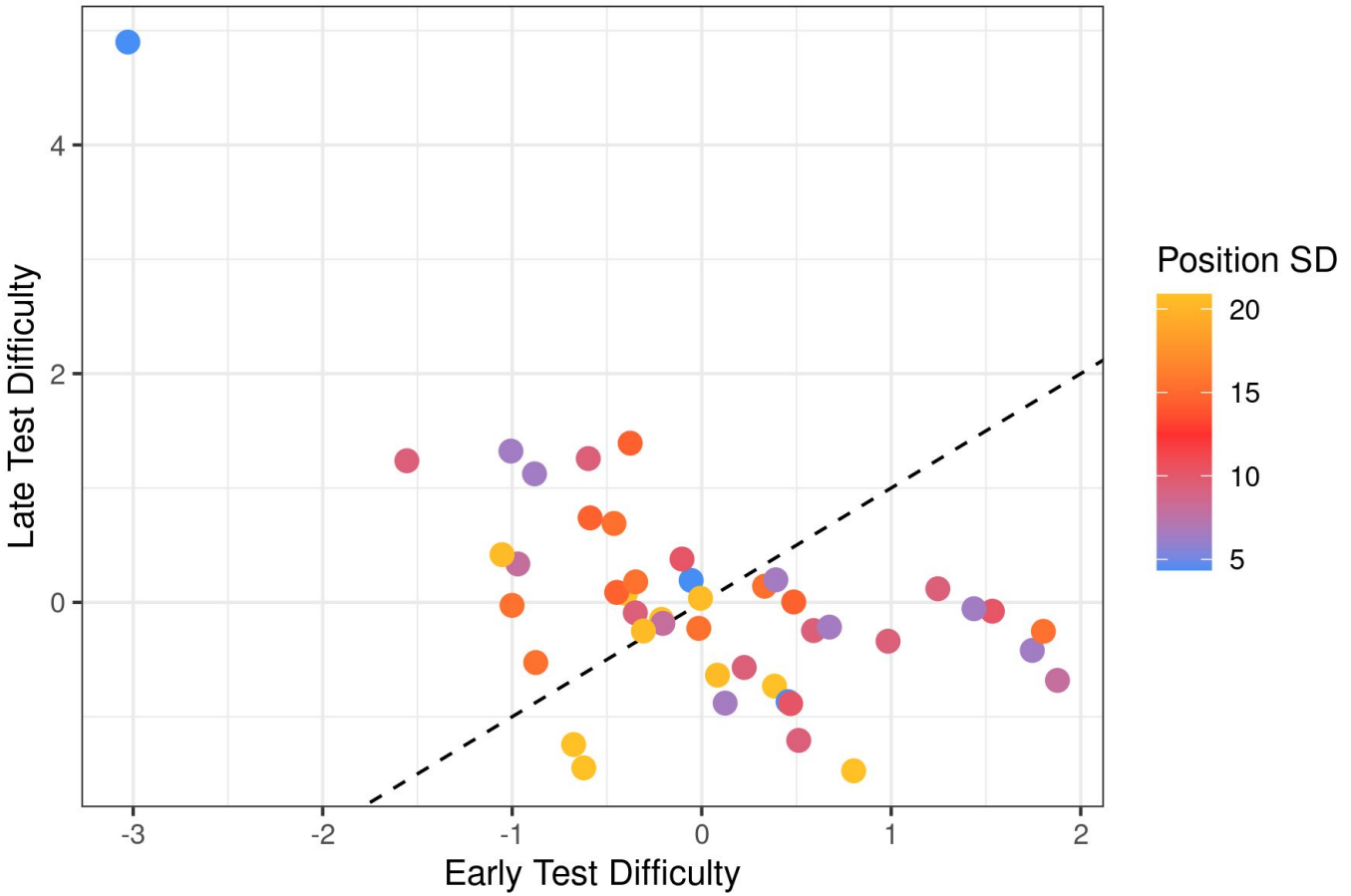
Position Sensitive Model Aligns Distributions

Misclassification is minimized



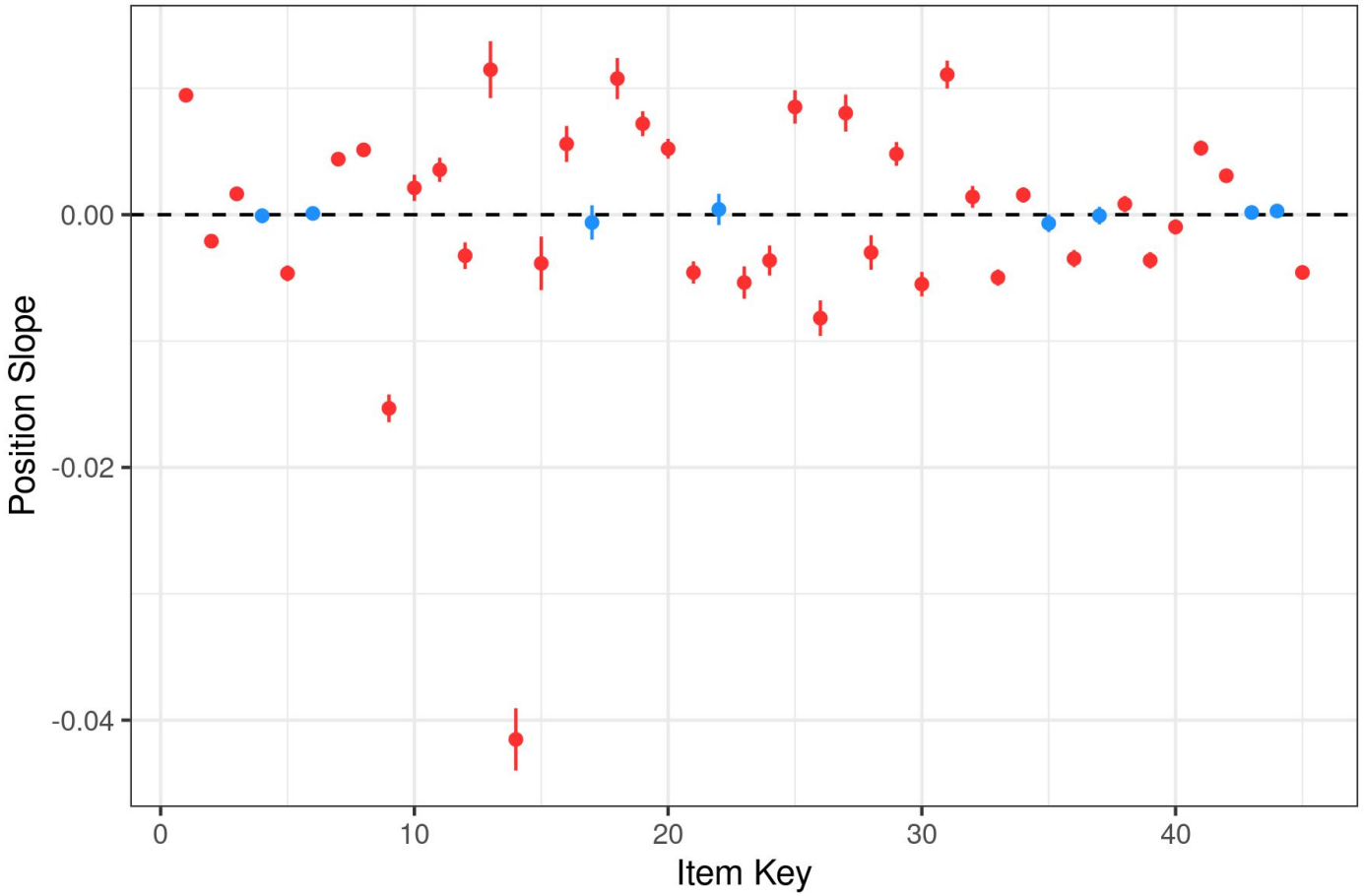
Comparison of Early and Late Item Parameters

Item-side position effects are not uniformly in the same direction



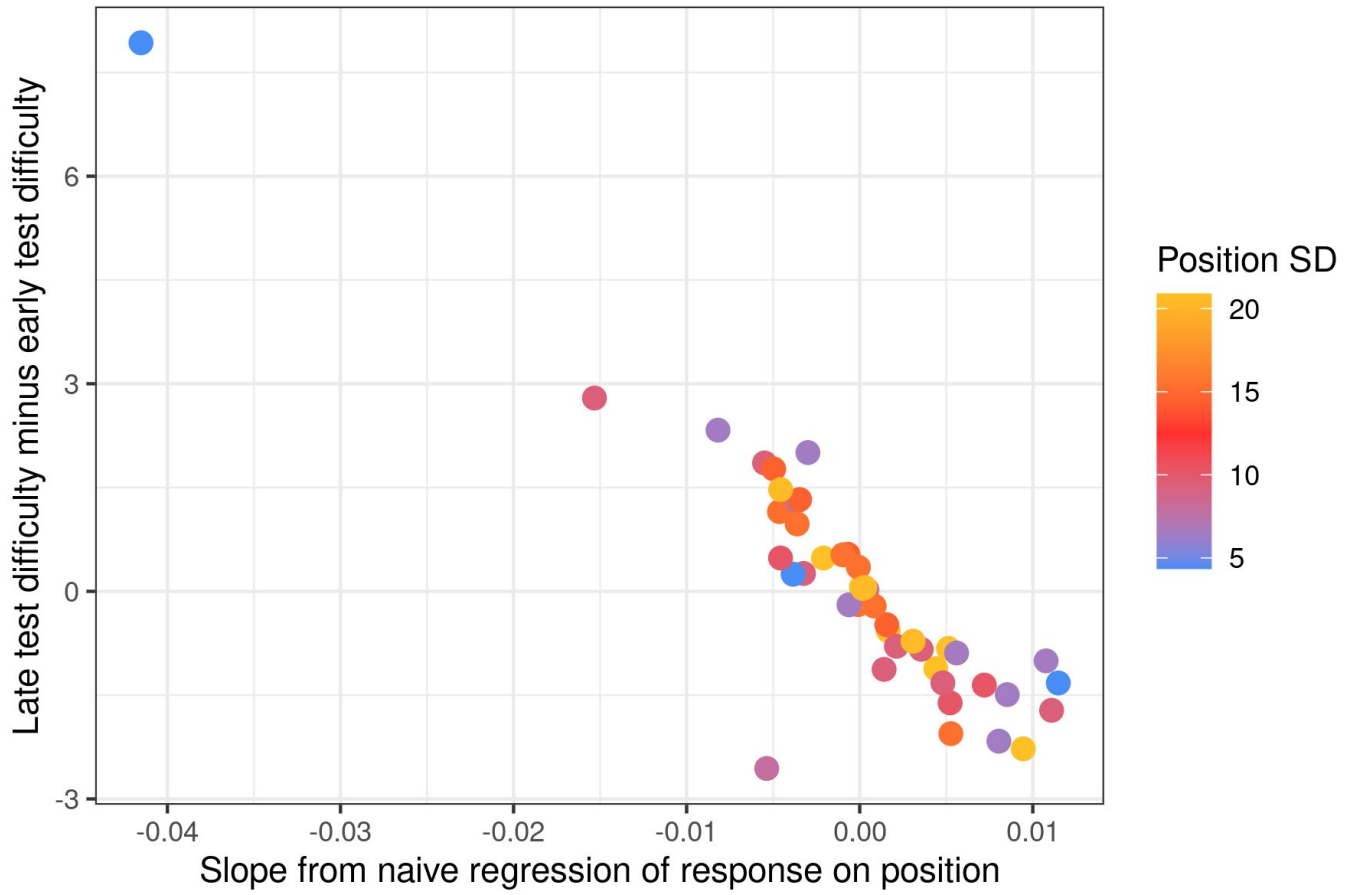
Heterogeneity in Position Effects

Naïve approaches to estimating item-level position effects tell a similar story



Magnitudes of Position Effects as Expected

Both position sensitive and naïve approaches agree on item-level effects



Conclusions and Implications

Model properties are robust to many situations where position effects may be at play

Self-Equating

Automatically produces scores individually adjusted for position effects

Interpretable Parameters

Differences in item difficulties reflect observed position effects

Safe for General Use

In simulation, model does not overfit and find non-existent position effects or spread item parameters with insufficient information

Investigative Value

Useful for adaptive and computerized scenarios when item position varies but effects may not be cleanly identified

Thank you!

Email: klint.kanopka@nyu.edu

X (RIP Twitter): @KlintKanopka

Bluesky: @klint.bsky.social