A Simple, Language-Independent Approach to Identifying Potentially At-Risk Introductory Programming Students

Brett A. Becker
University College Dublin
Ireland
brett.becker@ucd.ie

Catherine Mooney
University College Dublin
Ireland
catherine.mooney@ucd.ie

Amruth N. Kumar
Ramapo College of New Jersey
USA
amruth@ramapo.edu

Seán Russell
University College Dublin
Ireland
sean.russell@ucd.ie

ABSTRACT

For decades computing educators have been trying to identify and predict at-risk students, particularly early in the first programming course. These efforts range from the analyzing demographic data that pre-exists undergraduate entrance to using instruments such as concept inventories, to the analysis of data arising during education. Such efforts have had varying degrees of success, have not seen widespread adoption, and have left room for improvement.

We analyse results from a two-year study with several hundred students in the first year of programming, comprising majors and non-majors. We find evidence supporting a hypothesis that engagement with extra credit assessment provides an effective method of differentiating students who are not at risk from those who may be. Further, this method can be used to predict risk early in the semester, as any engagement – not necessarily completion – is enough to make this differentiation. Additionally, we show that this approach is not dependent on any one programming language. In fact, the extra credit opportunities need not even involve programming. Our results may be of interest to educators, as well as researchers who may want to replicate these results in other settings.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; **CS1**.

KEYWORDS

CS1; CS 1; CS-1; programming; introductory programming; introduction to programming; novice programming; extra credit

ACM Reference Format:
Brett A. Becker, Catherine Mooney, Amruth N. Kumar, and Seán Russell. 2021. A Simple, Language-Independent Approach to Identifying Potentially At-Risk Introductory Programming Students. In *Australasian Computing Education Conference (ACE '21), February 2–4, 2021, Virtual, SA, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3441636.3442318

ACE '21, February 2–4, 2021, Virtual, SA, Australia

ACM ISBN 978-1-4503-8976-1/21/02.
https://doi.org/10.1145/3441636.3442318

1 INTRODUCTION

In this study we present evidence supporting the hypothesis that engagement with extra credit opportunities can differentiate between not-at-risk students and those that may be at-risk, with high accuracy. Further, this method can predict which students are not at risk, or may be at risk, as the extra credit opportunities can be given at any time during a course. This approach has several advantages:

- Easy to implement – the approach only measures engagement with extra credit opportunities. These need not even be marked. Simply attempting an extra credit opportunity is enough to be effective.
- Language independent – the approach can be applied regardless of the language(s) utilised in the course.
- Flexible as to time – the approach can be applied at any point during a course.
- Can be used where there is a lack of historical data – the approach can be applied in the first semesters of university where prior performance data such as GPAs are not available.

We say language-independent for two reasons. First, when extra credit opportunities involve programming tasks, the extra credit is in the language of instruction. But, our approach does not dictate the language of instruction or of extra credit. In other words, any language can be used. Second, we also experiment with non-programming extra credit. This is truly language-independent as there is no programming language involved at all.

We define extra credit to be optional, low-stakes assessment for which the points awarded are not included in the 'regular' 100% of the points that make up the final course grades. Put another way, it is always possible for a student to achieve 100% without taking part in any extra credit assessments. In practice, in the rare cases that completion of extra credit opportunities results in a grade over 100%, a student's grade is capped at 100%.

Predicting student performance [10, 41, 48] and in particular identifying students in difficulty early [29, 39] has been a goal of many computing education researchers. Much of this focus stems from a perception that failure rates in introductory programming are too high, with multiple studies citing average failure rates of approximately 33% in introductory programming courses [6, 47]. While acknowledged as high, others would note that failure rates are not alarmingly high, and would seem to have changed little since much work was focused on the problem [47]. However, very recent

work convincingly demonstrates that computer science failure rates are not notably higher than other STEM disciplines [43]. Regardless, the ability to differentiate between at-risk and not-at-risk students is still an attractive goal as all students could conceivably benefit from distributing classroom resources to different students as required, most efficiently. Doing so as early as possible could help these students not only with their current class, but with subsequent classes and potentially prevent them from withdrawing.

Further, recent trends in many institutions include greater enrollments in computing [9] often leading to larger classes, and reduced staff to student ratios. This often translates to a greater workload for staff and less time per individual student. Unfortunately, many proposed methods for predicting student performance require an investment of time and effort on the part of the instructor, the students, or both. This presents a barrier that may deter many from implementing these methods.

In this paper we present the results of a study into the uptake of extra credit assignments by 433 students over the course of two years, to differentiate between students who have a very high likelihood of doing well and those who may be at-risk of failing. We describe our students and context further in Section 3.

1.1 Research Questions

Our research questions are:

- RQ1 Can voluntary, low-stakes extra credit opportunities be used to differentiate between not-at-risk and potentially at-risk students in an introductory programming course?
- RQ2 Will extra credit as a performance differentiator apply to both women and men similarly?
- RQ3 Does the nature of voluntary, low-stakes extra credit opportunities (i.e., programming vs. non-programming exercises) affect the ability to differentiate between not-at-risk and potentially at-risk students?

2 RELATED WORK

This study mainly concerns two topics, predicting novice programmer performance and the effects of extra credit activities and optional exercises. The first has been a focus of many researchers in computing education, while the second has seen little attention, even outside computing education compared to the large volume of assessment literature. This is set in the context that exactly what is expected of introductory programming students is often not stated explicitly [4, 32], and that despite many years of study and debate, some significant issues in teaching programming remain unresolved and probably unresolvable [34]. This leaves educators in a position where exploring their own context to find what works for them is important, and greatly beneficial if what works for them might in turn work for others.

2.1 Predicting Programming Performance

There is a very long history of interest in the topic of predicting novice programmer performance [42, 46]. The period 2010-present has seen a dramatic increase in activity, captured well in a large systematic review [18]. Predicting performance is a common question that practitioners have for researchers [13]. Early instances in introductory programming typically involved the use of standardised tests such as the IBM Programmer Aptitude Test and were sometimes aimed at reducing a perceived waste of training time in educating students unsuited to computer programming [20].

Various assessments and tests have been developed to assess CS1 performance including FCS1 [45] and SCS1 [37]. Although not designed to be predictive, these or similar instruments could possibly be used for that purpose. However, these approaches generally have several costs and drawbacks against their use as predictors. They must be explicitly administered, can be lengthy and time consuming, and are not language independent [15]. Bockmon et al. [8] highlighted the lack of valid and reliable cognitive instruments for predicting student performance. In particular, they highlighted the problems with SCS1 being a lack of formal validation and the time required for students to complete it.

As development tools improved, student behaviour could be captured using specially created programming environments or through the addition of plugins to existing integrated development environments. Predictions of student performance could now be made based on how students programmed, rather than the final result of their work. Watson et al. compared a number of predictors, both based on tests and programming behavior, and found that programming behavior based predictors typically outperformed those based on tests [48]. Other approaches for predicting performance while learning is happening include the use of analytics and micro-analytics from virtual learning environments [3] and interaction data such as keystroke analytics [11, 28].

Kuehn et al. developed an expert system for predicting student performance in introductory programming [21]. The expert system used as inputs, data relating to student's attitudes and behavior with regard to their studies and produced an estimated grade. The data was gathered using a single survey containing a number of questions which were grouped to capture factors that the authors believed would contribute to student success.

Other studies have noted that a large factor in performance is motivation. Students with good intrinsic motivation had less difficulty completing courses than students with poor motivation [1]. Others have suggested that as students' scores in assessments had an effect on motivation, the use of extra-credit activities itself can help keep motivation high by facilitating increases in grades [30].

In [40] Quille & Bergin detail the 13-year evolution of a model called PreSS (Predict Student Success) which aims to predict student success early in CS1. The PreSS model is based on a naïve Bayes classifier that uses three factors to predict student success a few hours into CS1: programming self-efficacy, mathematical ability and the number of hours per week a student plays computer games. A revalidation study looked at the performance of PreSS across 11 institutions, and found an accuracy of ~70%, and sensitivity of ~80% for detecting students at risk of failing or dropping out, while also looking at new factors for the model. The model is a real-time prediction system with a web interface that only takes about five minutes to administer.

Alternatives such as performance in games or other complex tasks have been suggested. Lorenzen and Chang tested students on their ability to reason about solutions to situations in the MasterMind© board game [31]. Students were encouraged to practice and participate by being given extra credit in the course for doing well.

A moderate correlation was found between students' grades and their performance in the activity. The authors noted that there was difficulty assessing students' performance.

2.2 Extra Credit and Optional Exercises

Work involving extra credit opportunities is much more scarce than that attempting to predict performance and is often from psychology studies, using psychology students. Researchers reported that extra credit activities had at least some positive effect or correlation with performance, attendance, and other factors. However, there is evidence that it is not (or, was not in 1993) a common practice [35]. Some studies reported positive effects (discussed below), along with some concerns. Harrison et al. [17] show gender and class-size bias that could benefit some students more than others. Hardy [16] (also providing positive results discussed below) advanced the position that instructors should reevaluate the amount of time and effort it takes instructors to assign, assess, and record such activity, which should be weighed carefully against the benefits it confers.

Harrison et al. [17] found in a sample of 508 students in several non-introductory psychology college courses, that those with existing higher grades were more likely to complete extra credit assignments than those with lower grades. They also found that female students were more likely than male students to complete extra credit assignments. This supported findings by Hardy [16] who also found that only "better" students completed extra credit.

Several authors have found offering extra credit to be beneficial. Padilla-Walker [36] found that extra credit performance was a strong predictor of exam score, above and beyond gender, college GPA, and ACT scores. Wilder et al. [49] found that the use of random extra credit quizzes increased student attendance by 10%. Dominguez et al. [14] found that students completing optional (but not for credit) review exercises improved academic performance.

However, extra credit assessment can also raise some issues, for instance in terms of equity, depending on the context and details of how it is applied. Pynes presented seven arguments against extra credit [38], however most of these only apply to very specific circumstances such as offering extra credit only to certain students, or as a means to compensate for failing or not submitting some other mandatory work. Our use of extra credit carefully adhered to several principles in order to mitigate the issues raised by Pynes that applied to our context: all students had equal opportunities to complete the extra credit; the marks were in addition to regular assessed work (so, students could still get full marks without the extra credit); marks were capped at 100% in the case that a students' total score exceeded that; and the potential gain was small – worth considerably less than any other mandatory assignment – to prevent students focusing on a 'reward' at the expense of 'risk', such as prioritizing extra work over required work.

3 APPROACH

In our study, data was collected from first year introductory programming classes over four semesters between September 2017 and June 2020. Although the first and second semesters of the year-long introductory programming sequence are administratively and logistically separate, they are both mandatory, form a two-course sequence, and together form our university's equivalent of the course that is traditionally referred to as CS1 [5, 19, 34].

Each year, students are separated into two sections based on their major. Students majoring in software engineering (SE) are placed in one section and students majoring in internet of things engineering (IOTE) or electronic and information engineering (EIE) are placed in another. We consider SE majors to be "CS majors". This degree is delivered by Computer Science faculty and conforms to ACM/IEEE joint curriculum for Computer Science. The IOTE and EIE majors are delivered by engineering faculty, but do feature some mandatory computing courses delivered by computing faculty, such as those we discuss in this study.

The courses are delivered by one of two lecturers depending on the section and the semester. However, lecturers are consistent from one year to the next. The breakdown of students is shown in Table 1. Groups 1 and 3 are a mix of IOTE and EIE students, while groups 2 and 4 are SE students. There were 433 students in total (~25% identifying as women, and ~75% as men). We do not report on non-binary gender identification due to low numbers and anonymity.

Table 1: Participant groups. N is number of students (total 433). XC Type is Extra Credit type: programming, or non-programming.

Group	Acad. Year	Lecturer	Semester	N	XC Type
1	2017/18	A	2	122	Prog
2	2018/19	B	1	91	Prog
3	2018/19	A	2	131	Prog
4	2019/20	B	1	89	Non-Prog

All the groups were given the opportunity to engage with extra credit opportunities during these courses. The extra credit work and the associated points that students could receive were discussed with students during a lecture early in the semester and on the learning management system. Students were informed that extra credit was completely optional and could only have a positive (or neutral) effect on their final grade. There was no class time set aside for completion of the work. Students were also given more traditional mandatory programming assignments to complete every week, and a traditional, written, summative exam at the end of each semester.

For groups 1-3, eight thematically appropriate problets and eplets were made available as extra credit as described in Section 3.1. These were made available at the start of the semester and students had until the end of the semester to complete them. Final grades were determined through a traditional weighted combination of mandatory coursework (programming assignments) and a paper final exam. For groups 1 and 3 the coursework / final exam split was 40% / 60%, while for group 2 it was 30% / 70%.

For group 4, students were asked to complete a non-programming assignment as described in Section 3.2. Mandatory programming assignments completed over the semester determined 20% of the students' final grade, with the remaining 80% determined by two terminal exams, one a supervised live programming exam and the other a traditional written exam.

3.1 Programming Extra Credit Assessments

For "programming" extra credit assessments, we used problets (problets.org) and epplets (epplets.org) software tutoring suites. Problets are problem-solving tutors on code-tracing concepts and epplets are based on Parsons puzzles. Both can be used by students in their own time to supplement their classroom instruction, but neither is designed to replace traditional programming assignments.

Problems presented in problets include evaluating expressions, debugging programs, tracing program behaviour, and predicting program state [22]. The tutors grade the student's answer and provide feedback explaining the correct solution. They adapt to student needs, presenting problems on only the concepts that the system determines needs reinforcement. The tutors are set up for mastery learning [33, 50]. They have a large, built-in resource of unidentical problems on each concept and present problems on each concept until the student demonstrates mastery of it. Evaluations have shown that the step-by-step explanation provided by the tutors helps students learn code-tracing concepts [23, 24] and practicing with problets helps students learn to write code [25, 26].

Epplets present Parsons puzzles [27] in which students are presented a description of a problem along with a 'scrambled' program to implement it. The program statements must be rearranged into their correct order and all other statements eliminated. Students cannot progress until they have correctly reassembled the complete program, but are given feedback on each incorrect attempt.

Once a tutor is completed, it presents a confirmation code to the student. Information detailing the tutor attempted, number of problems solved as well as date and time of completion are encoded into this confirmation code. At the end of the course, students were asked to submit a single text file containing the confirmation codes for the tutors that they had completed. These codes were then verified using the decoder provided with the tutor suite to verify that the students had indeed completed the work.

For groups 1, 2 and 3, an extra 0.5% credit was awarded to students for each of the tutors completed, allowing for a maximum extra credit of 4%. This was awarded regardless of the performance of the student on the tutors, which were designed to keep presenting problems until the student had demonstrated understanding of each of the included learning outcomes.

It is important to note that problets and epplets do not require students to write programs. They only involve answering programming questions, tracing code, Parsons problems, etc. They address application and analysis in Bloom's taxonomy [7], as compared to synthesis targeted by typical programming projects. Typically, students spent 30-90 seconds per problet problem and 4-5 minutes per epplet puzzle.

3.2 Non-Programming Extra Credit Assessments

For "non-programming" extra credit assignments, students were assigned an essay on an influential figure in computing. In order to take part in the extra credit assignment, students were required to make a request through the learning management system. In response to this request, the student was assigned a computing figure. It was originally planned that each student would receive a unique figure, however the level of uptake required that some students received the same ones.

The typical format of the assignment required the achievements of the figure to be discussed highlighting their greatest contributions and if applicable, to discuss something that was named after them. This required that students perform independent research outside of the course notes and textbook. Non-programming assignments targeted knowledge in Bloom's taxonomy [7].

Students in group 4 were awarded between 2% and 5% extra credit depending on the quality of the essay submitted. Of the 47 students that requested a topic for the extra credit assignment, 45 students submitted essays.

3.3 Extra Credit Engagement

For the purposes of this research, we discern between those that engaged with extra credit opportunities and those that did not. Engagement counted as any attempt at any extra credit opportunity. No differentiation was made between students in groups 1, 2, and 3 who completed a single problet or epplet and those who completed all of the available problets and epplets. Students from group 4 were deemed to have engaged if they had completed the essay.

4 RESULTS & DISCUSSION

In these results, final course grades do not take into account the 0-5 extra credit points available to students, except in the last figure. This decision was taken to fairly compare the performance of students who did and did not engage in extra credit, without increasing the gap between them (if there is any) due to the extra credit points students may have been awarded (if they did engage with extra credit). All statistical significance tests are Welsh's t-tests (two-tail), $\alpha = 0.05$. We use Cohen's d to report effect size. Our interpretations are: small ($0.2 \leq d < 0.5$), medium ($0.5 \leq d < 0.8$) and large ($d \geq 0.8$) [12, pp25-26]. We report Cohen's d and t-test results even when not significant for completeness and to allow for replications. We do not correct for multiple tests as this study is restricted to a small number of planned comparisons as outlined in [2]. Further, this work is exploratory, and our goal is to look for promising leads that can be followed up in subsequent studies, not to present definitive, generalizable findings [44].

4.1 Baselines: Grades and Gender

Figure 1 shows all 433 students (groups 1-4). Group 4 had the option of doing non-programming extra credit and are seen on the left side. Groups 1-3 who had programming extra credit options are combined on the right side. This does not include the extra credit points themselves and therefore illustrates one baseline, showing that the distribution of final grades between the group with non-programming extra credit available to them ($N = 89, M = 67, SD = 20$), and the three groups that had programming extra credit available to them ($N = 344, M = 64, SD = 21$), are not significantly different $t(143.17) = 1.13, p = 0.26$ (Cohen's $d = 0.13$).

Figure 2 shows that women have a significantly higher mean final grade ($N = 109, M = 70, SD = 17$) compared to men ($N = 324, M = 63, SD = 22$), $t(238.14) = -3.47, p < 0.001$ with Cohen's $d = 0.34$ (small). This provides another baseline.

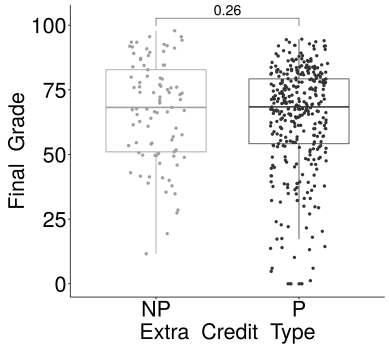

Figure 1: Final grades vs extra credit type *available* (programming (P) / non-programming (NP)) for all 433 students.

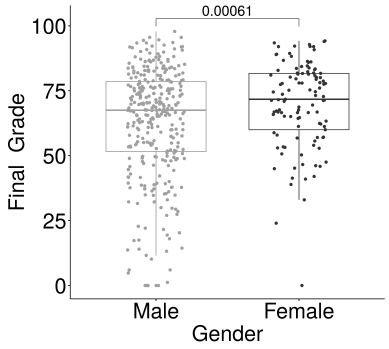

Figure 2: Final grades vs gender for all 433 students.

4.2 RQ1: Extra Credit as a Differentiator

Figure 3 shows that students who engaged in extra credit have higher mean final grades ($N = 117, M = 76, SD = 14$) than those that did not engage in any extra credit ($N = 316, M = 60, SD = 21$), $t(306.66) = -8.58, p < 0.001$. A Cohen's d of 0.78 indicates a (very high) medium effect size. It is notable that only two of 117 students that attempted extra credit received failing grades just below 40 (before including the extra credit points).

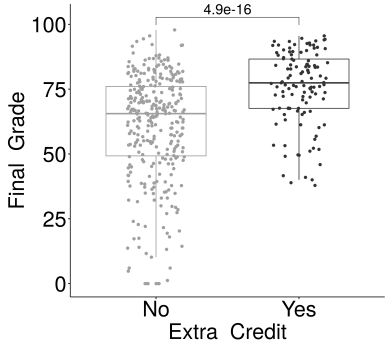

Figure 3: Final grades vs extra credit engagement (No / Yes).

RQ1 was: *Can voluntary, low-stakes extra credit opportunities be used to differentiate between not-at-risk and potentially at-risk students in an introductory programming course?* This evidence supports answering this question in the affirmative. We discuss more evidence for this later in this section.

These results also support Hardy [16] and Harrison et al. [17] who found that a minority of students completed extra credit opportunities. Harrison et al. found 39%. Our figure is 27%. Harrison et al. also found higher grades for these students, although their difference in means was 3 points, compared to 16 in our results.

4.3 RQ2: Gender

Figure 4 shows that women who attempted extra credit had higher mean grades ($N = 41, M = 78, SD = 15$) compared to women who did not attempt extra credit ($N = 68, M = 65, SD = 16$), $t(90.94) = -4.19, p < 0.001$. A Cohen's d of 0.81 indicates a large effect size. Similarly, men who attempted extra credit had higher mean grades ($N = 76, M = 75, SD = 14$) compared to men who did not ($N = 248, M = 59, SD = 22$), $t(198.30) = -7.21, p < 0.001$. A Cohen's d of 0.75 indicates a (high) medium effect size.

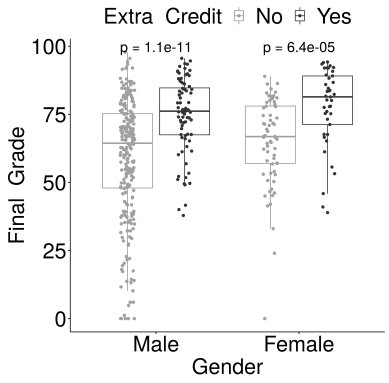

Figure 4: Final grades vs gender broken down by extra credit engagement (No / Yes).

RQ2 was: *Will extra credit as a performance differentiator apply to both women and men similarly?* Our results show that women and men who engaged with extra credit achieved statistically significantly higher mean final grades (16 points higher for men and 13 for women).

4.4 RQ3: Nature of Extra Credit

Figure 5 shows that in the semester where non-programming extra credit was offered (exclusively) those who attempted it had a higher mean final grade ($N = 45, M = 72, SD = 18$) compared to those who did not attempt extra credit ($N = 44, M = 61, SD = 21$), $t(84.43) = -2.58, p = 0.012$. A Cohen's d of 0.54 indicates a medium effect size.

Similarly, in the three semesters when programming extra credit was offered (exclusively) those who attempted it had a higher mean final grade ($N = 72, M = 78, SD = 11$) compared to those who did not attempt extra credit ($N = 272, M = 60, SD = 21$), $t(219.45) = -9.63, p < 0.001$. A Cohen's d of 0.91 indicates a large effect size.

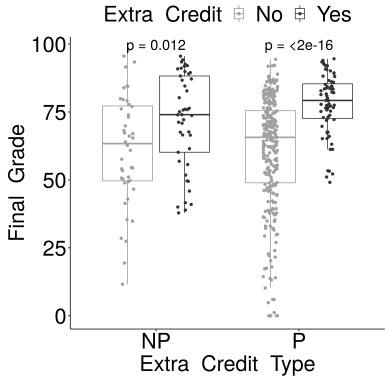

Figure 5: Final grades vs extra credit type (programming (P) / non-programming (NP)), broken down by extra credit engagement (No / Yes).

The mean increase in scores was 18 for programming extra credit groups and 11 for the non-programming extra credit group.

Figure 6 shows that students who did not engage in extra credit but had non-programming extra credit available to them (group 4) ($N = 44, M = 61, SD = 21$) did not have a statistically significant difference in their final grade distribution compared to those in groups 1-3 that did not engage in extra credit but had programming extra credit available to them ($N = 272, M = 60, SD = 21$), $t(58.90) = 0.33, p = 0.742$ (Cohen's $d = 0.05$). However, for students that did engage in extra credit opportunities, those that had programming extra credit available to them (groups 1-3) had a significantly higher mean final grade ($N = 72, M = 78, SD = 11$) than those that had non-programming extra credit opportunities available ($N = 45, M = 71, SD = 18$), $t(66.28) = -2.10, p < 0.040$. A Cohen's d of 0.44 indicates a (high) small effect size.

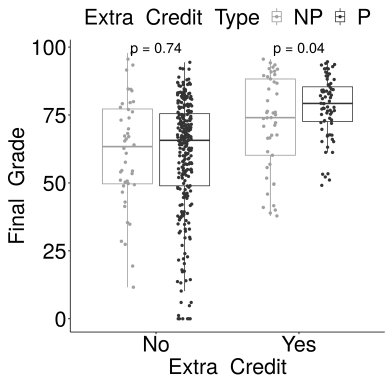

Figure 6: Final grades vs extra credit engagement (No / Yes) broken down extra credit type (programming (P) / non-programming (NP)).

RQ3 was: *Does the nature of voluntary, low-stakes extra credit opportunities (programming vs. non-programming exercises) affect the ability to differentiate between not-at-risk and potentially at-risk students?* Figures 5 and 6 show that students who engage with either type of extra credit have higher mean final grades. Further, those that engaged with programming extra credit had higher final grades than those that engaged with non-programming extra credit. These results show that the nature of the extra credit opportunities are both positive, but different, with programming extra credit being correlated with higher grades than non-programming extra credit.

Figure 7 shows the range of final grades including extra credit (worth ≤ 5 points). All the students that did not pass the course (final grade < 40) did not attempt extra credit. All the students who attempted extra credit passed the course and those that attempted programming extra credit passed by at least 10 points (final grade > 50). Means are shown by black lines. Interestingly, once extra credit points were factored in, the two students that did attempt extra credit but would have been below the passing grade without extra credit points did pass in the end, because of the extra credit.

Finally, all extra credit opportunities were made available early in the semester. Later, we could use this data not just to *differentiate* between students but to confidently *predict* which students might be at risk. For many students, we could predict as early as week 3 of a 16-week semester. However, we did not analyze our timing data, and can't definitively state how early we could make these predictions in general. This will be our first order of future work.

5 LIMITATIONS & CONSIDERATIONS

Although we found a correlation between engaging with extra credit opportunities and academic performance, this does not imply causation. In fact, it has been claimed that the students who "need" extra credit most are the ones who are unlikely to take advantage of the opportunity [17]. However, this differentiation (identifying likely not-at risk vs. more likely at risk) is what we use extra credit engagement to achieve. In other words, it is possible that the fact that weaker students tend to not engage in extra credit is where the power of this approach as a differentiator lies.

Our result showing that programming extra credit is correlated with higher final grades than non-programming extra credit is not necessarily robust. Non-programming extra credit was offered in only one semester, by one lecturer. Therefore we may also be observing differences in the lecturer, course content, or other environmental variables. Other factors include a general fear or dislike of "more programming" or a higher interest in non-programming activities, perceived time commitments required for completion, and a perceived cost/reward ratio. Nonetheless, for other comparisons, our results seem to generalize across semesters, lecturers, and majors, lending credit to the possibility they would generalize to other courses and institutions. Differences in assessment between groups, whether represented by different grading styles between lecturers or changes in assessment type or component breakdown of the course seem to have had no discernible effect on the efficacy of extra credit as a predictor of performance. However, replication of these results in other contexts is required to verify this.

6 CONTRIBUTIONS & MOVING FORWARD

Our contribution is a mechanism of administering extra credit assessment which is effective in differentiating between students with a negligible risk of failing, and those with a much higher chance of being at risk. The mechanism has the following advantages:

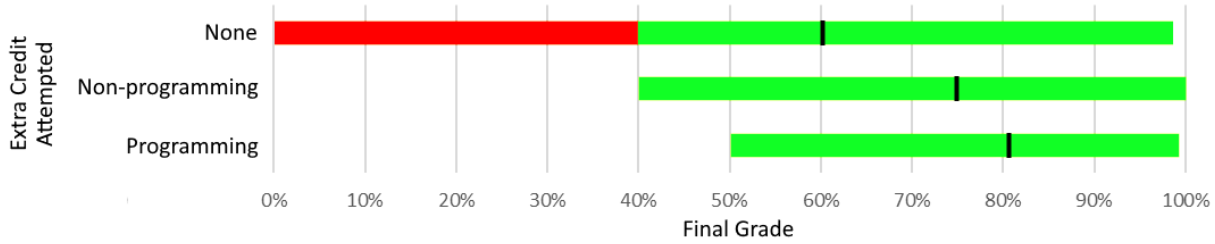

Figure 7: Range of final grades for students who attempted extra credit (programming and non-programming) vs those that did not attempt extra credit. Grades here include the extra credit points. Means are indicated by black lines.

- Easy to implement
- Language independent
- Flexible as to timing within the semester
- Useful where there is a lack of other data such as prior GPAs

Our supporting findings are:

(1) Students' willingness to attempt extra credit assignments was a low-cost and reliable performance differentiator in the introductory programming course. We did not even consider how well they did on the extra credit assignments, only that they had attempted them.
(2) The extra credit assignments do not have to involve programming. In the two options we considered, one involved solving programming problems (problets and epplets) and the other involved writing an essay on a famous computing figure. Neither involved actual programming for students. This further supports that it is not what students do for extra credit as much as whether they attempt extra credit that predicts their chances of success in the course.
(3) Students who engaged with programming exercises had significantly higher final grades than those engaging with non-programming essays. Although engaging in either type of extra credit was associated with higher final grades, some extra credit activities seem to correlate with student success more than others. The choice of activity likely requires a balance in learning value versus student motivation: how much students might learn from an extra credit activity versus how motivated they may feel in attempting the activity.
(4) Although men showed a larger gain (3 points) in final grade when attempting extra credit, our baseline showed that women had final grades 8 points higher than men regardless of engagement with extra credit. Thus the mean final grade for women was higher than that of men. We conclude that gender has an effect, but that it likely transcends more than what we measured in this research.
(5) Our results are supported by (and support) other work in extra credit assessment, specifically that weaker students engage in extra credit less than stronger students, and that this engagement correlates with higher performance.

One way that this work could be directly applied is assigning a single small-stakes, low-impact, easy to administer extra credit assessment early in the semester. Our results support the hypothesis that efforts to identify at-risk students should be concentrated on those that do not attempt this assessment. One line of future work is to see if making such assessments 'regular credit', not 'extra credit' – and just being very low-stakes – is enough to differentiate between those not at risk and those who potentially are.

Other future work could focus on multi-institutional studies to explore the robustness of this data and more fine-grained analysis of timing data to see how early this technique can be used to predict students that may be at risk. We believe that this work is of interest not only to introductory programming instructors, but also instructors teaching other computing courses.

ACKNOWLEDGMENTS

Partial support for this work was provided by the National Science Foundation under grants DUE-0817187, DUE-1432190 and DUE-1502564.

REFERENCES

[1] Raad A. Alturki. 2016. Measuring and Improving Student Performance in an Introductory Programming Course. *Informatics in Education* 15, 2 (2016), 183–204. https://eric.ed.gov/?id=EJ1117149
[2] Richard A. Armstrong. 2014. When to Use the Bonferroni Correction. *Ophthalmic and Physiological Optics* 34, 5 (2014), 502–508.
[3] David Azcona and Kevin Casey. 2015. Micro-analytics for Student Performance Prediction. *Int. J. Comput. Sci. Softw. Eng.* 4, 8 (2015), 218–223.
[4] Brett A. Becker and Thomas Fitzpatrick. 2019. What Do CS1 Syllabi Reveal About Our Expectations of Introductory Programming Students?. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1011âĂŞ1017. https://doi.org/10.1145/3287324.3287485
[5] Brett A. Becker and Keith Quille. 2019. 50 Years of CS1 at SIGCSE: A Review of the Evolution of Introductory Programming Education Research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 338–344. https://doi.org/10.1145/3287324.3287432
[6] Jens Bennedsen and Michael E Caspersen. 2007. Failure Rates in Introductory Programming. *SIGCSE Bull.* 39, 2 (jun 2007), 32–36. https://doi.org/10.1145/1272848.1272879
[7] B.S. Bloom, Engelhart M.D., E.J. Furst, W.H. Hill, and D.R. Krathwohl. 1956. *Taxonomy of Educational Objectives: Handbook I: Cognitive Domain.* Longmans Group.
[8] Ryan Bockmon, Stephen Cooper, Jonathan Gratch, and Mohsen Dorodchi. 2019. (Re)Validating Cognitive Introductory Computing Instruments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 552–557. https://doi.org/10.1145/3287324.3287372
[9] 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. https://doi.org/10.1145/2676723.2677333
[10] C K Capstick, J D Gordon, and A Salvadori. 1975. Predicting Performance by University Students in Introductory Computing Courses. *SIGCSE Bull.* 7, 3 (sep 1975), 21–29. https://doi.org/10.1145/382216.382483
[11] Kevin Casey. 2017. Using Keystroke Analytics to Improve Pass-fail Classifiers. *Journal of Learning Analytics* 4, 2 (2017), 189–211.

[12] Jacob Cohen. 1988. *Stastical Power Analysis for the Behavioral Sciences* (second ed.). Lawrence Erlbaum Associates.

[13] Paul Denny, Brett A. Becker, Michelle Craig, Greg Wilson, and Piotr Banaszkiewicz. 2019. Research This! Questions That Computing Educators Most Want Computing Education Researchers to Answer. In *Proceedings of the 2019 ACM Conference on International Computing Education Research (ICER '19)*. Association for Computing Machinery, New York, NY, USA, 259âĂŞ267. https://doi.org/10.1145/3291279.3339402

[14] César Domínguez, Arturo Jaime, Jónathan Heras, and Francisco José García-Izquierdo. 2019. The Effects of Adding Non-Compulsory Exercises to an Online Learning Tool on Student Performance and Code Copying. *ACM Trans. Comput. Educ.* 19, 3 (jan 2019). https://doi.org/10.1145/3264507

[15] Mark Guzdial. 2019. We Should Stop Saying Language Independent. We Don't Know How To Do That. http://bit.ly/WSSSLI

[16] Marjorie S Hardy. 2002. Extra Credit: Gifts for the Gifted? *Teaching of Psychology* 29, 3 (2002), 233.

[17] Marissa A Harrison, Denise G Meister, and Amy J Lefevre. 2011. Which Students Complete Extra-Credit Work? *College Student Journal* 45, 3 (2011), 550–555.

[18] Arto Hellas, Petri Ihantola, Andrew Petersen, Vangel V Ajanovski, Mirela Gutica, Timo Hynninen, Antti Knutas, Juho Leinonen, Chris Messom, and Soohyun Nam Liao. 2018. Predicting Academic Performance: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. ACM, New York, NY, USA, 175–199. https://doi.org/10.1145/3293881.3295783

[19] Matthew Hertz. 2010. What Do "CS1" and "CS2" Mean? Investigating Differences in the Early Courses. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. Association for Computing Machinery, New York, NY, USA, 199–203. https://doi.org/10.1145/1734263.1734335

[20] Aaron Katz and William W Helme. 1962. *Prediction of Success in Automatic Data Processing Programming Course*. Technical Report. Army Personnel Research Office Washington DC.

[21] Michael Kuehn, Jared Estad, Jeremy Straub, Thomas Stokke, and Scott Kerlin. 2017. An Expert System for the Prediction of Student Performance in an Initial Computer Science Course. In *2017 IEEE International Conference on Electro Information Technology (EIT)*. IEEE, 1–6. https://doi.org/10.1109/EIT.2017.8053321

[22] Amruth N Kumar. 2003. Learning Programming by Solving Problems. In *Informatics Curricula and Teaching Methods: IFIP TC3 / WG3.2 Conference on Informatics Curricula, Teaching Methods and Best Practice (ICTEM 2002) July 10–12, 2002, Florianópolis, SC, Brazil*, Lillian Cassel and Ricardo A Reis (Eds.). Springer US, Boston, MA, 29–39. https://doi.org/10.1007/978-0-387-35619-8_4

[23] Amruth N Kumar. 2005. Results from the Evaluation of the Effectiveness of an Online Tutor on Expression Evaluation. *SIGCSE Bull.* 37, 1 (feb 2005), 216–220. https://doi.org/10.1145/1047124.1047422

[24] Amruth N Kumar. 2006. Explanation of Step-by-step Execution as Feedback for Problems on Program Analysis, and its Generation in Model-based Problem-solving Tutors. *Technology, Instruction, Cognition and Learning (TICL) Journal* 4, 1 (2006), 65–107.

[25] Amruth N. Kumar. 2013. A Study of the Influence of Code-Tracing Problems on Code-Writing Skills. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '13)*. Association for Computing Machinery, New York, NY, USA, 183âĂŞ188. https://doi.org/10.1145/2462476.2462507

[26] Amruth N. Kumar. 2015. Solving Code-Tracing Problems and Its Effect on Code-Writing Skills Pertaining to Program Semantics. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15)*. Association for Computing Machinery, New York, NY, USA, 314âĂŞ319. https://doi.org/10.1145/2729094.2742587

[27] Amruth N Kumar. 2019. Helping Students Solve Parsons Puzzles Better. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 65–70. https://doi.org/10.1145/3304221.3319735

[28] Juho Leinonen, Krista Longi, Arto Klami, and Arto Vihavainen. 2016. Automatic Inference of Programming Performance and Experience from Typing Patterns. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 132âĂŞ137. https://doi.org/10.1145/2839509.2844612

[29] Soohyun Nam Liao, Daniel Zingaro, Michael A Laurenzano, William G Griswold, and Leo Porter. 2016. Lightweight, Early Identification of At-Risk CS1 Students. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. ACM, New York, NY, USA, 123–131. https://doi.org/10.1145/2960310.2960315

[30] Ed Lindoo. 2018. Back to the Basics in an Effort to Improve Student Retention in Intro to Programming Classes. *J. Comput. Sci. Coll.* 34, 2 (dec 2018), 72–79. https://dl.acm.org/doi/10.5555/3282588.3282599

[31] Torben Lorenzen and Hang-Ling Chang. 2006. MasterMind: A Predictor of Computer Programming Aptitude. *SIGCSE Bull.* 38, 2 (jun 2006), 69–71. https://doi.org/10.1145/1138403.1138436

[32] Andrew Luxton-Reilly. 2016. Learning to Program is Easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16)*. Association for Computing Machinery, New York, NY, USA, 284âĂŞ289. https://doi.org/10.1145/2899415.2899432

[33] Andrew Luxton-Reilly, Brett A. Becker, Yingjun Cao, Roger McDermott, Claudio Mirolo, Andreas Mühling, Andrew Petersen, Kate Sanders, Simon, and Jacqueline Whalley. 2017. Developing Assessments to Determine Mastery of Programming Fundamentals. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 388. https://doi.org/10.1145/3059009.3081327

[34] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory Programming: A Systematic Literature Review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018 Companion)*. Association for Computing Machinery, New York, NY, USA, 55–106. https://doi.org/10.1145/3293881.3295779

[35] John C Norcross, Heather S Dooley, and John F Stevenson. 1993. Faculty Use and Justification of Extra Credit: No Middle Ground? *Teaching of Psychology* 20, 4 (1993), 240–242. https://doi.org/10.1207/s15328023top2004_13

[36] Laura M Padilla-Walker. 2006. The Impact of Daily Extra Credit Quizzes on Exam Performance. *Teaching of Psychology* 33, 4 (2006), 236–239. https://doi.org/10.1207/s15328023top3304_4

[37] Miranda C Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. In *Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16)*. Association for Computing Machinery, New York, NY, USA, 93–101. https://doi.org/10.1145/2960310.2960316

[38] Christopher A Pynes. 2014. Seven Arguments Against Extra Credit. *Teaching Philosophy* 37, 2 (2014), 191–214.

[39] Keith Quille and Susan Bergin. 2018. Programming: Predicting Student Success Early in CS1. a Re-Validation and Replication Study. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2018)*. Association for Computing Machinery, New York, NY, USA, 15–20. https://doi.org/10.1145/3197091.3197101

[40] Keith Quille and Susan Bergin. 2019. CS1: How Will They Do? How Can We Help? A Decade of Research and Practice. *Computer Science Education* 29, 2-3 (2019), 254–282. https://doi.org/10.1080/08993408.2019.1612679

[41] Keith Quille, S Bergin, and Aidan Mooney. 2015. Programming: Factors That Influence Success Revisited and Expanded. In *International Conference on Engaging Pedagogy (ICEP), 3rd and 4th December, College of Computing Technology, Dublin, Ireland*, Vol. 10. 1047344–1047480.

[42] Nathan Rountree, Janet Rountree, Anthony Robins, and Robert Hannah. 2004. Interacting Factors That Predict Success and Failure in a CS1 Course. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR '04)*. Association for Computing Machinery, New York, NY, USA, 101âĂŞ104. https://doi.org/10.1145/1044550.1041669

[43] Simon, Andrew Luxton-Reilly, Vangel V Ajanovski, Eric Fouh, Christabel Gonsalvez, Juho Leinonen, Jack Parkinson, Matthew Poole, and Neena Thota. 2019. Pass Rates in Introductory Programming and in Other STEM Disciplines. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19)*. Association for Computing Machinery, New York, NY, USA, 53–71. https://doi.org/10.1145/3344429.3372502

[44] David L Streiner and Geoffrey R Norman. 2011. Correction for Multiple Testing: Is There a Resolution? *Chest* 140, 1 (2011), 16–18.

[45] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. Association for Computing Machinery, New York, NY, USA, 111–116. https://doi.org/10.1145/1953163.1953200

[46] Philip R Ventura Jr. 2005. Identifying Predictors of Success for an Objects-First CS1. (2005).

[47] 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. https://doi.org/10.1145/2591708.2591749

[48] Christopher Watson, Frederick W B Li, and Jamie L Godwin. 2014. No Tests Required: Comparing Traditional and Dynamic Predictors of Programming Success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 469–474. https://doi.org/10.1145/2538862.2538930

[49] David A Wilder, William A Flood, and Wibecke Stromsnes. 2001. The Use of Random Extra Credit Quizzes to Increase Student Attendance. *Journal of Instructional Psychology* 28, 2 (2001), 117–120.

[50] Daniel Zingaro, Michelle Craig, Leo Porter, Brett A. Becker, Yingjun Cao, Phill Conrad, Diana Cukierman, Arto Hellas, Dastyni Loksa, and Neena Thota. 2018. Achievement Goals in CS1: Replication and Extension. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. Association for Computing Machinery, New York, NY, USA, 687–692. https://doi.org/10.1145/3159450.3159452