Evaluating the Use and Effectiveness of Ungraded Practice Problems in an Introductory Programming Course

Caleb O'Malley
Department of Engineering Education
University of Florida
Gainesville, Florida, USA
calebomalley@ufl.edu

Ashish Aggarwal
Department of Engineering Education
University of Florida
Gainesville, Florida, USA
ashishjuit@ufl.edu

ABSTRACT

Educational researchers have been interested in finding out factors which are pivotal in a students' success within any course. However, less is known about students' engagement with optional course content and its effect on learning outcomes. Optional content is any ungraded component of the course available to students for additional practice. In our context, it is ungraded quizzes based on concepts discussed in an introductory programming course. In this paper, we present the methodology, analysis, and results of a study concerned with how students engage with optional quizzes and what effects this content may have on students' learning. We find that before the midterm exam, over half of all students completed at least one quiz of the four available, while a third of students completed all available quizzes. Leading up to the midterm exam, we observed a large increase in submissions. During the second half of the semester, overall participation decreased slightly. Again, leading up to the final exam, students' submissions became more frequent. When investigating correlations between quiz completion and student performance, notable differences were observed between the highest and lowest levels of quiz completion. The results of this study will help computer science educators in understanding how students utilize optional content similar to ours and further guide in improving the effectiveness of such content, especially in the context of introductory programming courses. These insights will help to guide the creation and implementation of optional practice problems, with the goal to improve the student's overall experience of the course.

CCS CONCEPTS

• **Social and professional topics ~** Computer science education; CS1

KEYWORDS

Optional programming exercises, CS1, course approach, non-majors, practice

ACE'20, February 3–7, 2020, Melbourne, VIC, Australia

ACM 978-1-4503-7686-0/20/02…$15.00
https://doi.org/10.1145/3373165.3373185

ACM Reference format:

Caleb O'Malley and Ashish Aggarwal. 2020. Evaluating the Use and Effectiveness of Ungraded Practice Problems in an Introductory Programming Course. In *Proceedings of 22nd Australasian Computing Education Conference (ACE'20). ACM, New York, NY, USA, 8 pages.* https://doi.org/10.1145/3373165.3373185

1 Background

Introductory programming courses are often cited for their high attrition rates and overall difficulty [3, 15]. In an effort to improve student learning outcomes, instructors and researchers attempt to understand what factors are crucial to a student's performance, and create new teaching tools and strategies in the process. It is essential for instructors who intend to incorporate new components into their courses to know what to expect. By providing more insight into how students interact with tools, as well as the impact these tools may have on student engagement and performance, instructors can make more informed decisions.

1.1 Teaching Tools

Various teaching tools and strategies have been studied for their ability to enhance the students' learning experience. Some of the most well-known among these include the flipped classroom [9], pair programming [12, 16], and different styles of short practice problems [1, 2, 8].

McDowell et al. [12] report that pair programming is an essential tool for improving student retention, programming confidence, and persistence in computer science related majors. Another study from Wood et al. [16] also finds that pair programming is a valuable tool for instructors in CS1 courses. The students involved in this study provided positive feedback about pair programming, and exhibited increased motivation, engagement, and performance.

In a study focused on unannounced, or "pop" quizzes, Cicirello [5] found that pop quizzes improved student performance on both exams and programming assignments. Effects also varied according to students' majors and class year.

Allen et al. [1] reported that the implementation of many small programs (MSPs), as opposed to one large program (OLP), into CS1 courses yielded happier students with better grades.

1.2 Optional Practice Problems/Content

Student interactions with course material have also been studied extensively, taking place in many different contexts.

Allen et al. [2] performed a follow-up study that described student usage of the MSPs, which observed that student interaction with the MSPs was also very positive. Students often

completed more problems than required, completed them earlier than OLPs, and used them for exam preparation.

Edgcomb et al. [7] studied the extent to which students completed assignments earnestly. They found that for the most part, students earnestly completed assignments and few "cheated the system". Although this material was not optional, they concluded that students are more likely to earnestly complete material if points are awarded towards their final grade, if the content is well-designed, and if the amount of work is sizable.

Additional research on the viewability of homework answers [17] finds that students will usually complete homework earnestly rather than looking at the answer for immediate credit, but that this earnestness decreases over the semester, likely due to fatigue and difficulty of the content.

Leppänen et al. [11] found, through machine learning methods, that a student's performance can be accurately predicted by their usage of online content within about 3-4 weeks of the start of the semester.

Edwards et al. [8] reported on the use of an open source drill-and-practice system consisting of short programming problems. Students appeared to enjoy using the system and were able to see the benefits that a system like this offered. Student participation in this optional content was found to be linked to exam performance, specifically for code-writing questions.

1.3 Motivation

Enrollment in computer science courses continues to see a boom, drawing in non-majors and students of varying levels of prior programming experience [4]. One study from Sax et al. [14] found that many non-computing majors are enrolling in introductory programming courses, many of which are female students. These students and many more find themselves enrolling in these courses due to interest, rather than a degree requirement.

CS educators must grapple with the challenge of making their courses more effective for all types of students. This can be difficult, as tailoring content and personalizing learning becomes more difficult with scale. An instructor may not want to require a great deal of content pertaining to a single concept for students who have prior programming experience, as they may already be quite familiar with the concept. However, this repetition might be useful for students who do not have as much prior programming experience. One solution is to provide some optional ungraded practice problems to students, so that they can be used by students who need extra practice, while simultaneously not requiring everyone to complete it. However, less is known about the usefulness and effectiveness of such optional content. Do the students who normally excel in their coursework overall tend to use it more? Or do the students who think they need more practice use it more often? This is important to know, because understanding the student's approach to the course helps educators decide what improves students' learning.

To provide more insight into this question, we are studying the usage of optional content in an introductory programming course for non-majors, and analyzing whether or not this extra practice benefits students. We believe that the results of this analysis will help instructors to better design their courses according to student expectations, providing a richer experience for all students.

1.4 Research Questions

This study seeks to answer the following research questions:

1. How do students interact with the optional content in an introductory programming course?
    a. How many students complete the content?
    b. When do students complete the content?
    c. Why do students complete the content?
2. How effective is this optional content in improving student learning outcomes?

2 Methods

2.1 Course

The study took place during the Spring semester of a two-credit introductory programming course designed for engineering students in a public R1 research institution in the Southeast U.S.A. The course is carried out both online and in person, providing traditional lectures which are recorded and then made available online immediately. This course uses MATLAB as its programming language. The workload of the course consists of weekly project-based homework assignments intended to be moderately challenging. Students are able to seek help from other students, teaching assistants, or the internet to complete these assignments. The students take two exams; a midterm and a final. For studying and learning purposes, students have access to practice exams as well as the optional online quizzes which are the focus of this study. A total of 169 students are taken into account by this study.

2.2 Demographics

A survey was made available online at the end of the semester, and students received a few extra credit points for completing it. The survey asked for the student's name, gender, class year, major, and amount of prior programming experience. The programming experience question was framed as time spent programming, and the answer choices are listed as follows: "No prior experience", "Between 1 to 10 hours", "Between 11 to 100 hours", "Between 101 to 500 hours", "I am a software developer", or "I invented a programming language". The students' responses are broken down below:

Table 1. Majors of student population.

Major	Number of Students
Mechanical Engineering	73 (43%)
Aerospace Engineering	42 (25%)
Civil Engineering	26 (15%)
Material Science Engineering	13 (8%)
Other Engineering	14 (8%)
Undeclared	1 (1%)

By major, the class is mostly mechanical engineering, aerospace, and civil engineering.

By gender, 78% of students are male and 22% are female. No students chose alternate answers.

By class year, 55% of students are freshmen, 30% are sophomores, 14% are juniors, and 1% are seniors.

Table 2. Prior programming experience of student population.

Prior Programming Experience	Number of Students
"No prior experience"	78 (46%)
"Between 1 and 10 Hours"	28 (17%)
"Between 11 and 100 Hours"	49 (29%)
"Between 101 and 500 Hours"	14 (8%)

In terms of prior programming experience, the class was split roughly halfway between having no prior experience and having at least some prior experience.

Overall, the class is heavily dominated by male students, and nearly all students are enrolled in an engineering major. Over three-fourths of the students are either first or second-year students, and the major categories of prior experience include "No prior experience" and "Between 11 and 100 hours".

2.3 Optional Quizzes

For this study, optional content is provided in the form of weekly optional quizzes. These practice quizzes are posted periodically throughout the semester on the course's learning management system (LMS). Four quizzes were released before the midterm exam, with the first quiz being released about a month before the exam and the fourth quiz being released one week before the exam. This timeline also holds for the other four quizzes which were released after the midterm, before the final exam.

Each quiz consists of roughly ten questions of increasing difficulty, all focused on a single concept. The quizzes are intended to solidify foundational concepts, and only include content already covered in lecture. The questions are created to test program reasoning skills. Most questions include a code snippet which students must trace through in order to answer the question, either providing the output of the program or providing some missing link to make the program function properly. The questions are similar to exam questions, but are not as difficult. Students are able to submit attempts for these quizzes an unlimited number of times, and the correct answers are not shown to them after submitting an attempt.

In our context, students are not shown the answers due to the fact that most questions contain a snippet of code. If a student is struggling with a question, they can use MATLAB to run the code and understand it in this way. This type of learning is encouraged over simply reading the correct answer from the screen.

Students were given unlimited attempts at the quizzes so that they would be able to use this resource whenever they felt appropriate, and however much they desired.

2.3.1 Exclusion of Data Points

While sorting through the quiz submission data, it became apparent that two primary types of submissions existed. Most submissions come from students simply completing the quiz and trying to earn a good score. Some other submissions, however, come from students who only answer one or two questions in order to check if they get those specific questions right. Because quiz completion is being studied as a possible factor of course performance in this study, we needed to guarantee that only meaningful interactions with the optional content were part of our data set.

To account for this, a criterion was created to ensure that the attempts being considered in our analysis were representative of students putting in effort to understand the concepts. In order to be considered a meaningful quiz attempt, the student must have answered at least half of the questions on the quiz. Nearly all of the attempts that ended up being excluded from analysis contained only one or two answers.

While this data was excluded from the more formal statistical analysis, the trends in this data is discussed in analysis section 4.1.4.

2.4 Data Collection

Data was collected in two primary ways. Throughout the semester, the students completed the optional quizzes online through the course's LMS. Reports were generated for each quiz, which included the number of attempts, time submitted, and scores. Data was also collected via the end-of-course survey, which allowed us to collect demographic information about the students.

The end-of-course survey was also a source of student feedback. Students provided responses to questions regarding attitudes towards programming and the course in general, as well as the resources they found most useful during the semester.

While students are still able to access old quizzes up until the end of the semester, only submissions that are completed before the upcoming exam are counted. This is because the students' quiz completion is being analyzed as a possible causal factor of student success.

3 Analysis

3.1 Quiz Usage

When analyzing how students interact with the optional quizzes, the data considered are the number of students completing each quiz, the number of attempts by students on each quiz, the general time at which the student completes the quizzes, and whether or not the student completed the quiz within twenty-four hours of the upcoming exam. Students with varying levels of prior programming experience were also analyzed to check if students with higher or lower levels of prior experience appear either more or less likely to complete optional quizzes.

3.1.1 Effect on Student Performance

A student's performance in the course is defined as their exam scores. Exam scores were chosen to represent student performance because other metrics, such as final grade or assignment average, are fairly high among most students. Students are given a few days to complete homework assignments, meaning they can seek help from peers, teaching assistants, and the internet. Therefore, many students' average assignment scores are quite high and are similar across the sample. The data from exam scores, on the other hand, contains more variation and is a better representation of a student's understanding of the content.

The variables used when analyzing student performance are quiz completion and prior programming experience. When looking at how many of the quizzes most students completed, it becomes apparent that three major groups exist. Students were categorized as completing all quizzes, some of the quizzes, or none of the quizzes.

Table 3. Quiz completion data for all quizzes throughout the semester.

Quizzes before midterm exam			Quizzes before final exam		
Quiz Number	Total Students Completing	First Attempt Within 24 hours of Midterm	Quiz Number	Total Students Completing	First Attempt Within 24 hours of Final
1	107	55 (51%)	5	84	34 (40%)
2	91	58 (64%)	6	81	39 (48%)
3	80	63 (79%)	7	79	44 (56%)
4	67	51 (76%)	8	67	45 (67%)

Prior programming experience (PPE) was also tested for its possible impact on student performance. A student with PPE = 0 reported having either "No prior experience" or "Between 1-10 hours" of experience. A student with PPE = 1 consists of all students who reported having more experience in programming.

3.2 Tests for Assumptions

Before performing statistical tests, the exam data was tested for assumptions of normality, homogeneity of variance, and independence. All statistical analysis was conducted in IBM-SPSS 25.

The assumption of normality was tested and was not met via examination of the residuals. The Shapiro-Wilk (S-W) test for normality (SW = 0.948, df = 169, $p < 0.001$) and skewness (-0.944) and kurtosis (1.341) statistics show that the data is negatively skewed for the exam average. The boxplot for this data demonstrated a similar negatively skewed shape, as did the histogram and Q-Q plot. These plots are not included for the sake of brevity. According to Levene's test, the homogeneity of variance assumption was satisfied [$F(5, 163) = 1.395$, $p = 0.229$]. The data's independence is limited due to the students being in the same course and institution.

3.3 T-tests Comparing Means

Before performing a t-test, the exam scores of each group of students is compared via an F-test. This test checks if the variances of the two groups are similar. The t-test follows the F-test and either assumes equal variances or unequal variances based on the results of the F-test. The t-stat used to determine significance is two-tailed, checking if each group of students performed either significantly better or worse than the other group. An alpha value of 0.05 was used to determine significance for all tests.

Students with PPE = 0 were compared to students with PPE = 1 to check for significant differences between the two groups' mean exam scores.

Students who completed all optional quizzes were compared to students who completed none of the quizzes to check for significant differences between the students' mean exam scores.

Students were also divided into subgroups. The group of students with PPE = 0 who completed all quizzes was compared to the group of students with PPE = 0 who completed none of the quizzes, and vice versa for students with PPE = 1.

3.4 Factorial ANOVA

In addition to the t-tests which compare the means between different groups of students, a factorial ANOVA was conducted to determine if the mean exam score achieved by students differed based on their prior programming experience (PPE = 0 or 1) and the students' level of quiz completion (None, some, or all). Line plots using the estimated marginal means of each group were also generated to visualize the interaction and effects of each factor on mean exam score.

One factorial ANOVA was performed for each exam, as well as for the average of the two exam scores. When performing the ANOVA for midterm scores, the quiz completion is based only on students' completion of the first four quizzes. Similarly, when conducting the ANOVA for final scores, only the students' completion of the last four quizzes was counted towards their quiz completion. For the average exam score ANOVA, all completed quizzes were counted.

4 Findings

4.1 Patterns of Usage

4.1.1 Before Midterm Exam

In the first half of the course, four quizzes were made available before the midterm exam.

Before the midterm exam, 113 students (67%) out of 169 completed at least one of the quizzes, and 59 of those students (35%) completed all of the optional quizzes. The remaining 56 students (33%) did not complete any of the optional quizzes.

Note that for every single one of these first quizzes (Table-3), at least half of all students interacted with the quiz for the first time less than twenty-four hours before they had to take the upcoming exam.

4.1.2 Before Final Exam

After the midterm, four more quizzes were made available. During this time, 93 students (55%) out of 169 completed at least one of the quizzes, while 60 of those (36%) students completed all of the optional quizzes. The remaining 76 students (45%) did not complete any of the optional quizzes.

Less students overall completed quizzes during the second half of the semester, and a smaller proportion of students completed them within one day of the exam.

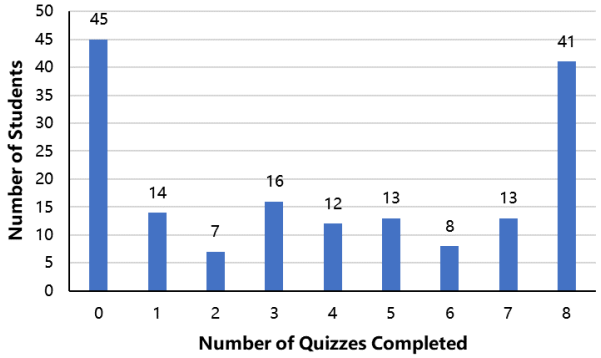

Figure 1. Number of optional quizzes completed by students.

The number of optional quizzes completed by the students over the entire semester (Figure-1) shows that many completed either none of the quizzes or all of them, and many chose to only complete some of them.

4.1.3 Prior Programming Experience

When considering what factors may affect quiz usage, prior programming experience is one possible candidate. Students with higher or lower levels of prior experience may seek out this content for different reasons.

Looking primarily at the students who completed either none or all of the quizzes, it is observed that about 35% of students with higher levels of prior programming experience chose not to complete any available quizzes. In contrast, only about 20% of students with little to no prior experience chose to complete none of the quizzes.

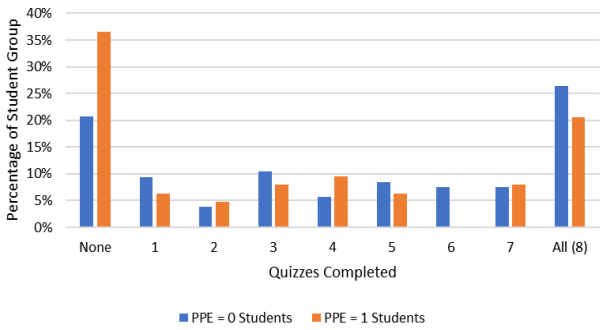

Figure 2. Number of optional quizzes completed by students.

Additionally, only about 20% of students with higher levels of programming experience chose to complete all quizzes, whereas over 25% of students with low prior experience completed all of the quizzes (Figure-2). This relationship does become foggy throughout the middle section, where the sample sizes are smaller.

4.1.4 Insights from Excluded Data

As mentioned in section 2.3.1, not all quiz submissions were complete. Many were received with only one or two responses entered, often following an initial fully-answered submission.

The average number of submissions excluded from any one quiz was 61. While this sounds like a lot of data to exclude, it is worth noting that on many of the quizzes, a single student was responsible for up to 10 submissions containing only a single answer each. When taking a deeper look at the features of this data set, most excluded submissions are preceded by a single attempt which was filled out completely. From this point, the student would submit a single response to each question, probing the quiz to find out which question was answered incorrectly. This phenomenon is a direct result of the choices to make answers hidden, even after submission, and to allow unlimited submissions.

What this tells us about the students is that many of them are seeking mastery of the quizzes. This is a possible direction of future work, which would help uncover more about the nature of students' submissions.

4.1.5 Number of Attempts

Students were able to submit an unlimited number of attempts to each quiz. On the first four quizzes, on average, students submit 1.97 attempts. After removing incomplete attempts, students submitted an average of 1.55 meaningful attempts.

On the last four quizzes, students submit an average of 1.94 attempts. This time, the average meaningful attempts submitted per student was 1.37.

The highest number of meaningful submissions from any student throughout the semester is eight. If including excluded submissions, several students submit upward of ten times.

These numbers tell us that, on average, students are very likely to use the quizzes more than once, and likely twice. Similar to the interpretation of the excluded attempts, this indicates that some students are seeking mastery while others are content with one submission.

4.2 Correlation with Student Performance

4.2.1 Midterm Exam

On the midterm exam, students who completed all four quizzes before midterm did not perform significantly different ($p = 0.786$) from those who completed none of the quizzes.

On the midterm exam, students with PPE = 1 performed significantly better **($p < 0.001$)** than students with PPE = 0.

On the midterm exam, students with PPE = 0 who completed all quizzes did not perform significantly different ($p = 0.559$) from students with PPE = 0 who completed none of the quizzes.

On the midterm exam, students with PPE = 1 who completed all four quizzes did not perform significantly different ($p = 0.671$) from students with PPE = 1 who completed none of the quizzes.

Table 4: Average midterm scores for each major category of students.

Category (N)	Avg. Midterm Score
All quizzes completed (59)	87.2
No quizzes completed (56)	86.6
PPE = 1 (63)	91.6
PPE = 0 (106)	83.5
All students (169)	**86.1**

The results from the factorial ANOVA agree with the findings from the t-tests. The interaction of quiz completion and prior programming experience is not statistically significant, but there is a significant main effect for prior programming experience ($F = 18.952$, $df = 1, 163$, **$p < 0.001$**). Effect size is small for both prior experience and quiz completion (partial $\eta_{ppe}^2 = 0.104$; partial $\eta_{quiz}^2 = 0.002$), and observed power is large for prior experience (0.991) and small for quiz completion (0.079).

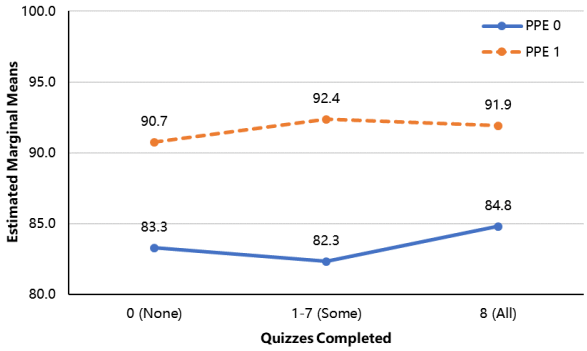

Figure 3. Prior programming experience and quiz completion are visualized with midterm scores using a line plot.

The line plot (Figure-3) of estimated marginal means for each group on midterm shows that prior programming experience was a more significant factor than quiz completion.

4.2.2 Final Exam

On final, students who completed all four quizzes before final performed significantly better **(p = 0.012)** than those who completed none of the quizzes.

On final, students with PPE = 1 performed significantly better **(p = 0.012)** than students with PPE = 0.

On final, students with PPE = 0 who completed all quizzes performed significantly better **(p = 0.037)** than students with PPE = 0 who completed none of the quizzes.

On final, students with PPE = 1 who completed all four quizzes did not perform significantly different (p = 0.056) from students with PPE = 1 who completed none of the quizzes.

Table 5: Average final scores for each major category of students.

Category (N)	Avg. Final Exam Score
All quizzes completed (60)	78.0
No quizzes completed (76)	71.9
PPE = 1 (63)	77.8
PPE = 0 (106)	71.9
All students (169)	**76.3**

The results from the factorial ANOVA elaborate on the differences found in final. The interaction between quiz completion and prior programming experience is not statistically significant, but there is a statistically significant main effect for both quiz completion (F = 3.836, df = 2, 163, **p = 0.024**) and prior programming experience (F = 5.432, df = 1, 163, p = 0.021). Effect size is small for both prior experience and quiz completion (partial η_{ppe}^2 = 0.032; partial η_{quiz}^2 = 0.045), and observed power is moderate for both prior programming experience (0.639) and quiz completion (0.690).

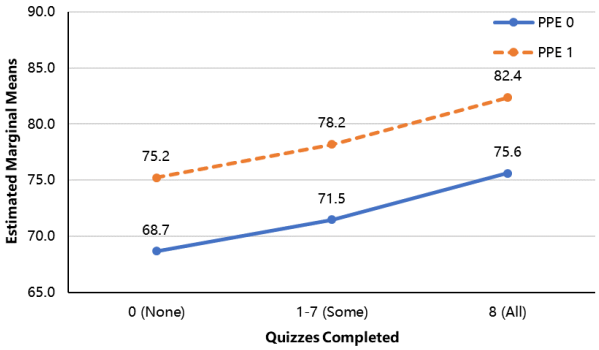

Figure 4: Prior programming experience and quiz completion are visualized with final scores using a line plot.

The line plot (Figure-4) of estimated marginal means for each group on the final provides evidence that both prior programming experience and quiz completion are significant factors for a students' exam score. The lines are sloped upwards to show that scores increase with higher levels of quiz completion.

4.2.3 Overall

When performing these tests on the students' average exam scores rather than just the midterm or final, the dominant factor appears to be prior programming experience. Although students who completed all quizzes during the second part of the course performed significantly better than those who completed none, overall this difference is not significant for average exam scores.

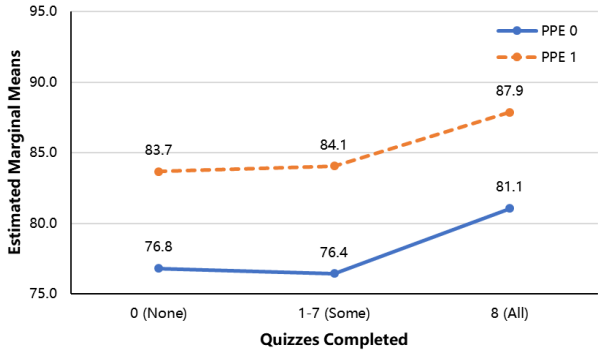

Figure 5: Prior programming experience and quiz completion are visualized with average exam scores using a line plot.

The results from the factorial ANOVA on the students' average exam scores provide additional evidence. The interaction of quiz completion and prior programming experience is not statistically significant, but there is a statistically significant main effect for prior programming experience (F = 13.647, df = 1, 163, **p < 0.001**). The main effect for quiz completion is not significant. Effect size is small for both prior experience and quiz completion (partial η_{ppe}^2 = 0.077; partial η_{quiz}^2 = 0.022), and observed power is large for prior programming experience (0.957) and small for quiz completion (0.386).

5 Discussion

5.1 Student Usage of Quizzes

When analyzing students' usage of the quizzes over time, it was observed that for most quizzes, around 50% of the students completing the quizzes submitted their first attempt within twenty-four hours of the upcoming exam. Student feedback about the quizzes indicates that the students viewed the lecture videos, lecture slides, and homework problems to be sufficient for their learning, and that the quizzes were simply a nice addition that was useful for exam preparation and detail checking. Many students provided positive reviews stating that the quizzes "helped in solidifying a foundation in basic concepts" and that they "targeted common mistakes and misconceptions". Because the quizzes were ungraded and allowed for unlimited submissions, the students were able to use the content in the manner which suited them best.

One byproduct of not displaying the correct answers and allowing many submissions is the observed pattern of seeking mastery through many single-answer submissions. Although this set of data was not conducive to the statistical methods used throughout this study, it provides an interesting caveat which will likely become the subject of future analysis.

5.2 Student Motivation

What is central to this discussion, however, is student motivation. Why do students choose to complete these quizzes? Once more, students' responses are vital to understanding this question. Students were asked to indicate the reason for not completing the quizzes on the survey at the end of the course.

Upon analyzing the responses, we found that the most common reason given is that students "thought the lecture videos were sufficient" and that many students "feel confident in [their] ability to understand and write code". Many students also reported that they were too busy overall to devote extra time to the course and complete the quizzes. Surprisingly, only one student cited the ungraded nature of the quizzes as a reason for not completing them.

To a limited extent, students with different levels of prior experience elected to complete either more or fewer quizzes on average. Students with more prior experience completed fewer quizzes on average than students with lower prior experience. However, this trend is only present at the two extreme ends of quiz completion. No such trend was observed among students completing between one and seven quizzes.

Motivation can also be viewed as a potential factor of success in the course. Although this was not studied nor tested for, it is a possibility that the students who completed the quizzes are simply more motivated, either intrinsically or extrinsically, to excel in their coursework, and that this is the primary reason for any observed differences between the quiz completion groups. Despite this possibility, the nearly unanimously positive reviews of the quizzes among students who completed them provide reason believe that the quizzes did play a role in the success of students, and provided many students with valuable learning experiences.

5.3 Differences Between Exams

On the midterm exam, only prior programming experience was found to be significant ($p < 0.001$). On the final, both prior programming experience ($p = 0.012$) and quiz completion ($p = 0.012$) were found to be significant factors. This leads us to believe that students may receive the most benefit from these quizzes on advanced content. Drawing from student feedback, multiple students specifically mentioned that the quizzes were most helpful when addressing more complex topics; One student noted that "the most helpful quizzes were on matrix math" and another student "found the quizzes for images particularly helpful". These two concepts are both covered during the latter part of the course. However, several students also reported the same for earlier concepts, such as loops. Overall, this tells us that the students found ways to use the quizzes in a variety of scenarios, from detail-checking specific examples minutes before an exam, to serving as a vital tool for solidifying fundamental programming ideas.

Upon studying the possible benefits received by students of varying levels of prior programming experience, very little difference was found. Students with lower incoming levels of prior experience were seen to exhibit the same trends in exam performance as students with higher levels of prior experience. This points to a possible area of future work, as the variable of prior programming experience has more depth to it. The applicability of certain programming experiences to a MATLAB course certainly varies.

6 Recommendations

Although it was found that on average, prior programming experience was more significant than quiz completion, student reviews of the quizzes provides reason to consider incorporating optional quizzes into coursework. When embedding content like this into a course, it is valuable to know that it is likely that around half of the class will be completing the quizzes, and that many of them will use this content as the exam draws near. Adaptations can also be made to better accommodate students with low prior programming experience. Ideally, all students would be performing well by the end of the course. This means that students with little to no prior experience in programming need extra support in order to perform at the same level as those with greater experience.

While reviews of the quizzes were generally positive, students did offer some constructive criticism which may be valuable for instructors considering integrating similar content. Students requested more difficult problem sets that would help better prepare them for the exam. A few students also noted that the quizzes helped mostly with code tracing skills, but not as much with code writing. These concerns can be taken into account by adding questions to the quizzes which differ in style and difficulty.

7 Threats to Validity

Before performing any analysis, tests were conducted for assumptions of normality, homogeneity of variance, and independence. The data set used did not meet the assumption of normality by the S-W test for normality. This means that we must be very careful when drawing conclusions from the factorial ANOVA data as well as the T-tests. Normally, the solution to this is to only accept smaller p-values as significant, or use Bonferroni Correction. In our case, almost all significant p-values were less than .001, which allowed us to be confident in our conclusions regarding prior programming experience and quiz completion. In addition, these conclusions were deduced from a combination of sources, and not simply the data from one or two statistical tests.

Students' responses to our survey question for prior programming experience is another small threat to validity. It can be difficult to quantify the amount of prior programming experience a student has, especially to the exact number of hours. For example, some students who responded "Between 11 and 100 hours" might only have 8 or 9 hours of programming experience, but simply thought they had more. This would result in some students potentially being placed in the wrong category. In addition, prior programming experiences vary from student to student, and may be more or less applicable to the course material. More work is being done to make the analysis of prior programming experience more granular.

Measuring student performance by considering only exam scores also slightly limits the study. With only two exam scores representing the whole of a students' performance in the course, some details and intricacies are certainly overlooked. If a student has one bad test day but completes all other assignments perfectly, earning an A, they could still be classified as a student who is performing poorly due to their one bad exam score. However, we still feel that the exams provide a very strong set of data to work with, especially in tandem with qualitative data.

8 Conclusions

The increasing diversity and size of CS1 enrollment, paired with the non-major context of this particular course, creates an environment consisting of students with a variety of educational needs. The usage and effectiveness of optional quizzes were studied as a possible solution, providing students of varying prior programming experience with a flexible learning tool. The students utilized the optional quizzes in the manner which suited them best, but many chose to use them as a last-minute study aid within twenty-four hours of the exams. Throughout the semester, roughly half of the students attempted at least one quiz, while a smaller portion (roughly one third) of students elected to

complete all of the quizzes. In line with prior research, prior programming experience proved to be significant factor of learning outcomes overall. However, students completing the quizzes were only seen to perform better during the second half of the course. Despite this, the students provided positive reviews of the quizzes and cited their value as a versatile tool in the course.

For instructors planning to incorporate more optional content into their courses, it will be important to know how students will likely interact with this content, and to what extent it might influence the students' learning outcomes. By understanding the usefulness and practicality of teaching tools such as optional quizzes, instructors' decisions regarding course design and content can be more informed. Future research directions include investigating student performance based on exam question type (tracing code versus writing code), a deeper look into prior programming experience according to types of experiences (self-taught versus formal learning, in different languages and styles), as well as a more qualitative study into student motivations for completing optional assignments.

REFERENCES

[1] Joe Michael Allen, Frank Vahid, Kelly Downey, and Alex Edgcomb. 2018. Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP). In Proceedings of 2018 ASEE Annual Conference & Exposition. DOI: https://peer.asee.org/31231

[2] Joe Michael Allen, Frank Vahid, Alex Edgcomb, Kelly Downey, and Kris Miller. 2019. An Analysis of Using Many Small Programs in CS1. In Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19). ACM, New York, NY, USA, 585-591. DOI: https://doi.org/10.1145/3287324.3287466

[3] Jens Bennedsen and Michael E. Caspersen. 2007. Failure rates in introductory programming. SIGCSE Bull. 39, 2 (June 2007), 32-36. DOI: https://doi.org/10.1145/1272848.1272879

[4] Tracy Camp, Stu Zweben, Ellen Walker, and Lecia Barker. 2015. Booming Enrollments: Good Times?. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 80-81. DOI: https://doi.org/10.1145/2676723.2677333

[5] Vincent A. Cicirello. 2009. On the role and effectiveness of pop quizzes in CS1. In Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09). ACM, New York, NY, USA, 286-290. DOI: https://doi.org/10.1145/1508865.1508971

[6] Suzanne L. Dazo, Nicholas R. Stepanek, Robert Fulkerson, and Brian Dorn. 2016. An Empirical Analysis of Video Viewing Behaviors in Flipped CS1 Courses. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16). ACM, New York, NY, USA, 106-111. DOI: https://doi.org/10.1145/2899415.2899468

[7] Alex Edgcomb, Frank Vahid, Roman Lysecky, and Susan Lysecky. 2017. Getting Students to Earnestly Do Reading, Studying, and Homework in an Introductory Programming Class. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). ACM, New York, NY, USA, 171-176. DOI: https://doi.org/10.1145/3017680.3017732

[8] Stephen H. Edwards, Krishnan P. Murali, and Ayaan M. Kazerouni. 2019. The Relationship Between Voluntary Practice of Short Programming Exercises and Exam Performance. In Proceedings of the ACM Conference on Global Computing Education (CompEd '19). ACM, New York, NY, USA, 113-119. DOI: https://doi.org/10.1145/3300115.3309525

[9] Hassan Khosravi and Kendra M.L. Cooper. 2017. Using Learning Analytics to Investigate Patterns of Performance and Engagement in Large Classes. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). ACM, New York, NY, USA, 309-314. DOI: https://doi.org/10.1145/3017680.3017711

[10] Celine Latulipe, Audrey Rorrer, and Bruce Long. 2018. Longitudinal Data on Flipped Class Effects on Performance in CS1 and Retention after CS1. In Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18). ACM, New York, NY, USA, 411-416. DOI: https://doi.org/10.1145/3159450.3159518

[11] Leo Leppänen, Juho Leinonen, Petri Ihantola, and Arto Hellas. 2017. Predicting Academic Success Based on Learning Material Usage. In Proceedings of the 18th Annual Conference on Information Technology Education (SIGITE '17). ACM, New York, NY, USA, 13-18. DOI: https://doi.org/10.1145/3125659.3125695

[12] Charlie McDowell, Linda Werner, Heather E. Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. Commun. ACM 49, 8 (August 2006), 90-95. DOI: https://doi.org/10.1145/1145287.1145293

[13] Aidan McGowan, Philip Hanna, and Neil Anderson. 2016. Teaching Programming: Understanding Lecture Capture YouTube Analytics. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16). ACM, New York, NY, USA, 35-40. DOI: https://doi.org/10.1145/2899415.2899421

[14] Linda J. Sax, Kathleen J. Lehman, and Christina Zavala. 2017. Examining the Enrollment Growth: Non-CS Majors in CS1 Courses. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17). ACM, New York, NY, USA, 513-518. DOI: https://doi.org/10.1145/3017680.3017781

[15] Christopher Watson and Frederick W.B. Li. 2014. Failure rates in introductory programming revisited. In Proceedings of the 2014 conference on Innovation & technology in computer science education (ITiCSE '14). ACM, New York, NY, USA, 39-44. DOI: http://dx.doi.org/10.1145/2591708.2591749

[16] Krissi Wood, Dale Parsons, Joy Gasson, and Patricia Haden. 2013. It's never too early: pair programming in CS1. In Proceedings of the Fifteenth Australasian Computing Education Conference - Volume 136 (ACE '13), Angela Carbone and Jacqueline Whalley (Eds.), Vol. 136. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 13-21.

[17] Joshua Sai Yuen, Alex Daniel Edgcomb, and Frank Vahid. 2016. "Will Students Earnestly Attempt Learning Questions if Answers are Viewable?". 2016 ASEE Annual Conference & Exposition, New Orleans, Louisiana, 2016, June. ASEE Conferences, 2016.