education sciences

MDPI

Article

Learners' Performance in a MOOC on Programming

Lidia Feklistova *, Marina Lepp and Piret Luik

Institute of Computer Science, University of Tartu, 51009 Tartu, Estonia; marina.lepp@ut.ee (M.L.); piret.luik@ut.ee (P.L.)

* Correspondence: lidia.feklistova@ut.ee

Abstract: In every course, there are learners who successfully pass assessments and complete the course. However, there are also those who fail the course for various reasons. One of such reasons may be related to success in assessment. Although performance in assessments has been studied before, there is a lack of knowledge on the degree of variance between different types of learners in terms of scores and the number of resubmissions. In the paper, we analyse the performance in assessments demonstrated by non-completers and completers and by completers with different engagement levels and difficulty-resolving patterns. The data have been gathered from the Moodle statistics source based on the performance of 1065 participants, as regards their completion status, the number of attempts made per each programming task and quiz, and the score received per quiz. Quantitative analysis was performed with descriptive statistics and non-parametric tests. Non-completers and completers were similar in resubmissions per quiz, but the former, expectedly, made more resubmissions per programming task and received lower quiz scores. Completers made more attempts per task than per quiz. They could provide a correct solution with a few resubmissions and receive good scores already at a pragmatic engagement level. At the same time, the increased use of help sources in case of difficulties was also associated with a higher number of attempts and lower quiz scores received. The study may have implications in understanding the role of assessments in dropouts and how completers with different engagement and difficulty-resolving patterns cope with assessments.

Keywords: MOOC on programming; performance in assessments; non-completers; completers; engagement styles; difficulty-resolving patterns; non-parametric tests

Citation: Feklistova, L.; Lepp, M.; Luik, P. Learners' Performance in a MOOC on Programming. *Educ. Sci.* **2021**, *11*, 521. https://doi.org/10.3390/educsci11090521

Academic Editors: Enrique Barra Arias, Aldo Gordillo and Sonsoles López-Pernas

Received: 3 August 2021
Accepted: 3 September 2021
Published: 8 September 2021

1. Introduction

Participation in a massive open online course (MOOC) usually does not require any prerequisite or predefined levels from learners. This aspect makes MOOCs particularly attractive to self-directed learners who are basically interested in improving their knowledge in a course field [1]. Participants have the opportunity to determine the learning goals for themselves as student-centred learning suggests, and as they progress through the course, participants may adapt approaches to reach their goals ultimately [2]. Course instructors provide a variety of activities to engage with [3,4] and different support mechanisms such as forums [5] or troubleshooters [6]. Adult learners find quizzes and problem sets useful, as they can be used to get feedback and verify learning results [1]. Completers' performance in assessment varies, but they all make at least an attempt to pass [7]. Repeated attempts (resubmissions) with improving solutions allow learners to have higher grades in MOOCs [8,9]. In the context of learning programming, Auvinen [10] theorised that a student makes a high number of attempts because exercise is difficult to solve on the first attempt and a student makes a low number of attempts because the solution is carefully checked before submission. At the same time, numerous attempts, even without getting the right solution, demonstrate a student's resilience—a key feature for programmers [11]. Giving a learner the possibility to have several attempts at assignments and quizzes is a helpful approach from the perspective of dropout prevention [12]. Dropout is one of the

biggest MOOC problems for which researchers have identified different reasons. Those who will ultimately be considered non-completers may enrol only to study a specific part of the course they are interested in [13], or they can have difficulties with effective time management [13,14]. Difficulties in understanding learning materials are also seen as an insuperable obstacle for them [14,15]. There is a lack of knowledge about how non-completers perform in assignments in comparison to completers in the same course. This study might shed light on this gap. Although, naturally, completers are likely to have different engagement and difficulty-resolving strategies that ultimately lead them to successful course completion, the current study might also provide insights into how those strategies contribute to completers' performance.

The paper tried to shed light on those aspects in the context of a MOOC on programming. In this study, descriptive statistics and non-parametric tests are employed to examine performance in programming tasks and quizzes by non-completers and completers, as well as by previously identified completers with different engagement styles [16] and difficulty-resolving patterns [17].

1.1. Literature Review

Learners' performance can be defined as temporary fluctuations in behaviour or knowledge that can be evaluated during the acquisition process or immediately after it [18]. To assess learners, course instructors use different types of assessments and allow resubmissions to provide correct responses. This section considers prior findings of learners' performance and the effect of multiple resubmissions on students' success. In addition, an overview of performance results in terms of engagement styles and difficulty-resolving patterns is presented.

1.1.1. The Effect of Multiple Attempts on Success

Studies both support and refute the value of multiple submissions. It has been stated that those who try several times are more likely to pass a task [19] and persist in a course [20]. Learners can achieve higher grades [8,9,21], and the probability of the correct answer increases with each new attempt to solve a task [22]. Others have claimed that a high number of attempts lead to lower grades [10] and might facilitate the guessing effect [23]. Some students hope in vain to improve performance by submitting solutions multiple times, but without concentrating on error sources [24]; some students do nothing in between attempts [25]. Instead of thinking hard and solving a problem on their own, learners start to abuse hints and automated check tools, which can ultimately lead to poor hands-on skills [20].

The most powerful disengaging factor in a programming course is a failure in hands-on assignments. It does not matter how many times a learner can submit a given assignment as he/she may quit the course even after the first unsuccessful submission, without attempting to improve a code and resubmit it [26]. Some learners present workable solutions on the first attempt; others make multiple attempts while improving solutions. The average number of submissions per each assignment on programming can vary [10,27,28]. Topics and levels of difficulty in computer science courses also matter. In a course for advanced users, the number of attempts to pass a task can be about twice as high as in a course for beginners [27]. In computer science courses, the submission of multiple assignments can be considered as an indicator of high engagement and effective behaviour [20,28,29].

1.1.2. Engagement Styles and Performance

The classification of engagement styles is based on what activities are undertaken and how often they are participated in by a learner throughout the course [30]. Studies [31–33] have demonstrated contradictory results regarding performance by learners with different engagement styles. On the one hand, the more learners engage with course activities, the better they perform [7]. Shi and Cristea [34] clarified that the more students visit learning materials, write comments and attempt to answer quiz questions, the better

performance results they achieve. The authors found that students with the highest level in the mentioned aspects made on average more attempts to answer a question than those whose engagement level was a little lower and who were less active in submitting comments. Among those who were least engaged with learning materials and were socially active, only about a quarter attempted to answer questions. Their results and the number of submissions were the lowest. However, Arora et al. [35] identified learners who showed better performance with lesser effort and engagement than those who used the maximum amount of course materials. Deng et al. [32] found no difference in academic performance by learners from clusters with multidimensional engagement patterns.

Through working with interactive activities, learning by doing, students find out more about the subject and get better results than their fellow students who merely watch videos or read materials [36]. There are also studies [37,38] where learners who demonstrated a high level of engagement with forum posts and watching videos achieved high results in quizzes. Abbakumov et al. [22] concluded that those who actively watch video lectures and are productive with formative assessments have a higher probability of solving items correctly.

1.1.3. Difficulty-Resolving and Performance

In programming courses, resilience, along with other factors (e.g., the accuracy of problem-solving and analysing and debugging compilation errors), can lead to better performance. Students with this trait do not easily give up, usually have good performance and pass a course [29]. In an interactive activity, designed for practising the obtained skills, learners may not achieve the required result right away but could return to it after they have watched a video or have read the text lesson [39]. It has been suggested that the strategy of viewing and reviewing lecture materials can bring positive results [25].

If a student spends an inordinate amount of time solving a programming assignment and makes errors, it could mean that he/she is having difficulty and needs help [40]. Learners who use computer-mediated contact with friends as a help-seeking strategy while working on assignments have the highest completion rate. However, those who ask friends face-to-face or do not use any help-seeking strategy at all are less successful. Learners who use an Internet Relay Chat (IRC) room provided by the platform fall in between the two [41]. Learners often ask lecturers for help when struggling with surmountable difficulties and are interested in synchronous help that could be provided beyond face-to-face office hours and email. As a result of this approach, a grade is positively correlated with the amount of received help—those who ask for help and receive more support get better results [42].

There are courses that provide automatic hints to resolve difficulties. A positive correlation has been found between the usage of troubleshooters and the number of attempts in programming tasks and quizzes. Those who had difficulties were more active in using this type of hint [6]. It is important to weigh the necessity and number of clues, as those who eventually failed the programming course were four times more likely to use those features compared to top students and were less enthusiastic about compiling code [43].

Auvinen [10] found it difficult to state that an inefficient learning strategy leads to poor performance, but Thomas et al. [44] suggested that some learning styles are more suitable for learning programming than others. Since different approaches can affect performance in different ways, they may help understand how different learners perform in assessment. The aim of the current paper is to analyse performance in assessments demonstrated by non-completers and completers, as well as by completers with different engagement styles and difficulty-resolving patterns. As such, the following research questions are considered in this study:

1. RQ 1: To what extent do the performances in assessments between non-completers and completers differ?
2. RQ 2: To what extent do the performances in assessments between completers grouped into engagement clusters differ?

3. RQ 3: To what extent do the performances in assessments between completers grouped into clusters of difficulty-resolvers differ?

The remainder of the article paper is structured as follows. Section 2 contains a description of the methodology. In Section 3, the data analysis and results of the study are provided. Section 4 discusses the findings, followed by concluding remarks in Section 5.

2. Methodology

2.1. Context of Study

The article addresses a 4-week Estonian-language MOOC About Programming (worth 1 the European Credit Transfer and Accumulation System (ECTS) credit). The course was developed and organized by a research group from the Institute of Computer Science at the University of Tartu (Estonia) several times since December 2014. In the course, the basic programming concepts and structures of Python were introduced, and overviews about IT-related subjects and examples were given [45]. Learning materials were presented in the form of texts and videos, demos and weblinks to additional resources were also provided. Weekly videos and emails were sent to learners to give an overview of what was happening in the course and encourage them to continue the course. Among used supported mechanisms, there were a forum, a helpdesk with a quick response within 8 h (even on weekends), and troubleshooters that contained hints of certain aspects provided for every programming task. There were also self-assessment questions and additional tasks, but they were not compulsory and their results were not stored. The course was designed in such a way that everyone who had a computer with Internet access could access materials and tasks.

The MOOC was basically intended for adults interested in programming, but children could also participate. Previous experience with programming was not required. Course participants and completers differed in terms of demographic and social background [46], motivation [47], interaction with provided activities [16], and resolved difficulties in programming tasks in different ways [17].

To receive a certificate of completion, learners had to provide workable solutions for all mandatory practical programming tasks and pass all quizzes provided in the Moodle environment. There were six programming tasks in total, and the learners had to write programming codes on their own. The learners were asked to write their Python codes in Thonny. This development environment helps beginners evaluate expressions step-by-step and analyse the process of programming [48]. The solutions had to be uploaded to the virtual programming lab and were assessed by an automatic assessment system as passed or failed. A learner had to resubmit the solution if, for example, any required solution variables or cycles were missing, there were spelling mistakes, or the output was a little different from the one required by the task. In addition, there were four quizzes with 10 questions each. The quizzes were based on theoretical content and included mainly multiple-choice questions. Learners were allowed to use all the materials. The answers were automatically checked by the Moodle, and a learner needed to get at least 9 points out of 10 to pass a quiz. The number of tries in the programming tasks and quizzes was not limited. Such multiple resubmissions of a task or quiz are referred to hereinafter as “attempts”.

2.2. Sample

The current paper considers the course started in September 2018. The MOOC had 1307 registrants, but 242 did not start the course. The certificate requirements were met by 773 completers. There were 292 non-completers who did not meet all certificate requirements but submitted at least one programming task solution and/or did at least one quiz with an intention to achieve a required threshold value (no-show registrants were excluded). The description of the study sample is provided in Table 1.

Table 1. Descriptive statistics of the sample.

	All Participants	Non-Completers	Completers
Sample size	1065	292	773
Female	54.3%	53.4%	54.7%
Average age	33.0 (SD = 10.81)	36.4 (SD = 10.47)	33.2 (SD = 10.94)
Age range	10–70	10–60	10–70

Previously, we revealed completers' clusters based on their reported engagement [16] and difficulty-resolving strategies [17]. Those clusters were formed using K-mean cluster analysis. The data were gathered from the questionnaires. Filling them out was voluntary (at the end of the course), and only answers from completers were analysed. Regarding engagement styles, completers self-evaluated their interaction (from *not at all* to *all*) with the listed course activities. Regarding difficulty-resolving patterns, the completers self-evaluated their activity level (from *not at all* to *always*) with the listed help sources. It should be noted that for the second and third research questions, the number of studied completers (575) was smaller compared to the total course completers (773), since only 580 completers filled in the questionnaires used to identify clusters and five completers who filled in the questionnaires asked not to use their performance data. The sample description by clusters is as follows:

- Engagement clusters [16]:
  - active knowledge collectors (N = 170) engaged with most provided activities, except reading forum posts and using troubleshooters. They were predominantly older completers with higher education;
  - for minimum knowledge collectors (N = 95), the main sources used were learning materials on programming, self-assessment questions, and provided demos. This cluster had a higher share of males and younger persons with lower education who had studied programming before;
  - pragmatic knowledge collectors (N = 201) concentrated mainly on the activities needed to complete the MOOC on programming. No distinctive differences by demographic and social characteristics were found;
  - support-required knowledge collectors (N = 114) engaged with all available course activities and were more active in using provided support mechanisms. Most of them had no previous experience in studying programming
- Difficulty-resolvers' clusters [17]:
  - bounded resolvers (N = 172) tended to re-read learning materials and were least likely to search for additional materials on the Internet. In this cluster, those who had studied programming before occupied a large portion;
  - moderate resolvers (N = 74) usually re-read learning materials and tried to find information on the Internet, forums, or troubleshooters. For most of them, this course was their first experience with a web-based course;
  - step-by-step resolvers (N = 124) coped with difficulties by using learning materials and troubleshooters. The members of this cluster were mostly inexperienced in studying programming;
  - social resolvers (N = 42) had the highest activity levels in re-reading the learning materials, using troubleshooters, and seeking help from forums and helpdesk. This cluster had the highest share of female learners, who had never studied programming nor participated in a web-based course before;
  - self-supporting resolvers (N = 168) used learning materials and were the most active in searching for additional materials on the Internet. Most of them were male and experienced in studying programming but had never participated in a web-based course.

2.3. Data Collection

In this study, we defined performance in assessment as the number of attempts per programming task and per quiz and the numerical score received in a quiz.

From the Moodle, we collected quantitative data about learners' completion status, the number of attempts made per each programming task and quiz, and the score received per quiz. It should be noted that the opportunity to upload a task solution or take a quiz once again was available after it was evaluated as passed. The numeric score was available only for quizzes. In the collected data, only the best score per quiz was stored.

Each learner who completed the course was assigned to one of the completers' clusters identified in previous studies about completers' engagement [16] and difficulty-resolving [17]. This manipulation was possible because the current study used the same dataset with the same sample, and the data about completers' cluster membership were stored.

2.4. Data Analysis

Quantitative data were analysed using IBM SPSS Statistics 26.0. The preliminary analysis of main variables did not reveal any missing data. In the data, there were outliers. Those high numbers of attempts were considered a natural part of the whole population, since there are always learners who make a lot of attempts. In addition, according to Frost [49], if the sample size is large enough (as it was in the current study), it is allowed to consider those extreme values as a normal part of the data distribution, and, therefore, should not be deleted from the study.

Descriptive statistics were used to calculate the average, standard deviation, and range of different performance variables, such as the number of attempts. The calculated average number of attempts per programming task for a completer was based on all six mandatory tasks; the calculated average number of attempts per quiz and the average received score per quiz were based on all four mandatory quizzes. The calculated average number of attempts per programming task for a non-completer was based on all completed tasks (at least one task, but there was a non-completers who did all six tasks); the calculated average number of attempts per quiz and the average received score per quiz were based on all submitted quizzes (at least one quiz, but there were several non-completers who did all four quizzes). The collected data did not fit a normal distribution, and therefore, non-parametric tests were applied. Those tests are also recommended [49] to cope with outliers, and the findings cannot be overly affected by those extreme values. The Kruskal–Wallis H-test was used to determine the existence of differences between the non-completers and the completers. The same test was used to determine the existence of differences between the completers from the engagement clusters and the difficulty-resolvers' clusters. The Mann–Whitney U-test was used to make pairwise comparisons between identified clusters.

3. Results

3.1. Performance in Assessments by Non-Completers and Completers

In the studied MOOC, 1004 participants submitted at least one programming task solution out of six. Throughout the course, the smallest number of attempts was 1, and the largest was 121. The course maximum was achieved by a completer (Figure 1). On average, a learner made 2.05 (SD = 3.931) attempts per programming task.

There were 1061 participants who did at least one quiz out of four. Throughout the course, the number of attempts ranged from 1 to 4, and only a small number of learners needed more than two attempts (Figure 2). The average number of attempts per quiz was 1.04 (SD = 0.218). Quiz scores ranged from 0 to 10, and the average score was 9.73 (SD = 1.063) per quiz.

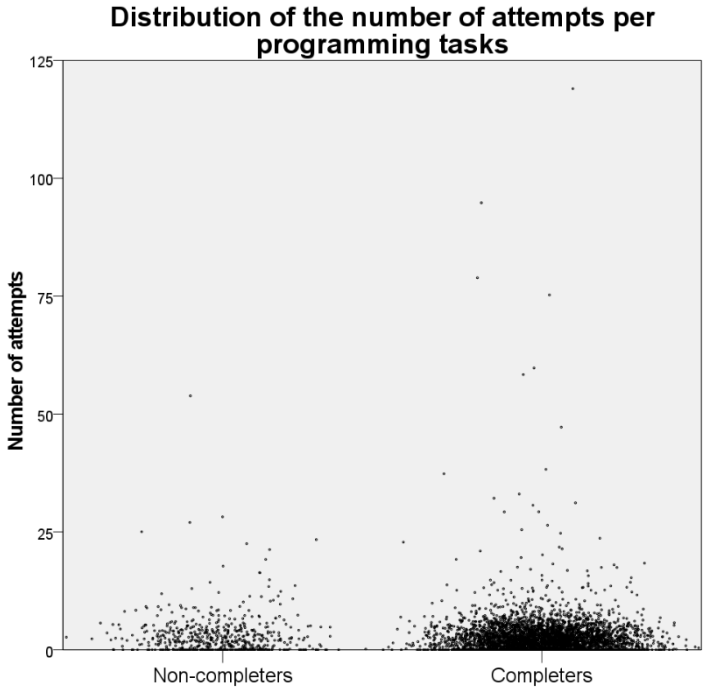

Figure 1. Distribution of the number of attempts per programming task while comparing non-completers and completers.

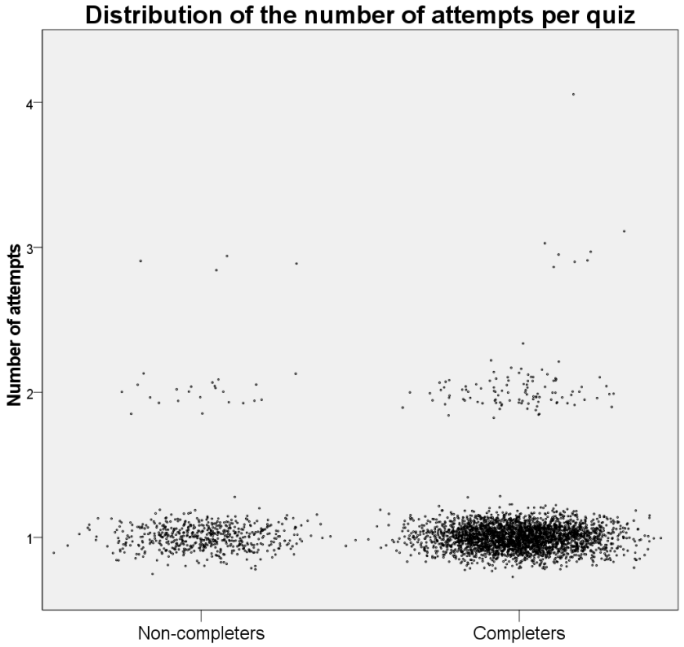

Figure 2. Distribution of the number of attempts per quiz while comparing non-completers and completers.

The Kruskal–Wallis H-test was employed to examine the difference between non-completers and completers (Table 2). The result showed that the learners in the non-completers' group made, on average, more attempts per programming task compared to completers ($p < 0.001$). No statistically significant difference was found regarding the average number of attempts to pass a quiz ($p > 0.05$). Non-completers received, on average, lower scores per quiz ($p < 0.001$).

Table 2. Attempts and scores by non-completers and completers across the course.

		Non-Completers	Completers	Kruskal–Wallis H
Attempts per programming task	n	231	773	
	range	1−55	1−121	
	mean (SD)	2.67 (4.107)	1.97 (3.902)	46.973 ***
Attempts per quiz	n	288	773	
	range	1−3	1−4	
	mean (SD)	1.05 (0.251)	1.04 (0.211)	1.130
Scores per quiz	n	288	773	
	range	0−10	9−10	
	mean (SD)	9.06 (2.423)	9.86 (0.345)	85.037 ***

*** $p < 0.001$.

It must be stressed that the number of learners in the group of non-completers who were trying to pass an assignment decreased steadily as the course progressed. Comparing non-completers and completers for each of the six programming tasks, we found that, only in the last task, there was no difference in the average number of attempts (H = 3.306; $p > 0.05$). For the last task, there were only 19 learners among non-completers who tried to solve it. In all other tasks, non-completers made on average more attempts (in all cases $p < 0.05$). In the context of each of the four quizzes, we found that only in the last quiz learners among non-completers made, on average, more attempts (H = 9.136; $p < 0.01$; n = 42 non-completers). Regarding the scores for each quiz, the learners in the non-completers' group always received lower scores compared to completers (in all cases, $p < 0.001$).

3.2. Performance in Assessments by Completers with Different Engagement Styles

Previously, we identified four engagement clusters based on completers' stated interaction with provided activities in the MOOC. The number of attempts per programming task varied between engagement clusters. As expected, in every engagement cluster, some completers passed the task on the first attempt. In the cluster of support required knowledge collectors there were completers who made up to 121 attempts; in all other clusters, this number was below 20 (Figure 3).

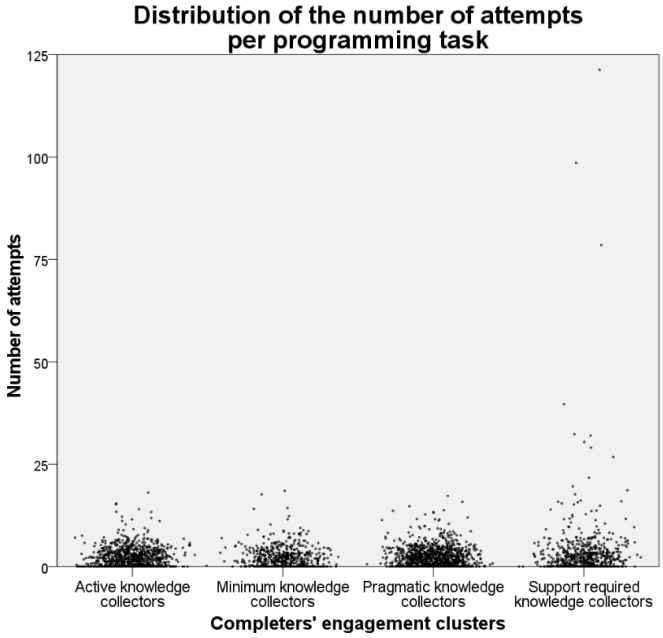

Figure 3. Distribution of the number of attempts per programming task while comparing completers with different engagement styles.

The Kruskal–Wallis test showed a difference between engagement clusters regarding the average number of attempts per programming task ($p < 0.001$; Table 3). The pairwise comparison revealed that support-required knowledge collectors made more attempts on average compared to all other clusters (with the Mann–Whitney U-test, $p < 0.01$ in all cases). Minimum knowledge collectors made more attempts on average compared to active and pragmatic knowledge collectors (U = 273,335.0 and 315,169.5, respectively; in both cases, $p < 0.05$). No significant difference was found between active and pragmatic knowledge collectors (U = 592,947.5; $p > 0.05$).

Table 3. Comparison of completers' engagement clusters across the course.

		Active Knowledge Collectors (n = 169)	Minimum Knowledge Collectors (n = 95)	Pragmatic Knowledge Collectors (n = 197)	Support-Required Knowledge Collectors (n = 114)	Kruskal–Wallis H
Attempts per programming task	range mean (SD)	1−19 1.62 (1.638)	1−18 1.77 (1.811)	1−15 1.63 (1.656)	1−121 2.91 (7.514)	45.500 ***
Attempts per quiz	range mean (SD)	1−3 1.02 (0.138)	1−3 1.06 (0.251)	1−2 1.03 (0.175)	1−4 1.07 (0.289)	17.662 ***
Scores per quiz	mean (SD)	9.92 (0.267)	9.81 (0.394)	9.90 (0.294)	9.82 (0.381)	47.774 ***

*** $p < 0.001$.

The number of attempts per quiz also varied between engagement clusters. Again, in every cluster, there were completers who passed the quiz on the first attempt. Support-required knowledge collectors made up to four attempts, while the completers from the pragmatic cluster were able to pass a quiz in two or fewer attempts. In all other clusters, this number was between one and three (Figure 4).

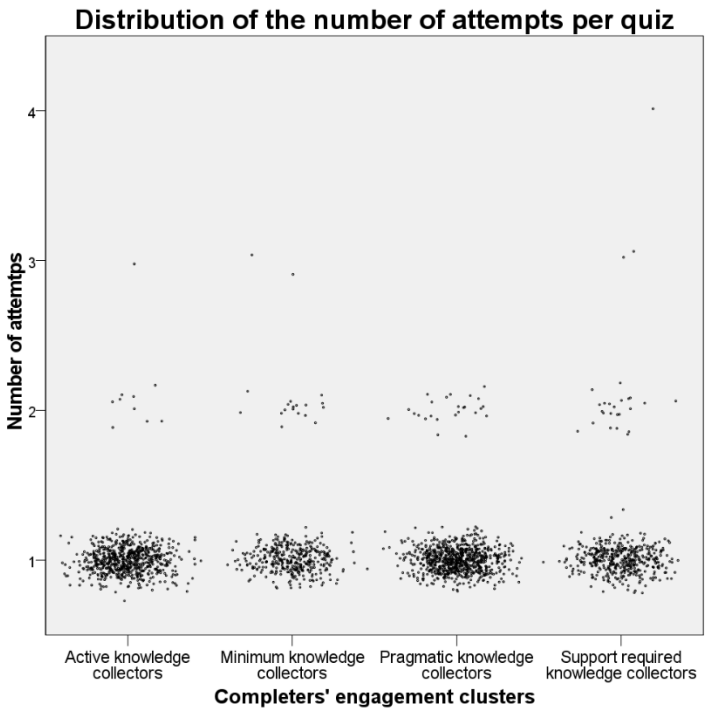

Figure 4. Distribution of the number of attempts per quiz while comparing completers with different engagement styles.

There was a difference found between engagement clusters in terms of the average number of attempts per quiz ($p < 0.001$; Table 3). The pairwise comparison revealed that active knowledge collectors made fewer attempts on average compared to all other clusters (with the Mann–Whitney U-test, $p < 0.05$ in all cases). Pragmatic knowledge collectors made fewer attempts on average compared to support required knowledge collectors

(U = 175,082.5; $p < 0.05$). For all other pairwise comparisons, no significant differences were found ($p > 0.05$).

Due to certificate requirements, the test scores ranged in all completers' clusters from 9 to 10 points (Table 3). Again, engagement clusters differed in terms of the received scores per quiz ($p < 0.001$). Active knowledge collectors achieved higher scores compared to minimum and support-required knowledge collectors (U = 113,646.0 and 138,944.0, respectively, in both cases $p < 0.001$). Pragmatic knowledge collectors received higher scores compared to completers from the minimum and support-required clusters (U = 135,208.0 and 165,244.0, respectively, in both cases $p < 0.001$). For all other pairwise comparisons, no significant differences were found ($p > 0.05$).

3.3. Performance in Assessments by Completers with Different Difficulty-Resolving Patterns

Previously, we identified five difficulty-resolvers' clusters in the MOOC. Each cluster had a set of preferred actions to resolve difficulties encountered in programming tasks. As expected, there were completers in every difficulty-resolvers' cluster, who passed the task on the first attempt. Up to 121 attempts were made by the completers in the cluster of step-by-step resolvers and up to 96 attempts in the cluster of social resolvers; in all other clusters, this number was below 40 (Figure 5).

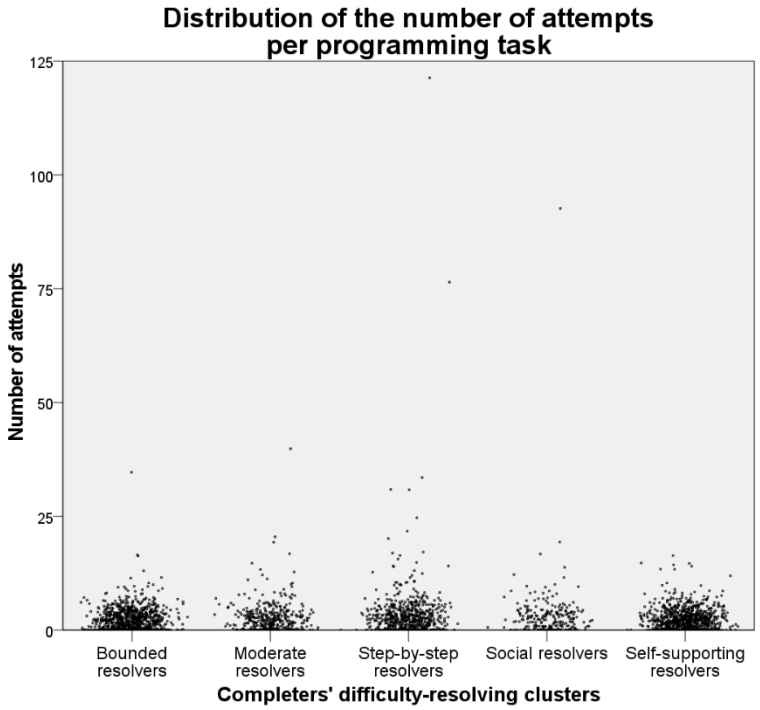

Figure 5. Distribution of the number of attempts per programming task while comparing completers with different difficulty-resolving patterns.

Difficulty-resolvers' clusters differed in the average number of attempts per programming task ($p < 0.001$; Table 4). Bounded resolvers made, on average, fewer attempts compared to all other clusters (with the Mann–Whitney U-test, $p < 0.05$ in all cases). Self-supporting resolvers made fewer attempts on average compared to moderate, step-by-step, and social resolvers (U = 205,785.0, 347,945.5, and 109,872.0, respectively; in all cases, $p < 0.05$). For all other pairwise comparisons, no significant differences were found ($p > 0.05$).

Table 4. Comparison of difficulty-resolvers' clusters across the course.

		Bounded Resolvers (n = 170)	Moderate Resolvers (n = 72)	Step-by-step Resolvers (n = 123)	Social Resolvers (n = 42)	Self-Supporting Resolvers (n = 168)	Kruskal–Wallis H
Attempts per programming task	Range	1−33	1−36	1−121	1−96	1−19	
	Mean (SD)	1.56 (1.766)	2.07 (2.904)	2.44 (6.095)	2.46 (6.330)	1.65 (1.583)	45.696 ***
Attempts per quiz	Range	1−2	1−2	1−4	1−3	1−3	
	Mean (SD)	1.01 (0.114)	1.02 (0.143)	1.06 (0.279)	1.08 (0.309)	1.04 (0.211)	21.648 ***
Scores per quiz	Mean (SD)	9.93 (0.259)	9.88 (0.327)	9.78 (0.416)	9.87 (0.338)	9.90 (0.296)	65.572 ***

*** $p < 0.001$.

The number of attempts per quiz also varied between difficulty-resolvers' clusters. Again, in every cluster, some completers passed the quiz on the first attempt. Step-by-step resolvers made up to four attempts and social and self-supporting resolvers were able to pass a quiz in three attempts, while the other resolvers made two or fewer attempts (Figure 6).

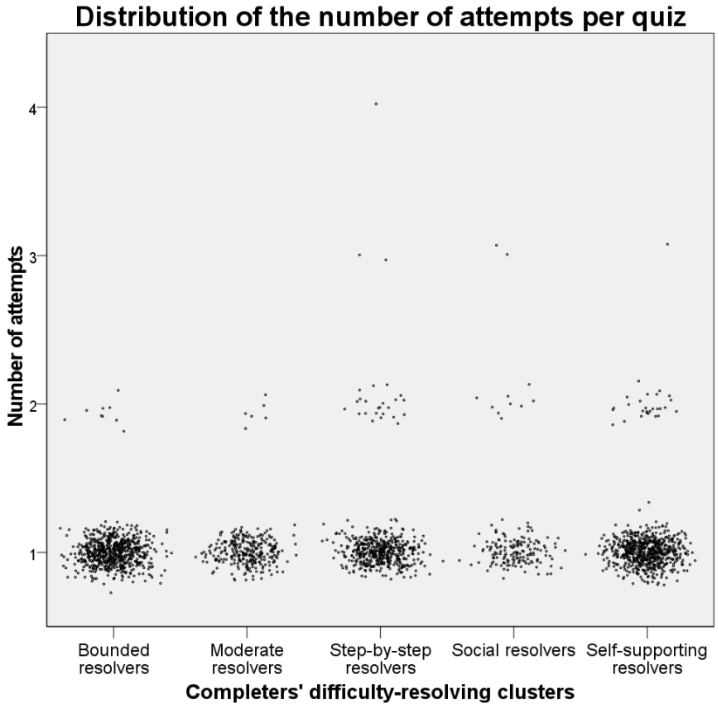

Figure 6. Distribution of the number of attempts per quiz while comparing completers with different difficulty-resolving patterns.

There was a difference between difficulty-resolvers' clusters in the average number of attempts per quiz ($p < 0.001$; Table 4). Bounded resolvers made on average fewer attempts compared to step-by-step, social, and self-supporting resolvers (with the Mann–Whitney U-test, $p < 0.01$ in all cases). Moderate resolvers made on average fewer attempts compared to step-by-step and social resolvers (U = 68,571.0 and 23,106.0, respectively, in both cases $p < 0.05$). For all other pairwise comparisons, no significant differences were found ($p > 0.05$).

Due to certificate requirements, the quiz scores ranged in all clusters from 9 to 10 points (Table 4). Difference was found between difficulty-resolvers' clusters regarding the received scores per quiz ($p < 0.001$). Bounded resolvers achieved higher results compared to moderate, step-by-step, and social resolvers (U = 93,076.0, 142,274.0, and 53,756.0, respectively; in all cases, $p < 0.05$). Step-by-step resolvers received lower scores compared to moderate, social, and self-supporting resolvers (U = 63,762.0, 37,584.0, and 144,678.0, respectively; in all cases, $p < 0.05$). For all other pairwise comparisons, no significant differences were found ($p > 0.05$).

4. Discussion

The paper examines how different groups of learners in a MOOC on programming vary in performance. Performance in assessment was defined as the number of attempts per programming tasks and per quiz and the numeric score received in a quiz.

4.1. Performance in Assessments by Non-Completers and Completers

First, we compared the performances in assessments between completers and non-completers. There was no difference in the average number of attempts per quiz between the two groups. One reason might be that learning materials could be used during the quizzes. However, non-completers received on average lower scores in quizzes than completers. It is possible that some non-completers did not aim for higher grades and stopped as soon as they reached the 90% threshold of correct answers per quiz. The findings of the Rõõm et al. [50] suggested that when non-completers struggle with a quiz, they tend to read materials more thoroughly and test their understanding with self-assessment questions, which help them achieve at least the minimum score required for passing the quiz. While Chen et al. [20] showed that completers in computer science courses make more submissions in problem sets, our results indicated that non-completers made on average more attempts per programming task than completers. A possible explanation would be that non-completers resubmit programming task solutions multiple times, without concentrating on the source of error [24], or they do nothing between the attempts [25]. In addition, Rõõm et al. [51] have found that if learners do not understand the logic of writing programming code, they solve it by trial and error.

4.2. Performance in Assessments by Completers with Different Engagement Styles

Next, we focused on studying completers' performance in assessments, since their strategies allowed them to pass the course successfully. We were interested in comparing performance in assessments between completers from different engagement clusters previously identified by Feklistova et al. [16]. Prior studies [31–33] have provided contradictory results about learners with different engagement styles. In our study, completers in each cluster engaged differently and might make different efforts to pass an assignment. Thus, the results indicated differences in performance between engagement clusters. Although the difference in the average number of attempts per quiz was quite small, it was still statistically significant. The findings indicated that in every cluster there were completers who provided perfect solutions or answers right off and those who were satisfied with achieving only the threshold value required for passing. At the same time, some completers might not achieve the required threshold value on the first attempt, or some completers like to make a perfect submission that could require several attempts. Previous studies have also found variation in the number of attempts [10,26–28].

We observed that completers, on average, made more attempts per programming task than per quiz. The highest number of attempts per programming task and a tendency to make more attempts per quiz, combined with a lower score per quiz, were seen among completers from the cluster of support required knowledge collectors who were more active in using different support mechanisms, including forums, and were highly engaged with the provided activities. This contradicts previous findings [33,37,38] that those who are the most active in discussion forums and have a high engagement level with watching videos demonstrate better overall performance and receive better scores in quizzes. It is possible that support-required knowledge collectors find the topic of programming more challenging to understand immediately or that the writing style of learning materials does not help them provide a solution on the first attempt. In spite of difficulties, they look for support and clues in all possible sources. Perhaps, the hints received from a forum or a helpdesk were interpreted by those completers in a wrong way, as many of them were inexperienced in programming and they may need additional attempts to fix errors. Since support-required knowledge collectors were active in using the learning materials, they

were able to find the correct answers to quizzes that were mainly focused on theoretical aspects covered in the materials.

In our study, two clusters of completers (active and pragmatic knowledge collectors) were similar: they made a low number of attempts per programming task and received better scores in quizzes. However, those clusters differed in engagement. The cluster of active knowledge collectors had a high engagement level with most provided activities, except the tools mainly used for asking support. The cluster of pragmatic knowledge collectors was focused mostly on the activities needed to complete the MOOC [16]. Some authors [21,34] have argued that the more students are visiting learning materials, writing comments and attempting to answer quiz questions, the better performance they demonstrate. Our findings suggest that in addition to those active learners with a moderate level of engagement demonstrated by pragmatic knowledge collectors, it is possible to pass a programming task with a small number of attempts and achieve good quiz scores. It is interesting that in terms of attempts per quiz, the pragmatic knowledge collectors made on average a little more attempts than active knowledge collectors. It might be because the active knowledge collectors engaged with most provided activities, including additional materials, and were able to get a broader overview of the topic, expanding and enriching their understanding of the subject field, and could use the additional knowledge in solving quizzes.

Our results indicated quite interesting findings about minimum knowledge collectors whose engagement level was quite low. Despite many of them having experience in programming, their average number of attempts per quiz and quiz scores did not differ from those who actively used support mechanisms and had, for the most part, never studied programming before. In addition, our results indicated that in comparison to active and pragmatic knowledge collectors, minimum knowledge collectors made on average more attempts per programming task, received lower quiz scores and tended to make on average more attempts per quiz. It is an interesting finding, as some authors have suggested that students with prior knowledge in programming [52,53] and lesser engagement with course materials due to previous knowledge in the subject field [35] tend to demonstrate good performance. Perhaps in a programming task, minimum knowledge collectors rely on prior subject knowledge and skills, but in most theoretical quizzes, they cannot achieve good results without reading materials and watching videos properly. Completers with a low engagement level might do just enough to pass the course, demonstrating the best results might not be their primary aim. Completers from the cluster of minimum knowledge collectors might approach assessments such as a challenge, since the number of allowed resubmissions was unlimited and could pass a task or quiz in several attempts without being afraid to fail the course.

4.3. Performance in Assessments by Completers with Different Difficulty-Resolving Patterns

In the current study, we also compared performance in assessments between completers with various difficulty-resolving patterns previously identified by Feklistova et al. [17]. Our results indicated that completers with various difficulty-resolving patterns differ in performance in assessments. Although the difference in the average number of attempts per quiz between different clusters is quite small, it is still statistically significant. It is logical, as different approaches might lead to different outcomes in resolving difficulties and, consequently, might influence the number of attempts per assessment and the result. Like the clusters of completers based on engagement styles, each cluster of difficulty-resolvers included both completers who achieved the required threshold value for passing the assessment on the first attempt and those who made several resubmissions to provide an acceptable programming task solution. The former approach is consistent with the open form of MOOCs where learners are not expected to receive a perfect grade [32].

For most beginners, it is quite usual to experience difficulties while solving a programming task. Learners may use a combination of help sources to cope with a problem. While Carter et al. [42] argued that those who receive more support get better grades, our

results indicate that those (moderate, step-by-step, and social resolvers) who actively used different help sources in case of difficulties tended to make on average more attempts to provide a workable programming task solution and received lower quiz scores. In our study, social resolvers who re-read the learning materials and were the most active in using troubleshooters and seeking help from forums and helpdesk tended to make on average more attempts per programming task and quiz and receive lower quiz scores. This finding contradicts previous results that the highest activity in a forum allows learners to receive the highest quiz scores [37] and demonstrate overall higher performance [33].

Similarly to Malekian et al. [25], our results indicate that reviewing learning materials is a good approach to obtaining a positive result. Two groups in our study (bounded and self-supporting resolvers) who rarely used troubleshooters or forums did not write to helpdesk or ask a friend for help. The first group, bounded resolvers, who mainly just re-read materials, demonstrated on average the smallest number of attempts per programming task and a tendency to receive better quiz scores. Those completers might use materials to refresh memory or clarify a point, and this is sufficient to achieve good results in a few attempts. In addition, completers from the cluster of bounded resolvers might apply a "first think then act" strategy as Karavirta et al. [24] have suggested. The second group, self-supporting resolvers, re-read learning materials and actively searched for materials on the Internet. Their performance in the programming task was second only to that of bounded resolvers, but their performance in the quizzes did not differ from moderate or social resolvers. Perhaps self-supporting resolvers relied too much on Internet resources and did not pay attention to learning materials where course instructors might provide practical help hints.

Step-by-step resolvers, besides learning materials actively used detailed guidelines such as troubleshooters, made the highest number of attempts per assessment and received the lowest quiz scores. This result contradicts the findings of Karavirta et al. [24] that those who use the resubmission option more than others finally get good exercise scores. Perhaps, step-by-step resolvers might refer to troubleshooters without any effort to correct an error themselves and submit a new version with an intention to check if the improvement is workable or not. However, as has been suggested in some studies [20,43], learners may abuse hints and get poor hands-on skills.

Our results indicate that the approach demonstrated by moderate resolvers in using various help resources leads to an average performance as sometimes those completers are as good as bounded resolvers and sometimes they are similar to the social ones. This group is also described in the work conducted by Karavirta et al. [24], where learners who received an average students' grade were not very active in resubmissions since their grade was good enough and they had done pretty much what they were expected to do to pass a course.

5. Conclusions

In the paper, we aimed to analyse the performance of non-completers and completers in assessments in a MOOC on programming. Firstly, we found that non-completers made on average more attempts per programming task than completers but no statistically significant difference between those groups was found in relation to the quiz. Completers received higher quiz scores than non-completers. Secondly, our study provided insight into understanding how different strategies of engagement and difficulties resolving contribute to MOOC learners' performance. Active engagement with support mechanisms leads to more attempts per assessment and lower quiz scores. Very good performance in a course on programming can be achieved already at the pragmatic engagement level and with reasonable usage of help resources. We found that the more different sources of help are actively used in case of difficulties the more attempts tend to be made to provide a workable programming task solution and the lower quiz scores are received. Completers on average made more attempts per programming task than per quiz.

Our findings can serve as the foundation to guide MOOC instructors. They can consider these results while developing courses, for example, in supporting certain groups of participants. For instance, those who actively use support mechanisms and appreciate guidelines often make only minor changes in a code at a time, and by resubmitting it they verify the correctness of the improvements made. Thus, giving more attempts or even an unlimited number of attempts can help those participants complete MOOCs successfully. In addition, instructors can use the findings to inform participants that it is normal for learners with different engagement styles and difficulty-resolving patterns to make different numbers of attempts to provide workable solutions and correct answers as well as to achieve different results in quizzes. Course instructors can encourage newcomers that previous course runs' completers have engaged and resolved difficulties in different ways and have passed the course successfully. Taking into account a variety of MOOC participants, course instructors may also consider the need to differentiate assignments or provide task descriptions with varying degrees of detail.

The novelty of this study was not only in investigating not solely non-completers and completers in a MOOC as was done previously, but also in relating the performance to the different completers' strategies. In addition, the originality of this study lies in analysing different learner groups' performance in two assessment types (programming tasks and quizzes) simultaneously used in the same course run. We found the following interventions improved the understanding of how two different learning aspects—engagement and difficulty-resolving—contribute to completers' performance in assessments.

The present study has some limitations that have to be pointed out when generalising the findings. We studied completers of one Estonian-language MOOC on programming on Python only, and the course lasted only four weeks. It should be noted that some completers did not consent to the use of their performance data. This caused the reduction of the sample size used to study performance demonstrated by completers from different engagement and difficulty-resolving clusters. In the study, we analysed how many attempts on average learner made per programming task and per quiz and how high his/her average quiz score was. It would be useful for future research to find out how the numbers of attempts and quiz scores by non-completers and completers in each cluster change during the course, in dynamics, and how different learners cope with each programming task and quiz.

Author Contributions: Conceptualization, L.F., M.L. and P.L.; data curation, L.F., M.L. and P.L.; formal analysis, L.F. and P.L.; investigation, L.F., M.L. and P.L.; methodology, L.F., M.L. and P.L.; resources, L.F.; writing of the original draft, L.F.; writing—review & editing, L.F., M.L. and P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was approved by the Ethics Committee of the University of Tartu, Estonia (protocol code 341/T-2, 17 May 2021).

Informed Consent Statement: Informed consent was obtained from all course participants involved in the study.

Data Availability Statement: Data are available from the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Janakiraman, S.; Watson, S.L.; Watson, W.R. Adult learners use of self-directed learning strategies in a massive open online course. *J. Ethnogr. Qual. Res.* **2018**, *13*, 122–133.
2. Hannafin, M.J.; Hannafin, K.M. Cognition and Student-Centered, Web-Based Learning: Issues and Implications for Research and Theory. In *Learning and Instruction in the Digital Age*; Spector, J.M., Ifenthaler, D., Isaias, P., Kinshuk-Sampson, D., Eds.; Springer: Boston, MA, USA, 2010; pp. 11–23. [CrossRef]
3. Brooker, A.; Corrin, L.; de Barba, P.; Lodge, J.; Kennedy, G. A tale of two MOOCs: How student motivation and participation predict learning outcomes in different MOOCs. *Australas. J. Educ. Technol.* **2018**, *34*, 73–87. [CrossRef]

4. Soffer, T.; Cohen, A. Students' engagement characteristics predict success and completion of online courses. *J. Comput. Assist. Learn.* **2019**, *35*, 378–389. [CrossRef]
5. Hadi, S.M.; Rawson, R. Driving Learner Engagement and Completion within MOOCs: A Case for Structured Learning Support. In Proceedings of the European Stakeholder Summit on Experiences and Best Practices In and Around MOOCs, Graz, Austria, 22–24 February 2016; pp. 81–93.
6. Lepp, M.; Palts, T.; Luik, P.; Papli, K.; Suviste, R.; Säde, M.; Hollo, K.; Vaherpuu, V.; Tõnisson, E. Troubleshooters for Tasks of Introductory Programming MOOCs. *Int. Rev. Res. Open Dis.* **2018**, *19*, 56–75. [CrossRef]
7. Kizilcec, R.F.; Piech, C.; Schneider, E. Deconstructing Disengagement: Analyzing Learner Subpopulations in Massive Open Online Courses. In Proceedings of the Third International Conference on Learning Analytics and Knowledge, Leuven, Belgium, 8–13 April 2013; pp. 170–179. [CrossRef]
8. DeBoer, J.; Breslow, L. Tracking Progress: Predictors of Students' Weekly Achievement During a Circuits and Electronics MOOC. In Proceedings of the First ACM Conference on Learning @ Scale Conference, Atlanta, GA, USA, 4–5 March 2014; pp. 169–170. [CrossRef]
9. Do, C.B.; Chen, Z.; Brandman, R.; Koller, D. Self-driven mastery in massive open online courses. *MOOCs Forum* **2013**, *1*, 14–16. [CrossRef]
10. Auvinen, T. Harmful Study Habits in Online Learning Environments with Automatic Assessment. In Proceedings of the 2015 International Conference on Learning and Teaching in Computing and Engineering, Taipei, Taiwan, 9–12 April 2015; pp. 50–57. [CrossRef]
11. Pereira, F.; Oliveira, E.; Fernandes, D.; Carvalho, L.S.G.; Junior, H. Otimização e Automação da Predição Precoce do Desempenho de Alunos Que Utilizam Juízes Online: Uma Abordagem Com Algoritmo Genético. In Proceedings of the Simpósio Brasileiro De Informática Na Educação, Brasília, DF, Brasil, 11–14 November 2019; pp. 1451–1460.
12. Chen, Y.; Zhang, M. MOOC Student Dropout: Pattern and Prevention. In Proceedings of the ACM Turing 50th Celebration Conference, Shanghai, China, 12–14 May 2017; pp. 1–6. [CrossRef]
13. Chang, R.I.; Hung, Y.H.; Lin, C.F. Survey of learning experiences and influence of learning style preferences on user intentions regarding MOOCs. *Br. J. Educ. Technol.* **2015**, *46*, 528–541. [CrossRef]
14. Zheng, S.; Rosson, M.B.; Shih, P.C.; Carroll, J.M. Understanding Student Motivation, Behaviors and Perceptions in MOOCs. In Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, Vancouver, BC, Canada, 14–18 March 2015; pp. 1882–1895. [CrossRef]
15. Hew, K.F.; Cheung, W.S. Students' and instructors' use of massive open online courses (MOOCs); motivations and challenges. *Educ. Res. Rev.* **2014**, *12*, 45–58. [CrossRef]
16. Feklistova, L.; Lepp, M.; Luik, P. Completers' Engagement Clusters in Programming MOOC: The Case of Estonia. In Proceedings of the 12th Annual International Conference of Education, Research and Innovation, Seville, Spain, 11–13 November 2019; pp. 1119–1126. [CrossRef]
17. Feklistova, L.; Luik, P.; Lepp, M. Clusters of Programming Exercises Difficulties Resolvers in a MOOC. In Proceedings of the 19th European Conference on e-Learning, Berlin, Germany, 28–30 October 2020; pp. 563–569. [CrossRef]
18. Soderstrom, N.; Bjork, R. Learning versus performance: An integrative review. *Perspect. Psychol. Sci.* **2015**, *10*, 176–199. [CrossRef]
19. Cristea, A.; Alshehri, M.; Alamri, A.; Kayama, M.; Stewart, C.; Shi, L. How Is Learning Fluctuating? FutureLearn MOOCs Fine-Grained Temporal Analysis and Feedback to Teachers and Designers. In Proceedings of the 27th International Conference on Information System Development, Lund, Sweden, 22–24 August 2018; Available online: https://dro.dur.ac.uk/25775/1/25775.pdf (accessed on 25 May 2021).
20. Chen, C.; Sonnert, G.; Sadler, P.M.; Malan, D.J. Computational thinking and assignment resubmission predict persistence in a computer science MOOC. *J. Comput. Assist. Learn.* **2020**, *36*, 581–594. [CrossRef]
21. de Barba, P.G.; Kennedy, G.E.; Ainley, M.D. The role of students' motivation and participation in predicting performance in a MOOC. *J. Comput. Assist. Learn.* **2016**, *32*, 218–231. [CrossRef]
22. Abbakumov, D.; Desmet, P.; Van den Noortgate, W. Measuring student's proficiency in MOOCs: Multiple attempts extensions for the Rasch model. *Heliyon* **2018**, *4*, 1–15. [CrossRef]
23. Rushkin, I.; Chuang, I.; Tingley, D. Modelling and using response times in online courses. *J. Learn. Anal.* **2019**, *6*, 76–89. [CrossRef]
24. Karavirta, V.; Korhonen, A.; Malmi, L. On the use of resubmissions in automatic assessment systems. *Comput. Sci. Educ.* **2006**, *16*, 229–240. [CrossRef]
25. Malekian, D.; Bailey, J.; Kennedy, G. Prediction of Students' Assessment Readiness in Online Learning Environments: The Sequence Matters. In Proceedings of the Tenth International Conference on Learning Analytics and Knowledge, Frankfurt, Germany, 23–27 March 2020; pp. 382–391. [CrossRef]
26. Sharma, K.; Kidzinski, Ł.; Jermann, P.; Dillenbourg, P. Towards Predicting Success in MOOCs: Programming Assignments. In Proceedings of the European Stakeholder Summit on Experiences and Best Practices in and around MOOCs, Graz, Austria, 22–24 February 2016; pp. 135–147.
27. Edwards, S.H.; Snyder, J.; Pérez-Quiñones, M.A.; Allevato, A.; Kim, D.; Tretola, B. Comparing Effective and Ineffective Behaviors of Student Programmers. In Proceedings of the Fifth International Workshop on Computing Education Research Workshop, Berkeley, CA, USA, 10–11 August 2009; pp. 3–14. [CrossRef]

28. Kennedy, G.; Coffrin, C.; de Barba, P.; Corrin, L. Predicting Success: How Learners' Prior Knowledge, Skills and Activities Predict MOOC Performance. In Proceedings of the Fifth International Conference on Learning Analytics and Knowledge, Poughkeepsie, NY, USA, 16–20 March 2015; pp. 136–140. [CrossRef]
29. Pereira, F.D.; Oliveira, E.H.T.; Oliveira, D.B.F.; Cristea, A.I.; Carvalho, L.S.G.; Fonseca, S.C.; Toda, A.; Isotani, S. Using learning analytics in the Amazonas: Understanding students' behaviour in introductory programming. *Br. J. Educ. Technol.* **2020**, *51*, 955–972. [CrossRef]
30. Anderson, A.; Huttenlocher, D.; Kleinberg, J.; Leskovec, J. Engaging with Massive Online Courses. In Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, 7–11 April 2014; pp. 687–698. [CrossRef]
31. Byrne, P.; Lyons, G. The Effect of Student Attributes on Success in Programming. In Proceedings of the 6th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, Canterbury, UK, 25–27 June 2001; pp. 49–52. [CrossRef]
32. Deng, R.; Benckendorff, P.; Gannaway, D. Linking learner factors, teaching context, and engagement patterns with MOOC learning outcomes. *J. Comput. Assist. Learn.* **2020**, *36*, 688–708. [CrossRef]
33. Tseng, S.; Tsao, Y.-W.; Yu, L.-C.; Chan, C.-L.; Lai, K.R. Who will pass? Analyzing learner behaviors in MOOCs. *Res. Pract. Technol. Enhanc. Learn.* **2016**, *11*, 1–11. [CrossRef]
34. Shi, L.; Cristea, A. In-depth exploration of engagement patterns in MOOCs. In *Web Information Systems Engineering—WISE 2018*; Hacid, H., Cellary, W., Wang, H., Paik, H.Y., Zhou, R., Eds.; Springer: Cham, Switzerland, 2018; Volume 11234, pp. 395–409. [CrossRef]
35. Arora, S.; Goel, M.; Sabitha, A.S.; Mehrotra, D. Learner groups in massive open online courses. *Am. J. Distance Educ.* **2017**, *31*, 80–97. [CrossRef]
36. Koedinger, K.R.; Kim, J.; Jia, J.Z.; McLaughlin, E.A.; Bier, N.L. Learning Is Not a Spectator Sport: Doing Is Better Than Watching for Learning From a MOOC. In Proceedings of the Second (2015) ACM Conference on Learning @ Scale, Vancouver, BC, Canada, 14–18 March 2015; pp. 111–120. [CrossRef]
37. Kahan, T.; Soffer, T.; Nachmias, R. Types of participant behavior in a massive open online course. *Int. Rev. Res. Open Dis.* **2017**, *18*, 1–18. [CrossRef]
38. Khalil, M.; Ebner, M. Clustering patterns of engagement in massive open online courses (MOOCs): The use of learning analytics to reveal student categories. *J. Comput. High.* **2017**, *29*, 114–132. [CrossRef]
39. Wilkowski, J.; Deutsch, A.; Russell, D.M. Student Skill and Goal Achievement in the Mapping with Google MOOC. In Proceedings of the First ACM Conference on Learning @ Scale Conference, Atlanta, GA, USA, 4–5 March 2014; pp. 3–10. [CrossRef]
40. Murphy, C.; Kaiser, G.; Loveland, K.; Hasan, S. Retina: Helping Students and Instructors Based on Observed Programming Activities. In Proceedings of the 40th ACM Technical Symposium on Computer Science Education, Chattanooga, TN, USA, 4–7 March 2009; pp. 178–182. [CrossRef]
41. Nelimarkka, M.; Hellas, A. Social Help-Seeking Strategies in a Programming MOOC. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Baltimore, MD, USA, 21–24 February 2018; pp. 116–121. [CrossRef]
42. Carter, J.; Dewan, P.; Pichiliani, M. Towards Incremental Separation of Surmountable and Insurmountable Programming Difficulties. In Proceedings of the 46th ACM Technical Symposium on Computer Science, Kansas City, MO, USA, 4–7 March 2015; pp. 241–246. [CrossRef]
43. Estey, A.; Coady, Y. Can Interaction Patterns with Supplemental Study Tools Predict Outcomes in CS1? In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, Arequipa, Peru, 11–13 July 2016; pp. 236–241. [CrossRef]
44. Thomas, L.; Ratcliffe, M.; Woodbury, J.; Jarman, E. Learning Styles and Performance in the Introductory Programming Sequence. In Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, Cincinnati, KY, USA,, 27 February–3 March 2002; pp. 33–37. [CrossRef]
45. Lepp, M.; Luik, P.; Palts, T.; Papli, K.; Suviste, R.; Säde, M.; Tõnisson, E. MOOC in Programming: A Success Story. In Proceedings of the 12th International Conference on e-Learning, Orlando, FL, USA, 1–2 June 2017; pp. 138–147.
46. Luik, P.; Feklistova, L.; Lepp, M.; Tõnisson, E.; Suviste, R.; Gaiduk, M.; Säde, M.; Palts, T. Participants and Completers in Programming MOOCs. *Educ. Inf. Technol.* **2019**, *24*, 3689–3706. [CrossRef]
47. Luik, P.; Suviste, R.; Lepp, M.; Palts, T.; Tõnisson, E.; Säde, M.; Papli, K. What Motivates Enrolment in Programming MOOCs? *Br. J. Educ. Technol.* **2019**, *50*, 153–165. [CrossRef]
48. Thonny. Python IDE for Beginners. Available online: https://thonny.org/ (accessed on 25 August 2021).
49. Frost, J. Guidelines for Removing and Handling Outliers in Data. Available online: https://statisticsbyjim.com/basics/remove-outliers/ (accessed on 25 August 2021).
50. Rõõm, M.; Luik, P.; Lepp, M. Learners' Sequence of Course Activities During Computer Programming MOOC. In Proceedings of the 19th European Conference on e-Learning, Berlin, Germany, 28–30 October 2020; pp. 452–459. [CrossRef]
51. Rõõm, M.; Luik, P.; Lepp, M. Learners' Use of Time in MOOCs About Programming. In Proceedings of the 12th International Conference on Education and New Learning Technologies, Online Conference, 6–7 July 2020; pp. 4380–4387. [CrossRef]

52. Hagan, D.; Markham, S. Does It Help to Have Some Programming Experience Before Beginning a Computing Degree Program? In Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland, 11–13 July 2000; pp. 25–28. [CrossRef]
53. Morrison, M.; Newman, T.S. A Study of the Impact of Student Background and Preparedness on Outcomes in CS I. In Proceedings of the Thirty-Second SIGCSE Technical Symposium on Computer Science Education, Charlotte, NC, USA, 21–25 February 2001; pp. 179–183. [CrossRef]