Assessment of programming pre-requisites and interventions for student success in an aerospace curriculum

Dr. Kathryn Anne Wingate, University of Colorado at Boulder

Dr. Kathryn Wingate is an instructor at University of Colorado Boulder, where she teaches design and mechanics courses. She holds her PhD in mechanical engineering, and worked at NGAS as a materials scientist.

Dr. Aaron W. Johnson, University of Colorado Boulder

Aaron W. Johnson is an Instructor in Smead Aerospace Engineering Sciences at the University of Colorado Boulder. He teaches courses in structures and vehicle design, and his research focuses on how mathematical models are taught in undergraduate engineering science courses and how these models are used in analysis and design. Before CU he was a postdoctoral research fellow at the University of Michigan and the Tufts University Center for Engineering Education and Outreach. He received his Ph.D. in Aeronautics and Astronautics from the Massachusetts Institute of Technology in 2014 and a bachelor's degree in aerospace engineering from the University of Michigan in 2008.

Miss Lyndsay Rose Ruane
Dennis Akos

Introduction

Complex aerospace systems increasingly rely on integrated software to function, resulting in an industry demand for software-savvy aerospace engineering graduates. To respond to this demand, a significant number of MATLAB programming assignments have been incorporated across the aerospace curriculum at the Ann and H.J. Smead Department of Aerospace Engineering and Sciences at the University of Colorado, Boulder. Lab and homework assignments require students to write a significant amount of MATLAB code starting the first semester of sophomore year in the statics and computational methods courses. MATLAB programming assignments become increasingly difficult as students progress through the undergraduate program, culminating in a year-long senior design course where students design, build, and validate an aerospace system, with at least 30% of the work being software-centric.

Discussions with students and faculty suggest that MATLAB proficiency may be a critical barrier to success in the sophomore and junior years, possibly resulting in student attrition from the program. To prepare for the computer programming demands in the curriculum, students are required to take a computer science course in the CS department their freshmen year (typically taken first semester). However, between transfer and non-transfer students, a wide variety of computer science courses focusing on a number of programming languages are approved for this prerequisite. This, combined with the vast range of student programming experiences in high school and second semester freshman year results in an incoming sophomore class with a wide spectrum of baseline MATLAB ability.

The first intense MATLAB assignment occurs during the first month of sophomore year in the statics course, and requires students to parse an input file, assign variables, and utilize loops and conditional statements to develop a script which can solve various static equilibrium problems. In previous years, this assignment was completed in groups of three students. However, anecdotal evidence suggested that typically a single student with confidence in programming would write the entire code, while the other students would assist with the report. This year the assignment was revised to be an individual assignment, with the goal that every student would gain critical programming experience. The assignment was split into three coding scripts over three weeks, and scripts were auto-graded using MATLAB grader.

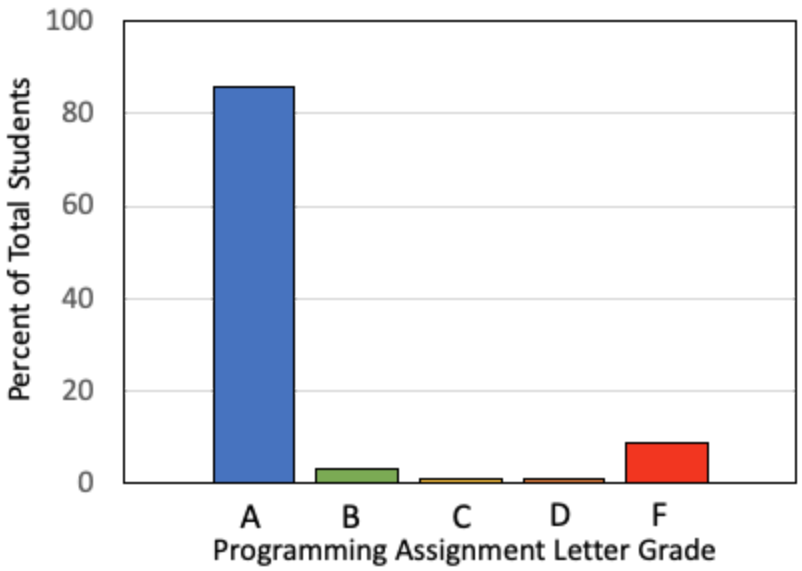

Figure 1. Histogram showing individual student grades in statics programming assignment. N = 237.

Overall, students generally did well in the programming assignment. However, as can be seen in Figure 1, roughly 10% of students did not pass the lab, scoring a 'D' (70%) or lower. We are interested in why some students struggled with the lab while others succeeded. In this paper we conduct statistical analyses to answer the following research questions:

Is there a relationship between students' grades on the programming assignment and their

- RQ1. Gender?
- RQ2. Grades in the freshman computer science course?
- RQ3. Programming experience in high school or undergraduate education, beyond freshman computer science?
- RQ4. Grades in the linear algebra course?

Literature Review

Prior research has investigated a number of factors that have been hypothesized to influence students' performance in computer science (CS). Three of the most-studied factors are students' 1) innate aptitude, 2) environment, and 3) learning strategies in their introductory programming course. Regarding students' aptitude, there are anecdotal claims for the existence of a "geek gene" where certain students have an innate aptitude for CS while others do not [1]. This hypothesis explains the bimodal distribution of grades that many CS faculty see in their intro-level CS courses, although evidence suggests that the presence of bimodal grade distributions may be overstated [1,2]. However, research has pushed back against this idea of a "geek gene" and has found no set of cognitive factors that strongly and reliably predict students' programming ability [3]. As Robins writes, "Maybe there is no 'geek gene'/innate capacity or combination of cognitive or other factors that predicts or accounts for success or failure at programming, any more or less than for other domains of learning" [4, p. 351].

Other research has hypothesized that students' performance in CS is influenced by their environment -- their background, their exposure to CS in high school, and their support in learning CS. Much of the literature focuses on student demographics, such as race or gender, as a means to highlight pre-college opportunity gaps between these underrepresented students and majority students. Research has shown that there are racial and gender disparities in access to high school CS instruction, qualified computing teachers, and the physical resources to study CS [5,6]. These high school opportunity gaps can then lead to consequential performance gaps in undergraduate CS [7,8]. Traditional social expectations can also result in girls being encouraged to pursue CS less than boys [9] or girls experiencing microaggressions that discourage them from further pursuing CS [10]. Along these lines, the ethnographic study of an undergraduate programming course conducted by Secules et al. shows the combined impact of several "mundane and seemingly innocuous" aspects of the course, such as the seating arrangement in lecture and the status hierarchy when an experienced student was paired with an inexperienced student for a group lab [11]. These aspects of the course all combined to create a culture that projected the implicit message that the focus of the study--a female student from a vocational high school--could not succeed in engineering.

Lastly, studies have shown that students' performance in CS is influenced by their learning strategies in their introductory CS course, which are often dictated top-down by the pedagogy and structure of the course. Research has shown the most difficult aspects of learning CS for novices are not related to the syntax of the language, but rather their mental models of a program--the strategies for designing and implementing a program to solve a certain task [4,12,13]. However, as Robins writes, "declarative knowledge (e.g., being able to state how a "for" loop works) [...] receives the most explicit attention in typical textbooks and CS1 courses, which usually focus on presenting knowledge about a particular language" [4, p. 337]. As a result, multiple studies have shown that passing an introductory programming course does not imply future programming success. Soloway et al. found that 38% of students who had completed one semester of programming were able to complete an assignment that asked them to write a loop to compute an average of a given data set [14]. Similar results were found by McCracken et al., who gave a language-agnostic CS assessment to 216 students at 8 international institutions at the end of their introductory programming course [15]. In general, students did much more poorly than the authors expected. As they state, "We did answer the question we asked in the Introduction section: Do students in introductory computing courses know how to program at the expected skill level? The results from this trial assessment provide the answer 'No!' and suggest that the problem is fairly universal" [15, p. 132].

Of our four research questions, three map to the environment. Gendered effects of performance in CS are influenced by the social environment, whereas programming experience in high school or undergraduate education beyond introductory CS is reflective of the opportunity that students have in their pre-college environment. We also take students' grades in the linear algebra course to be reflective of their environment, rather than their ability, because the programming assignment we study involved forming an $Ax = b$ matrix from a set of equations. Based on students' difficulty with this concept in class, we hypothesized that having prior knowledge of linear algebra would be helpful on this programming assignment. And therefore, students who had the opportunity to take advanced math in high school--and therefore the opportunity to take linear algebra during their freshman undergraduate year--would be at an advantage. The last research question, the relationship between grades in the freshman CS course and the

grades on the programming assignment, is indicative of students' ability because we do not have the knowledge about students' learning strategies in this freshman CS course.

Programming Assignment Background

The programming assignment analysed in this paper is the first major programming assignment given to students in the aerospace curriculum, and occurs in the statics course during the first month (September) of their sophomore year. This assignment was originally developed by Prof. Kurt Maute, and was modified for auto-grading purposes by the first author. The overarching goal of the programming assignment is for students to create a series of MATLAB scripts which parse a .txt input file of data describing a 3D static equilibrium problem with known external forces/moments and unknown reactions forces/moments. The MATLAB script must then use the parsed data to formulate a matrix with the six independent force and moment static equilibrium equations, and solve this matrix to find the reaction forces and moments of the system. Student code must be robust to solve any statically determinate 3D equilibrium problem. To do this, students must understand basic MATLAB data types, know how to read in and convert data types, be able to index, utilize conditional statements, set up loops, and perform very basic linear algebra operations. Note that the prerequisites for the statics course did include a basic programming course, but did not include any linear algebra courses. Therefore, this was the first time the majority of students has seen any type of matrix multiplication or formation. The programming assignment was individual, given over three weeks, and auto-graded in MATLAB grader. Students had two hour lab periods once a week to work on the assignment with an instructor and team of TAs present. Prior to each lab period, students were required to watch a 30 minute pre-lab video that reviewed fundamental principles necessary to complete the coding assignment for that week. During the first week, students had to write a script that could parse the .txt file and define variables for future use. During the second week, students had to write a script to take the defined variables, organize them into the six independent force and moment equilibrium equations, form a $Ax = b$ matrix with these equations, and solve the matrix for x (the reaction forces and moments). The pre-lab video for the second week taught the students the basics of matrix multiplication, how to format a system of equations into $Ax=b$ form, and worked through a few examples on this topic. During the third week, students had to put the two scripts together and solve a number of 3D equilibrium problems with 'blind' input files to ensure their code was robust to a wide variety of boundary conditions, external forces, and external moments. For each script students were given six attempts in MATLAB grader to submit the correct code, but were allowed to test their code in MATLAB as many times as needed. 5 hours of lab office hours were offered a week. MATLAB grader checks all submitted scripts to ensure certain defined variables are present and the correct value. Therefore, MATLAB grader can be vulnerable to hardcoding, in which students 'hardcode' certain variables to be the correct value to 'trick' the grader. For example, if students were required to create a loop and iteratively add numbers until $x = 100$, a student could hardcode by simply defining $x = 100$. MATLAB grader allows the instructional team to see all submitted code, and the TA team carefully checked all submitted code to ensure 'hardcoding' was not present. Students who submitted 'hardcoded' code received 0 credit for that particular week's submission.

Data Collection

We were the instructors for the sophomore statics course, and therefore had access to all students' programming assignment grades. The College of Engineering Office of Data Analytics provided the

authors with secure protected files with student grades from calculus 1, calculus 2, calculus 3, linear algebra, and freshman computer science, as well as student gender, race, and ethnicity. We did not consider the race and ethnicity variables in our analysis because of low sample sizes for certain groups. While it was clear what classes students had transferred from other institutions, the grade for any transfer class was not available. The authors developed an optional Qualtrics survey for students in the statics course which asked questions regarding student's pre-college coding experience, the computer science courses they had completed, and their current confidence levels with MATLAB. Students were given this survey at the end of the semester, and were offered 2 extra credit points on their lowest homework score if they completed this survey. Of the 249 students who finished the course, N = 185 students completed this survey. A separate optional survey was sent out after the programming assignment was submitted, and asked students about the time they had spent on the assignment, and the most difficult aspects of the assignment. N = 65 students completed this survey.

Results

To answer RQ1, the relationship between students' gender and their grade on the programming assignment, we performed a Welch's t-test to test whether the mean grade on the assignment was different between women and men. Figure 2 shows the difference in grades between women and men. While the mean grade on the assignment was slightly higher for women than for men (93.4% for women, 89.9% for men), we found no evidence to reject the null hypothesis ($p = 0.21$). Therefore, the data does not support the hypothesis that there is a relationship between students' gender and their grades on the programming assignment. While a null result, this is encouraging given the social factors that dissuade women from pursuing computer science.

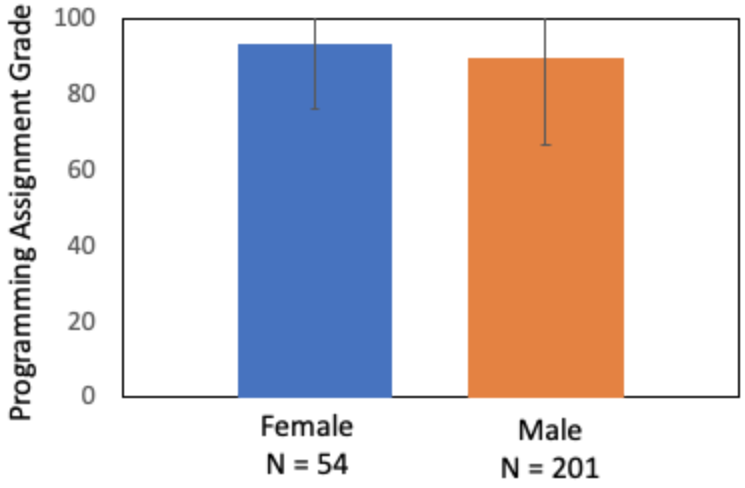

Figure 2. Effect of gender on students' programming assignment grade. N = 255. Error bars show standard deviation.

To answer RQ2, the relationship between students' grades in their freshman computer science course and their grade on the programming assignment, we plotted students' grades on the programming assignment vs. grade in the freshman computer science course, where 4.0 is an 'A,' 3.7 is an 'A-,' 3.3 is a 'B+,' 3.0 is

a ‘B,’ etc (Figure 3). The size of each dot represents the number of students with that combination of assignment grade and computer science grade. As is expected, the largest dot represents the 67 students who got an A in freshman computer science and a 100% on the assignment. However, there was no evidence to support the hypothesis that there is a correlation between students’ programming assignment grade and computer science grade (Kendall’s rank correlation, $p = 0.07762$, $\tau = -0.1$). Note that 21 students who received an ‘A’ or ‘A-’ in the freshmen computer science course failed the statics programming assignment.

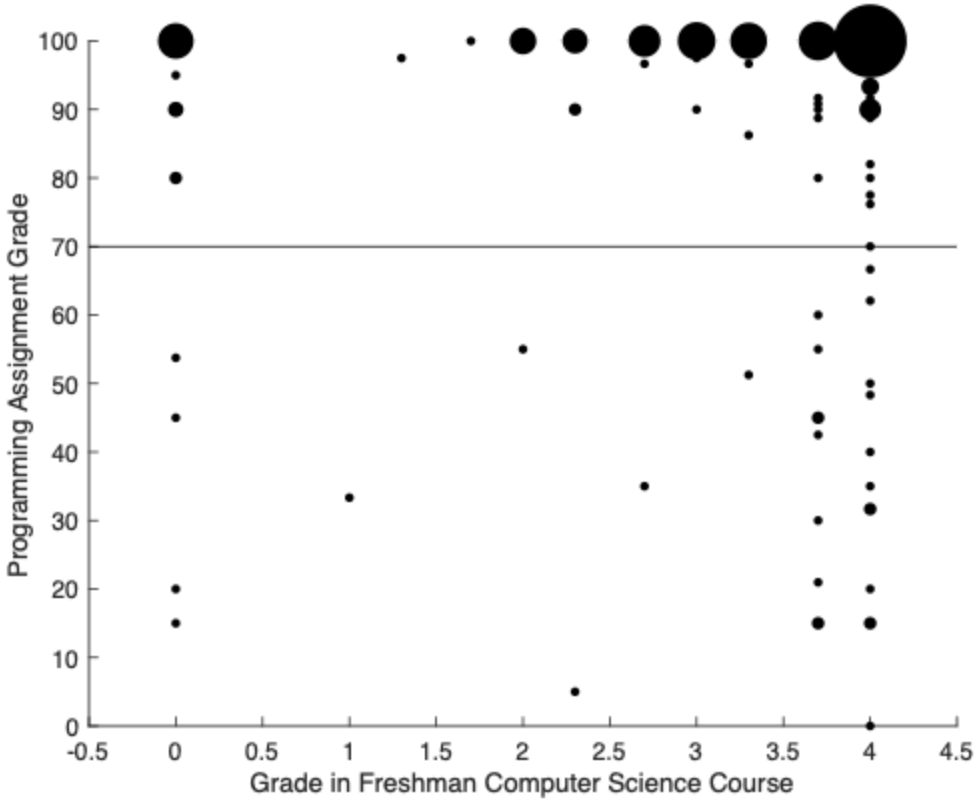

Figure 3. Effect of grade in freshman computer science course on students’ programming assignment grades. N = 240. Size of each dot represents the number of students with that combination of assignment grade and freshman computer science grade.

To answer RQ3, the relationship between students’ programming experience outside of the freshman computer science course and their grade on the programming assignment, we used data from the Qualtrics survey given at the end of the semester. Of the students who responded, 180 gave information about their computer science course history. We grouped students into one of three categories: little or no high school computer science experience (70% of respondents), 1 or more semesters of high school computer science experience (20%), and 1 or more semesters of undergraduate computer science beyond the required freshman course, regardless of how much, if any, computer science they took in high school (10%). Figure 4 shows that all 17 students with advanced undergraduate computer science experience got a perfect grade on the programming assignment. However, as expected, the standard deviation of students’ programming assignment grades increases as students’ computer science experience decreases.

Initial statistical analyses found no difference between groups, however, these statistical results are inconclusive due to small sample sizes and a high risk of a type 2 error (accepting the null hypothesis when there is indeed a significant difference between groups).

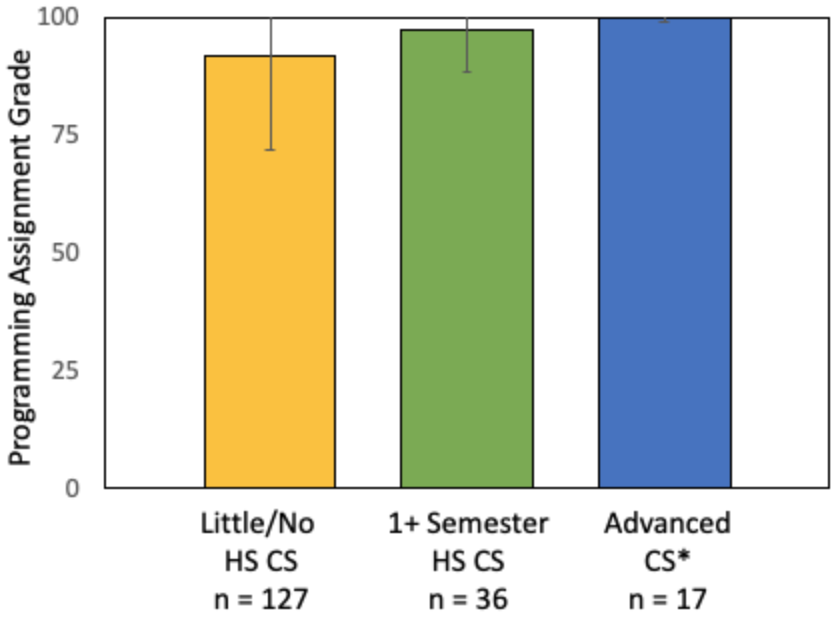

Figure 4. Effect of programming experience on students' programming assignment grade. N = 180. Error bars show standard deviation.

Finally, to answer RQ4, the relationship between students' grades in linear algebra and their grade on the programming assignment, grouped students into 5 categories. Of the 237 students for which we had data, 54.0% had not yet taken linear algebra and 27.0% were currently taking linear algebra. These two groups encompassed our first two categories. The remaining 3 categories captured the grade distribution of the 19.1% of students who had completed linear algebra, with 6.8% of the 237 students receiving an 'A', 7.2% of students receiving a 'B', and 5.1% of students receiving a 'C.' Figure 5 shows that, similar to the effect of programming experience, the standard deviation is very small for students who got an 'A' in linear algebra and much larger for all other categories. Bartlett's test found the standard deviations between groups to be significantly different, $p = 0.00$. Statistical analysis did find the group of students who earned an 'A' in linear algebra had a significantly higher programming assignment grade than students who had not yet taken linear algebra. (One-Way ANOVA with Games Howell Post Hoc, $p = 0.001$).

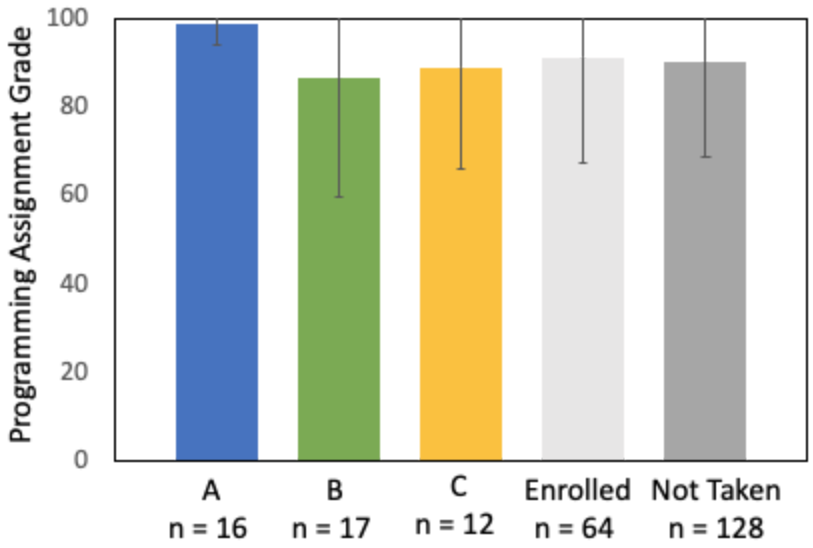

Figure 5. Effect of grade in linear algebra on students' programming assignment grade. N = 237. Error bars show standard deviation.

Discussion

Overall, the statistical tests we ran do not directly indicate why some students did poorly on the programming assignment. However, the statistical results do shed light on what variables may have impacted some students' success. For example, earning an 'A' in linear algebra is beneficial- these students have a significantly higher average grade on the programming assignment, with significantly lower variation. As the programming assignment requires setting up and manipulating an Ax = b matrix, it is not surprising that students who have a solid grasp of matrix theory did well in the lab. Note that students who have completed linear algebra are a full year ahead in math, and transferred in both calculus 1 and calculus 2 (both AP calculus AB and BC exams). Further, the applied math department, which teaches the linear algebra class, is known for the rigor of their courses. Therefore, students who have completed and earned an 'A' in linear algebra likely went to a top-tier high school or transferred in significant community college work, and are likely high performing students in general. It is difficult to separate if the student's grasp of linear algebra allowed them to succeed in this lab, or if this subset of students will succeed in the majority of assignments thrown at them due to a strong academic foundation and excellent study skills. However, a grasp of basic linear algebra matrix formation concepts is important; in an optional class survey (N = 65) the majority of the students found formulating the Ax = b matrix to be the most difficult aspect of the assignment. Further, we did review the Ax=b concept in lecture, and included a Ax=b problem on the exam given immediately after the programming assignment was due. Out of all of the problems on the exam, students struggled the most with the Ax=b problem. While some literature indicates a positive correlation between math and programming skills [16], other researchers have found that once general ability is factored out no relationship exists between a person's overarching math ability and their success in computer science [17].

Figure 4 suggests a trend of increasing programming assignment grades with increasing computer science (CS) instructional experience. We were surprised that prior to enrollment in the university, 75% of

students had either severely limited or no prior computer programming experience. Even more surprising, there is no significant association between a student's grade in their freshmen computer programming course and their grade in the sophomore programming assignment (Figure 3). There were 21 students who got an 'A' or 'A-' in freshman CS but failed the programming assignment with a failing grade below 70%. The required first year CS course is half C+ and half MATLAB, requires no programming experience, and focuses on basic programming techniques. Students typically take this course the fall of their freshmen year, and take the statics course the fall of their sophomore year. It is possible that 8 weeks of MATLAB instruction is simply not enough for some students to tackle the parsing and matrix formation challenges in this programming assignment, as in Soloway et al. [14] and McCracken et al. [15]. Or, it could be that this programming assignment is too open-ended for some students who have only completed a single CS course. Note that while we did have access to the detailed freshmen CS course syllabi, we were not able to discuss at length with freshmen CS teaching team how open-ended their homework and exam coding assignments were. Finally, the students who struggled may have had personal or academic challenges external to the statics course which impacted their performance.

Overall, we were encouraged to see no difference in the programming assignment grades between females and males. We were unable to examine differences in student performance due to race and ethnicity as sample sizes across certain groups were quite low. However, the literature found that differences in programming performances amongst various races/ethnicities can often be directly linked to a lack of access to good computing instruction and resources [5,8]. Through the survey data, we were able to look at a student's programming experiences prior to this course. The sample sizes in some of the post-college programming experience groups were low, and did not allow for statistical analysis to be performed with adequate power levels to map results to the larger population. However, trends did appear, and further work with large sample sizes could be done.

Conclusions

This paper reports a first attempt at understanding the underlying variables that impact a student's computer programming ability on sophomore coding assignments in the [name blinded] aerospace department. Interesting initial findings include:

- 75% of students in [name blinded] aerospace department enter the university as freshmen with little to no computer programming experience.

- Overall, most students were able to complete the programming assignment with an 'A.'

- There are a non-trivial number of students who got an 'A' or 'A-' in freshman computer science but failed the first programming assignment in their sophomore statics course.

- There was no statistically significant difference in programming assignment grades between male and female students.

- A strong understanding of linear algebra (as represented by an 'A' grade) appears to be helpful in this particular programming assignment.

Overall, this study highlighted the diversity of student programming experiences in our program. In our sophomore aerospace courses we have both students that are very experienced in computer science (CS) and have completed multiple university level CS courses, as well as students whose only CS experience was the required freshmen computing course taken a year ago. While sample sizes of students in certain "programming experience" bins (Figure 4) were too low for statistical analysis, the trends suggest that more CS experience is correlated with higher scores on the programming assignment. Therefore, we think that an important take-away from this study is that future aerospace CS assignments must be carefully designed to have programming and mathematical concepts that are challenging but doable for students across this spectrum.

In future work we would like to expand this study by analyzing a larger sample size of students to better understand the impact of high school and freshmen CS experiences on aerospace programming assignment performance. We hope to differentiate the impacts of 'good study skills' and 'strong programming background' on final student performance. We hypothesize that while many students were able to complete the assignment with an 'A,' the time that students spent on the assignment was highly variable and correlated with their programming experiences. An in-depth investigation would require additional quantitative data (e.g. detailed surveys of all students) or even qualitative research (e.g. interviews with students) to fully understand the variables that impact student programming performance in this aerospace department.

References

1. Lister, R. (2010). Geek genes and bimodal grades. *ACM Inroads*, 1(3), 16–17.
2. Patitsas E., Berlin, J., Craig, M. & Easterbrook, S. (2020). Evidence That Computer Science Grades Are Not Bimodal. *Communications of the ACM*, 63(1), 91-98.
3. Robins, A. V. (2010). Learning edge momentum: A new account of outcomes in CS1. *Computer Science Education*, 20, 37–71.
4. Robins, A. V. (2019). Novice Programmers and Introductory Programming. In S. A. Fincher & A. V. Robins (Eds.), *Cambridge Handbook of Computing Education Research* (pp. 327–376). Cambridge, UK: Cambridge University Press.
5. Margolis, J., Estrella, R., Goode, J., Jellison-Holme, J., & Nao, K. (2008). *Stuck in the Shallow End: Education, Race, and Computing*. Cambridge, MA: MIT Press.
6. Google Inc., & Gallup Inc. (2015). *Searching for computer science: Access and barriers in U.S. K–12 education*. Retrieved from http://g.co/cseduresearch
7. Milner IV, H. R. (2012). Beyond a test score: Explaining opportunity gaps in educational practice. *Journal of Black Studies*, 43(6), 693–718.
8. Lewis, C. M., Titterton, N., & Clancy, M. (2012). Using collaboration to overcome disparities in Java experience. In *Proceedings of the Ninth Annual International Conference on International Computing Education Research* (pp. 79–86). New York: ACM.

9. Google Inc., & Gallup Inc. (2017). *Encouraging Students Toward Computer Science Learning. Results From the 2015–2016 Google–Gallup Study of Computer Science in U.S. K–12 Schools* (Issue Brief No. 5). Retrieved from https://goo.gl/iM5g3A
10. Lewis, C. M., Shah, N. & Falkner, K. (2019). Equity and Diversity. In S. A. Fincher & A. V. Robins (Eds.), *Cambridge Handbook of Computing Education Research* (pp. 481–510). Cambridge, UK: Cambridge University Press.
11. Secules, S., Gupta, A., Elby, A., & Turpen, C. (2018). Zooming out from the struggling individual student: An account of the cultural construction of engineering ability in an undergraduate programming class. *Journal of Engineering Education*, 107(1), 56-86.
12. Davies S. P. (1993). Models and theories of programming strategy. *International Journal of Man–Machine Studies*, 39, 237–267.
13. Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14–18.
14. Soloway, E., Ehrlich, K., Bonar, J., & Greenspan, J. (1983). What do novices know about programming? In B. Shneiderman & A. Badre (Eds.), *Directions in Human–Computer Interactions* (pp. 27–54). Norwood, NJ: Ablex.
15. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33, 125–180.
16. Byrne, P., & Lyons, G. (2001). The effect of student attributes on success in programming. In *Proceedings of the 6th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 49-52). New York: ACM.
17. Pea, R. D., & Kurland, D. M. (1984). *On the Cognitive Prerequisites of Learning Computer Programming*. Technical Report No. 18. New York: Bank Street College of Education.