Analyzing Pre-Existing Knowledge and Performance in a Programming MOOC

Hannah Burd
Dartmouth College
Hanover, NH
hannah.j.burd.22@dartmouth.edu

Erik Hemberg
MIT CSAIL
Cambridge, MA
hembergerik@csail.mit.edu

Ana Bell
MIT
Cambridge, MA
anabell@mit.edu

Una-May O'Reilly
MIT CSAIL
Cambridge, MA
unamay@csail.mit.edu

ABSTRACT

Massive Open Online Courses (MOOCs) are accessible to anyone with a device that can connect to the internet. MOOCs aim to increase the accessibility of higher-level knowledge and skills, such as programming. To understand how students are performing and struggling in the course, we investigate a popular MITx MOOC that teaches introductory programming. We look at problem set questions and examine students with different levels of pre-existing knowledge. Specifically, we study the number of attempts of each group per question and the mean final accuracy of each group per question. We find that for nearly all questions, students with no programming experience struggle more than students with prior programming experience. Moreover, we observe a potential turning point in the course where students of all experience levels begin to struggle. Our findings both show that two groups of MOOC students perform differently and inform question design in MOOCs by demonstrating which question types are particularly arduous.

Author Keywords

Student background; problem sets; question design; learning analytics; edX; MOOCs.

CCS Concepts

•**Applied computing** → **E-learning;** •**Human-centered computing** → **User studies;** •**Social and professional topics** → *Computational thinking;*

INTRODUCTION

Massive Open Online Courses (MOOCs) make higher-level knowledge accessible to those outside of traditional classrooms; however, they also face several new challenges that are seldom present in a traditional setting. For example, MOOCs typically experience extremely high drop out rates [5], lack the physical presence of a professor, and are composed of a heterogeneous group of students with varying language proficiency, study time, pre-existing programming experience, goals, etc.[3]. The COVID-19 pandemic and the global shift to online classrooms has further demonstrated the need for and potential of effective remote learning environments. Therefore, it is vital that we continue to study existing e-learning courses, like MOOCs, to improve remote learning for all.

It is intuitive that the group of students who take the course and have some level of programming experience will generally perform better in the course than the group with no experience. However, this difference becomes concerning when one group significantly and consistently outperforms the other, as this may suggest a group of learners is not being supported.

By observing how each group (with/without programming experience) performs on tasks such as problem sets, we can analyze the success patterns of each group outside of a single grade. Moreover, if the discrepancy between groups only exists for certain questions on that problem set, analysis may indicate a correctable flaw in question design that is causing divergence among groups.

Related Work

Linear regression models have shown that number of distinct attempts per question is correlated with problem difficulty [2]. Previous work that examined student programming background suggests that the importance of practice in positive learning outcomes is influenced by question topic and student pre-existing knowledge [4].Furthermore, no models have definitively shown that pre-existing knowledge effects the method one uses to answer a question [1].

Research Questions

Do students with no pre-existing knowledge about programming struggle significantly more than students that have any level of pre-existing knowledge? Furthermore, are there identifiable points in the course where one or both groups begin

L@S '20, August 12–14, 2020, Virtual Event, USA.

ACM ISBN 978-1-4503-7951-9/20/08 ...$15.00.
http://dx.doi.org/10.1145/3386527.3406728

Experience	*Count*	*Percentage of Total*
No Response	7,882	11.66
Absolutely None	16,374	24.21
Other Language	30,552	45.18
Know Python	11,336	16.76
Veteran	1,483	2.19
Total	67,627	100.00

Table 1. Number of submission histories in 6.00.1x with each type of pre-existing experience. We also show the per-category percentage of all submission histories. Numbers shown are for all 21 questions and therefore include duplicate users.

Experience	*Count*	*Percentage of Total*
No Response	408	11.72
Absolutely None	853	24.50
Other Language	1,568	45.04
Know Python	577	16.58
Veteran	75	2.16
Total	3,481	100.00

Table 2. Number of unique users in 6.00.1x with each type of pre-existing experience. We also show the per-category percentage of all users.

to struggle more? We hypothesize that students with no pre-existing knowledge about programming do struggle more than students with experience. We quantify student performance through their distinct attempts and final accuracy for analysis.

COURSE CONTEXT AND DATASET

Our data comes from 2016 Term 2 and 2017 Term 1 of MIT's 6.00.1x *Introduction to Computer Science and Programming Using Python* on edX. In this course, students learn to code through video lectures, short finger exercises, discussion threads, exams, and open-ended problem set questions at the end of each unit. In this session of 6.00.1x, there are 21 distinct problem set questions, described in this paper with unit and problem number (i.e. 1-1 is Unit 1 Problem 1).

We use data from a pre-course survey in which students self-identify experience level as well as student submission history data collected by the edX platform (Table 1 and Table 2). The pre-existing experience categories are *No Response, Absolutely None, Other Language, Know Python* or *Veteran*. We proceed with a data analysis that omits the *No Response* group and joins the *Other Language, Know Python, and Veteran* groups into a single *Know Programming* category. After applying these changes, there are 59,745 total question submission histories and 3,073 unique students in the data set.

METHODS

Distinct Attempts

We examine the number of distinct attempts per question per student with a pre-existing experience type. We run a one-sided Mann-Whitney U test for each question to check if students with *Know Programming* experience use fewer attempts per question than those with *Absolutely None* experience. We also visualize the number of distinct attempts for all questions for all pre-existing experience types (Figure 1).

Final Accuracy

We examine the final accuracy per question per student with a pre-existing experience type. Final accuracy is 1 or 0, where 1 means the student ultimately got the question correct and 0 means the student did not get the answer correct. We run a one-sided Mann-Whitney U test for each question to check if students with *Know Programming* experience are more likely to arrive at a correct answer than those with *Absolutely None* experience. We also visualize the final accuracy values for all questions for all pre-existing experience types (Figure 2). There are 59,745 total question submission histories and 3,073 unique students in the data set (N *Know Programming* = 2220, N *Absolutely None* = 853.

OBSERVATIONS AND ANALYSIS

We visualize student performance with comparative plots in terms of average number of distinct attempts (Figure 1) and average final accuracy (Figure 2).

The figures show that students with programming experience generally use fewer attempts on a question and have a higher mean accuracy than those with no programming experience. Initially, mean attempts and mean accuracy are relatively consistent; however, as the course progresses, the performance of each group becomes more variable. The high standard deviations are a result of both extreme outliers and variability in individual students. Additionally, there appears to be a potential turning point in the course at 4-5, as nearly all mean number of attempts are greater than the cumulative mean and nearly all final accuracy means are below the cumulative mean. However, 1-3, 2-1, and 3-4 also appear to either follow this pattern or have a large difference in mean across groups.

Statistical analysis of distinct attempts and final accuracy reveals that students with pre-existing experience do outperform students without experience in number of distinct attempts and/or final accuracy for most questions. In Table 3, raw p-values are shown. After the turning point in the course, the p-values are more likely to show a statistically significant difference among groups (after a post hoc correction), supporting the notion that the turning point also indicates a point where the discrepancy in the performance of groups becomes more distinct. An additional Mann Whitney U test revealed that there is also an overall significant difference between distinct attempts and final accuracy for each experience type.

DISCUSSION

Figure 1 and Figure 2 support the hypothesis that those with pre-existing programming experience generally outperform those with no pre-existing experience in regard to number of distinct attempts and final accuracy. However, there are high standard deviations, potentially limiting major conclusions.

The figures also suggest that for questions that appear more difficult than average (as determined by number of attempts and whether or not the student eventually arrives at a correct answer), it is more likely that the difference in performance between students with varying experience will be greater than questions that are less difficult than average. In other words, after the turning point of the course (and with challenging questions like 3-4), students with no pre-existing experience

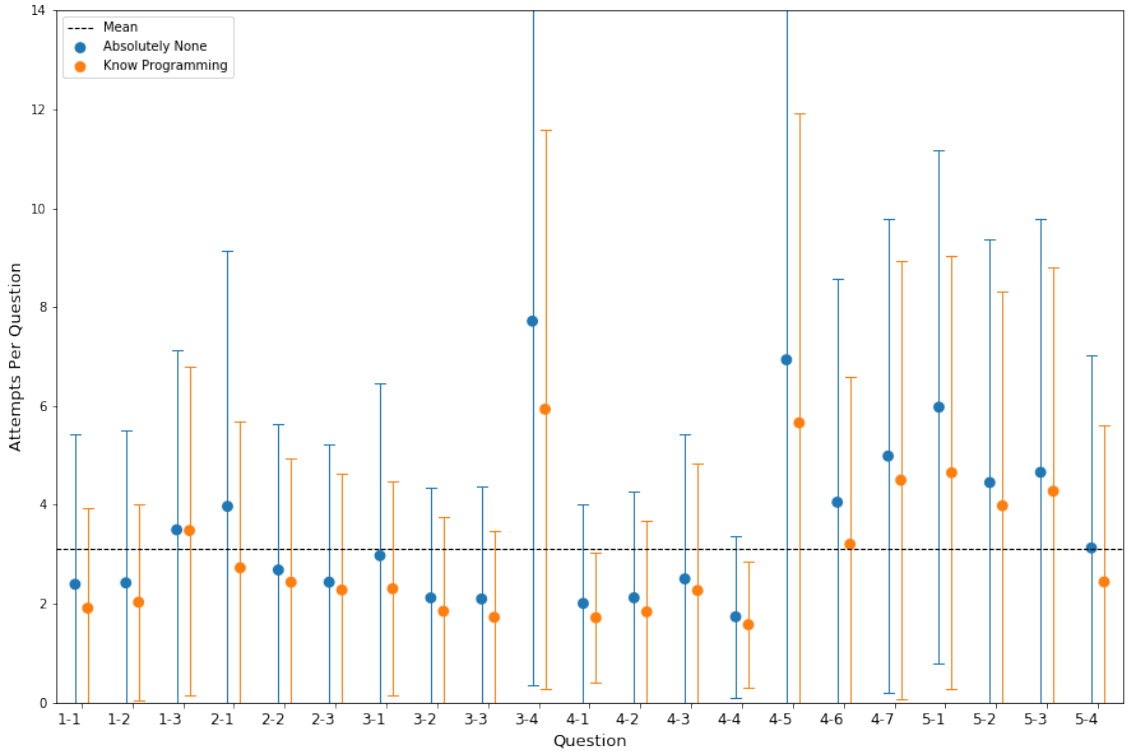

Figure 1. Mean number of attempts per question (mean = 3.1237, n = 3,073). The error bars show +/- one standard deviation.

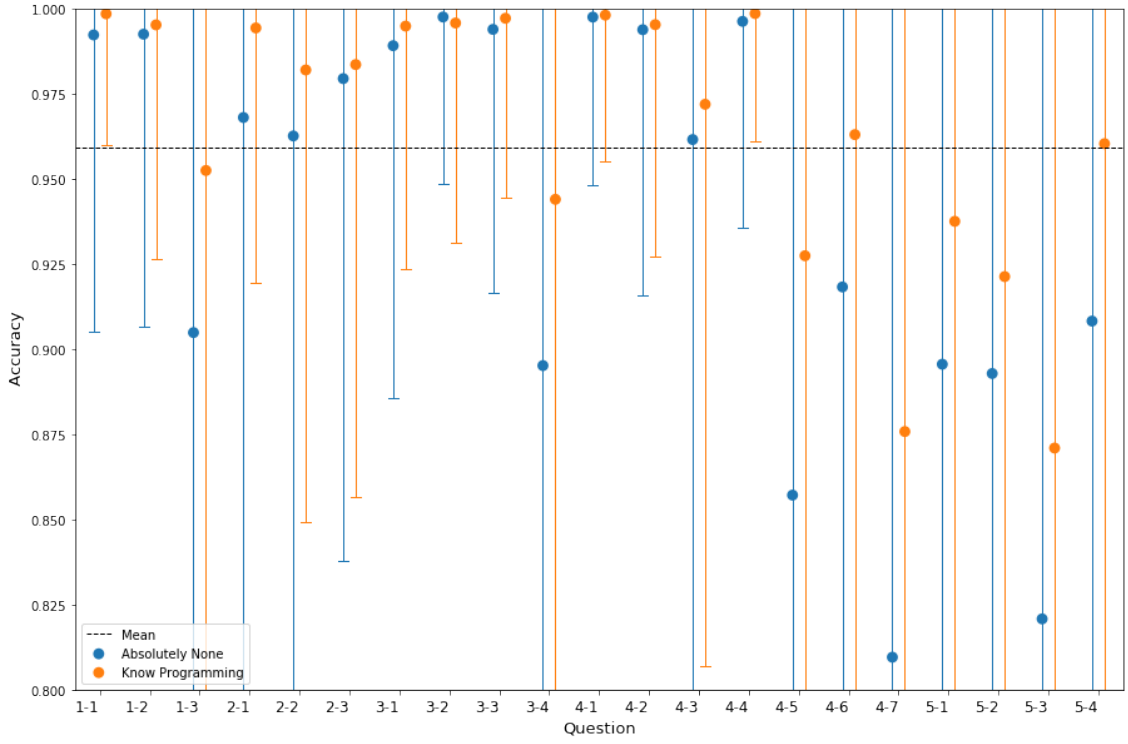

Figure 2. Mean final accuracy per question (mean = 0.9594, n = 3,073). The error bars show +/- one standard deviation.

Question	Distinct Attempts	Final Accuracy
1-1	0.011	0.004
1-2	0.037	0.186
1-3	0.909	<<0.001
2-1	<<0.001	<<0.001
2-2	0.024	<0.001
2-3	0.273	0.227
3-1	<<0.001	0.042
3-2	<<0.001	0.762
3-3	<<0.001	0.094
3-4	<<0.001	<<0.001
4-1	<0.001	0.376
4-2	<<0.001	0.306
4-3	0.020	0.074
4-4	0.030	0.109
4-5	<<0.001	<<0.001
4-6	<<0.001	<<0.001
4-7	0.004	<<0.001
5-1	<<0.001	<<0.001
5-2	0.119	0.011
5-3	0.127	<<0.001
5-4	<<0.001	<<0.001

Table 3. Raw p-values comparing number of distinct attempts per question or final accuracy per question for the *Absolutely None* and *Know Programming* groups. The black bar indicates the potential turning point in the course suggested by Figure 1 and Figure 2. A darker highlight indicates a significant value after a post hoc Bonferroni correction (alpha = 0.0024), a lighter highlight indicates a significant value without a post hoc correction (alpha = 0.05), and a lack of highlight indicates a statistically insignificant p-value.

were more likely to invest more attempts and give up/not arrive at a correct answer on the question than those with programming experience.

Looking at some of the questions that appear more difficult than average, a few patterns begin to emerge. These questions (3-4, 4-5 through 4-7, and Unit 5) involve taking multiple smaller pieces of code and putting them together to make one longer piece of code (often a game) that meets several specific criteria. In Unit 5, the student does not have to get the previous piece of code correct to be able to proceed with the questions. These are the only problem set questions that have this structure.

The one question that has neither a significant difference in attempts nor a significant difference in final accuracy among groups for any alpha is 2-3, "Using Bisection Search to Make the Program Faster". As bisection is a topic that requires both computational thinking and knowledge of syntax, substantial time and effort was put into the design of this question and teaching of the topic. There is a lecture video and a finger exercise dedicated to the topic, the question is heavily detailed, and the question even contains test cases for the student. This may explain how, despite the question containing some similar characteristics to the more difficult than average questions, it has a lower than average mean distinct attempts and higher than average final accuracy among both groups and there is no difference in the performance of the two groups.

NEXT STEPS

Our observations support the notion that those without pre-existing programming experience generally struggle more than those with pre-existing programming knowledge. This is unsurprising, and while it is unlikely that an *Absolutely None* group and *Know Programming* group will ever consistently perform equally, efforts should be made in learning design to close the gap. This means aiming for improved performance in the *Absolutely None* group, not expecting those with experience to decrease performance. By adequately taking into account prior student knowledge, a course can be better designed to support students at an appropriate pace.

In the future, it is important to experimentally explore how question design and teaching could be improved to better educate those without pre-existing programming experience. Moreover, considerations should be given to question design that will help all students connect several small coding concepts to one larger project.

REFERENCES

[1] Ayesha Bajwa, Ana Bell, Erik Hemberg, and Una-May OReilly. 2019. Student Code Trajectories in an Introductory Programming MOOC. In *L@S '19*. ACM, Chicago, IL, USA. https://dl.acm.org/doi/10.1145/3330430.3333646

[2] Sagar Biswas, Erik Hemberg, Nancy Law, and Una-May OReilly. 2019. Investigating Learning Design Categorization and Learning Behaviour in Computational MOOCS. In *Sixth ACM Conference on Learning @ Scale*. ACM, Chicago, IL, USA. https://doi.org/10.1145/3330430.3333664

[3] Jennifer DeBoer, Glenda Stump, Lori Breslow, and Daniel Seaton. 2013. Diversity in MOOC Students' Backgrounds and Behaviors in Relationship to Performance in 6.002x.

[4] Jitesh Maiyuran, Ayesha Bajwa, Ana Bell, Erik Hemberg, and Una-May OReilly. 2019. How Student Background and Topic Impact the Doer Effect in Computational Thinking MOOCs. In *2019 IEEE Learning With MOOCS*. 47–52. DOI: http://dx.doi.org/10.1109/LWMOOCS47620.2019.8939643

[5] Justin Reich and José A Ruipérez-Valiente. 2019. The MOOC pivot. 363 (01 2019), 130–131. DOI: http://dx.doi.org/10.1126/science.aav7958