Comparing Ebook Student Interactions With Test Scores: A Case Study Using CSAwesome

Hisamitsu Maeda, Barbara Ericson, Paramveer Dhillon
University of Michigan, Ann Arbor, MI, USA
{himaeda|barbarer|dhillonp}@umich.edu

ABSTRACT

Interactive ebooks have better learning gains than static ebooks, students prefer them, and more computing courses are using interactive ebooks. The number of computing ebooks on the open-source Runestone interactive ebook platform increased from one in 2011 to over 30 in 2020. Runestone currently serves over two million page views a week. It includes books for CS0, CS1, CS2, data science, and web development and supports coding in Java, Python, and C++. The ebooks include instructional material: text, images, videos, and interactive practice problems with immediate feedback: multiple-choice, fill-in-the-blank, write code (active code), mixed-up code (Parsons), clickable code, and matching. User interaction with the ebooks is timestamped and logged. This information includes page views, video plays, video completion, Parsons moves, problem answers, and learner written code. This fine grained data may help us automatically identify struggling students. This paper reports on several analyses comparing student activities to the midterm score from one of the Runestone ebooks, CSAwesome. Specifically, we compared the major types of log file entries to the midterm score and also conducted an in-depth analysis of mixed-up code (Parsons) problem data.

Keywords

e-book, data mining, Parsons problem

1. INTRODUCTION

Research has shown that interactive ebooks have better learning gains than static ebooks [20]. In addition, most students report that the interactive features help them learn and want to use interactive ebooks in future courses [15]. A 2013 working group predicted that traditional CS textbooks would be replaced by interactive ebooks [13].

Runestone is an open-source ebook platform that has grown from serving one interactive ebook [15] in 2011 to over 30 free ebooks in 2021. It supports several languages, including Python, Java, C++, and SQL [10]. During the 2020-21 academic year, Runestone had 69,400 registered users and served an average of over two million page views a weeks.

Runestone interactive ebooks log timestamped user interactions[1]. For the analyses in this paper, we use the **CSAwesome** interactive ebook. CSAwesome has been endorsed by the College Board for the Advanced Placement (AP) Computer Science A (CSA) course [4]. Advanced Placement courses are taken by secondary students for college credit and/or placement. The AP CSA course is equivalent to a first course for majors (CS1) at the college level and covers object-oriented programming in Java. Our CSAwesome data is from custom courses. Instructors can create a custom course for any of the free ebooks on Runestone and have their students register for that custom course.

Interactive ebooks provide rich data that could be leveraged to improve instruction [23]. We may be able to identify struggling students in order to provide help. Our research questions were 1) what student activities correlate with scores on the midterm and 2) what Parsons problem activities correlate with the midterm score? To answer these questions, we performed regression analyses at both the higher level based on the major types of activities and at the lower level with a more in-depth analysis of Parsons problem data.

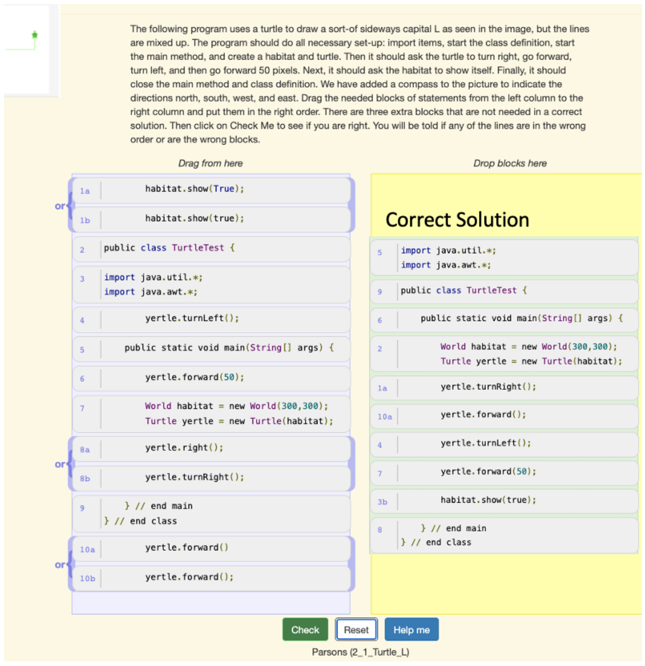
2. RELATED WORK

We are interested in the relationship between students' activities in the ebook and their pretest and midterm scores. In a related study, Pollari-Malmi et al. [20] found an increase in use, motivation, and learning gains from use of an interactive ebook versus an equivalent static ebook. Ericson et al. [6] found that teachers who used more of the interactive features in an ebook had higher gains in confidence and higher scores on the final posttest. Parker et al. [18] found that students and teachers used an interactive ebook differently with teachers showing more characteristics of expert learners. Akçapınar et al. [1] used student reading behavior in an ebook to predict at-risk students. Park et al. [17] used statistical change detection techniques to detect changes in student behavior from clickstream data in a Learning Management System (LMS). They found that students who increased their reviewing activity relative to other students in the course had a higher probability of passing the course.

[1]Researchers can request an anonymized logfile from Brad Miller, the founder of Runestone.

Several researchers [3, 24, 21, 22, 9, 19, 2] have been exploring Parsons problems as both an alternative to writing code from scratch and as a summative assessment. In Parsons problems learners must place mixed-up blocks of code in the correct order. They may also have to correctly indent the blocks. Parsons problems can also have distractors, which are code blocks that are not needed in a correct solution. Helminen et al. [12] visualized students' problem-solving process on Parsons problems using a graphical representation. They detected several different approaches to solving Parsons problems including top-down, control structures first, and trial and error. Some learners got stuck in circular loops and repeated the same incorrect solution. Ericson et al. [8] found that more learners attempted Parsons problems than nearby multiple-choice questions in an interactive ebook. Maharjan et al. [14] proposed an edit distance trail to show students' solution paths in Parsons problems in a simpler fashion than a graphical representation. They also discussed potential issues with studying Parsons problems using descriptive statistics methods. Morrison et al. [16] found that subgoal labels help students solve Parsons problems. Du et al. [3] conducted a review of recent studies on Parsons problems.

Ericson invented two types of adaptation for Parsons problems [7, 5]. In intra-problem adaptation the problem can be dynamically made easier by removing a distractor, providing indentation, or combining two blocks into one. The adaptation is triggered by clicking a "Help" button. In inter-problem adaptation the difficulty of the next problem is automatically modified based on the learner's performance on the last problem. The problem can be made easier by pairing distractor and correct code blocks or by removing distractors. It can be made more difficult by using all distractors and randomly mixing the distractor blocks in with the correct code blocks. Learners are nearly twice as likely to correctly solve adaptive Parsons problems than non-adaptive ones and report that solving Parsons problems helps them learn to fix and write code [5]. A randomized controlled study provided evidence that solving adaptive Parsons problems takes significantly less time than writing the equivalent code and with similar learning gains from pretest to posttest [7].



The following program uses a turtle to draw a sort-of sideways capital L as seen in the image, but the lines are mixed up. The program should do all necessary set-up: import items, start the class definition, start the main method, and create a habitat and turtle. Then it should ask the turtle to turn right, go forward, turn left, and then go forward 50 pixels. Next, it should ask the habitat to show itself. Finally, it should close the main method and class definition. We have added a compass to the picture to indicate the directions north, south, west, and east. Drag the needed blocks of statements from the left column to the right column and put them in the right order. There are three extra blocks that are not needed in a correct solution. Then click on Check Me to see if you are right. You will be told if any of the lines are in the wrong order or are the wrong blocks.

Drag from here

```
1a          habitat.show(True);
or
1b          habitat.show(true);

2   public class TurtleTest {

3   import java.util.*;
    import java.awt.*;

4           yertle.turnLeft();

5       public static void main(String[] args) {

6           yertle.forward(50);

7           World habitat = new World(300,300);
            Turtle yertle = new Turtle(habitat);

8a          yertle.right();
or
8b          yertle.turnRight();

9       } // end main
    } // end class

10a         yertle.forward()
or
10b         yertle.forward();
```

Drop blocks here

Correct Solution

```
5   import java.util.*;
    import java.awt.*;

9   public class TurtleTest {

6       public static void main(String[] args) {

2           World habitat = new World(300,300);
            Turtle yertle = new Turtle(habitat);

1a          yertle.turnRight();

10a         yertle.forward();

4           yertle.turnLeft();

7           yertle.forward(50);

3b          habitat.show(true);

8       } // end main
    } // end class
```

Figure 1: Example Parsons Problem with Paired Distractors

3. DATASET

The log file used in our analysis comes from the CSAwesome interactive ebook on the open-source Runestone platform. This book was revised by Beryl Hoffman of Elms College and the Mobile CSP project in 2019 for the 2019 AP CS A exam [4]. The log includes page views, video interaction, and the results from interactive practice problems: multiple-choice, write code (active code), and mixed-up code (Parsons problems).

An active code is a traditional programming exercise in which students write/edit/run code in the ebook. Many active code exercises have unit tests that students can use to verify that their code is correct. Students can also use a slider to view previous versions of the code. The "Show CodeLens" button on the active code will display a program visualizer (CodeLens), allowing students to step through their code line by line and visualize the variables. It is a version of Python Tutor [11]. A Parsons problem provides mixed-up code blocks that the learner must place in the correct order [19] as shown in Figure 1. The Parsons problems in this ebook were adaptive. While some adaptive systems use selection adaptation in which the next problem is selected from a set of possible problems based on the learner's performance, this system modifies the difficulty of the current or next problem in the ebook based on the learner's performance.

We analyzed five categories of log file entries: page views, video (play, pause, and completion), active code interaction (run, edit, slide, and unit test), Parsons problem block moves and answers, and answers to multiple-choice questions. Overall, the log file contained data from 1,893 students in 57 custom classes. Most of these were high school classes, but some of them were college classes. This paper analyzed a subset of the log data from 505 students who took both the pretest and midterm in their course. The summary statistics of the data are shown in Table 1. Over 37% of the log file activities were interaction with active code. This included running the code, editing the code, sliding the history to view different versions of the code, and running unit tests. The AP CSA exam includes 40 multiple-choice questions and four free response questions where the student must write Java code to solve a problem. The ebook is broken into 10 content-based units and five practice units. At the end of each content-based unit there are at least 10 multiple-choice questions, mixed-up code (Parsons) problems, and write code problems [4].

We were also interested in the effect of class size on students' performance. For this analysis, we divided the students into two groups: classes with less than 30 students, which is the typical maximum class size in high school, and classes with more than 30 students—224 students were in large classes, and 281 were in small classes.

Event Type	Count	Percentage
active code	1,020,735	37.66%
page view	651,136	24.02%
Parsons move and answer	524,206	19.34%
multiple choice answer	261,321	9.64%
video	83,892	3.09%
other	169,363	6.25%
Total	2,710,653	100%

Table 1: Various types of events in our dataset.

4. METHODOLOGY

4.1 Regression analysis of student activities

In this section, we analyze the impact of student activities on student performance. First, we explore the data and check the relationship between different student activities. Figure 2 shows that the percentage correct on the midterm has a positive correlation with the percentage of correct for each activity. Multiple-choice problems have the highest correlation with the midterm. This could be because the midterm is a set of 20 multiple choice questions. On the other hand, there is a negative correlation between the midterm and the number of some other activities (i.e. the number of times that CodeLens was used and the percent of videos that were completed).

Next, we employ a linear regression model to this data with the percentage of correct answers on the midterm exam as the dependent variable. The count and type of student activities were used as independent variables. Since the activity variables have a skewed distribution, we log-transformed them as $x \rightarrow log(x+1)$ and standardized them before running the regression analysis. Also, since there is a difference in the test results based on the class size (as can be seen from Figure 3), we add the class size as a dummy variable in our regression model. The regression results are shown in Table 2.

The percent correct on the midterm was negatively correlated with being in a larger class, perhaps because there is less one-on-one interaction with the instructor and, as a result, lower learning outcomes. Midterm results were also negatively correlated with the number of interactions with the CodeLens and the number of videos completed. This may indicate that both of these activities were more likely to be used by struggling students. Midterm results were positively correlated with the percent correct on the pretest, percentage correct on other multiple-choice questions, the number of page views, and the number of videos played. It is interesting that the midterm score is positively correlated with the number of videos played, but negatively correlated with the number of videos completed. It could be that stronger students watch a video till they find what they need and then quit.

4.2 Regression Analysis of Parsons Problems

In this section, we conduct an in-depth analysis of Parsons problems. As described earlier, these Parsons problems used both intra-problem and inter-problem adaptation. In intra-problem adaptation, if a student submits at least three incorrect solutions, they are notified that they can use a "Help" button to make the problem easier. Each time the student

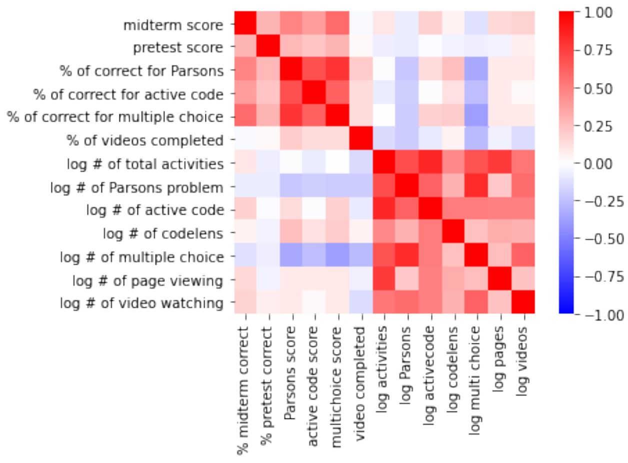


Figure 2: Correlation structure between the different activities.

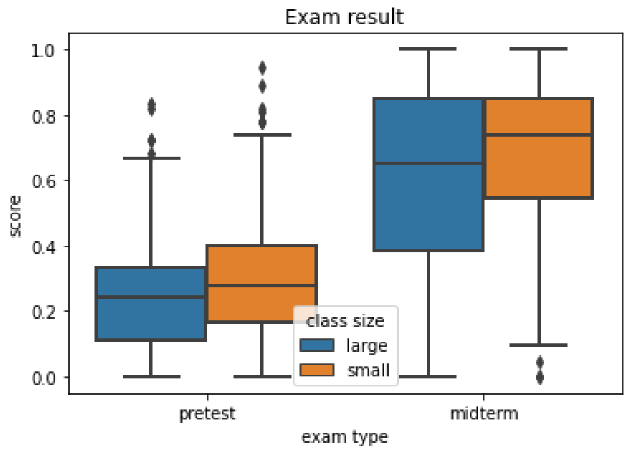


Figure 3: Figure showing results for the pretest and midterm exams.

clicks the "Help" button, the ebook will remove a distractor block from the solution, provide the indentation, or combine two blocks into one, hence providing an implicit hint.

First, we pre-process the log data to gather detailed information on the Parsons interactions. As shown in Figure 4, students have to move from a state where all the blocks are jumbled to the state in which all the blocks are placed correctly. The final state is the correct solution. We count each step taken by the students and the number of failures incurred until a student finds the correct ordering. We count the number of times a student got help (clicked the "Help" button). We also count the number of steps and time until a student asked for help. In order to tease apart the effect of "getting help," we add an interaction term with the "help flag" in our regression.

The regression result is shown in Table 3. As can be seen from the result, being in a large class is negatively related to the midterm score, as we found in our previous regression analysis. We also found that the number of steps before a

Variable	Coefficient
Large Class (or not)	-0.362***
	(0.00)
Percentage correct for pretest	0.1045***
	(0.009)
Percentage correct for multiple choice	0.5366***
	(0.000)
Number of active code interactions	0.0901*
	(0.09)
Number of CodeLens interactions	-0.1403***
	(0.002)
Number of page views	0.0931**
	(0.03)
Number of videos played	0.158***
	(0.005)
Number of videos completed	-0.08**
	(0.03)
N	417
R^2	0.416
$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$	

Table 2: Regression result: The outcome variable is the midterm score and the independent variables are the various student activities. Since the activity variables have a skewed distribution, we log-transformed them as $x \to \log(x + 1)$ and standardized them before running the regression analysis. p-values are shown in parenthesis.

Variable	Coefficient
Large Class (or not)	-0.0862***
	(0.000)
Parsons problem correct	0.0797*
	(0.09)
Number of incorrect submissions	-0.0136***
	(0.000)
Number of times help is used	-0.0279**
(when the help is used)	(0.03)
Number of steps before getting help	0.413*
(when the help is used)	(0.07)
Elapsed time before getting help	-0.3472*
(when the help is used)	(0.08)
N	402
R^2	0.141

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Table 3: Regression analysis of Parsons problems. The dependent variable is the midterm score and the independent variables are the various student activities pertaining to Parsons problems. p-values are shown in parenthesis.

student got help from the software was positively correlated with the midterm test results. This could be because the students who received help from the software after performing more correct steps were more motivated to succeed in the course in the first place. In other words, stronger student learners could figure out more of the problem before they asked for help. On the other hand, the time taken to get support was negatively associated with the test score. This implies that students who took too long to get support scored poorly on the midterm. In addition, students who received more support, i.e., received more help, did not score as well on the midterm.

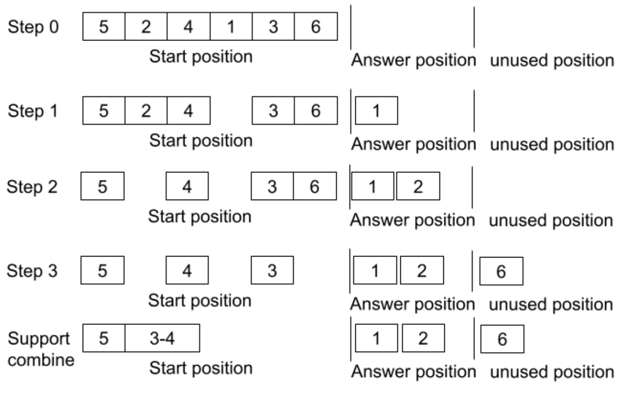


Figure 4: The process of solving the Parsons problem

Next, for a particular problem that the learners solved, drawing a sideways L with a turtle in Unit 2 as shown in Figure1, we also analyzed the number of block moves and the time elapsed before the learner found the correct answer for each problem. As discussed earlier, we counted the number of steps it took from the initial state to the correct order. We define "extra steps" as the number of code-block moves beyond the required number of moves to a correct answer. For example, if a solution can be reached in just 10 steps, and a student took 12 steps, this would be two extra steps. Additionally, we also measured the amount of time it took students to correct the jumbled code blocks in the Parsons problem. Figure 5 shows the number of extra steps taken as a function of the time taken for students in both large and small classes.

Figure 6 compares the extra steps taken as a function of the class size and the midterm test score. As can be seen, there is a significant relationship between students with good midterm test scores and the number of extra steps taken (t-statistic 5.082, p < 0.001). On the other hand, there is no relationship between class size and the number of extra steps taken (t statistic 1.151, p > 0.1).

Figure 7 shows the result comparing the time by class size and midterm test score. Unlike the number of "extra steps," there is no significant association between students with good midterm test scores and the time taken by the learners (t statistic 1.7843, p < 0.07). Also, there is no relationship between class size and time, t statistics 0.683, p > 0.1.

From this analysis, it appears that students who took fewer extra steps while solving this Parsons problem have better midterm scores. A similar trend is also observed in another problem from Unit 4 as shown in Figure 8. This could mean that taking extra steps and/or taking longer to solve a Parsons problem indicates that a student is struggling.

5. LIMITATIONS

The log file data was from a random selection of custom courses on the Runestone platform. We do not have any additional information about these courses, such as which items were assigned, final grades, or student demographics.

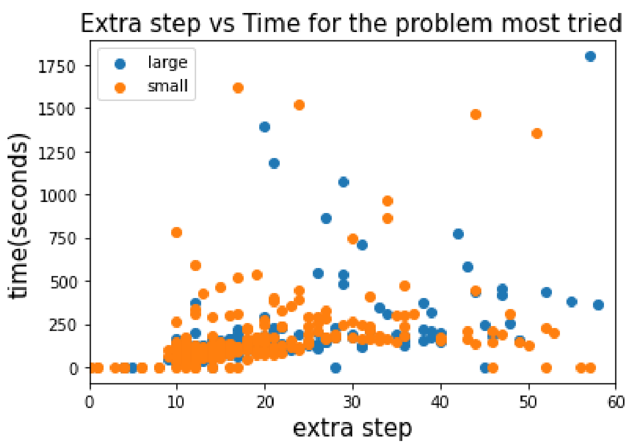


Figure 5: Relationship between extra steps and time

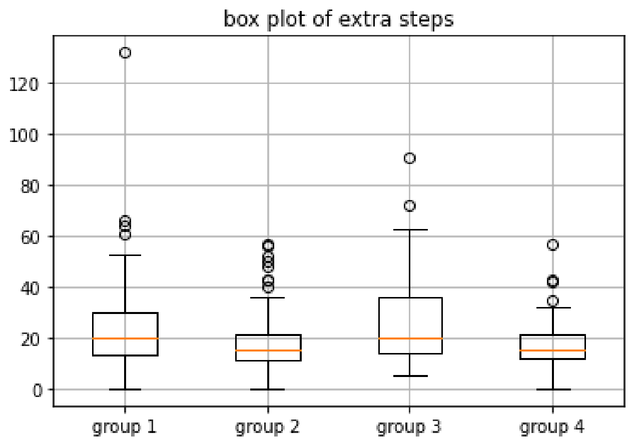


Figure 6: Comparison extra step by midterm score and class size. 1. small class and midterm score less than median, 2. small class and midterm score greater than median, 3. large class and midterm score less than median, and 4. large class and midterm score greater than median

In the summer of 2021 we will be receiving log file data for this same ebook from teachers who attended professional development with the Mobile CS Team. That data should allow for a more in-depth analysis.

6. CONCLUSION

In this paper, we performed several quantitative analyses on the clickstream data from student interaction with the CSAwesome ebook. We analyzed the relationship between students' activities on the ebook and their midterm scores. We found several positive and negative correlations. In a regression analysis the most highly weighted variable with a positive correlation was the percentage correct on other multiple choice questions and most highly weighted variable with a negative correlation was being in a large class.

We also analyzed the learner interaction patterns on the mixed-code (Parsons) problems. Specifically, we examined the impact of the number of steps taken, time taken, as well as the frequency of help on the students' midterm scores.

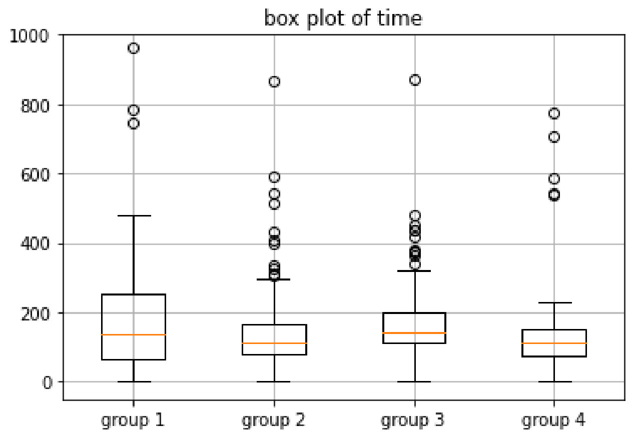


Figure 7: Comparison time by midterm score and class size. 1. small class and midterm score less than median, 2. small class and midterm score greater than median, 3. large class and midterm score less than median, and 4. large class and midterm score greater than median

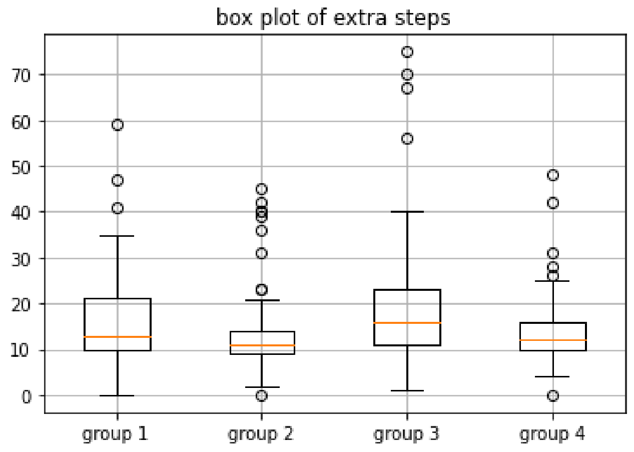


Figure 8: Comparison extra step by midterm score and class size of another problem in Unit 4 on nested loops

The results show a positive association between the number of correctly completed Parsons problems and the learners' midterm scores. Further, there was a negative association between the midterm scores and the time the learner took to get help on a Parsons problem as well as a negative association between the class size and the midterm score. A close look at two Parsons problems showed a negative correlation with the number of extra steps and time to solve a Parsons problem and the midterm score.

While our analyses uncover subtle patterns in students' interactions with the CSAwesome ebook, it will be interesting to test the robustness of our findings and see if they generalize to other interactive ebook platforms or across different programming courses, e.g., C++ or Python. If Parsons problems can help detect struggling students early in a course it may be possible to intervene to improve student performance.

7. REFERENCES

[1] G. Akçapınar, M. N. Hasnine, R. Majumdar, B. Flanagan, and H. Ogata. Developing an early-warning system for spotting at-risk students by using ebook interaction logs. *Smart Learning Environments*, 6(1):4, 2019.

[2] P. Denny, A. Luxton-Reilly, and B. Simon. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*, pages 113–124, 2008.

[3] Y. Du, A. Luxton-Reilly, and P. Denny. A review of research on parsons problems. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, ACE'20, page 195–202, New York, NY, USA, 2020. Association for Computing Machinery.

[4] B. Ericson, B. Hoffman, and J. Rosato. Csawesome: Ap csa curriculum and professional development (practical report). In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education*, WiPSCE '20, New York, NY, USA, 2020. Association for Computing Machinery.

[5] B. Ericson, A. McCall, and K. Cunningham. Investigating the affect and effect of adaptive parsons problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, pages 1–10, 2019.

[6] B. Ericson, S. Moore, B. Morrison, and M. Guzdial. Usability and usage of interactive features in an online ebook for cs teachers. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, pages 111–120, 2015.

[7] B. J. Ericson, J. D. Foley, and J. Rick. Evaluating the efficiency and effectiveness of adaptive parsons problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, pages 60–68, 2018.

[8] B. J. Ericson, M. J. Guzdial, and B. B. Morrison. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 169–178, 2015.

[9] B. J. Ericson, L. E. Margulieux, and J. Rick. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, pages 20–29, 2017.

[10] B. J. Ericson and B. N. Miller. Runestone: A platform for free, on-line, and interactive ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 1012–1018, 2020.

[11] P. J. Guo. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 579–584, 2013.

[12] J. Helminen, P. Ihantola, V. Karavirta, and L. Malmi. How do students solve parsons programming problems? an analysis of interaction traces. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research*, ICER '12, page 119–126, New York, NY, USA, 2012. Association for Computing Machinery.

[13] A. Korhonen, T. Naps, C. Boisvert, P. Crescenzi, V. Karavirta, L. Mannila, B. Miller, B. Morrison, R. R. Rodger, Susan H, and A. Shaffer, Clifford. Requirements and design strategies for open source interactive computer science ebooks. In *Proceedings of the ITiCSE working group reports conference on Innovation and technology in computer science education-working group reports*, pages 53–72, 2013.

[14] S. Maharjan and A. Kumar. Using edit distance trails to analyze path solutions of parsons puzzles. In *EDM*, 2020.

[15] B. N. Miller and D. L. Ranum. Beyond pdf and epub: toward an interactive textbook. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pages 150–155, 2012.

[16] B. B. Morrison, L. E. Margulieux, B. Ericson, and M. Guzdial. Subgoals help students solve parsons problems. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pages 42–47, 2016.

[17] J. Park, K. Denaro, F. Rodriguez, P. Smyth, and M. Warschauer. Detecting changes in student behavior from clickstream data. In *Proceedings of the Seventh International Learning Analytics amp; Knowledge Conference*, LAK '17, page 21–30, New York, NY, USA, 2017. Association for Computing Machinery.

[18] M. C. Parker, K. Rogers, B. J. Ericson, and M. Guzdial. Students and teachers use an online ap cs principles ebook differently: Teacher behavior consistent with expert learners. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, pages 101–109, 2017.

[19] D. Parsons and P. Haden. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, pages 157–163, 2006.

[20] K. Pollari-Malmi, J. Guerra, P. Brusilovsky, L. Malmi, and T. Sirkiä. On the value of using an interactive electronic textbook in an introductory programming course. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, pages 168–172, 2017.

[21] W. Wang, R. Zhi, A. Milliken, N. Lytle, and T. W. Price. Crescendo: Engaging students to self-paced programming practices. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pages 859–865, 2020.

[22] N. Weinman, A. Fox, and M. A. Hearst. Improving instruction of programming patterns with faded parsons problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[23] H. Yan, F. Lin, et al. Including learning analytics in the loop of self-paced online course learning design. *International Journal of Artificial Intelligence in Education*, pages 1–18, 2020.

[24] R. Zhi, M. Chi, T. Barnes, and T. W. Price. Evaluating the effectiveness of parsons problems for block-based programming. In *Proceedings of the 2019 ACM Conference on International Computing*

Education Research, pages 51–59, 2019.