Open IDE Action Log Dataset from a CS1 MOOC

Juho Leinonen[*]
University of Helsinki
juho.leinonen@helsinki.fi

ABSTRACT
Studying students' time-related behavior is a common research topic within educational data mining. One way to study time-related behavior is by collecting log data from the use of pedagogical tools or systems. In computer science in particular, such data has often been collected from students' use of integrated development environments (IDEs). In this work, we present and openly release a dataset containing IDE logs from an introductory programming MOOC. The dataset contains information on when actions in the IDE were performed in relation to deadlines over the different parts of the course. One exceptional aspect of the dataset is that part of the logs have been gathered at the keystroke level, allowing for fine-grained insight into the learning process. In addition to the IDE logs themselves, the dataset has information on whether students included in the data passed the course. This can facilitate further research that analyzes how time-related behavior correlates with performance in introductory programming courses.

Keywords
open data, IDE logs, log data, integrated development environment, CS1, keystroke data, programming process data, learning analytics, educational data mining

1. INTRODUCTION
One recurring research theme in educational data mining is studying student time-related behavior such as earliness of work [4, 6, 12, 15, 17], procrastination [9, 10, 18], and time-on-task [13, 14, 21]. Studies in this area typically utilize logs gathered from pedagogical tools or learning management systems. In computer science educational data mining (CSEDM) in particular, studying logs from integrated development environments (IDEs) has been common [8].

In this work, we present and make publicly available a dataset [16] containing IDE logs from an introductory programming massive open online course (MOOC). The logs contain information on actions students perform in the IDE while working on course exercises. Notably, the data does not contain source code etc. and can be thought of as "metadata" related to the students' learning processes. Such data has potential, for example, in developing time management based interventions, for example, by following the IDE-based learning analytics process model by Hundhausen et al. [7].

The data presented here is partly at the keystroke level, which allows for a fine-grained look into students' learning [11]. This data contributes to the growing body of open IDE log datasets [1, 3, 5], which can both support future research utilizing IDE logs and help replicate earlier findings; lack of replication has been noted as a major challenge in computing education research [8].

2. CONTEXT
The context of the data is a massive open online course (MOOC) on introductory programming (CS1). The programming language used in the course is Java. The course is split into seven parts, where each individual part introduces a few new programming concepts to the students. The concepts covered in the course are typical CS1 topics, beginning with variables, conditionals, loops and lists, which are followed by methods, parameters, classes and objects.

The course uses online materials. In the online material, concepts are introduced with examples that focus on how the concepts can be used in programs. The introduction of the concept ("theory part") is followed by a couple of exercises where students are expected to practice the concept. Each part has a few larger exercises at the end of the material that typically combine earlier concepts and the new concepts introduced in the respective part of the course. The course uses the many small exercises approach [2, 22], where students work on anywhere from ten to thirty exercises in each part of the course.

Exercises in the course are completed using an integrated development environment (IDE) with the custom Test My Code -plugin [23]. While exercise descriptions are found in the online material, students can download exercise templates directly into the IDE. The templates typically contain boilerplate code, and in some cases some starter code. The exercises are accompanied by instructor-created unit test suites that students can test their program against directly in the IDE. When students believe their code to be

[*]Part of the work conducted while at Aalto University.

J. Leinonen. Open IDE Action Log Dataset from a CS1 MOOC. In B. Akram, T. Price, Y. Shi, P. Brusilovsky, and S. Hsiao, editors, Proceedings of the 6th Educational Data Mining in Computer Science Education (CSEDM) Workshop, pages 88–91, Durham, United Kingdom, July 2022.

correct, they can submit their program from the IDE to the automated assessment system used in the course. The system often includes a few additional "hidden tests" that are not included in the unit test suite in the IDE to validate that students have not just written code that is hard-coded to pass the unit test suite used for local code evaluation in the IDE. Since students could run the instructor-created unit tests locally on their computer (not logged), there are likely many students who only have a single submission event for each exercise (they only submit after the tests pass locally).

Each part of the course has a deadline after which students can no longer submit their program for evaluation nor get the points for the exercises of that part. They can continue working on the exercise after the deadline if they so desire; however, events that occur after the deadline are not included in the data. The first four parts of the course had the same deadline to allow students to join the course late. For the fifth, sixth and seventh part of the course, the deadlines were one, two and three weeks later (compared to the deadline for the first four parts), respectively. The course was released 42 days before the deadline of the first four parts, so there are students who worked close to when the course was released and others who worked closer to the deadlines. Since the course is a MOOC, some of the students who started later might have only heard about the course after it was released, so starting closer to the deadlines might not indicate procrastination.

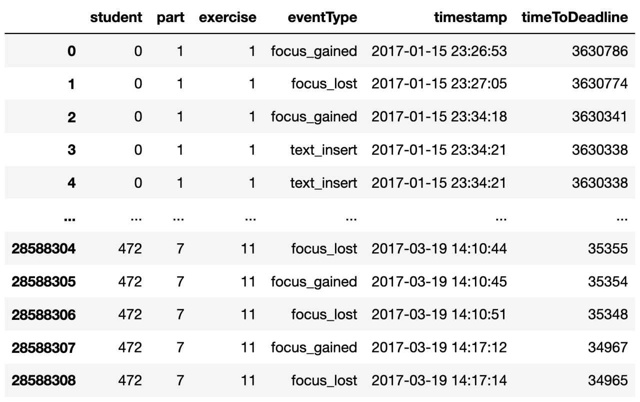
In addition to the exercises, there were three online exams. The online exams were completed in the IDE similar to course exercises with the exception that there was a time limit of two hours to complete all exercises of the exam from the time when the student started the exam. The three exams were situated at different points in the course. The first exam was at the beginning of the third part and covered topics from parts 1 and 2. The second exam was at the beginning of part 5 and covered the previous four parts. The third exam was at the end of part 7 and covered all course topics.

Passing the course was determined based on both points received from the exercises (55% of the total points) and points received from three online exams (45% of the total points). Students could receive 8% of the points by completing every exercise in each of the seven parts of the course (except for the first part, where they could get only 7% of the points). The three exams were worth 10%, 15% and 20% of the points. Students had to receive a total of 51% or more of the total course points in order to pass the course. In addition, students were required to receive at least half of the points from the last exam in order to pass the course, regardless of any other points received. Some students might not have many IDE logs even if they passed the course as it is possible they partly worked offline in which case the logs were never sent to the server, or they might have disabled logging intermittently.

The IDE used in the course collects data from the students' programming process. Only data from students who consented for their data to be used for research is included in the dataset. Additionally, students could turn logging on and off at will within the IDE.

Event type	Event count
text_insert	19,652,100
text_remove	5,582,299
text_paste	166,601
focus_gained	1,427,004
focus_lost	1,383,954
run	323,749
submit	52,602
total	28,588,309

Table 1: Number of each type of event in the data.



	student	part	exercise	eventType	timestamp	timeToDeadline
0	0	1	1	focus_gained	2017-01-15 23:26:53	3630786
1	0	1	1	focus_lost	2017-01-15 23:27:05	3630774
2	0	1	1	focus_gained	2017-01-15 23:34:18	3630341
3	0	1	1	text_insert	2017-01-15 23:34:21	3630338
4	0	1	1	text_insert	2017-01-15 23:34:21	3630338
...	...	...	...	...	...	...
28588304	472	7	11	focus_lost	2017-03-19 14:10:44	35355
28588305	472	7	11	focus_gained	2017-03-19 14:10:45	35354
28588306	472	7	11	focus_lost	2017-03-19 14:10:51	35348
28588307	472	7	11	focus_gained	2017-03-19 14:17:12	34967
28588308	472	7	11	focus_lost	2017-03-19 14:17:14	34965

Figure 1: A few example rows of the IDE log data.

The data collection followed the guidelines of the university that organized the course, country level guidelines for ethical research[1], and applicable legislation. The dataset contains only metadata about the learning process and no direct contributions from students (such as source code or keys pressed), which is a deliberate choice to preserve the privacy of the students included in the data.

3. DATASET DESCRIPTION

The dataset contains two files, `IDE_logs.csv` and `passed.csv`. `IDE_logs.csv` contains IDE action logs from the course and `passed.csv` contains information on whether the student passed the course.

The IDE logs consist of 1) a student identifier (identifier matches `passed.csv`), 2) which of the seven parts and 3) which exercise was the event related to, 4) information on which event happened (brief explanations of the events below), 5) a timestamp, and 6) how long before the deadline the event happened (in seconds). There is one row in the data for each event that was logged. The part and exercise identifiers are sequential: for example, exercise "5" means that it was the fifth exercise (in the material) for the specific part of the course[2]. The data is not in the ProgSnap2 [20]

[1]In particular, ethical review for research is only needed in specific cases, which our research does not fall under. See `https://tenk.fi/en/ethical-review/ethical-review-human-sciences` for details (accessed July 20th 2022).

[2]Students could complete exercises in any order they wished, so it is possible to have events from a latter exercise before even starting an earlier exercise.

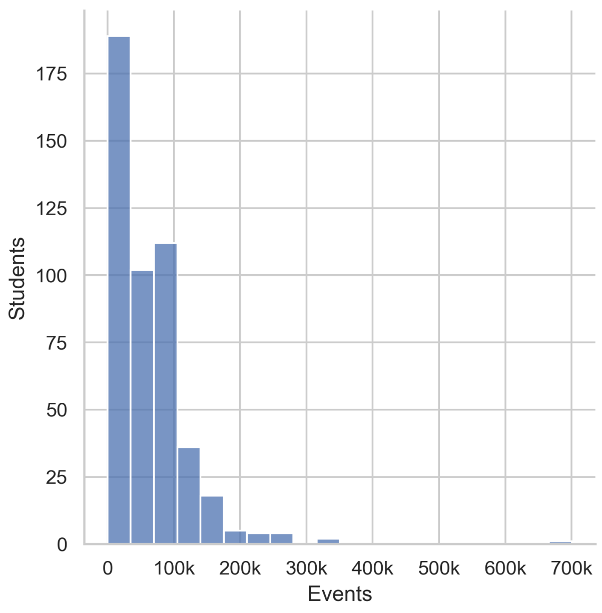


Figure 2: Distribution of logged actions per student.

programming snapshot format as the ProgSnap2 format requires sharing the source code ("CodeStateID" is a required column for ProgSnap2), which we are unable to share at this time due to privacy concerns.

There are seven types of events in the data which are as follows.

- **text_insert** – text was added into the IDE (at the keystroke level)
- **text_remove** – text was removed from the IDE
- **text_paste** – text was pasted into the IDE
- **focus_gained** – the IDE gained focus (student clicked into the IDE)
- **focus_lost** – the IDE lost focus (student clicked away from the IDE)
- **run** – student executed their program
- **submit** – student submitted their program

The data has a total of 28,588,309 rows (i.e. logged IDE actions) from a total of 473 students. The mean number of events per student is 60,440 (with a standard deviation of 61,228 and a median of 50,810). The number of each type of event in the data is outlined in Table 1. A few example data rows are shown in Figure 1 and the distribution of logged actions per student is visualized in Figure 2.

The `passed.csv` has one row for each student. The only columns are a student identifier and "passed" where the value is either `True` (student passed the course) or `False` (student did not pass the course).

Both `IDE_logs.csv` and `passed.csv` can be found here: `https://doi.org/10.5281/zenodo.6903968`

4. POTENTIAL RESEARCH QUESTIONS

There are many research questions that could be studied utilizing this type of data. As the data does not contain the source code or source code related information such as compilation status or correctness, the main type of research that the data is useful for is related to students' behavior in IDEs. Here, we outline a few potential research questions and areas where such data could be used.

One stream of research that can utilize this type of log data is studies related to student time-on-task. Recent works have, for example, examined how the granularity of data affects time-on-task estimates [13] and how time-on-task estimates derived from log data correlate with students' performance [14]. Potential research questions that remain are, for example, how individual students' times-on-task differ between exercises – is it always the same students who spend a lot of time on exercises, or are the exercise-specific differences between students in how much time they spend on the exercise?

Another common area of research in computing education where the data could be used is studying student time management such as how early (or late) students work on exercises, and procrastination. Prior work has found correlations between starting early and performing better in the course [6, 12]. Future work in this area could study, for example, whether time management related behavior varies across the course and how well can log-derived time management metrics predict student performance in the course. Additionally, one could try clustering the data to try identify whether there are clear student subpopulations with different time-related behavioral patterns, for example, try replicate prior work studying "tinkerers", "movers", and "stoppers" [19].

Lastly, a relatively under-explored aspect of student work is examining at what time of the day students work on exercises, and how this correlates with performance in the course. A recent study by Zavgorodniaia et al. [24] clustered students based on when they were working in an IDE and found clusters that roughly correspond to different "chronotypes", i.e. people's propensity to sleep at certain times of day and correspondingly, be alert at other times.

There are two main limitations related to the data. First, there is the possibility of self-selection bias as only data from students who both chose to allow action logging within the IDE and who consented to their data being used for research is included. It is possible that students who chose to do so differ from other students in the course. Second, the data does not include the source code or e.g. an AST representation of it. As part of our future work, we are exploring ways of sharing source code and code states while preserving the privacy of the students who have supplied the data.

Altogether, there are multiple research directions where IDE action logs could be utilized. Releasing this dataset openly hopefully facilitates future research in this area.

5. REFERENCES

[1] CodeBench dataset. `https://codebench.icomp.ufam.edu.br/dataset/`. Accessed: 2022-06-23.

[2] J. M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller. An analysis of using many small programs in cs1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 585–591, 2019.

[3] N. C. C. Brown, M. Kölling, D. McCall, and I. Utting. Blackbox: A large scale repository of novice programmers' activity. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 223–228, 2014.

[4] P. Denny, J. Whalley, and J. Leinonen. Promoting early engagement with programming assignments using scheduled automated feedback. In *Australasian Computing Education Conference*, pages 88–95, 2021.

[5] J. Edwards. Computer Programming Keystroke Data. `https://dataverse.harvard.edu/dataverse/cskeystrokes`, 2022.

[6] S. H. Edwards, J. Snyder, M. A. Pérez-Quiñones, A. Allevato, D. Kim, and B. Tretola. Comparing effective and ineffective behaviors of student programmers. In *Proceedings of the fifth international workshop on Computing education research workshop*, pages 3–14, 2009.

[7] C. D. Hundhausen, D. M. Olivares, and A. S. Carter. IDE-based learning analytics for computing education: a process model, critical review, and research agenda. *ACM Transactions on Computing Education (TOCE)*, 17(3):1–26, 2017.

[8] P. Ihantola, A. Vihavainen, A. Ahadi, M. Butler, J. Börstler, S. H. Edwards, E. Isohanni, A. Korhonen, A. Petersen, K. Rivers, et al. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports*, pages 41–63. 2015.

[9] A. M. Kazerouni, S. H. Edwards, T. S. Hall, and C. A. Shaffer. DevEventTracker: Tracking development events to assess incremental development and procrastination. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 104–109, 2017.

[10] A. M. Kazerouni, S. H. Edwards, and C. A. Shaffer. Quantifying incremental development practices and their relationship to procrastination. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*, pages 191–199, 2017.

[11] J. Leinonen. *Keystroke Data in Programming Courses*. PhD thesis, University of Helsinki, 2019.

[12] J. Leinonen, F. E. V. Castro, and A. Hellas. Does the early bird catch the worm? earliness of students' work and its relationship with course outcomes. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, pages 373–379, 2021.

[13] J. Leinonen, F. E. V. Castro, and A. Hellas. Fine-grained versus coarse-grained data for estimating time-on-task in learning programming. In *Proceedings of The 14th International Conference on Educational Data Mining (EDM 2021)*. The International Educational Data Mining Society, 2021.

[14] J. Leinonen, F. E. V. Castro, and A. Hellas. Time-on-task metrics for predicting performance. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 871–877, 2022.

[15] J. Leinonen, P. Denny, and J. Whalley. A comparison of immediate and scheduled feedback in introductory programming projects. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 885–891, 2022.

[16] J. Leinonen and A. Hellas. IDE Action Log Dataset from a CS1 MOOC, July 2022.

[17] L. Leppänen, J. Leinonen, and A. Hellas. Pauses and spacing in learning to program. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, pages 41–50, 2016.

[18] F. D. Pereira, E. H. Oliveira, D. B. Oliveira, A. I. Cristea, L. S. Carvalho, S. C. Fonseca, A. Toda, and S. Isotani. Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British journal of educational technology*, 51(4):955–972, 2020.

[19] D. N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons. Conditions of learning in novice programmers. *Journal of Educational Computing Research*, 2(1):37–55, 1986.

[20] T. W. Price, D. Hovemeyer, K. Rivers, G. Gao, A. C. Bart, A. M. Kazerouni, B. A. Becker, A. Petersen, L. Gusukuma, S. H. Edwards, and D. Babcock. ProgSnap2: A flexible format for programming process data. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 356–362, 2020.

[21] J. Spacco, P. Denny, B. Richards, D. Babcock, D. Hovemeyer, J. Moscola, and R. Duvall. Analyzing student work patterns using programming exercise data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 18–23, 2015.

[22] A. Vihavainen, M. Paksula, and M. Luukkainen. Extreme apprenticeship method in teaching programming for beginners. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 93–98, 2011.

[23] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using Test My Code. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 117–122, 2013.

[24] A. Zavgorodniaia, R. Shrestha, J. Leinonen, A. Hellas, and J. Edwards. Morning or evening? an examination of circadian rhythms of cs1 students. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 261–272. IEEE, 2021.