Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses

David H Smith IV
Western Washington University
smithd77@wwu.edu

Qiang Hao
Western Washington University
qiang.hao@wwu.edu

Filip Jagodzinski
Western Washington University
filip.jagodzinski@wwu.edu

Yan Liu
University of British Columbia
yan.liu@ubc.ca

Vishal Gupta
University of British Columbia
vishal.gupta@alumni.ubc.ca

ABSTRACT

Computer literacy and programming are being taught increasingly at the K-12 level with more students than ever matriculating in college with prior programming experience. Accurately assessing student programming skills acquired in high school can inform college faculty about the range of competencies in introductory programming courses. The tool predominantly-used for assessing past CS knowledge and skills is a survey, which lacks quantitative rigor. This study aims to (1) quantify the effects of prior knowledge in entry-level programming courses and (2) compare the different measurement approaches of student prior knowledge in programming, including surveys and aptitude tests. The results of this study reveal that a discrepancy exists between the results of surveys and aptitude tests. Consistent with prior survey studies, our survey results showed that the effects of student prior programming knowledge faded gradually during the course period. In contrast, the aptitude test results indicated that the effects of student prior knowledge did not weaken over time. The accuracy of both measurements and implications for instructors were further discussed.

CCS CONCEPTS

• **Applied computing → Education**;

KEYWORDS

CS1; prior knowledge; assessment; performance prediction

ACM Reference Format:
David H Smith IV, Qiang Hao, Filip Jagodzinski, Yan Liu, and Vishal Gupta. 2019. Quantifying the Effects of Prior Knowledge in Entry-Level Programming Courses. In *ACM Global Computing Education Conference 2019 (CompEd '19), May 17–19, 2019, Chengdu,Sichuan, China.* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3300115.3309503

1 INTRODUCTION

K-12 students today have unprecedented, and ever increasing, access to Computer Science (CS) courses and resources. A recent poll conducted by Gallup and Google concluded that 76% of U.S. schools offered some form of CS learning opportunity, 60% offered at least one CS course, and 40% offered a class that involved programming [1]. From this, it can be expected that an increased proportion of students enrolling in undergraduate computer science courses have some prior knowledge related to programming and/or CS fundamentals. However, the survey also noted that although many computer science related activities are offered in modern high schools, great variances exist in terms of what is offered by such activities, including general computer classes, introductory programming courses as well as a wide variety of clubs [1]. Furthermore, considering the lack of curricula consistency across K-12 in the United States, students' prior knowledge in computer science is likely to be heterogeneous and challenging to measure [2].

Studies investigating this topic generally divide the broader concept of "prior knowledge" into a number of factors asking students to self-evaluate their abilities and/or experiences. These factors are then compared to students' scores with the goal of determining which play the largest role in predicting students performance. Previous studies have found that students with prior CS or general STEM related knowledge are more likely to outperform their peers [3–6]. Other factors explored by these studies include factors such as "personal comfort", and how a student perceives their own programming abilities. Students exhibiting a strong sense of personal comfort as well as confidence in their own abilities have been shown to outperform their peers [6, 7].

Despite the relatively consistent findings, it is worth noting that all the prior studies used self-answering surveys as their primary means of data collection. Such data collection methods are only effective for determining where a student first came in contact with a subject, how long they stayed in contact with it, as well as a self-evaluation of their abilities. Such surveys neither validate nor quantify a student's perceived abilities. Given the great variance of computing education at the high school level, successful completion of a high school CS course may mean very different things to students from different schools or school districts (e.g., whether students have taken a CS course in high school is a frequently-used survey question). Consequently, the reliability and generalizability of the survey results might be questionable. To the best of our knowledge, alternative measurements to surveys, such as validated aptitude tests, are rarely used to measure student prior knowledge in programming.

To fill this gap, this paper aims to investigate different measurements of prior programming knowledge on student performance in the context of introductory programming courses (CS1). This study also explores using a validated aptitude test to measure prior knowledge. The results of this study provide insights into the accurate quantification of student prior knowledge in programming. These insights could be used by instructors to gain the most accurate view of their classes' prior knowledge and thus aid them in their course design and calibration.

2 RELATED WORK

Students develop their knowledge by interpreting incoming information through the lens of their existing knowledge, beliefs, and assumptions [8]. As a result, sufficient, appropriate and accurate prior knowledge aids in the learning process, whereas insufficient, inappropriate and inaccurate prior knowledge may have the opposite effect [9]. These aforementioned studies further discuss why teachers should take into consideration the extent of students' prior knowledge and structure their courses appropriately with the goal of building new knowledge atop that which already exists. Research conducted in a variety of fields on the impact of prior knowledge has concluded that prior knowledge is a key factor in student learning [10–14].

Factors involving or related to prior knowledge, and its effectiveness in predicting CS1 students' performance, explored by prior studies include:

- Exposure to and performance in math classes
- Exposure to and performance in core science classes (chemistry, physics, biology, etc.)
- Previous exposure to computer programming and CS concepts (high school, college, online, club, etc.)
- Previous non-programming computer experience
- Student programming comfort levels
- Perception of personal ability

The roles of both mathematical prowess and prior programming experience have been the most investigated by far with nearly all studies pointing to their significance in predicting students success. Bergin and Reilly [7] performed a replication study seeking to reproduce the significance of existing math, science, and programming knowledge in predicting students performance in a CS1 course. Their findings corroborate previous studies where math and/or prior programming experience were found to influence most future perfromance [6, 15–18]. Wilson and Shrock [6] investigated 12 factors, with math and science courses included among them, that might affect the learning ability of students in introductory computer science courses. They concluded that each of these factors plays a significant role in predicting student performance. Hagan and Markham investigated the effects of prior programming experience as measured by the number of programming languages a student had previously used [18]. Their findings point to students previous knowledge as having initially positive performance benefits. The level to which students with prior experience outperformed their peers was related to the number of programming languages a student had experience with.

There is a long-standing belief that there exists a correlation between student lecture attendance and their subsequent course performance[4]. Veerasamy *et al.* investigated the connection between prior programming knowledge, lecture attendance, and course performance and compared their results against the longstanding idea that lecture attendance plays a significant role in predicting student performance[4]. Their results not only show that students with prior programming knowledge outperformed their peers, but that those same students had significantly lower lecture attendance.

In addition to simply looking at the level of a student's prior experience, many studies investigate the impacts of factors such as self-efficacy and student comfort levels. The effects of student cognitive skills and self-efficacy were investigated alongside prior knowledge by Bergin and Reilly [7]. They found a student's perception of their own abilities to be the most significant factor in predicting their performance, even above that of prior knowledge. In Wilson and Shrock's investigation of 12 factors that may affect learning ability, they also investigated the effects of self-efficacy [6]. Their findings corroborate with that of Bergin and Reilly in that a student's perception of their own abilities had a significant impact on their performance throughout thea course.

The effects of prior knowledge and experience by separate genders has also been investigated by multiple studies [5, 15, 19]. Wilcox and Lionelle administered surveys prior to the start of term and as well as after the term's completion with questions primarily focusing on prior programming experience and personal comfort levels within the entire student body as well as between genders [5]. Their findings reaffirm the significance of student's comfort levels and prior knowledge in performance prediction. Female students were shown to have lower rates of both when compared to their male peers. A study also investigating the roles of aptitude in math and science as well as prior programming knowledge in student performance found that female students achieved scores consistent with their male peers [15]. It should be noted the student body consisted of male and female students with similar levels of prior knowledge which may have skewed results. With regard to the student body as a whole, students with prior programming knowledge significantly outperformed their peers. Additionally, the link between aptitude in math and science and performance in introductory programming courses was reaffirmed.

Given the significance of prior knowledge related to STEM concepts in predicting student performance in CS1 courses, the question arises as to the duration of their effects. Morrison [20] investigated this with regard to prior knowledge gained through high school computer science courses. Their findings show that students who had completed high school level computer science courses significantly outperformed their peers on both the courses' validated pretest as well as assessments performed throughout the course. The performance gap between students with high school experience and those without had narrowed to the point of obsolescence by CS2. Wilson and Shrock [6] suggest the effects of prior knowledge may be even more short-term than suggested by Morrison[20]. Their study showed the effects of prior programming experience to be limited to the midterm, disappearing by the time of the final exam.

All prior studies discussed in this section used self responding surveys as their primary means of data collection. Although this means of evaluation is effective for establishing the means by which a student initially came in contact with concepts, as well as a self evaluation of their skills, it does not quantify their current knowledge. This study seeks to add to the literature on this subject by using both a survey and a validated assessment to evaluate a student body and compare their results to their exam performance.

3 RESEARCH DESIGN

3.1 Participants and Contexts

This study was conducted on a group of students enrolled in a large university in the North American Pacific Northwest. A total of 62 students taking a CS1 over a regular academic term participated in this study. Students were expected to complete a set of programming assignments individually and two comprehensive programming projects collaboratively. In addition, two exams, including one midterm and one final, were administered.

CS1 in this study is being used in consistency with most prior computing education studies conducted in North America, which refers to the first core programming course [21, 22]. At the institution in question, students taking CS1 typically have declared their major intentions as CS, but are not yet in the CS major. Whether a student can be admitted to the CS major is dependent on their performance in CS1 and a set of sequential core courses. Many CS departments also offer a set of elective CS courses aiming at growing student interests in CS, such as "computer science and society" and "computational thinking", which involve little to no learning and teaching of programming. Such courses are referred to as "**college-level CS courses prior to CS1**" in this study. It is worth noting that college-level CS courses prior to CS1 are not necessarily more advanced than computer science courses being offered in high school.

3.2 Measurements

Both a survey and an aptitude test were given to students at the beginning of the semester to measure their prior knowledge in programming. The survey, composed of seven questions, was adopted from a study done by Wilcox and Lionelle [5].

The aptitude test, known as Programming Aptitude Test (PAT), was developed and validated by Tukiainen and Monkkonen [23]. The PAT evaluates a students grasp of fundamental programming concepts as well as conceptual solution design, but does not involve actual programming. There have been many efforts in developing language-independent assessments of CS1 knowledge in the last decade, such as Foundamental CS1 Assessment (FCS1) and Second CS1 Assessment (SCS1) [21, 24]. However, such instruments were developed to serve the goal of measuring student learning performance after taking CS1, but not prior to taking CS1. To our best knowledge, PAT developed by Tukiainen and Monkkonen [23] is the only instrument aiming at student prior knowledge before taking CS1. Different from FCS1 and SCS1 that involve complex programming concepts, PAT measures mainly generic problem solving strategies and capabilities. Given the specific goals of this study, PAT was chosen to measure student prior programming knowledge prior to taking CS1.

In addition, student performance on the midterm and final exams were collected and regarded as their academic performance. Both midterm and final exams were developed by the instructor. Although neither midterm nor final exam was validated, both of them carefully mapped the tested concepts of FCS1 developed by Tew and Guzdial [21] to language-dependent questions.

4 RESULTS

4.1 Descriptive Summary

Participants consisted of 71% male and 29% female students ranging in age from 18 to 43 years old (male: M = 20.20, SD = 3.90; female: M = 19.67, SD = 1.50). Students largely consisted of young adults under the age of 20 with numbers dwindling as age increased. In terms of prior experience, it was found that, of the pool of 17 female students enrolled in the class, 64.7% had no high school CS experience with the remaining 35.3% having had taken one or more classes. As for the college experience, all had taken at least one or more prior CS elective courses (e.g., computational thinking). Male participants showed slightly higher participation in high school CS courses with 44.2% having taken at least one or more. Male students also showed a high level of participation in prior college-level CS courses with 93.0% of male participants having taken one or more.

There were 48 course participants at or under the age of 20 with the remaining 14 falling above that age line. Students at or under the age of 20 tended to have had more experience via high school courses with 47.9% having taken at least one or more compared to only 28.6% of those over 20. Both groups had high levels of participation in college-level computer science courses with 93.8% of those under the age of 21 having completed at least one or more.

4.2 Results of Survey Responses

Linear regression was applied to examine the effects of surveyed factors on student midterm exam performance. As is seen in Table 1, the surveyed factors in total accounted for 18.32% of the variance in academic performance. Student completion of high school level computer science courses (t = 2.434 , p < 0.05) as well as amount of time dedicated to self study (t = 2.253, p < 0.05) were found to be significant factors in the linear model. The completion of college-level computer-science-related courses or a students level of confidence prior to beginning the course were not found statistically significant.

Table 1: Linear Regression of Survey Responses on the Midterm Exam

	R^2	R^2_{adj}	Δ F	β	t
Midterm	0.2636	0.1832	3.281		
CS in College				-2.932	-1.155
CS in High school				5.974*	2.434
Self Learning Time				5.813*	2.253
Confidence				2.202	0.779
Age				-3.823	-1.544
Gender				5.443	1.021

* p <0.05; **p <0.01; ***p <0.001

When the same factors were applied to a linear model for final exam performance, it was found that the surveyed factors only explained 8.068% of the observed variance in student performance (Table 2). None of the factors, however, were found to have any significance in predicting the exam's results. In other words, the influences of the surveyed factors became much weaker on student final exam performance than their midterm performance.

Table 2: Linear Regression of Survey Responses on the Final Exam

	R^2	R^2_{adj}	Δ F	β	t
Final	0.1711	0.08068	1.892		
CS in College				-3.473	-1.172
CS in High school				1.170	0.409
Self Learning Time				5.483	1.822
Confidence				2.272	0.689
Age				-5.472	-1.894
Gender				6.302	1.014

* p <0.05; **p <0.01; ***p <0.001

4.3 Results of the Programming Aptitude Test

Same as the analysis on student survey responses, linear regression was applied to explore student PAT performance on their midterm and final exam performance. Age and gender were controlled for analysis consistency (Table 3). The results showed that student PAT performance along age and gender explain 22.54% of the variance in midterm performance. Student PAT performance was found to be highly significant in predicting their midterm performance (t = 3.903, $p < 0.001$).

Table 3: Linear Regression of PAT Performance on the Midterm Exam

	R^2	R^2_{adj}	Δ F	β	t
Midterm	0.2635	0.2254	6.918		
PAT				9.091***	3.903
Age				-5.326*	-2.308
Gender				-1.487	0.291

* p <0.05; **p <0.01; ***p <0.001

When the same factors were tied to final exam results (Table 4) the model was found to explain 18.49% of the variance in performance. Student PAT performance was once again found to be a highly significant predictor of their final exam performance (t = 3.317, $p < 0.01$). Different from the surveyed factors, the predictive power of student PAT performance was found consistently strong on both their midterm and final exam performance.

Table 4: Linear Regression of PAT Performance on the Final Exam

	R^2	R^2_{adj}	Δ F	β	t
Final	0.225	0.1849	5.612		
PAT				8.7147**	3.317
Age				-5.9625*	-2.290
Gender				0.6099	0.106

* p <0.05; **p <0.01; ***p <0.001

4.4 Correlation Analysis

Given the significant differences in predictive power between survey factors and PAT performance on student learning, correlational analysis among survey factors and PAT performance was further conducted (Table 5).

Surprisingly, the correlations between PAT performance and any other survey factor is below 0.2, and none of them were found to be significant. In other words, taking CS courses (prior to CS1) in high school or college does not necessarily have a positive effects on the mastery of programming-language-independent problem solving. In contrast, confidence was found significantly correlated with both taking CS courses prior to CS1 in college (r = 0.353, $p < 0.01$) and self-learning time (r = 0.361, $p < 0.01$). In other words, taking CS courses prior to CS1 and self-study may boost student confidence in successful completion of CS1.

Table 5: Pearson Correlations Between Survey Factors and PAT Performance

	CSC	CSH	Confid	SLT	PAT
CS in College	1				
CS in High school	0.0190	1			
Confidence	0.353**	-0.111	1		
Self Learning Time	0.0910	0.0698	0.3607**	1	
PAT	-0.0897	0.0522	-0.0393	0.1130	1

CSC: CS in College, CSH: CS in High school, Confid: Confidence, SLT: Self Learning Time

5 DISCUSSION

5.1 Consistency of Findings

Surveys, as the conventional measurement approach of student prior knowledge, have been used widely in prior studies. Overall, these studies [5, 15, 20] have found that student prior knowledge in programming significantly influences their performance. However, the significance of these influences became smaller over time, whether it is over the course of one semester or across two different entry-level courses.

Such findings were consistent with those of the survey results of this study. It was found that the number of computer science courses taken at high school and self-learning time had significant influences on student midterm exam performance, but the effects were no longer detectable for their final exam performance. Although the findings on the gradually weakened effects of student prior knowledge were consistent with that of prior studies, it is worth noting that such factors had limited power in predicting student academic performance. If survey results are to be trusted and used as guidance for course design, instructors can rest assured that given some time every student will catch up and perform equally well. However, considering the low predictive power of the survey factors, the conclusion that students with all levels of prior knowledge can perform equally well might not be so easily reached.

5.2 Which Measurement Approach is Better?

The second research focus of this study was to compare the results of both surveys and aptitude tests. As opposed to surveyed factors, student PAT performance was found to be the most significant influencing factor in midterm and final exam performance.

The significance of these results could indicate the PAT as being a more accurate measurement of student prior knowledge in programming when compared to survey. A fundamental source of error existing within the survey collection method is the ambiguity held within its answers. For instance, in the question of self-study time, there exists no objectively quantifiable amount of time that would constitute "a lot". Even the question of prior courses is open to ambiguity as it fails to consider two important factors, course content and student performance in the course. The validated aptitude test eliminates such errors by providing students with a range of question on fundamental programming concepts, all with objectively correct or incorrect answers, and grading their performance.

The correlation analysis seems to corroborate that PAT is a more accurate measurement of student prior knowledge in programming. Taking CS courses prior to CS1 in college has low correlation with student PAT performance but high correlation with confidence in successfully completing CS1. One possible explanation for this finding is that taking courses such as computing and society or computational thinking can help boost student overall confidence in CS, but does not necessarily help improve their problem-solving capabilities in programming. The variations and lack of consistency of such courses may further contribute to such a result.

Though the PAT proved to be a superior predictor of performance it lacks the indications of where students acquired prior knowledge that surveys provide. The combined use of the PAT and survey results as a means of prior knowledge evaluation produces a more complete picture of the various factors that influence expected performance. The collection of such data affords instructors a better idea of their class's overall prior knowledge and therefore their expected performance. This information can then be used to better calibrate course content/structure to the given student bodies expected capabilities.

6 LIMITATIONS AND FURTHER DIRECTIONS

A few limitations may hinder the generalizability of this study. First, this study was conducted on a sample of 62 students at a single institution. The findings may be different if multiple classes taught by different instructors from different institutions are involved. Future studies may consider replication at a larger scale. Second, the adopted validated test, Programming Aptitude Test, was developed and validated in 2002 [23]. Considering the curriculum development in computer science in the last two decades, the results of this test may not provide an accurate portrayal of student capabilities today. Future studies may investigate the necessity of developing a new instrument to accurately measure student prior knowledge in programming. Furthermore, this study did not investigate the influences of student prior programming knowledge in their long-term performance (e.g., performance in sequential programming courses). To provide a complete picture of the impacts of student prior programming knowledge on their achievement, the understanding of both its short-term and long-term impacts are both necessary. Longitudinal studies adopting both surveys and validated tests may serve this purpose in the future.

7 CONCLUSIONS

Although the effects of prior knowledge have been examined in a variety of fields, most such studies opted to use self-reported surveys as the only measurement approach. The survey can indeed shed light as to whether or not a student has been previously exposed to related topics, but may not accurately measure the amount of knowledge a student has retained. This is especially true given that great variance exists in terms of what students may have previously learned, as happens to be the case with introductory programming courses. This study seeks to bridge this gap by comparing two measurements of student prior programming knowledge, including both surveys and aptitude tests. Although prior findings through surveys were consistent with those in this study, we detected a significant difference between the survey responses and aptitude test results. Survey results show the effects of previous experience, specifically that gained from high school, to have an impact only on midterm performance. Aptitude test performance, however, was shown to have significant impacts on both the midterm and final. Given the importance of student prior knowledge in course design and delivery, more studies exploring its role should be conducted using a standardized test as a means of evaluation.

8 APPENDIX

A PROGRAMMING APTITUDE TEST

Question 1:
The company has information of their employers in three different lists. You have acquired all three lists, which all have little bit different information depending in the purpose of the list. The lists have following information of the employees:

- List 1: the number, the name, the occupation and the department (*List 1 is ordered by the number of employee to ascending order*)
- List 2: the name, the number, the address, the phone number, and the SSN (*List 2 is organized to alphabetic order by the name*)
- List 3: the number, the SSN, salary and some secret information (*List 3 is ordered by the number to ascending order*)

Your job is to make a report of those employees whose salary is greater than $2,000. The report has to display the name, the address, the department and the salary of the employees. Describe how you would solve the problem.

Question 2:
Let i and j be both integers between 0 and 10 (inclusive). List all values of i and j, that make the expression to be always true:

- (i>=1) and (i<=5)
  The expression is true when i has values:
- (j>=7) or (j<=3)
  The expression is true when j has values:

Question 3:
Try to determine the general form of the series by the given series of words, and write a word sequence, that is next on the series.

- bce , bbcde , bbbcdde , ...
- bcace , bcacacace , bcacacacacace , ...
- abcccdd , abbccccdd , abbbcccccdd ...

Question 4:
Your job is to sum up 50 numbers and at the end report the sum and the count of numbers that were positive numbers (>0). Describe how you would solve the problem or write in a pseudo-code.

B SURVEY

Question 1:
Did you take computer science related courses in college?

- I took one CS related course in college
- I took more than one CS related courses in college
- Never

Question 2:
Did you take computer science courses in high school?

- I took one CS course at high school
- I took more than one CS courses at high school
- Never

Question 3:
How much time did you spend on self-learning & teaching of programming outside of school?

- A lot of time
- Some time
- Little time

Question 4:
How would you like to describe yourself in terms of programming?

- Have a lot of experience in programming
- Have some experience in programming
- Have very limited experience in programming

Question 5:
How confident are you in your ability of programming?

- Very confident
- Somewhat confident
- Not confident at all

Question 6:
What is your gender?

- Male
- Female
- Other

Question 7:
What is your age?

REFERENCES

[1] Google Inc. & Gallup Inc. Trends in the state of computer science in u.s. k-12 schools. 2016. URL http://goo.gl/j291E0.
[2] André Schäfer and Rainer Brück. Teaching strategies for undergraduate laboratories with students having heterogeneous prior knowledge. In *Global Engineering Education Conference (EDUCON), 2013 IEEE*, pages 112–117. IEEE, 2013.
[3] Anya Tafliovich, Jennifer Campbell, and Andrew Petersen. A student perspective on prior experience in cs1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 239–244, New York, NY, USA, 2013. ACM.
[4] Ashok Kumar Veerasamy, Daryl D'Souza, Rolf Lindén, and Mikko-Jussi Laakso. The impact of prior programming knowledge on lecture attendance and final exam. *Journal of Educational Computing Research*, 56(2):226–253, 2018.
[5] Chris Wilcox and Albert Lionelle. Quantifying the benefits of prior programming experience in an introductory computer science course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 80–85, New York, NY, USA, 2018. ACM.
[6] Brenda Cantwell Wilson and Sharon Shrock. Contributing to success in an introductory computer science course: A study of twelve factors. In *Proceedings of the Thirty-second SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '01, pages 184–188, New York, NY, USA, 2001. ACM. ISBN 1-58113-329-4. doi: 10.1145/364447.364581. URL http://doi.acm.org/10.1145/364447.364581.
[7] Susan Bergin and Ronan Reilly. Programming: Factors that influence success. *ACM Sigcse Bulletin*, 37:411–415, 01 2005. doi: 10.1145/1047124.1047480.
[8] James V Wertsch and C Addison Stone. The concept of internalization in vygotsky's account of the genesis of higher mental functions. *Lev Vygotsky: Critical assessments*, 1:363–380, 1999.
[9] Marie K. Norman, Marsha C. Lovett, Michael W. Bridges, Michele di Pietro, Susan A. Ambrose, and Richard E. Mayer. *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Jossey-Bass, 2010.
[10] Jeffrey Alan Greene, Lara-Jeane Costa, Jane Robertson, Yi Pan, and Victor M. Deekens. Exploring relations among college studentsâĂŹ prior knowledge, implicit theories of intelligence, and self-regulated learning in a hypermedia environment. *Computers & Education*, 55(3):1027 – 1043, 2010.
[11] Gregor Kennedy, Carleton Coffrin, Paula de Barba, and Linda Corrin. Predicting success: How learners' prior knowledge, skills and activities predict mooc performance. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, LAK '15, pages 136–140, New York, NY, USA, 2015. ACM.
[12] R. M. Rias and W. K. Yusof. Animation and prior knowledge in a multimedia application: A case study on undergraduate computer science students in learning. In *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pages 447–452, 2012.
[13] Stergios Tegos and Stavros Demetriadis. Conversational agents improve peer learning through building on prior knowledge. *Journal of Educational Technology & Society*, 20(1):99–111, 2017.
[14] Belle Selene Xia and Elia LiitiÃďinen. Student performance in computing education: an empirical analysis of online learning in programming education environments. 42:1–13, 11 2016.
[15] Pat Byrne and Gerry Lyons. The effect of student attributes on success in programming. *SIGCSE Bull.*, 33(3):49–52, June 2001.
[16] R. R. Leeper and J. L. Silver. Predicting success in a first programming course. *SIGCSE Bull.*, 14(1):147–150, February 1982.
[17] Laurie Honour Werth. Predicting student performance in a beginning computer science class. pages 138–143, 1986.
[18] Dianne Hagan and Selby Markham. Does it help to have some programming experience before beginning a computing degree program? *SIGCSE Bull.*, 32(3): 25–28, July 2000.
[19] University of Kent. Computer programming aptitude test. https://www.kent.ac.uk/ces/tests/computer-test.html, 2018.
[20] Briana B. Morrison, Adrienne Decker, and Lauren E. Margulieux. Learning loops: A replication study illuminates impact of hs courses. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*, ICER '16, pages 221–230, New York, NY, USA, 2016. ACM.
[21] Allison Elliott Tew and Mark Guzdial. The fcs1: a language independent assessment of cs1 knowledge. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 111–116. ACM, 2011.
[22] Cynthia Taylor, Daniel Zingaro, Leo Porter, Kevin C Webb, Cynthia Bailey Lee, and Mike Clancy. Computer science concept inventories: past and future. *Computer Science Education*, 24(4):253–276, 2014.

[23] Markku Tukiainen and Eero Monkkonen. Programming aptitude testing as a prediction of learning to program. In *Proceedings of PPIG*, pages 45–57, 2002.
[24] Miranda C Parker, Mark Guzdial, and Shelly Engleman. Replication, validation, and use of a language independent cs1 knowledge assessment. In *Proceedings of the 2016 ACM conference on international computing education research*, pages 93–101. ACM, 2016.