Biased Programmers? Or Biased Data? A Field Experiment about Algorithmic Bias

Bo Cowgill, Fabrizio Dell'Acqua
Graduate School of Business

Nakul Verma, Samuel Deng and Augustin Chaintreau
Department of Computer Science

Columbia University

Draft

Abstract

Why does "algorithmic bias" occur? The two most frequently cited reasons are "biased programmers" and "biased training data." We quantify the effects of these using a field experiment on a diverse group of AI practitioners. In our experiment, machine learning programmers are asked in a natural setting to predict math ability for a representative sample of over 20,000 OECD residents. One group of engineers is given perfectly representative training data, and the other is given a "dataset of convenience" – a biased training sample containing who confirm to common stereotypes about who is good at math. The engineers are also randomly exposed to reminders and technical advice about the presence and solution to algorithmic bias. We evaluate our experimental results in particular with respect to stereotypes about women and math.

We find that biased algorithms are mostly caused by biased data. However some interventions are effective appear at decreasing algorithmic errors. In particular, simple reminders about biased data are almost half as effective as providing completely unbiased data. By contrast, technical advice produced mixed results on algorithmic bias. Programmers who are able to understand and implement technical advice successfully reduce bias. However, many cannot follow the advice, resulting in algorithms that are actually *worse* than the programmers' who were given a simple reminder.

We also find that female programmers are more aware of the potential for algorithmic bias in the exercise, but ultimately do not produce less biased algorithms. We similarly find no evidence that programmers' subconscious attitudes about gender and math (measured by an IAT, implicit association test) are not correlated with bias their code. Lastly, we find evidence that incentives may reduce algorithmic bias.

1 Introduction

Why does "algorithmic bias" exist? Several high-profile cases allege algorithmic bias in criminal sentencing (Angwin et al., 2016), lending (Tan et al., 2017), recruiting (Lambrecht and Tucker, 2016), hiring (Ajunwa et al., 2016) and ad targeting (Datta et al., 2015) – raising policy questions about transparency, interpretability and discrimination. Scholars have proposed various plausible reasons, but almost no literature measures the underlying cause of algorithmic discrimination, or the effectiveness of real-world interventions to reduce it.

Our experiment aims to measure the two most frequently cited reasons for algorithmic bias, and two common proposed policy interventions. The first possible reason ("biased programmers") alleges that programmers developing algorithms have inherent biases that are passed onto the algorithms they write. As biases are correlated with demographics, a common version of this hypothesis claims the high-tech workforce at machine learning companies discriminates because it lacks diversity.

Most engineers in the technology sector are white or Asian men. These engineers may not be deliberately biased, but their approach to developing an algorithm may be influenced by their life experiences. For example, they may pay less attention to variables in the training data that are correlated with underserved demographics. Or they may look at particular training examples, measures of accuracy, or empirical applications of their algorithm that could bias the algorithm during development.[1]

The second theory is "biased training data" (Mullainathan and Obermeyer, 2017; Cowgill, 2017). Most machine learning applications are developed using historical data about outcomes. Such data may reflect historical biases about who is more (or less) likely to be labeled a criminal, or to be considered a future engineer or candidate for a high-paying job. Insofar as the real world is biased, data coming from it would reflect and perpetuate this bias. "Biased training data" may particularly come from sample selection problems. Suppose a bank wants to use machine learning to predict who will repay loans. If their historical dataset contains few loans to ethnic minorities, these algorithms may accidentally sustain, or even exacerbate, historically biased lending decisions. So-called datasets of convenience might be especially biased, and they would be exactly what companies use.

We believe that both hypotheses are probably responsible for algorithmic bias to some degree. However they are not necessarily equally responsible. It is possible that *neither* hypothesis is a particularly strong contributor to algorithmic bias. Programmer biographies might not affect the code or predictions they write. It is similarly possible that engineers given "unrepresentative data" can machine-learn their way out of this problem through a variety of computational techniques. For example, simple cross-validation may significantly help combat the problem of unrepresentative training data.

The goal of our field experiment aims to measure the sizes of these contributors so they can be directly compared on a common scale. There is very little research quantifying or comparing these

[1] One alleged example of this may have taken place in 2015, when Google released a photos product that offensively mislabeled African Americans (Forbes.com, 2015). Had Google's product developers had been more diverse, perhaps the problem would have been avoided.

contributing factors. Knowing the relative sizes of these biases will give practitioners practical guidance about combating algorithmic bias. If the dominant problem is "biased programmers," then the solution is to invest in diversifying the high-tech workforce and possibly training programmers to counteract subconscious biases. Alternatively: If the problem is "biased training data," the solution is to seek out higher quality data, and to invest in developing and teaching technical methods to model and adjust for biases.

Insofar as the second hypothesis ("biased training data") is important, a growing technical literature in computer science seeks to *reduce* algorithmic bias through technical solutions. Our experiment includes a fourth condition to test the effectiveness of technical guidance. Like our other hypotheses, we suspect that lack of technical training may explain bias. Technical guidance could plausibly reduce algorithmic bias enormously.

However, it could also fail to contribute very much. Programmers may not understand, could ignore or incorrectly implement the new techniques. Standard techniques such as simple cross-validation could go a long way, even without the new methods. In addition, companies could alternatively gather higher quality training data, and use standard techniques rather than utilizing new computational methods. The experiment is designed to measure the relative effectiveness of these potential solutions.

2 Experimental Design

In our experiment, we asked 260 machine learning engineers to complete a prediction task: Estimate math literacy for a representative sample of OECD citizens under different experimental conditions. We structured this as a programming contest, exploiting a setting that provides useful recognition for the engineers and job market credentials. The experiment was "blind"; that is, engineers were not told they are part of a research experiment. They were given instructions as if they are estimating the productivity of potential workers at math-related tasks for a company.

Math literacy is an ideal topic for studying algorithmic bias. Math skills are useful many occupations, particularly engineering, finance and other high-paying careers. Employers measures math aptitude directly and indirectly in job screening. Math performance is also the topic of common stereotyping suggesting that women are worse at math than men. This stereotype introduces the possibility of algorithmic bias if historical data is used carelessly.

In addition, high-quality training and test data about math literacy is collected by the OECD. Numeracy is measured internationally on a common scale by experts in OECD's *Programme for the International Assessment of Adult Competencies* (PIAAC). The PIACC data contains quantitative measures of math and other skills for a representative sample of the working-age population 24 countries.

The total PIAAC dataset contains over 200,000 citizen observations and over 500 covariates per citizen.[2] We asked our subjects to predict math skills as a function of biographical covariates, and evaluated them on a test set of 20,000 observations. We structured this as an individual task: the

[2]These characteristics include the citizens' demographic, educational, employment and other background characteristics.

objective of each programmer was to develop the most accurate predictions, i.e., those with the best mean squared error on a representative test set.

We recruited a relatively diverse group of engineers. One third of our subjects were women, two thirds men. With respect to ethnicity, sixty percent were East Asian; twenty-one percent white; fourteen percent South Asian and five percent African-American or Hispanic. All of the subjects had studied in top engineering schools, and about half of them were engineering majors.

Subjects in our experiment were randomly assigned to one of four different conditions, using stratified random assignment to balance ethnicity, major, school, degree, and gender. In the first condition, engineers received unbiased training data. In the second, third and fourth conditions, engineers received biased training data. In all cases, they were told a company was seeking advice on who to hire, and would have used their help. Programmers believed they were receiving a company dataset. We detail each experimental condition below.

Engineers in the first condition received 20,000 randomly selected people from the entire population. This training data is unbiased, as it is a completely representative subset of the PIACC data, constructed to resemble the OECD population as a whole. That is, this training dataset is free of sample-selection bias: the programmers have example outcomes for training on a truly representative group. This will allow programmers to utilize unbiased training data in their algorithms. Importantly, engineers were told their data was unbiased.

In the second condition, participants received a similar dataset as above. However, this dataset contained realistic sample-selection bias. In particular, programmers received 20,000 observations randomly selected among people who already were in jobs involving Math. Performance is still measured in an unbiased way, as the dataset contains actual math performance for these 20,000 observations. This simulates a real world situation in which a firm has performance outcomes *only* for workers it hired historically, possibly in a biased or unrepresentative way. Engineers in this condition were told the data contained this sample-selection: potentially, they could realize it was a problem and simply adjust their algorithms.

The third condition received the same training data as the second, and exactly the same information, but with the addition of a reminder that the training data is unrepresentative. We report the reminder below,

Note: As a reminder, our recruiters expectations about who to hire may have been systematically wrong. Performance data is available only for workers like the ones weve hired in the past. This may reflect our recruiters stereotypes or biases about who is good at math.

As you write your algorithm, please be mindful that your training dataset may originate in a biased social system. *Adjusting your algorithm to account for discrimination in hiring, self-sorting of applicants, or other sources of such bias could improve your accuracy on the test set. You will be evaluated only on the accuracy of your predictions on the test set.*

Please note: For the workers weve hired in the past, we are confident their performance outcomes in your training dataset have been labeled accurately. Performance is based on math and there are objective measures of quality. However, you only have this data about the candidates we previously hired.

The fourth condition received the same training data and the same reminder that the training data is unrepresentative as the third condition, *as well as* some technical training about avoiding

algorithmic bias from unrepresentative training datasets. In particular, programmers received a few pages of simple technical advice, outlining sample selection theory, and explaining some ways to correct for sample selection bias (e.g., sampling with replacement). This technical advice also contained links to further reading. The additional text, specific to this fourth condition, is reported below.

Here is a white paper containing an overview, some citations and instructions about this kind of adjustment. If the white paper doesnt help, feel free to ignore it. You will be evaluated only in terms of the accuracy of your predictions on the test set.

These four groups were then asked to score a test set of 20,000 new observations. The observations used for evaluation were drawn from a representative distribution, including both those with jobs requiring math and not. Even if programmers were given training data with sample selection issues, they would be evaluated using a sample without selection bias.[3]

We structured our task so that subjects developed their algorithms on their own. To do so, we had to avoid two elements: first, that they looked for external sources, and second, that they cooperated among themselves. The PIAAC dataset is public and we had to avoid they found it and used it to find their answers. Therefore, we substantially obfuscated variable names and descriptions. We also removed about 100 variables, as they are typically not observable at the time of hiring, and we claimed the dataset was coming from a company. In order to avoid cooperation, we claimed programmers had different parts of the problem to address, and we sent them datasets with minor differences, even when they were subjects in the same group. We also created graphically different documents (changing fonts and page breaks) for each engineer.

Programmers in all four conditions were evaluated according to the same standards: Awards were given according to the accuracy of predictions. Programmers could therefore earn greater rewards by contradicting stereotypes when this is justified based on high-quality, representative ground-truth data in the PIACC.[4]

We gave engineers two levels of accuracy they should have aimed at, in order to satisfactorily complete the task. The first accuracy level was the minimum required for a 90 percent task completion, while the second accuracy level was the level required for a 100 percent task completion. We still rewarded engineers with worse accuracy, but we incentivize them to reach these objectives. Additionally, we gave them extra incentives to improve on their objective: if they achieved better accuracy scores than the second (best) accuracy level, they obtained extra points, which worked as a stronger signal of their quality.

We used these accuracy thresholds to experimentally test the effects of different incentives. In fact, we might think that having thresholds that are more difficult to reach has an impact on how well engineers address sample selection: more rewarding incentives could convince them to exert extra effort and thus help them overcome issues coming from biased data. In order to test this

[3]We feel that this is how many real-world applications work: Models trained using unrepresentative datasets (sometimes "datasets of convenience") are often deployed in practice on a larger population. For example, the hiring algorithm in Hoffman et al. (2017) was based on training data on employees only, but was applied to make assessments about all job applicants, including ones very different than the firms' historical set of employees. Cowgill (2017) contains a theory model about when and why this may (or may not) create problems.

[4]We prefer *not* to structure the task as a contest because of the risk-seeking behavior that contests and tournaments may encourage (Rosen, 1988). Instead, we prefer to rewarding coders per unit of accuracy.

hypothesis, we randomly assigned one of two threshold levels within all four experimental groups: one was a higher threshold, and the other a lower one. We made no distinction between the two threshold levels: each engineers either had both thresholds higher or lower. In theory, those with a threshold that was more difficult to reach should have exerted extra effort. To further test the effect of incentives, we randomized how easy was to obtain extra bonuses, once the more accurate threshold was passed. One group had an easier time achieving the bonuses, and for the other was harder. This randomization was independent from the one of thresholds, and they were both independent from the four main group assignments. That is, within each one of the four groups, there were four sub-groups: some programmers had lower thresholds to reach, and an easier time obtaining bonuses; some had a lower threshold, and more difficult bonuses to obtain; some had a higher threshold and easy bonuses; some had a higher threshold and difficult bonuses. These randomizations did not affect any other part of the experiment.

In order to complement our analysis with a more in-depth understanding of our subjects, and the way they worked on their algorithms, we developed a survey, which engineers received after they submitted their work. Although we clarified that the content of their responses did not matter, programmers had to answer this survey to complete their task. We used the survey to collect some technical information on how they built their algorithms (e.g., their functional form, how they selected their features, what libraries they used), as well as some behavioral dimension (e.g., what accuracy they expected on the test set, how many hours they worked on the task).

We dedicated much of the survey to understand how engineers dealt with the issue of sample selection in the data. We asked questions to understand whether they were concerned their data might have been biased against a particular group, whether they implemented a sample-selection correction strategy, and, in case, what type of strategy. In case they tried to favor a certain group in their algorithm, we asked them why they did so - whether they were concerned with more social considerations (e.g., they want to help certain groups overcome historical adversity), or they were only trying to maximize their algorithms accuracy.

Finally, we explored psychological dimensions related to bias. In fact, individuals have different levels of bias, and for the biased programmers hypothesis it is central that we understand whether programmers are biased against women, and by how much. To get at this dimension, we developed a standard Implicit Association Test (Carpenter, 2018). This is a measure in social psychology aimed at detecting how strong are automatic associations (Greenwald, 1998): it measures peoples associations, rather than their beliefs. It is a relative measure, as it compares pairs of associations and tests which pairs are stronger in subjects minds. In particular, we studied the association between Math and Men, versus Math and Women, and we used English as the subject in opposition to Math. So, engineers had to associate Math/Women vs English/Men and also Math/Men vs English/Women. If the engineers associated the pairs Math/Men and English/Women in a stronger way than the opposite ones, we could claim there is a potential bias against women in math. The IAT gives us individual-level bias measures: every engineer was tested, and for all of them we have a measure of the strength of their associations.

References

Ajunwa, Ifeoma, Sorelle Friedler, Carlos E Scheidegger, and Suresh Venkatasubramanian, "Hiring by algorithm: predicting and preventing disparate impact," 2016.

Angwin, Julia, Jeff Larson, Surya Mattu, and Lauren Kirchner, "Machine bias: There's software used across the country to predict future criminals. and it's biased against blacks," *ProPublica, May*, 2016, *23*.

Cowgill, Bo, "Automating Judgement and Decisionmaking: Theory and Evidence from Résumé Screening," *Working paper*, 2017.

Datta, Amit, Michael Carl Tschantz, and Anupam Datta, "Automated experiments on ad privacy settings," *Proceedings on Privacy Enhancing Technologies*, 2015, *2015* (1), 92–112.

Hoffman, Mitchell, Lisa B Kahn, and Danielle Li, "Discretion in hiring," *The Quarterly Journal of Economics*, 2017, *133* (2), 765–800.

Lambrecht, Anja and Catherine E Tucker, "Algorithmic Bias? An Empirical Study into Apparent Gender-Based Discrimination in the Display of STEM Career Ads," 2016.

McGhee, Debbie E.; Schwartz Jordan L.K. Greenwald Anthony G.;, "Measuring Individual Differences in Implicit Cognition: The Implicit Association Test," *ournal of Personality and Social Psychology*, 1998, *74* (6), 14641480.

Mullainathan, Sendhil and Ziad Obermeyer, "Does Machine Learning Automate Moral Hazard and Error?," *American Economic Review*, 2017, *107* (5), 476–480.

Rosen, Sherwin, "Promotions, elections and other contests," *Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für die gesamte Staatswissenschaft*, 1988, *144* (1), 73–90.

T., Pogacar R. Pullig C. Kouril M. Aguilar S. LaBouff J. P. Chakroff A. Carpenter, "Conducting IAT Research within Online Surveys: A Procedure, Validation, and Open Source Tool," *PsyArXiv*, 2018.

Tan, Sarah, Rich Caruana, Giles Hooker, and Yin Lou, "Detecting Bias in Black-Box Models Using Transparent Model Distillation," *arXiv preprint arXiv:1710.06169*, 2017.

Tables and Figures

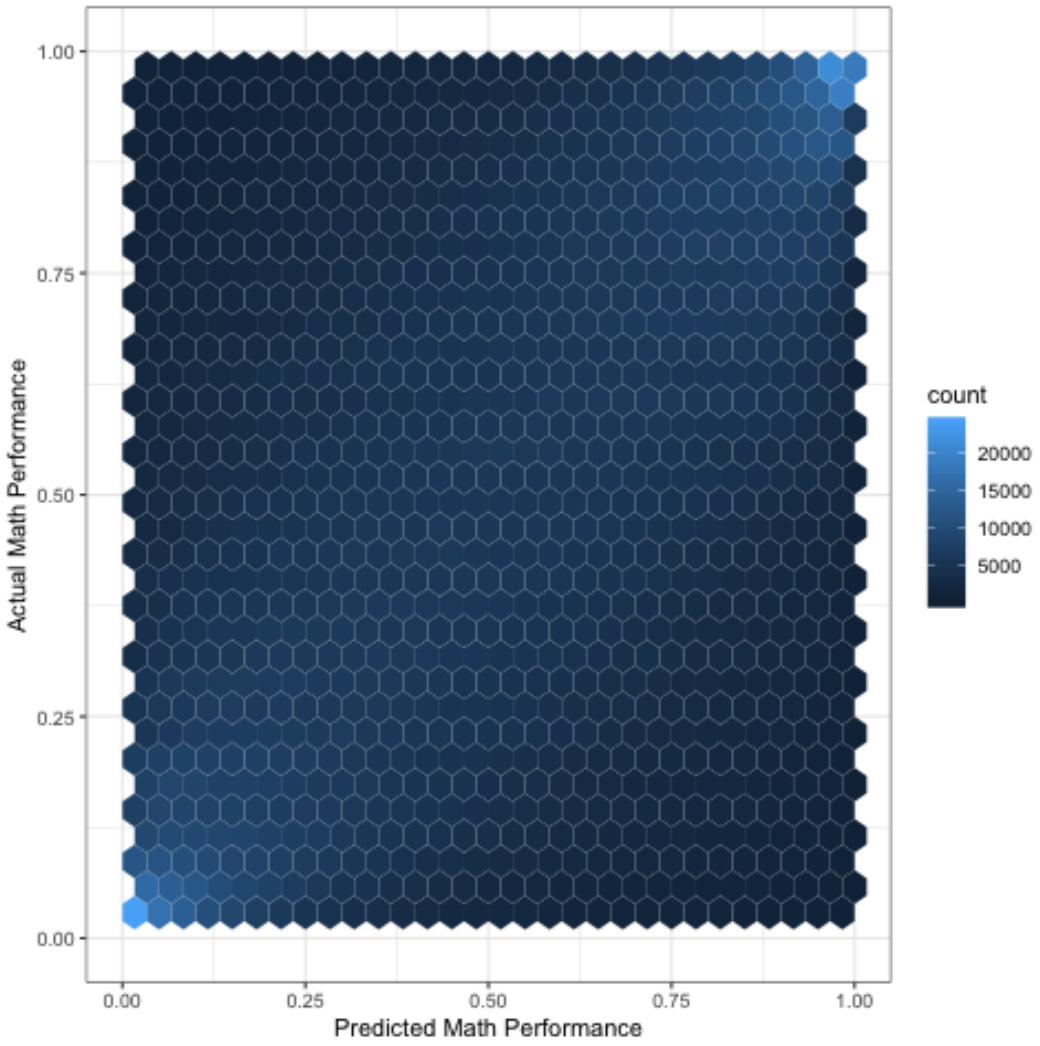


Figure 1: Kernel Density Plot: Predicted and Actual Math Scores

Table 1: **Characteristics of Engineers (Subjects)**

	Mean	SD
White	0.204	0.404
Asian	0.612	0.488
Indian	0.138	0.346
Male	0.681	0.467
Undergraduate	0.477	0.500
CS Major	0.508	0.501
Completed Task	0.904	0.295
Completed Survey	0.904	0.295
Previuos Assessment Score	0.874	0.087
Observations	260	

Table 2: **What Effects Level of Algorithmic Bias**

	Bias	Bias
unbiased_data	-108002.9***	-97757.6**
	(37065.9)	(38250.8)
reminder	-70668.3*	-66752.2*
	(36598.1)	(38812.3)
techadvice	90840.4***	89796.1***
	(34342.5)	(33089.1)
implemented_ssc	-26029.8	-31792.9
	(42327.3)	(41029.5)
male		-61348.5**
		(30120.0)
asian		-11267.0
		(42199.8)
indian		-11381.8
		(57096.4)
csmajor		41769.3
		(26091.6)
undergrad		25394.8
		(31285.4)
Observations	4541441	4541441