Computing Trajectories: Pathways into Computer Science and Programming Experience in the First Year

Darren K. Maczka

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Engineering Education

Jacob R. Grohs, Chair
Walter C. Lee
Marie C. Paretti
Manuel Pérez Quiñones

July 1, 2019
Blacksburg, Virginia

Keywords: computer science education, first-year engineering, MATLAB, broadening participation

Computing Trajectories: Pathways into Computer Science and Programming Experience in the First Year

Darren K. Maczka

(ABSTRACT)

Universities across the United States have been experiencing an increased demand for computer science majors. Adjusting curriculum to meet demand runs the risk of hindering ongoing efforts to broaden participation in computer science. To manage growth, and increase the representation of women and underrepresented minorities in the field, we must first understand current patterns for participation, and factors that impact access and persistence.

Universities with common first-year engineering programs present an opportunity for addressing some of the barriers that have traditionally limited access to computer science to certain groups. In particular, common first-year programs could provide early positive experiences with computer programming which encourage more students to consider computer science as a viable major.

To better understand how a common first-year engineering program may impact matriculation and persistence in computer science, I conducted studies to identify high-level patterns of participation in computer science, as well as how students experience programming instruction in an introductory engineering course. All studies share the same context: a large public research institution with a common first-year engineering program.

Results indicate that women are leaving computer science at all points of the curriculum, contributing to a reduced representation of women earning CS degrees. In contrast, URM and first-generation students have higher representation in the group earning CS degrees than in the group first declaring a CS major.

Computing Trajectories: Pathways into Computer Science and Programming Experience in the First Year

Darren K. Maczka

(GENERAL AUDIENCE ABSTRACT)

Many universities across the United States have been experiencing an increased demand for computer science majors. Adjusting curriculum to meet demand runs the risk of damaging efforts to increase the diversity of the computer science workforce. To manage growth and increase the representation of women and underrepresented minorities (students who are not white or East Asian) in the field, we must first understand who currently studies computer science, and factors that lead to their success in the major.

Universities with general first-year engineering programs present an opportunity for addressing some of the barriers that have traditionally discouraged women and underrepresented minorities from pursuing computer science. In particular, these programs could provide early positive experiences with computer programming which encourage more students to consider computer science as a possible major.

To better understand how experiences during students' first-year transition to college may impact decisions to major in computer science, I conducted studies to explore who enters computer science, and how they succeed in the major, as well as how students experience programming instruction in an introductory engineering course. All studies share the same context: a large public research institution with a general first-year engineering program.

Results indicate that women are leaving computer science at all points of the curriculum, contributing to a reduced representation of women earning CS degrees. In contrast, underrepresented minority students and students with parents who did not receive a college degree, make up a higher percentage in the group graduating with a CS degree than in the group who declare CS as their first major.

Dedication

for deadlines

Acknowledgments

To my advisor, Dr. Jacob Grohs: Thank you for your continued mentorship and support through this process. Our conversations have been immensely valuable in helping me focus my ideas. Thank you also for your unwavering support of all of my service activities, and the reminder of the value of a holistic professional experience.

To members of my dissertation committee, Drs. Marie Paretti, Walter Lee, and Manuel Pérez Quiñones: I have learned so much from each of you and your feedback through this process. Each of you has helped shape my research in unique ways, and the combination is truly greater than the sum of the parts.

To members of my research group: Michelle Soledad, Andrew Gillen, Tawni Paradise, and Stacey Kelly, I will always value the support you have provided and encouragement you continue to give. I would not be where I am today without all of you.

This work was partially supported by National Science Foundation grant No. DUE-1712089, as well as the Graduate Research Development Program administered by the Graduate Student Assembly at Virginia Tech. Any opinions, conclusions, or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the National Science Foundation, or the Graduate Student Assembly at Virginia Tech.

Attributions

The three manuscripts that make up this dissertation are the product of original work in which I had primary authorship and research responsibilities. However, I acknowledge that each includes intellectual contributions from other researchers within the Department of Engineering Education, who will receive credit as co-authors when the manuscripts are submitted for publication.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

Universities across the United States have been experiencing an enrollment boom in Computer Science (CS) majors over the past several years (NAE 2018). In parallel, there has been continued concern about the under-representation of women and racial minority students pursuing computer science (Google Inc. and Gallup Inc. 2016). Taken together, this poses a dilemma: increased enrollment provides an opportunity to attract and retain historically underrepresented groups to CS, but also raises the risk that adjustments to curriculum to accommodate the growth will disproportionately benefit those who are already in the majority (Roberts 2016).

While computer programming is often associated with the field of computer science, it is also recognized as a core skill that all engineers should learn (Azemi and Pauley 2008; Kramer 1994; Reid and Reeping 2014, e.g.). This presents an opportunity for engineering programs to address some of the identified barriers that have contributed to the lack of diversity in computer science. Universities with common first-year engineering (FYE) programs may be particularly well suited to make a positive impact on broadening participation in computer science. Because students in common first-year engineering programs do not declare an engineering major until after their first year (Orr et al. 2012), this is an opportune time to provide experiences that help students develop confidence and interest in computer science. Further, first-year programs have been shown to improve persistence rates in engineering as well as reduce major switching within engineering (Orr et al. 2012). Thus, by providing positive early experiences with programming, first-year programs can potentially level the playing field between students who have pre-college experience with programming and those that do not, as well as affect students' perception of the nature of programming work. Together, this could encourage students that otherwise would not have considered CS a

viable option to choose to major in the field.

Introductory programming experiences are recognized as an influential factor in deciding to major (Lewis, Yasuhara, and Anderson 2011) and persisting in computer science (Barker, McDowell, and Kalahar 2009). For these reasons, programming experienced introduced in a common FYE program is an important leverage point to attract students to the major. For students that decide to pursue computer science, the introductory disciplinary programming courses, referred to as CS1 and CS2, have been identified as a potential control point to encourage a more diverse population to persist in computer science (Cohoon and Tychonievich 2011).

1.1 Need for this Research

While first-year engineering programs have the potential to play an important role in helping a more diverse student body find their way to computer science, it is not clear if this is happening, or what factors may contribute to student perceptions of computer science as a potential degree option.

While programming experiences are common in introductory engineering programs, there are notable pedagogical challenges with teaching programing in this context. First, teaching programing is challenging in general, even for CS departments (Sleeman 1986), and second, there is evidence that taking a discipline-based approach to teaching programming is more effective (Magana, Falk, and Reese Jr 2013). The latter finding presents somewhat of a dilemma for designing first-year general engineering programming experiences because a good introduction for non-CS-majors will generally be different from a good introduction for CS-majors (Forte and Guzdial 2005).

To determine the efficacy of utilizing the first-year experience to broaden participation in computing, we first need to understand existing patterns of matriculation and persistence in

computer science, with a focus on the courses students are taking during this time.

1.2 Overview of the Study

This dissertation consists of three manuscripts, each a study that identifies potential leverage points to help broaden participation in computer science. All three studies share the same context: a large public research institution, referred to as Large Research Institution (LRI), with a common first-year engineering program, and where computer science is included in the college of engineering.

1.2.1 Site Description

LRI is a large public land grant institution located in a rural part of the mid-Atlantic region of the United States. According to 2018 census data, the surrounding county is 86% white and 52% male. At LRI, the undergraduate enrollment is between 20 and 30 thousand students: 65% white and 57% male. Engineering majors are a significant portion of the institution's population: degrees from the College of Engineering (COE) made up 27% of the bachelor's degrees awarded by LRI between 2014 and 2017.

Like many other large land-grant institutions (Chen et al. 2013), LRI has a common first-year engineering (FYE) program comprising a common curriculum, as well as first-year advising and other services designed to support first-year engineering students and provide information to help select a disciplinary major. Upon acceptance to the COE, students are not immediately admitted to a disciplinary major, but instead are considered General Engineering (GE) students. During a mandatory orientation session, students are requested to complete a survey in which they indicate their intended choice of major, this does not guarantee a major, but helps advisors provide recommendations for course enrollment.

Also during the orientation session, students attend a presentation of the engineering majors

available, and have access to academic advisors. Once enrolled, students receive a general email with time-specific information from an assigned academic advisor every two weeks during the semester during their time in the General Engineering program. During a student's second term in the GE program, they are asked to indicate their top three choices for major. Students who earn a 3.0 GPA are guaranteed their first choice of major, students with lower GPAs are assigned either their 1st, 2nd, or 3rd choice of major based on their GPA competitiveness and the available seats. With an average GPA of 3.34 for admitted students, CS is one of the more competitive majors in the COE at LRI.

There are a number of support services available to COE students, including a center with a mission to run programs to support diverse populations, as well as strong living learning communities (LLCs) including gender segregated communities for engineering students. These types of support services, and LLCs in particular, can contribute to a greater sense of belonging (Inkelas 2008), and encourage persistence.

The common curriculum for GE students includes a two-semester introductory course sequence FYE1 and FYE2. Like other introductory engineering courses (Reid and Reeping 2014), FYE1 and FYE2 cover a broad range of topics including communication skills, global awareness, design skills, and working with engineering tools. While acquiring skill in computer programming is not a listed required course outcome, MATLAB instruction makes up a significant portion of the curriculum and aligns with the learning outcome related to the use of engineering tools.

The FYE program is indented to provide students with sufficient experiences and support to make an informed decision about a disciplinary major by the end of their second semester. However, according to the published timetable for the computer science program, CS majors require an introductory programming course sequence CS1 and CS2, and CS1 is shown to be taken in a student's second semester, while they are still GE students. This may appear

to students that if they are to major in CS, they must come to this decision prior to their second term so that they can enroll in CS1 for an on-time graduation. It is worth noting that the requirements of this course sequence are flexible enough that it is possible to begin CS1 in semester 3 and still graduate with a CS degree in a 4-year time-frame. However, students may not realize this if they do not seek help from their academic advisor.

Student and Faculty Demographics. There is evidence that faculty representation can have a positive impact on retention of women (Bettinger and Long 2005), and underrepresented ethnic and racial minorities in STEM fields (Price 2010). Tables in Appendix A describe student demographics in both the COE as well as in CS across intersections of gender, URM, and first-generation status. For example, Table A.1 indicates that in the college, a higher percentage of URM students are also first-generation (31.1%) than non-URM students (13.3%). This relationship holds in CS as well (Table A.4) with 29.8% of URM CS students also being first-generation compared to only 14.6% of the Non-URM CS population.

Table 1.1 shows representation of faculty in the COE (N=300-400) and CS (N=20-40) in both 2009 and 2016.

Table 1.1: Faculty Representation in the COE and CS

	2009		2016	
	COE	CS	COE	CS
Female	12%	6.9%	16.2%	13.5%
African-American	3.4%	0	1.9%	0
Asian-American	19.3%	24.1%	25.6%	37.8%
Hispanic	3.7%	3.4%	3.3%	0
Unknown	0	0	5.8%	5.4%

While race and ethnicity information are recorded by different systems for faculty and students, it is clear that the combine representation of African-American and Hispanic faculty in the college during both 2009 and 2016 (7.1% and 5.2%) is less than representation of

URM students in the college (9.8%). The representation gap in CS is even greater, with no African-American or Hispanic faculty in 2016. Representation of women increases in the faculty between 2009 and 2016 in both the college, and in CS, though CS has a lower representation of women faculty than the college for both years. Both CS and the college have a lower representation of women in the faculty than in the student population.

1.2.2 Manuscript 1: Patterns of Participation in Computer Science

A first step towards understanding how a common first-year program may impact matriculation and persistence in CS is the exploration of how representation in the college, and in CS changes from the time students arrive at the university, to when they graduate. This study investigates patterns of matriculation, persistence, and graduation in CS across several demographic variables. It addresses the research questions

1. Who majors in Computer Science (CS) compared to other majors in the college?
2. How do metrics of success for populations of interest in CS compare to other Engineering majors?
3. For people who leave CS without a degree, when do they do so?

This study leverages existing population data for all engineering students at Large Research Institution and descriptive analysis to identify patterns that inform future efforts to improve matriculation and persistence rates for underrepresented populations in computer science. Findings indicate that representation of women, URM, and first-generation students change after the common first-year. Representation of women drops during this time, while representation of URM and first-generation students increases.

1.2.3 Manuscript 2: Migration Pathways into Computer Science

Using the same dataset as in Manuscript 1, this study shifts the focus to two introductory programming courses, CS1 and CS2. It addresses the research questions:

1. What associations exist between gender, URM, and first-generation status and performance in CS1 and CS2?
2. How does performance in CS1 affect matriculation rates into CS?
3. How does performance in CS1 and CS2 affect graduation rates and time-to-degree in CS?

Centered around a framework that models how performance in introductory courses may impact persistence in a discipline (Suresh 2006), this study uses multinomial and binomial logistic regression to explore the effects of several predictor variables on the performance in CS1 and CS2, as well as matriculation and graduation rates in CS.

1.2.4 Manuscript 3: MATLAB Instruction in a First-Year Engineering Course

This study takes a qualitative approach to understand how students in a first-year engineering course perceive instruction related to MATLAB. As the only programming experience common to all incoming engineering students, MATLAB as introduced in the FYE course sequence serves as a potential leverage point to provide early positive programming experiences for students who may not have had access to such experiences in high school. Because MATLAB itself is not associated with computer science, it is unclear if students would make connections between computer science skills and the MATLAB instruction they receive during their first year. The purpose of this study is to explore this possibility by addressing the following research questions:

1. How do first-year engineering students describe MATLAB instruction?
2. What are the contextual factors that mediate student experiences with MATLAB instruction?

The objective of this study is to shed light on the challenges in the use of a general engineering introduction to programming as a formative programming experience with the potential to generate interest in majoring in CS.

1.3 Contributions

Findings from the three studies can inform efforts to leverage first-year experiences to broaden participation in computer science.

1.3.1 Manuscript 1

This research provides context for understanding how representation in computer science changes at different points in the curriculum at a university with a common first-year engineering program. It finds that representation of women starts low, and continues to drop at successive decision points. This pattern is consistent with other researcher's findings regarding women's participation in computer science. In contrast to the pattern seen in women, this study finds the reverse in URM and first-generation college students. The representation of these two groups starts out low, but increases after the first year suggesting that some aspect of the experience during the first year of these students is encouraging them to pursue computer science.

1.3.2 Manuscript 2

This research finds that while performance in CS1 and CS2 likely have some impact on matriculation and persistence rates in computer science, their explanatory power alone is

weak and therefore it is likely there are additional factors that are not accounted for in the data. Additionally, it finds that of the students that are high performing in CS1 as determined by final grades, women matriculate at higher rates into CS than the average. This finding is complicated by the fact that CS1 is not an exclusive prerequisite for computer science, and therefor suggests future research needs to be done to better understand who is enrolling in CS1 and why.

1.3.3 Manuscript 3

This research provides insights into students' perspective of the MATLAB instruction they experience in their first-year engineering course. Findings suggest that many contextual factors interact such that similar instructional strategies are experienced quite differently by different students. To help understand the findings, the Identity-based Motivation (Oyserman et al. 2017) is used as a framework to discuss the findings and identify promising strategies to guarantee productive experiences for more students.

References

Azemi, Asad and Laura L. Pauley (2008). "Teaching the Introductory Computer Programming Course for Engineers Using Matlab." In: *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*. IEEE, T3B–1 (cit. on p. 1).

Barker, Lecia J., Charlie McDowell, and Kimberly Kalahar (2009). "Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major." In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA). SIGCSE '09. New York, NY, USA: ACM, pp. 153–157. DOI: 10.1145/1508865.1508923 (cit. on p. 2).

Bettinger, Eric P. and Bridget Terry Long (2005). "Do Faculty Serve as Role Models? The Impact of Instructor Gender on Female Students." In: *American Economic Review* 95.2, pp. 152–157 (cit. on p. 5).

Chen, Xingyu et al. (2013). "A Taxonomy of Engineering Matriculation Practices." In: 2013 ASEE Annual Conference & Exposition, pp. 23.120.1–23.120.13 (cit. on p. 3).

Cohoon, James P. and Luther A. Tychonievich (2011). "Analysis of a CS1 Approach for Attracting Diverse and Inexperienced Students to Computing Majors." In: *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. New York, NY, USA: ACM, pp. 165–170. DOI: 10.1145/1953163.1953217 (cit. on p. 2).

Forte, Andrea and Mark Guzdial (2005). "Motivation and Nonmajors in Computer Science: Identifying Discrete Audiences for Introductory Courses." In: *Education, IEEE Transactions on* 48.2, pp. 248–253 (cit. on p. 2).

Google Inc. and Gallup Inc. (2016). *Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics* (cit. on p. 1).

Inkelas, Karen (2008). *National Study of Living-Learning Programs: 2007 Report of Findings* (cit. on p. 4).

Kramer, K. A. (1994). "Using C Programming as a Vehicle to Overcome Barriers." In: *Proceedings of 1994 IEEE Frontiers in Education Conference - FIE '94*. Proceedings of 1994 IEEE Frontiers in Education Conference - FIE '94, pp. 30–33. DOI: 10.1109/FIE.1994.580463 (cit. on p. 1).

Lewis, Colleen M., Ken Yasuhara, and Ruth E. Anderson (2011). "Deciding to Major in Computer Science: A Grounded Theory of Students' Self-Assessment of Ability." In: *Proceedings of the Seventh International Workshop on Computing Education Research*. ICER '11. New York, NY, USA: ACM, pp. 3–10. DOI: 10.1145/2016911.2016915 (cit. on p. 2).

Magana, Alejandra J., Michael L. Falk, and Michael J. Reese Jr (2013). "Introducing Discipline-Based Computing in Undergraduate Engineering Education." In: *ACM Transactions on Computing Education (TOCE)* 13.4, p. 16 (cit. on p. 2).

NAE, National Academy of Engineering (2018). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. National Academies Press. DOI: 10.17226/24926 (cit. on p. 1).

Orr, M. K. et al. (2012). "Engineering Matriculation Paths: Outcomes of Direct Matriculation, First-Year Engineering, and Post-General Education Models." In: *2012 Frontiers in Education Conference Proceedings*. 2012 Frontiers in Education Conference Proceedings, pp. 1–5. DOI: 10.1109/FIE.2012.6462357 (cit. on p. 1).

Oyserman, Daphna et al. (2017). "An Identity-Based Motivation Framework for Self-Regulation." In: *Psychological Inquiry* 28.2-3, pp. 139–147. DOI: 10.1080/1047840X.2017.1337406 (cit. on p. 9).

Price, J. (2010). "The Effect of Instructor Race and Gender on Student Persistence in STEM Fields." In: *Economics of Education Review* 29.6. 901, pp. 901–910. DOI: 10.1016/j.econedurev.2010.07.009 (cit. on p. 5).

Reid, Kenneth and David Reeping (2014). "A Classification Scheme for "Introduction to Engineering" Courses: Defining First-Year Courses Based on Descriptions, Outcomes, and Assessment." In: *American Society for Engineering Education Annual Conference & Exposition. Indianapolis, IN (1-11). Washington DC: American Society for Engineering Education* (cit. on pp. 1, 4).

Roberts, E. (2016). "A History of Capacity Challenges in Computer Science." In: *Retrieved June* 1, p. 2016 (cit. on p. 1).

Sleeman, Derek (1986). "The Challenges of Teaching Computer Programming." In: *Commun. ACM* 29.9, pp. 840–841. DOI: 10.1145/6592.214913 (cit. on p. 2).

Suresh, Radhika (2006). "The Relationship between Barrier Courses and Persistence in Engineering." In: *Journal of College Student Retention: Research, Theory & Practice* 8.2, pp. 215–239. DOI: 10.2190/3QTU-6EEL-HQHF-XYF0 (cit. on p. 7).

Chapter 2: Manuscript 1

Patterns of Participation in Computer Science

Target Journal: IEEE Transactions on Education

Abstract

With growing demand for computer science graduates, and an ongoing concern that those who enter and graduate with computer science degrees are not representative of the population at large, there is a need to understand patterns of matriculation and graduation within CS. This study employs quantitative analysis of institutional data and archived survey data from the first-year engineering program of a large research institution to explore patterns of participation in computer science. We explore differences between gender, underrepresented minorities, and first-generation college students with respect to their rates of persistence and graduation with Computer Science Degrees. Consistent with other studies, we find that women matriculate into CS at lower rates then men, and leave CS at higher rates. URM and first generation students, however, have relatively higher matriculation and persistence rates in CS compared to their peers.

2.1 Introduction

As pressure to broaden participation within science, technology, math, and engineering (STEM) fields continues, there is mounting evidence that variation in participation within STEM fields warrants discipline-specific analysis to guide broadening participation initiatives. Computer science has a particularly volatile history with regard to participation of women in the field: in the 1970s and early 1980s the percentage of women earning a bachelors degree in CS was increasing more than women's participation in other engineering and

professional fields (Henn 2014). The percentage peaked at 34% in 1984 and has been in decline since; in 2016, just under %19 of the CS bachelors degrees awarded were earned by women (NSF 2019). This reverse of the historical trend of gendered participation has been one reason efforts to broaden participation within CS have been focused on gender. However, some research suggests that in many cases there may be more variation in the factors thought to impact participation in CS within gender groups than across (Larsen and Stubbs 2005), so to understand who participates in CS and better design broadening participation initiatives we must take a more holistic view of the field and include factors beyond gender in our analysis.

In this study I explore patterns in matriculation and graduation rates in computer science at a large research institution. At this institution the Computer Science department is housed within the College of Engineering and students admitted to the college proceed through a common first-year program before matriculating into a degree major. While not universal, this model of a common first-year engineering experience is replicated in many large research institutions, and has been shown to improve retention (Orr, Brawner, Ohland, et al. 2013). Because of the first-year experience at this institution, many students form or refine their choice of major during the first year. These major switches, along with persistence and graduation data provide information about three critical decision points to understand pathways into, and out of, CS across different demographics. Illuminating patterns in participation in computer science will help guide retention efforts, as well as inform future research to understand how and why different demographics tend to experience different pathways. In particular, the context of a common first-year engineering program may present an opportunity to broaden participation in CS by providing positive perceptions of the field to students who arrive knowing little about it. To this end, this study addresses the following research questions:

1. Who majors in Computer Science (CS) compared to other majors in the college?
   (a) Who arrives at the university intending to major in CS?
   (b) Who declares a major of CS after their first year?
   (c) Who eventually graduates with a CS degree?
2. How do metrics of success for populations of interest in CS compare to other Engineering majors?
3. For people who leave CS without a degree, when do they do so?

2.2 Literature Review

Language of "broadening participation" in engineering is often cast in terms of increasing diversity, or percent representation of people who have traditionally been underrepresented in the field, for example women and people of color. Broadening participation has been understood through several paradigms: 1) the pipeline–who shows up, 2) pathways, where do they go, and 3) ecosystems–what types of experiences they have on the way (Lee 2019). This work is situated around the pipelines metaphor, which despite having received much criticism (e.g. Fealing, Lai, and Myers Jr 2015; Miller 2015), still provides value for identifying systemic problem areas associated with matriculation and graduation rates to study further (Lee 2019; Lord et al. 2019). Figure 2.1 depicts a simplified computer science pipeline. Work situated around the pipeline-metaphor typically looks at who is attracted to the field, who is retained, and who graduates. Identifying where along the pipeline people are being pushed out, i.e. leaking, can help identify systemic issues that need to be addressed.

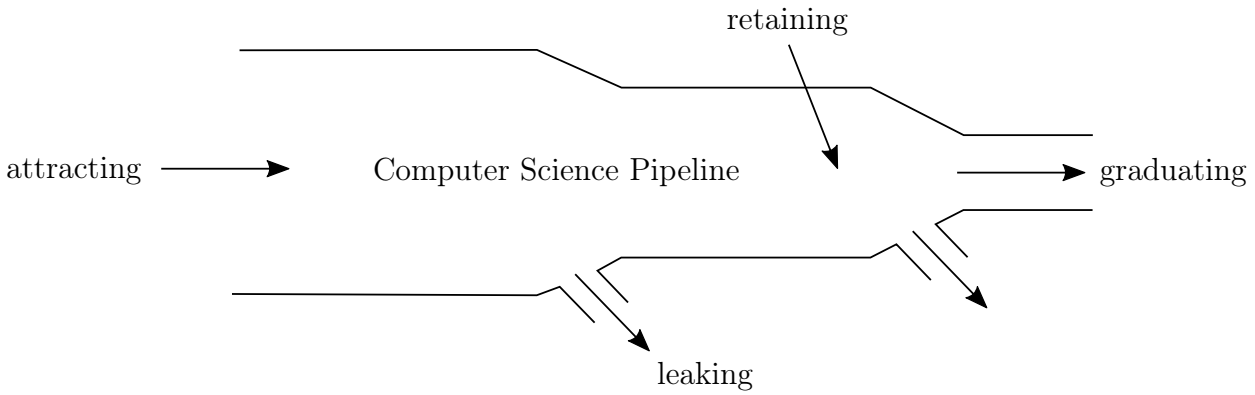

Figure 2.1: The pipeline model can draw attention to deficiencies in the system if we start from the premise that it should be working equally well for everyone. Adapted from Lord et al. (2019)

Fealing, Lai, and Myers Jr (2015) note that "a pipeline is understood to be a favored or privileged route", one that has historically worked for the dominant group. With this in mind, we can take a critical view of a particular pipeline with regard to where members of the non-dominant group are disproportionately leaving. These breaks in the pipeline can be seen as a fault of the system itself, rather than as an indicator of deficiencies of the people passing through it. Such an approach is useful in its ability to highlight barriers to access and participation via traditional routes (Lee 2019). In the context of computer science, non-dominant groups include women, underrepresented minorities (URM), and first-generation students.

2.2.1 Women in Computer Science

From 2004 to 2014, the percentage of computer science bachelors degrees awarded to women dropped from 25% to 18% (NCSE 2018). Historically, the participation of women in computing has been much greater than present day and as such the decline of women in computer science has been the subject of much analysis.

Providing Historical Context. A criticism of the pipeline metaphor with regard to women in computing is that it tends to overlook important social and historical contexts of computing (Vitores and Gil-Juárez 2016). However, drawing on literature of migratory flows, Lord et al. (2019) suggest that understanding how migratory patterns are rooted in historical and social structures can aid in the use of the pipeline metaphor as a framework for understanding variation in migratory patterns. This is particularly advantageous in the case of understanding current trends within Computer Science as the decline in representation of women in the field is a relatively recent phenomenon; prior to 1984, the representation of women in computer science was increasing at rates consistent with other professional fields.

Compared to other professional fields, computer science is fairly young, and changes in women's participation can be broadly explained by the maturation of the field through the assimilation into existing domains of business and academia (Abbate 2012). In the early days of computing, high demand for computer operators and programmers coupled with the perception of the work associated with these jobs transferred from the work they were replacing–mainly clerical–contributed to a sharp increase in women earning Computer Science degrees in the 1960s and 1970s. As computing became more heavily integrated into day-to-day business operations, the perceived value of the work increased, as did the desire to provide a path to management from entry level computing careers, which began to force women into dead-end data-entry jobs (Ensmenger 2010). Meanwhile, computer science programs at universities had long been structured around the assumption that most new computer science students had never programmed, or even seen a computer before. With the introduction of the personal computer in the 1980s, and an enrollment boom in computer science programs, departments began adjusting their curriculum to assume a certain degree of prior experience programming computers. The marketing of the personal computer as exclusively boys' toys ensured that young men were disproportional more likely to have

had this prior experience. Women, who were less likely to have had access to a personal computer before college, struggled with introductory course expectations while their male peers appeared to get by with relative ease (Margolis and Fisher 2002b).

Efforts to Understand and Reverse the Decline. The decline in women's participation in Computer Science in the 1980s has inspired a large body of work in an attempt to understand, and reverse the trend. In their critical review of research on women's participation in computer science, Cohoon and Aspray (2008) conclude that women are not avoiding or leaving computer science due to academic reasons, or lack of drive, but instead are leaving due to perceptions of the culture of computing, lack of role models, and lack of peer support. These findings are largely consistent with others that look at women's participation in STEM more broadly, (e.g. Lichtenstein et al. 2014; Seymour 1995). Broader efforts to understand women's participation in STEM fields indicate that gender plays a particularly strong role in participation in decisions to leave computer science compared to most other STEM fields (Christopher Strenta et al. 1994).

Cohoon (2001) found that retention of women CS students varied by type of institution and department characteristics, and some departments retained women at rates on par with men. The departments that exhibited equal retention rates between men and women tended to have at least one women faculty member, strong institutional support, a relatively high percentage of women students in the department, and value teaching. Conducting a study of persistence in computer science at Carnegie Mellon University, Margolis and Fisher (2002b) found that women persisted with computer science both for the expected reasons of identifying with the work and having a history of access to a computer, but also for reasons external to the field itself, such as a desire for a stable economic future. This finding is further expanded upon by Larsen and Stubbs (2005), who find that in terms of perceptions of the field there is more variability in how students talk about computer science within gender groups

than across, suggesting a complex interaction of factors influence women's decisions to stay or leave computer science.

Even when women persist through graduation and earn a computer science degree, Main and Schimpf (2017) found they are 2.5 times more likely than men to leave computing careers, "due to work-family conflict, the pervasive occupational culture, and lack of mentoring and networking opportunities" (p.302). Thus, consistent with reasons for leaving computer science during school, the forces that tend to drive women out are largely cultural, rather than a lack of affinity or skill for the work (Beyer and Haller 2006).

2.2.2 Beyond Gender

While the underrepresentation of women in computer science has received much focus, gender is not the only way to view diversity in CS. Some have observed that computer science attracts a rather homogeneous population, and while most women may not identify with the stereotype (Lewis, Anderson, and Yasuhara 2016), many men do not fit the expectations of a "typical CS student" as well (Larsen and Stubbs 2005). This view could be used to help understand underrepresentation of other groups, such a URM and first-generation students. Because prior experience with programming, as well as familiarity with computing fields impact decisions to major in computer science, this could indicate barriers to participation of URM and first-generation students as their home and school environments may not provide access to these experiences and knowledge to the degree that non-URM and continuing-generation students have.

Chen and Carroll (2005) found that while first generation students took *computing* courses no less frequently than continuing generation students–those with at least one parent who graduated from college, first generation students were less likely to take *computer science* courses. That is, while they are showing an interest in the topic and skills related to com-

puting, they may be avoiding the computer science profession. The authors speculated that this was related to the tendency for first generation students to focus on vocational degrees and the perception that computer science as a major was more abstract. It is important to note that this study was published in 2005 and thus perceptions of computer science may have changed, namely due to increased awareness of the role of computer science in today's economy via high profile companies such as Google and Facebook, and in popular media such as David Fincher's *The Social Network* (2010).

It is likely that parental involvement and support plays a significant role in college major choice (Ma 2009), but while first-generation families are supportive, in terms of guidance into specific majors, first-generation students seek direction from their teachers, and guidance-counselors, who play a strong role in the decision for first-generation students to major in engineering (Trenor 2009). Once in an engineering major, continued support from family, as well as performance measures influence first-generation students to persist in engineering (Garriott, Navarro, and Flores 2017).

2.2.3 Broadening Participation in Computing

There is continued interest in both increasing the number of people who acquire computer science skills, as well as broadening participation within the field of Computer Science. Reasons include an anticipated growing need for computing skills as well as the recognition that a diverse workforce is necessary to produce high quality and equitable products and services. Most work in this area leverage national datasets to demonstrate the under representation of various groups graduating with computer science degrees, or entering the workforce (Peckham et al. 2007). Rather than to take for granted that the current institution of focus is representative of national trends in the under-representation of certain populations within computer science, the objective of this study is to describe the current state of participation

in computer science at a particular which can serve as a foundation for subsequent efforts to broaden participation at this institution. In particular, the local context of a common first-year engineering program is an under-studied environmental factor in existing work that investigates the representation in computer science.

2.3 Methods

A thorough descriptive analysis of the existing landscape is a necessary prerequisite to any intervention design. Descriptive analysis can help identify phenomenons of interests and generate new research questions (Loeb et al. 2017). In particular, in the context of the literature reviewed, a descriptive analysis will help illuminate who migrates to and away from computer science at a large research institution, at what critical time periods, and help direct future lines of inquiry. Given the lack of literature exploring the effects of a common-first year program on participation in computer science, this analysis will provide help fill this gap and provide evidence to assess whether institutions with common first-year programs could provide unique opportunities for broadening participation within computer science that may not be possible or feasible at other institutions.

2.3.1 Site Description

Large Research Institution (LRI) is a large public research institution with a national reputation for a strong engineering program. Like many institutions, at LRI Computer Science is housed within the College of Engineering (COE). To understand how LRI may be representative of other CS degree granting institutions we can compare with results of the Taulbee survey, which LRI participates in, sent by the Computing Research Association each year to all PhD-granting departments of Computer Science, Computer Engineering, and Information in North America (CRA 2015). Table 2.1 compares CS degrees awarded by Taulbee institutions in 2016 to those awarded by LRI.

Table 2.1: CS Degrees awarded 2016-2017 by Taulbee institutions and LRI

	Taulbee	LRI
Female	7143	67
Male	31511	344

Results of a χ^2 test indicate that the gender ratio at LRI is not significantly different than the Taulbee schools, and thus findings related to gender and CS at LRI may generalize to other Taulbee schools. At LRI just 10% of enrolled CS majors are identified as URM which is significantly lower than URM representation across Taulbee institutions ($\chi^2(1) = 51.01, p < 0.001$), so results pertaining to URM students may not generalize to other Taulbee schools if demographic representation at the institution level is an important factor in the participation of URM students in CS.

Like other institutions across the country (National Academies of Sciences 2017), LRI has experienced a large increase in the number of students enrolling as CS majors since 2012, as seen in Table 2.2. This corresponds with increased national pressure to grow the computer science workforce and rising awareness of the field through popular culture (National Academies of Sciences 2017)

Table 2.2: Number of students who are enrolled in CS by year.

AY	n	% growth
2011	491	0.00
2012	576	17.31
2013	671	36.66
2014	686	39.71
2015	749	52.55
2016	832	69.45
2017	933	90.02

Enrollment for a given year is determined if a student has their major declared as CS for that year.

Because we have population data for all students at LRI, inferential statistics are not nec-

essary. Findings are representative of LRI and any generalizations beyond should consider similarities between LRI and the institution of choice. Characteristic features of the computer science pathway at LRI is that 1) the Computer Science (CS) department is housed withing the College of Engineering (COE), and all students within the college participate in a first-year engineering (FYE) program prior to matriculating to a degree-granting major.

Like other schools with a FYE program (Chen, Brawner, et al. 2013), students admitted to the COE at LRI participate in a common first-year curriculum of general engineering (GE) coursework, activities to learn about the different engineering majors offered at LRI, and have access to first-year advisors who can assist students with course planning and provide guidance in selecting courses to meet requirements for students' desired major. As part of the required coursework, FYE students are instructed in the use of MATLAB as a programming environment. For many students, this may be the first experience they have had with programming of any kind. While MATLAB itself is not associated with computer science, it is possible students may have a positive experience with programming which could lead to an interest in computer science.

After completing the FYE requirements, students request up to three majors to begin in the following semester. Students with a GPA of 3.0 or higher are guaranteed to be placed in their first choice of major, while students with lower GPAs compete for limited seats. At LRI, Computer Science is one of the more competitive majors in the college with most of the students entering the major with a GPA of 3.0 or higher. As a consequence, students who express interest in computer science but have a GPA lower than 3.0 are encouraged by their academic advisors to consider a second choice of major.

Another consideration students who wish to major in CS at LRI must make is that unlike other majors in the COE, CS requires a prerequisite course that is typically taken during the first year, before students have matriculated into their majors. Thus, students who discover

CS as an interest during their first year may view this as a barrier if they had not planned their course-load around this requirement. Academic advisors are aware of these challenges and are adept at helping students interested in CS develop 4-year graduation plans even if they have to wait to take the first CS prerequisite course in their second year.

2.3.2 Population Description

Using institutional records obtained through NSF grant No. DUE-1712089 we obtained population data for students enrolled at LRI from Fall 2009 through Spring 2017. These data include majors enrolled in, courses taken, and degrees obtained, along with demographics such as gender, first-generation status, underrepresented minority (URM) status, and tuition status (in-state or out-of-state). URM is defined as having a non-White and non-East-Asian race. In addition to these data collected at the institutional level, the first-year program collects survey data from incoming students that ask about intended major. The survey is administered by the first-year advisors as part of student orientation and has an annual response rate ranging from 61-89% over the years for which we have data. This information about intended major is the only non-population level data used in this study.

To provide a broad understanding of who matriculates and persists in Computer Science, percentages of students associated with Computer Science were computed for three 'epochs': *incoming*, when students enter the college, *declared*, when students select their first degree-granting major, and *degree*, when students graduate and earn a degree. Students are counted in CS for each of these epochs as follows:

Incoming Students who are admitted to the college of engineering and indicate an intent to major in CS on their pre-orientation survey.

Declared Students who declare CS as their major of choice sometime after their first

semester in the college. For students who change their major this includes students who's last major change is to CS.

Degree represents the students who graduate with CS degrees.

2.4 Results

Figure 2.2 visualizes percent representation of female students, first generation students, and URM students in Computer Science compared to the college for each demographic. Representation of female students starts lower than the college average with pre-orientation interest and drops at each successive epoch. While the representation of female students in the college as a whole drops from the Declared epoch to the Degree epoch, the drop in representation in computer science is greater. The pattern in first generation and URM students is reverse, and in fact at pre-orientation first generation student representation in interest in computer science is below that of the college but representation of first-generation students increases to above the college representation once students declare majors. At the degree epoch, CS degrees has a higher representation of first-generation students than the average for all COE degrees.

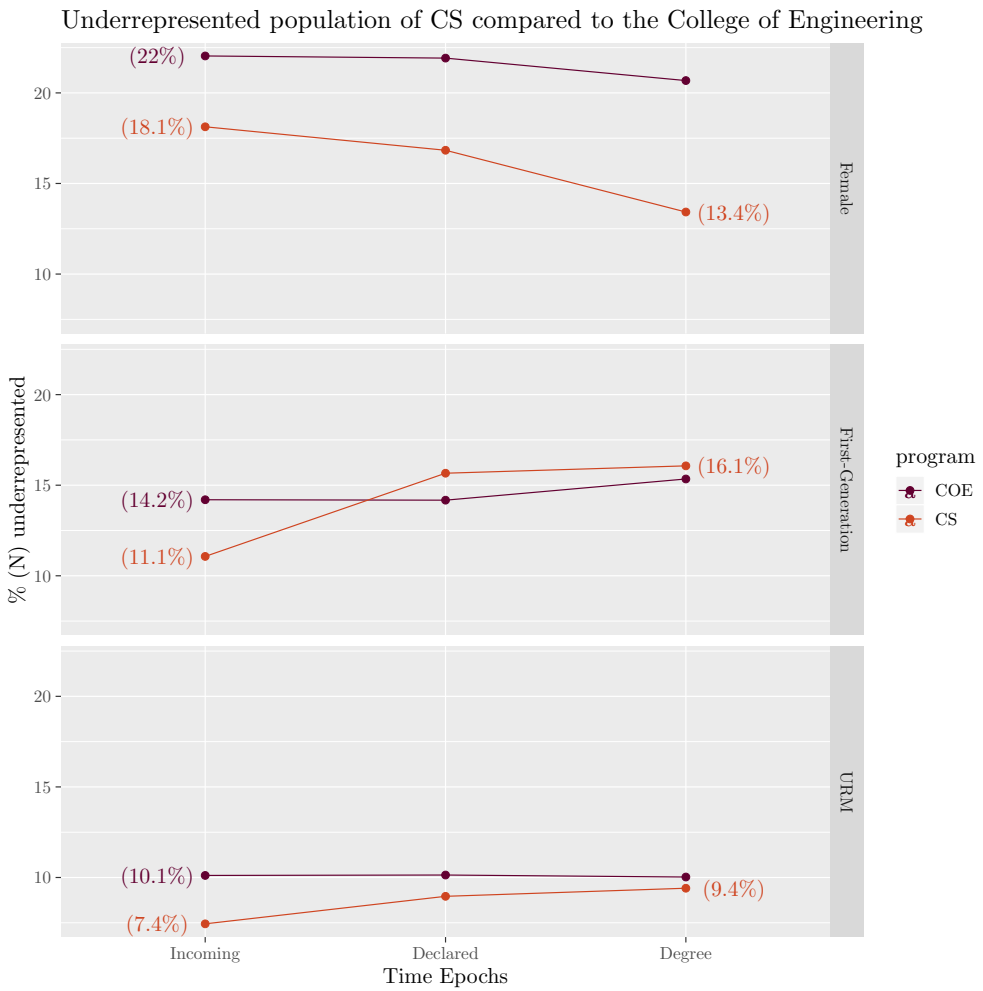

Figure 2.2: Computer Science Demographic changes at different time points compared to the College of Engineering.

Table B.1 in Appendix B contains the same information in Figure 2.2, but in table form and includes number of students in each epoch. Numbers of students may go up or down independently of the relative percentage compared to the college average. For example, from incoming to declared, the number of female students in CS increases from 95 to 201 while the percentage of female students falls. This is due to disproportionately more male students declaring CS as a major compared to those that declared an interest as incoming students.

2.4.1 Persistence and Yield

Since the pipeline metaphor places a focus on inputs and outputs, matriculation and graduation rates are unsurprisingly a common way to measure and report success within STEM and CS, as well as quantify diversity within these fields (Main and Schimpf 2017; Su 2010). Most studies that seek to understand representation challenges in computer science, or determine the effect of an intervention, identify the problem by citing data summaries in national reports such as NCSE (2018). While these serve as an important starting points, summaries of graduation rates by demographic in general cannot illuminate institution-specific patterns in matriculation and retention that are important for context-aware retention interventions.

Ohland et al. (2011) argue it is critical to examine multiple measures of success when comparing groups. They use the 8-semester persistences measure and 6-year graduation rate measure. They observe that while the 8-semester persistences has been demonstrated to be a strong predictor of 6-year graduation, by decomposing degree rate into 8-semester persistence and *yield*–the percentage of those that persist who go on to earn degrees within 6 years–important information about *how* graduation rates are achieved are revealed. The expressions for these quantities are shown in equation (2.1).

$$\begin{aligned} \%\text{ persistence} &= \frac{\#\text{ persisting to 8th term in COE, 6th term in major}}{\#\text{ matriculating into any COE major}} \\ \%\text{ yield} &= \frac{\#\text{ graduating in 6 years}}{\#\text{ persisting to 8th term in COE, 6th term in major}} \\ \text{6-year graduation rate} &= \%\text{ yield} \times \%\text{ persistence} \end{aligned} \tag{2.1}$$

While Ohland et al. (2011) use an 8-semester persistence rate and 6-year graduation rate from engineering, we would like to compare success within engineering majors. This is not as sim-

ple as conducting the same analysis within each major because LRI has a common first-year engineering program: students matriculate into the college some time before matriculating into disciplinary majors, so while it is reasonable to expect 8-semesters of persistence in the *college*, typical persistence within any given major would be less. For this analysis we use 8-semester persistence within the COE, combined with 6-semester persistence in a first declared major. This is a sub-population of all the students within the college as some enter the college but leave before declaring a major within the college. Students who do not persist for 8 semesters in the college and 6 semesters in their first major may still go on to earn a degree in the college if they change majors to another in the college. For example, if a student first selected CS and remained in that major for 1 semester before changing majors to CpE and remained in that major for 6 semesters, the student would have counted towards the lowering of the persistence rate for CS, but would not have contributed to the persistence rate of CpE. This analysis considers only first-time-in-college students. Transfer students, as well as those that switch to engineering after their first year at LRI, are excluded. Only the first undergraduate degree a student earned is included, thus if a student earns an undergraduate degree in EE in one term, and CS in a later term, only their EE degree and EE persistence will be included. The graduation time-window is 6 years from the first term at LRI, rather than first term in the College. Students who earn dual degrees in the same term are counted as having a degree in their matriculation major if either of their degrees matches their matriculation major.

Our data contain term-by-term records for each student where each record contains which college and major a student is enrolled in for a given term. To calculate length of time we need a common starting point. A reasonable strategy is to count from a student's first semester in the COE.

Gender Success Rate. Figure 2.3 shows persistence and yield rates for CS and other COE students grouped by gender. The size of each dot is proportional to the number of students in the respective group. Male CS majors are towards the high end of graduation rates with generally high persistence and yield rates. Looking at female students in the college, CS majors fall on the low end of graduation rates, primarily due to one of the lowest persistence rates of all majors. The yield rate for female CS students however is higher than for male CS students. In fact, all women who persist in the college for 8 semesters and in CS for 6 go on to graduate. It is important to keep in mind that the students that do not graduate with a CS degree may still be graduating with another degree.

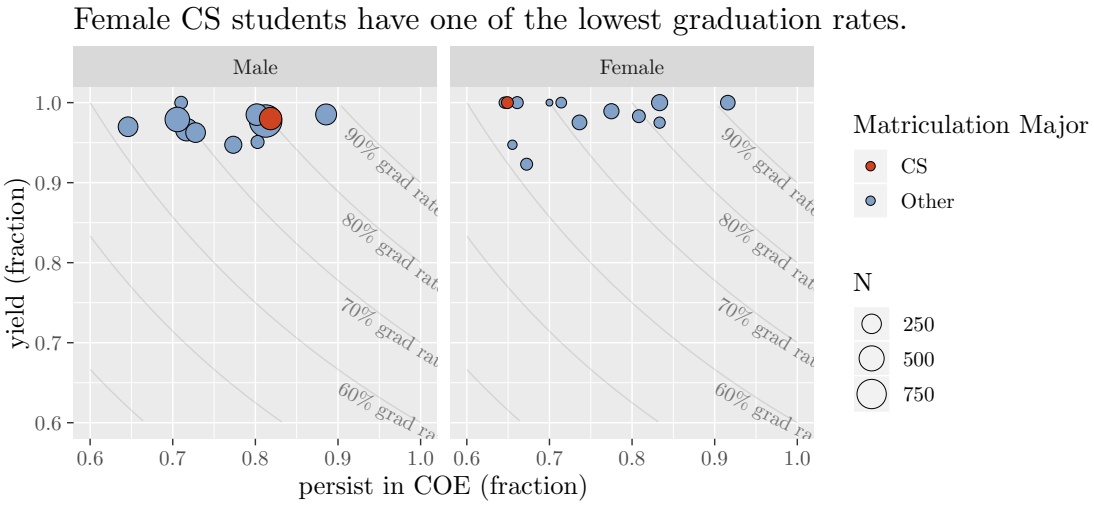

Figure 2.3: CS compared to other COE majors by gender. Dot size is proportional to number of students.

URM Success Rates. Figure 2.4 shows persistence and yield rates for URM CS students compared to those in other COE majors. Like Female students in this data set, all of URM students who persist in the college for 8 semesters and CS for 6 go on to earn a CS degree. Unlike Female students, URM CS students have a higher persistence rate than non-URM peers and as a result graduate at higher rates than their peers. Graduation rates for URM students in other COE majors are lower compared to their peers, a trend that as been

reported in other studies, (e.g. Ohland et al. 2011). With a few exceptions, URM students have lower yield rates than their peers across the college, but in CS the yield rate of URM students is higher than for non-URM students.

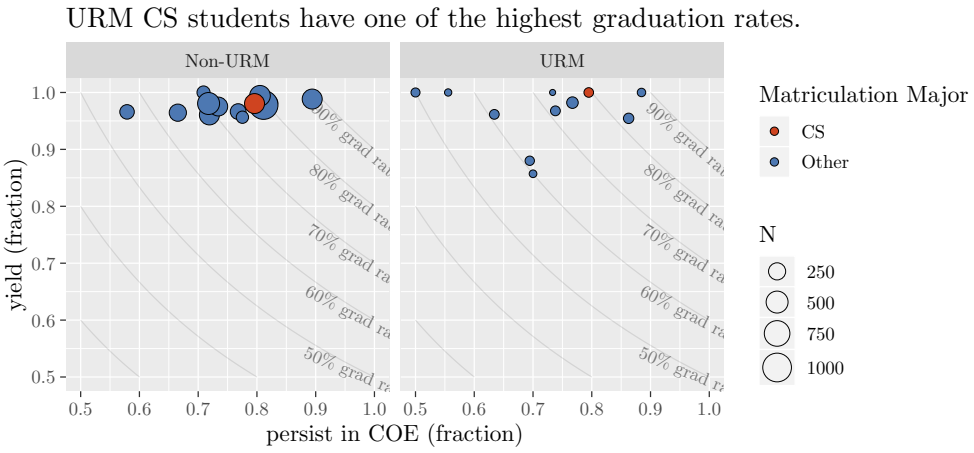

Figure 2.4: Comparing URM students across majors in the college. Dot size is proportional to number of students.

First Generation Success Rates. Figure 2.5 compares the 8 semester COE, 6 semester CS persistence and 6 year yield for first-generation and continuing-generation students. First-generation CS students have one of the highest graduation rates in the college, with the highest persistence rate. For first-generation students in other majors there is a general lowering of yield rates compared to continuing-generation students. This is consistent with other studies of first-generation students in engineering (e.g. Engle and Tinto 2008).

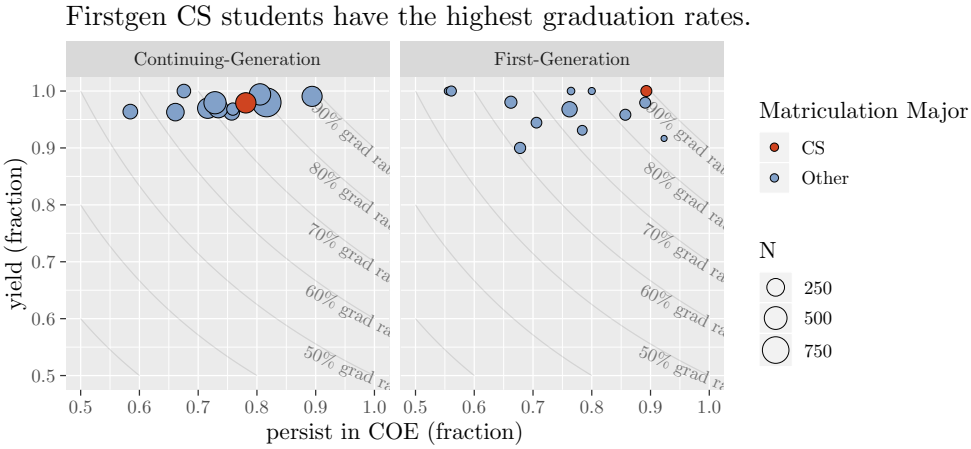

Figure 2.5: CS compared to other COE majors by first-generation status. Dot size is proportional to number of students.

2.4.2 Time to Exit

The previous analysis provides a snap-shot view of persistence and yield patterns, but does not provide any insight into time differences between groups. Event history analysis (EHA) is a method to estimate likelihood of an event of interest happening within some time period and is commonly used in medical research where it is usually referred to as survival analysis. As longitudinal educational datasets have become more readily available, so too as the use of event history analysis in education research (e.g. Min et al. 2011; Plank, DeLuca, and Estacion 2008; Yang 2017).

In the simplest case of event history analysis, there is a single event of interest that people are guarantee to experience at some point in time. People for whom the event has not occurred by the end of the study are considered "censored" and are still included in the probability estimations. While including all students in the analysis, even those for whom we do not have 6-years worth of data for, is beneficial, for many of the events we are interested in it is not appropriate to assume all students will experience them. For example, students may leave a degree program without a degree, or graduate with a degree, in addition to not having

experienced either event within the data. To analyze this type of data, a more general form of EHA called competing risks analysis is used. As the name implies, this analysis handles the case where mutually-exclusive events may occur. For example, Yang (2017) uses competing risks analysis to study the effect of variables such as family background, academic ability, and school characteristics, affect the probability of students dropping out of high school, or graduating. The analysis is not restricted to only two competing events, Scott and Kennedy (2005) use this technique to investigate the probability of students in associates programs to either dropout, graduate with a terminal AA degree (i.e., not go on to enroll in a bachelors program, or graduate and transfer to a bachelors program.

Because EHA can include participants for whom an event has not occurred yet, this analysis includes all available data, including students who's first enrolled term is less than 6 years prior to our most recent data. While we are still working with population data, since the analysis is estimating probability of an event occurring, including participants for whom the event has not occurred yet, there is uncertainty associated with the predictions.

For this analysis, I investigate the time-to-event for either graduating with a CS degree, or leaving the CS program. Leaving the program includes both switching to a different degree program, within or out of the COE, as well as leaving the university entirely.

Gender. Figure 2.6 describes the probability of either graduating with a CS degree, or leaving the CS degree program, across gender. These results are consistent with the persistence and yield analysis and indicate no critical time for female CS students compared to male students. All students leave or graduate in similar times, male students are just more likely to graduate, and less likely to leave at any particular time. The gap widens with time as female students become increasingly likely to leave CS if they do not graduate by their 8th term.

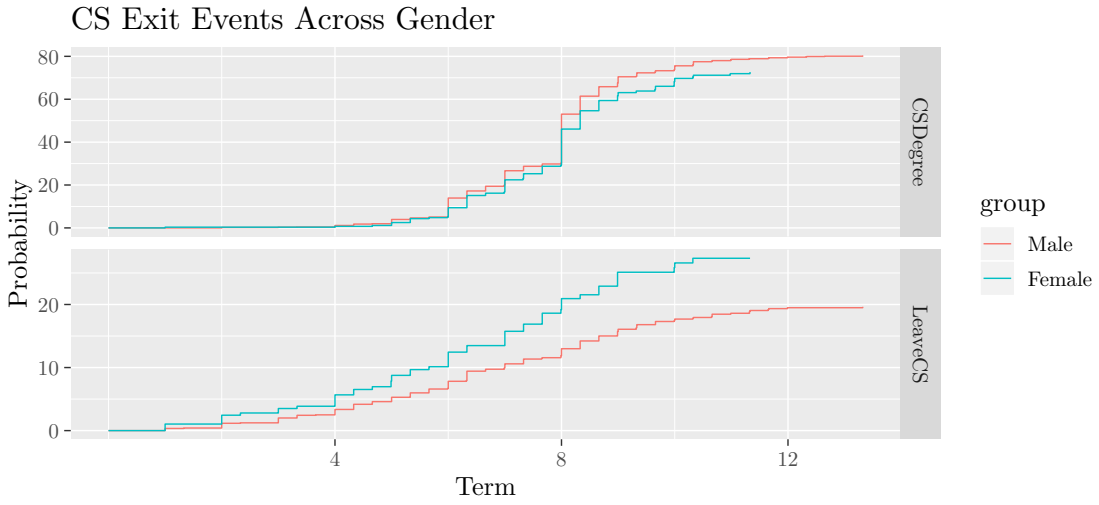

Figure 2.6: Probability of CS exit events across gender.

2.4.3 URM Exit Times

Figure 2.7 describes the results of competing risk analysis for CS exit events compared by URM status. While there are no striking differences between the probability to earn a CS degree for URM and non-URM students, URM students appear to maintain a lower probability of leaving CS before the 9th semester compared to non-URM students.

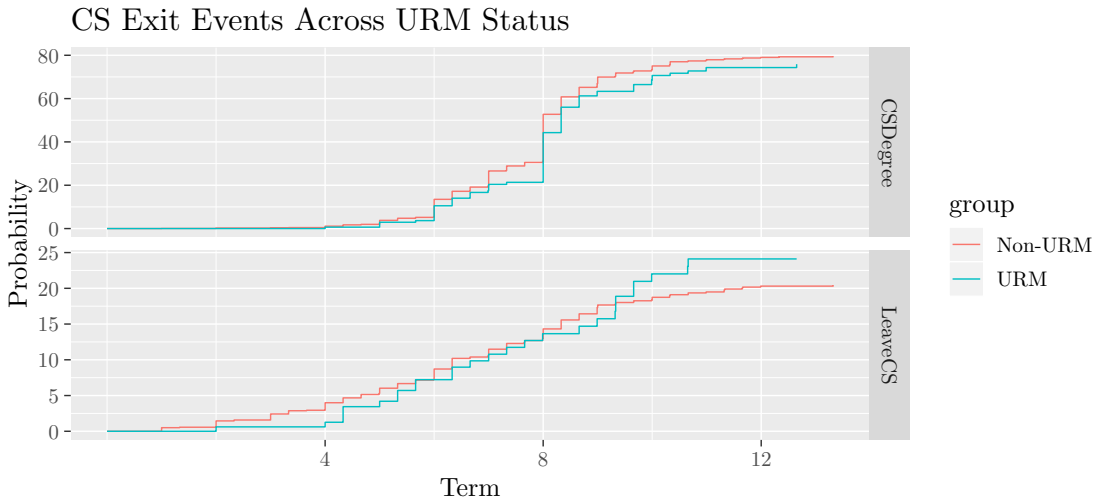

Figure 2.7: Probability of CS exit events across URM status

2.4.4 First Generation Exit Times

Compared to continuing-generation students, first-generation students are more likely to graduate with a CS degree. This result is consistent with the persistence and yield analysis above. Additionally, the competing risks analysis described in Figure 2.8 show that first-generation CS students are more likely to graduate sooner than continuing-generation students, with notably higher probability of graduating before the 8th semester. This early graduation may be due to disproportionately more first-generation students entering CS as transfer students, thus they come in with more credits towards graduation in their 1st term compared to students who start at LRI immediately after high school. After the 8th semester, the relative probables of graduation between first-generation and continuing-generation remains fairly constant.

Given the graduation rates, it is unsurprising that overall, first-generation CS students are less likely to leave the program without a degree, however the competing risks analysis adds nuance to this understanding as there appears to be relatively large increases in the probability of first-generation students leaving CS between terms 8 and 10 compared to continuing-generation students. Though this is not enough to reverse the general pattern, it does suggest pressures to leave are disproportional affecting first-generation students who continue past 8 semesters without a degree.

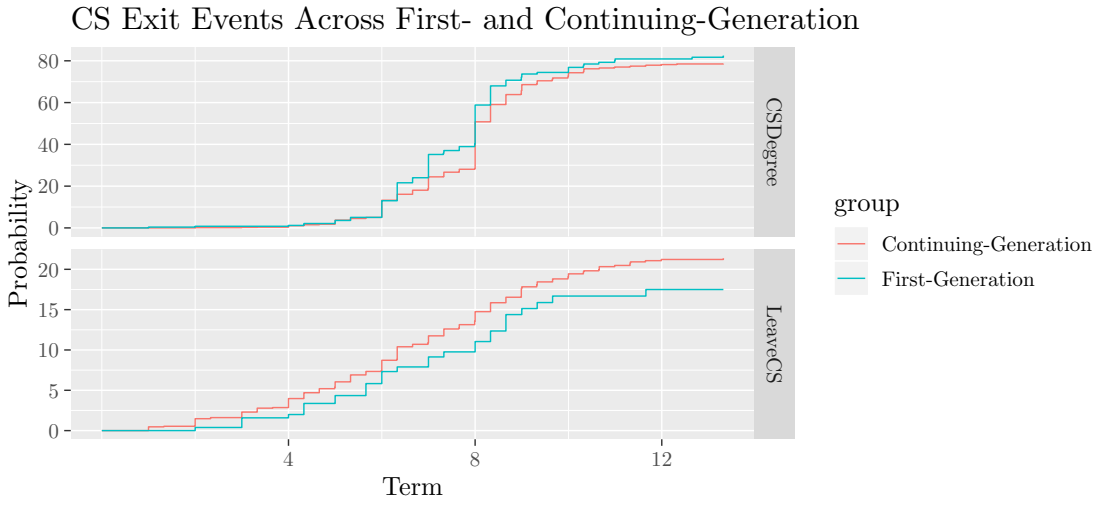

Figure 2.8: Probability of CS exit events across continuing-generation and first-generation students.

2.5 Discussion and Implications

Using the pipeline metaphor as a theoretical framework prompts questions related to critical points along the pipeline at which students may be matriculating into CS, or leaving the field.

2.5.1 Representation of Women Decreases with Time

The relative drop in representation of women declaring CS as a major compared to those declaring an interest in it suggests that the first two semesters in the college of engineering, before students have declared a major, is when disproportionately more women than men are turning away from CS. This suggests that efforts to make CS a more accessible choice during the first-year could help broaden participation.

Given the reverse trend for first generation and URM students, future work is needed to explore what perceptions or experiences are attracting these populations to CS during the first year to better understand how the experience of women may be improved. Some possible

factors may be advising practices, or perception of information sessions related to CS.

The continued drop in representation of women from those majoring to those earning a degree in 6 years suggests women are more likely than men to leave CS once matriculated. Compared with other majors within the college, female CS students are less likely to persist for 8-semesters than female students in other majors. This is consistent with results of other studies that indicate female students tend to leave CS at higher rates than their male peers (Cohoon 2001; Main and Schimpf 2017). It is important to note that CS is not the only engineering major that demonstrates this pattern, but the persistence gap between men and women in CS appears large compared to other majors. The higher yield rate of female CS students indicates that those that do persist in the college for 8 semesters are more likely to earn a degree within 6 years. This suggests that to make adjustments to improve female graduation rates we should first understand barriers that may lead female CS students to leave the college before 8 semesters. The results of the event history analysis indicate there is no clear time at which female students are leaving, the migration away from CS happens gradually over the course of enrollment. This pattern is consistent with other reports that students leave CS due to a "chilly climate" (Christopher Strenta et al. 1994) or feeling out of place in the disciplinary culture (Lewis, Anderson, and Yasuhara 2016) rather than specific factors like challenging introductory coursework.

2.5.2 Relative success of URM and First-Generation students

One surprising finding in this analysis was that URM and First-Generation students in CS had higher graduation rates than their counterparts. This is the reverse of the pattern seen in engineering in general, and indeed in most other degree programs at LRI. It is relevant to note the gender distribution in both URM and first-generation populations, described in Tables A.2 and A.3 in Appendix A. The URM population in the college had a slightly

higher representation of women than the non-URM population, and the first-generation population a slightly lower representation compared to the continuing-generation population. The differences are not great enough to suggest that the pattern observed in these populations was due to gender.

Viewing the persistence and yield data in the context of the epoch data, we see that first-generation students who matriculate into CS have higher 8-semester persistence rates, slightly higher yield rates, and CS attracts a disproportionate number of first-generation students after they have arrived. The pattern of attracting more students into the major after arriving also appears to hold for URM students, but to a lesser degree—percentages of CS degrees going to URM students are still lower than the percentage of URM students in the college. One possible explanation for this phenomenon is that both first-generation and URM students may have less awareness of the different engineering majors, including computer science, before arriving at college. Knowledge of skills and attributes associated with a major is likely a pre-requisite to choosing a major (Beggs, Bantham, and Taylor 2008). It's possible that URM and first-generation have less access to information about CS prior to arriving at college, but upon arriving they learn something about the major that particularly appeals to them.

The persistence and graduation rates are consistent with the EHA results that show URM and first generation students, once matriculating to CS, generally have a lower probability of leaving the than their peers. In the case of URM students, this relationship holds only to the 9th semester, but first generation students maintain a lower probability of leaving over the whole time period.

While these patterns are interesting, the data do not explain why they occur, and in particular if they are a result of the context at LRI, or the Computer Science discipline itself. Performing the same analysis on a multi-institutional dataset would allow us to answer whether

these patterns exist across institutions, or are specific to LRI. Future work will explore the possibility of applying similar analysis to MIDFIELD data, though a direct comparison may not be possible as those data do not contain first-generation status.

Additionally, there is of course variability within both URM and first-generation populations that are not visible in this analysis. For example, Lohfink and Paulsen (2005) found that first-generation students from higher income families were more likely to persist from their first to second year at a 4-year institutions than first-generation students from lower income families. Thus, we must accept the possibility of confounding factors, for example, perhaps first-generation students who gravitate towards computer science tend to come from higher-income families than those who choose other majors.

If the pattern exists across institutions. If it turns out this pattern appears across multiple institutions, we could conclude that something about CS as a major makes it more appealing to first-generation students, and more likely for first-generation students to persist. One possible explanation is that with recent coverage of success and subsequent perception of high number of well-paying job opportunities at software-based companies, for example Facebook and Google, there is a strong association between a CS degree and secure financial future. In some cases, programming work can easily be done remotely, granting individuals access to high-paying jobs without requiring them to move far from home. If these favorable views of computer science were to explain the increased interest in first-generation and URM populations, we might expect a higher percentage to arrive at LRI with an interest in majoring in CS. The fact that there is an increase in participation after the first year suggests that first-generation and URM students are having experiences in the FYE program that lead them to computer science at rates higher than continuing-generation and non-URM students. It would be interesting to explore this migration to CS in more detail: is the gain in first-generation and URM CS students due to initially undecided students choosing CS,

or changing from a different major of interest?

2.6 Limitations and Future Work

Existing literature on broadening participation indicates that community college pathways are of critical importance for supporting traditionally underrepresented students navigate into 4-year instiutitons (Lyon and Denner 2019). Our analysis excluded transfer students, thus, we miss out on a large portion of the populations we are interested in better understanding. To address this shortcoming, we are exploring alternatives to the 8-semester-persistence metric used in this analysis. One viable alternative would be to use completed credit hours as a proxy for persistence.

There were significant perturbations to the patterns reported here when including students who did not enter the college of engineering as first-year students. A sizable sub-population of students enrolled in the college began in other areas of the university and entered the college some time after their first year. It is possible that this population exhibits patterns different than those shown here. For example, due to the competitive nature of being accepted into some engineering majors, students who matriculate in at later points may have effectively experienced a more extensive pre-screening process before being admitted. This sub-population is also important for understanding the participation of non-dominant groups in computer science as studies indicate that women and other underrepresented groups often enter computer science through non-conventional means (Vitores and Gil-Juárez 2016). This is a case where the pipeline metaphor indeed inhibits understanding as it does not account for alternate pathways into the pipeline. Additional work is needed to explore variation in persistence and yield across different pathways into computer science.

As stated previously, there are likely variations within demographic groups that would enhance our understanding of the patterns seen here. For example, there may be significant

differences between the first-generation students that matriculate to Computer Science compared to those that matriculate to other majors. The data provide limited access to exploring some of these variations, though analysis would be limited by small sample size of dividing the first-generation population into subgroups.

2.6.1 Effects of a General Engineering Program

Thus far we have not found any literature that specifically address questions of computer science matriculation and retention at universities with common first-year engineering programs. Research on first-year engineering programs suggest that a common first year of general engineering increases retention rates in engineering, as well as reduces the number of students changing majors–and potentially delaying graduation–compared to direct-admit programs (Orr, Brawner, Lord, et al. 2012). In this context then, the pipeline metaphor is a particularly salient tool to identify weaknesses in the first-year experience. If we assume that a perfect first-year program would provided students with sufficient information they need to make an informed choice of major, we would expect very low rates of major changes and leaving engineering after the first year. Given the current results, the FYE program does appear to be a time disproportionately more women than men are turning away from CS, and disproportionately more first-generation and URM students are turning towards the major. It is not clear whether this is due to experiences specific to the FYE program itself, such as coursework or advising, or independent of it. A direction for future work would be to better understand students experiences with coursework and other academic support services during the first year of general engineering that could influence major choice.

2.7 Conclusion

The findings related to gender are consistent with other studies that suggest students are forced out of the CS due to perceptions of a poor fit, which could also account for the rel-

atively low percentage of these students declaring an interest in CS to begin with. This suggests efforts to broaden participation among women at LRI should be focused on adjusting the culture of Computer Science to be more inclusive: demonstrating to students that computer science is a broad field, rich in opportunities for collaboration, and can have a positive social impact (Margolis and Fisher 2002a).

More work to understand why URM and first generation students demonstrate relative success in Computer Science compared to other COE majors may provide additional clues to improve the experience of women as well. A multi-institution analysis is needed to determine if these patterns are general to computer science, or specific to LRI.

References

Abbate, Janet (2012). *Recoding Gender: Women's Changing Participation in Computing*. MIT Press (cit. on p. 16).

Beggs, Jeri Mullins, John H. Bantham, and Steven Taylor (2008). "Distinguishing the Factors Influencing College Students' Choice of Major." In: *College Student Journal* 42.2, pp. 381–395 (cit. on p. 36).

Beyer, Sylvia and Susan Haller (2006). "Gender Differences and Intragender Differences in Computer Science Students: Are Female Cs Majors More Similar to Male Cs Majors or Female Nonmajors?" In: *Journal of Women and Minorities in Science and Engineering* 12.4. DOI: 10.1615/JWomenMinorScienEng.v12.i4.50 (cit. on p. 18).

Chen, Xianglei and C. Dennis Carroll (2005). *First-Generation Students in Postsecondary Education: A Look at Their College Transcripts. Postsecondary Education Descriptive Analysis Report. NCES 2005-171*. ED Pubs, P (cit. on p. 18).

Chen, Xingyu, Catherine E. Brawner, et al. (2013). "A Taxonomy of Engineering Matriculation Practices." In: 2013 ASEE Annual Conference & Exposition, pp. 23.120.1–23.120.13 (cit. on p. 22).

Christopher Strenta, A. et al. (1994). "Choosing and Leaving Science in Highly Selective Institutions." In: *Research in Higher Education* 35.5, pp. 513–547. DOI: 10.1007/BF02497086 (cit. on pp. 17, 35).

Cohoon, J. McGrath (2001). "Toward Improving Female Retention in the Computer Science Major." In: *Communications of the ACM* 44.5, pp. 108–114 (cit. on pp. 17, 35).

Cohoon, Joanne and William Aspray (2008). *A Critical Review of the Research on Women's Participation in Postsecondary Computing Education*. MIT Press (cit. on p. 17).

CRA (2015). *CRA Taulbee Survey*. URL: https://cra.org/resources/taulbee-survey/ (visited on 2018) (cit. on p. 20).

Engle, Jennifer and Vincent Tinto (2008). *Moving Beyond Access: College Success for Low-Income, First-Generation Students*. Pell Institute for the Study of Opportunity in Higher Education (cit. on p. 29).

Ensmenger, Nathan (2010). "Making Programming Masculine." In: *Gender Codes*. Red. by Professor Thomas J. Misa. Wiley, pp. 115–141 (cit. on p. 16).

Fealing, Kaye Husbands, Yufeng Lai, and Samuel L. Myers Jr (2015). "Pathways vs. Pipelines to Broadening Participation in the STEM Workforce." In: *Journal of Women and Minorities in Science and Engineering* 21.4 (cit. on pp. 14, 15).

Garriott, Patton O., Rachel L. Navarro, and Lisa Y. Flores (2017). "First-Generation College Students' Persistence Intentions in Engineering Majors." In: *Journal of Career Assessment* 25.1, pp. 93–106. DOI: 10.1177/1069072716657533 (cit. on p. 19).

Henn, Steve (2014). *When Women Stopped Coding*. URL: http://www.npr.org/sections/money/2014/10/21/357629765/when-women-stopped-coding (visited on 2015) (cit. on p. 13).

Larsen, Elizabeth A. and Margaret L. Stubbs (2005). "INCREASING DIVERSITY IN COMPUTER SCIENCE: ACKNOWLEDGING, YET MOVING BEYOND, GENDER." In: *Journal of Women and Minorities in Science and Engineering* 11.2. DOI: 10.1615/JWomenMinorScienEng.v11.i2.20 (cit. on pp. 13, 17, 18).

Lee, Walter C. (2019). "Pipelines, Pathways, and Ecosystems: An Argument for Participation Paradigms." In: *Journal of Engineering Education* 108.1, pp. 8–12. DOI: 10.1002/jee.20241 (cit. on pp. 14, 15).

Lewis, Colleen M., Ruth E. Anderson, and Ken Yasuhara (2016). ""I Don'T Code All Day": Fitting in Computer Science When the Stereotypes Don'T Fit." In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ICER '16. New York, NY, USA: ACM, pp. 23–32. DOI: 10.1145/2960310.2960332 (cit. on pp. 18, 35).

Lichtenstein, Gary et al. (2014). "Retention and Persistence of Women and Minorities along the Engineering Pathway in the United States." In: *Cambridge handbook of engineering education research*, pp. 311–334 (cit. on p. 17).

Loeb, Susanna et al. (2017). "Descriptive Analysis in Education: A Guide for Researchers. NCEE 2017-4023." In: *National Center for Education Evaluation and Regional Assistance* (cit. on p. 20).

Lohfink, Mandy Martin and Michael B. Paulsen (2005). "Comparing the Determinants of Persistence for First-Generation and Continuing-Generation Students." In: *Journal of College Student Development* 46.4, pp. 409–428. DOI: 10.1353/csd.2005.0040 (cit. on p. 37).

Lord, Susan M. et al. (2019). "Beyond Pipeline and Pathways: Ecosystem Metrics." In: *Journal of Engineering Education* 108.1, pp. 32–56. DOI: 10.1002/jee.20250 (cit. on pp. 14–16).

Lyon, Louise Ann and Jill Denner (2019). "Chutes and Ladders: Institutional Setbacks on the Computer Science Community College Transfer Pathway." In: *ACM Trans. Comput. Educ.* 19.3, 25:1–25:16. DOI: 10.1145/3294009 (cit. on p. 38).

Ma, Yingyi (2009). "Family Socioeconomic Status, Parental Involvement, and College Major Choices—Gender, Race/Ethnic, and Nativity Patterns." In: *Sociological Perspectives* 52.2, pp. 211–234. DOI: 10.1525/sop.2009.52.2.211 (cit. on p. 19).

Main, J. B. and C. Schimpf (2017). "The Underrepresentation of Women in Computing Fields: A Synthesis of Literature Using a Life Course Perspective." In: *IEEE Transactions on Education* 60.4, pp. 296–304. DOI: 10.1109/TE.2017.2704060 (cit. on pp. 18, 26, 35).

Margolis, Jane and Allan Fisher (2002a). "Computing With a Purpose." In: *Unlocking the Clubhouse: Women in Computing*. MIT press, pp. 49–60 (cit. on p. 40).

— (2002b). *Unlocking the Clubhouse: Women in Computing*. MIT press (cit. on p. 17).

Miller, David (2015). *A Metaphor to Retire*. URL: https://www.insidehighered.com/views/2015/03/03/essay-calls-ending-leaky-pipeline-metaphor-when-discussing-women-science (visited on 2019) (cit. on p. 14).

Min, Youngkyoung et al. (2011). "Nonparametric Survival Analysis of the Loss Rate of Undergraduate Engineering Students." In: *Journal of Engineering Education* 100.2, pp. 349–373. DOI: 10.1002/j.2168-9830.2011.tb00017.x (cit. on p. 30).

National Academies of Sciences, Engineering (2017). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. DOI: 10.17226/24926 (cit. on p. 21).

NCSE (2018). *Women, Minorities, and Persons with Disabilities in Science and Engineering*. National Center for Science and Engineering Statistics, NSF (cit. on pp. 15, 26).

NSF (2019). *Women, Minorities, and Persons with Disabilities in Science and Engineering: 2019 | NSF - National Science Foundation*. URL: https://ncses.nsf.gov/pubs/nsf19304/digest/field-of-degree-women#computer-sciences (visited on 2019) (cit. on p. 13).

Ohland, Matthew W. et al. (2011). "Race, Gender, and Measures of Success in Engineering Education." In: *Journal of Engineering Education* 100.2, pp. 225–252 (cit. on pp. 26, 29).

Orr, M. K., C. E. Brawner, S. M. Lord, et al. (2012). "Engineering Matriculation Paths: Outcomes of Direct Matriculation, First-Year Engineering, and Post-General Education Models." In: *2012 Frontiers in Education Conference Proceedings*. 2012 Frontiers in Education Conference Proceedings, pp. 1–5. DOI: 10.1109/FIE.2012.6462357 (cit. on p. 39).

Orr, Marisa K., Catherine E. Brawner, Matthew W. Ohland, et al. (2013). "The Effect of Required Introduction to Engineering Courses on Retention and Major Selection." In: 2013 ASEE Annual Conference & Exposition, pp. 23.1192.1–23.1192.9 (cit. on p. 13).

Peckham, Joan et al. (2007). "Broadening Participation in Computing: Issues and Challenges." In: *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. ITiCSE '07. New York, NY, USA: ACM, pp. 9–13. DOI: 10.1145/1268784.1268790 (cit. on p. 19).

Plank, Stephen B., Stefanie DeLuca, and Angela Estacion (2008). "High School Dropout and the Role of Career and Technical Education: A Survival Analysis of Surviving High School." In: *Sociology of Education* 81.4, pp. 345–370. DOI: 10.1177/003804070808100402 (cit. on p. 30).

Scott, Marc A. and Benjamin B. Kennedy (2005). "Pitfalls in Pathways: Some Perspectives on Competing Risks Event History Analysis in Education Research." In: *Journal of Educational and Behavioral Statistics* 30.4, pp. 413–442 (cit. on p. 31).

Seymour, Elaine (1995). "The Loss of Women from Science, Mathematics, and Engineering Undergraduate Majors: An Explanatory Account." In: *Science Education* 79.4, pp. 437–473. DOI: 10.1002/sce.3730790406 (cit. on p. 17).

Su, Lester K. (2010). "Quantification of Diversity in Engineering Higher Education in the United States." In: *Journal of Women and Minorities in Science and Engineering* 16.2 (cit. on p. 26).

The Social Network (2010). biography, drama.

Trenor, Julie Martin (2009). "A Phenomenological Inquiry of the Major Choice Processes of an Overlooked Demographic: First Generation College Students in Engineering." In: *Proceedings of the 2009 Research in Engineering Education Symposium* (cit. on p. 19).

Vitores, Anna and Adriana Gil-Juárez (2016). "The Trouble with 'Women in Computing': A Critical Examination of the Deployment of Research on the Gender Gap in Computer Science." In: *Journal of Gender Studies* 25.6, pp. 666–680. DOI: 10.1080/09589236.2015.1087309 (cit. on pp. 16, 38).

Yang, Fan (2017). *A Competing Risks Survival Analysis of High School Dropout and Graduation: A Two-Stage Model Specification Approach*. ProQuest LLC (cit. on pp. 30, 31).

Chapter 3: Manuscript 2

Migration Pathways into Computer Science

Target Journal: Computer Science Education

Abstract

In the context of the current enrollment boom in Computer Science majors, many universities are struggling to keep up with demand. While increased enrollment puts a strain on finite institutional resources, it also has the potential to reinforce barriers–such as assumptions of prior programming experience–that prevent traditionally underrepresented populations, women and underrepresented minorities, from gaining entry into the field. Introductory programming courses serve as gatekeeper courses that can act as a filter to those who persist in computer science. In this study, we investigate how performance in the introductory programming course sequence at a large public research institution correlates with student demographics and CS matriculation and graduation rates.

3.1 Introduction

For the past few years, colleges in the United States have been experiencing an enrollment boom in computer science. Many colleges and universities are struggling to keep up with the demand in computer science (NAE 2018). At universities where Computer Science is housed in the college of engineering, this has prompted debates to split the degree program into its own school to help attract more faculty and consolidate computing courses (Basken 2018). During this same time there has been a continued push to broaden participation in computing, among other STEM fields (AIR 2012).

Historically, enrollment booms in computer science, and the strains they place on institu-

tional resources, have resulted in action by universities and departments to be more selective of who they admit (Roberts 2016). Strategies to address high demand may also impact curricular decisions in ways that increase barriers to certain populations. For example, in response to the enrollment boom in the 1980's, which coincided with the rise of personal computers in the home, computer science programs adjusted the curriculum in their introductory programming courses to assume prior programming experience. Since personal computers were marketed primarily to young men (Margolis and Fisher 2002a), this change effectively limited women's access to computer science programs, resulting in a downward decline in their participation in the field (Margolis and Fisher 2002b).

In situations of enrollment that has the potential to overextend resources, introductory courses can play a significant role in controlling who is admitted to a major. These courses, referred to as gatekeeper, *barrier courses*, or weed-out courses, occur early in a curriculum and often have a high failure rate (Suresh 2006). The experience students have in introductory programming courses in particular have been shown to influence decisions to major in computer science (Lewis, Yasuhara, and Anderson 2011), and these courses have been viewed as barrier courses to computer science (Bennedsen and Caspersen 2007).

In this study I investigate the possibility that introductory programming courses, CS1 and CS2, that are early prerequisites for computer science, act as barrier courses to the major at a large public research institution in the United States with a common first-year engineering (FYE) program. To explore the possibility of CS1 and CS2 acting like barrier courses I address the following research questions:

1. What associations exist between gender, URM, and first-generation status and performance in CS1 and CS2?

2. How does performance in CS1 affect matriculation rates into CS?

3. How does performance in CS1 and CS2 affect graduation rates and time-to-degree in CS?

3.2 Literature Review

Much work in understanding attrition in CS focuses on the impact of so-called CS1 and CS2, the first and second introductory programming courses offered by many departments. Studies have shown that affective factors such as student interests, values, and experiences, are correlated with grades in CS1 (McKinney and Denton 2004) and that CS1 is a formidable time for prospective CS students to construct beliefs about their computing self-efficacy (Kinnunen and Simon 2011) and ability to complete a CS degree (Lewis, Yasuhara, and Anderson 2011). An important finding by Lewis, Yasuhara, and Anderson (2011) is that students form their understanding of their own abilities in large part by comparing their perceived performance to that of their peers in CS1. Thus, if students with little pre-college programming experience observe that they are expending far more effort completing assignments than their classmates they may conclude that they are at a disadvantage and choose not to major in CS.

Given the impact CS1 can have on matriculation into CS, efforts to broaden participation in Computer Science often focus on this course. For example, Cohoon and Tychonievich (2011) describe a section of CS1 designed specifically for inexperienced programmers. They attribute an increased departmental enrollment of both women and racial minorities to the success of their alternative CS1. In a similar fashion, but with different motivations, to meet a university requirement that all incoming students take an introductory programming course, Georgia Tech created an alternative to the CS1 for Non-CS majors. They found that by framing the course around media computation, some non-majors were encouraged to continue to engage with CS topics and skills beyond the required service course (Guzdial

and Forte 2005).

Introductory programming courses have also been viewed as a critical site for forming beliefs about the field of computer science that govern students' sense of belonging in the field (Benbow and Vivyan 2016). For example, Sax et al. (2018) found that women experienced a small decline in sense of belonging during an introductory computing course, but that underrepresented minority (URM) students actually experience a higher sense of belonging than their peers. They conclude however that department-level experiences were a more significant predictor of sense of belonging than what students experienced in the classroom.

3.3 Theoretical Framework

To understand how experiences in barrier courses, such as CS1, can affect persistence and success in computer science, I adapted a model, shown in Figure 3.1, originally developed by Suresh (2006). The original model links performance in barrier courses to persistence in engineering and identifies several factors which impact performance in barrier courses. The introductory programming courses CS1 and CS2 have qualities that suggest they may act as barrier courses to CS: they are early requirements for the major and studies of these courses at other institutions suggests that in some cases they exhibit a high failure rate (Bennedsen and Caspersen 2007).

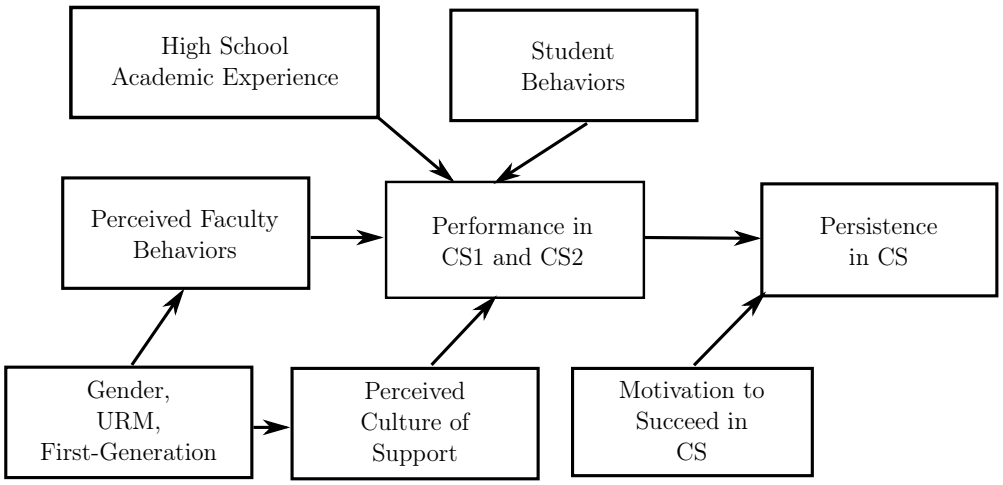

Figure 3.1: Model of variables that may impact performance in introductory programming courses CS1 and CS2 and persistence in Computer Science. Derived from barrier course model by Suresh (2006).

Demographic variables such as gender, URM, and first-generation status are not explicitly part of this model, however the social constructs "perceived culture of support" and "perceived faculty behavior" may have variation across different demographic groups due to sense of belonging in a particular social environment. For example, there may be differences in how women experience the perceived culture of support compared to men (Sax et al. 2018). Therefor, I extend Suresh's model to include demographic variables as underlying factors.

3.4 Methods

3.4.1 Overview

To explore relationships between CS1 performance and CS matriculation and graduation, population-level data for engineering students at Large Research Institution (LRI) were obtained as part of a larger study supported by NSF No. DUE-1712089 related to gatekeeper courses across engineering.

3.4.2 Site Description

LRI is a large public research institution serving between 20,000 to 30,000 undergraduate students. Engineering is a strong program at LRI, contributing over 25% of all degrees earned at the institution. At LRI, Computer Science is housed within the College of Engineering (COE). To understand how LRI may be representative of other CS degree granting institutions we can compare with results of the Taulbee survey sent by the Computing Research Association each year to all PhD-granting departments of Computer Science, Computer Engineering, and Information in North America (CRA 2015). The gender distribution in CS at LRI is consistent with that across Taulbee institutions, while the proportion of URM students enrolled in CS at LRI is lower than Taulbee institutions, 8% compared to 18% (Chapter 2).

Like other institutions across the country (National Academies of Sciences 2017), Large Research Institution has experienced a large increase in the number of students enrolling as CS majors since 2012. This corresponds with increased national pressure to grow the computer science workforce and rising awareness of the field through popular culture (National Academies of Sciences 2017).

The nature of the first-year engineering program at LRI presents a somewhat different environment for studying CS migration and attrition compared to schools without a common first year, or where CS is housed outside of the college of engineering. The introductory programming course, CS1, is generally taken during a student's second term in the college, before students are expected to declared a major, while the second computer science course CS2 is taken the third term, after most students have declared a major. This timing is at odds with the intent of the FYE program to provide students time to make an informed decision on major choice since it assumes an intent to major in CS, and thus enrolling in

CS1, prior to the completion of the first year. It is possible that incoming students who are not already considering computer science as a major, but become interested in it late in the FYE program, may perceive the timing of CS1 as a barrier to entry. While the first-year advisors are adept at helping students who discover CS in their second semester arrange their courses to meet the entry requirements and still graduate in 4-years, many students may not know to seek out their advisors in this case.

Additionally, enrollment management practices govern which majors students are admitted to. Students who earn a GPA of 3.0 or higher during their first year are guaranteed their first choice of engineering major, while those who earn lower than a 3.0 are placed based on available seats. At LRI, CS is one of the more competitive majors, and so students who select it as a first choice but earn a GPA lower than 3.0 may be placed in their second choice instead.

3.4.3 Population Description

Using institutional records we obtained population data for students enrolled at Large Research Institution from Fall 2009 through Spring 2018. These data include majors enrolled in, courses taken, and degrees obtained, along with demographics such as gender, first-generation status, underrepresented minority (URM) status, and tuition status (in-state or out-of-state). In addition, we have course records that include courses taken and grades received. These course records also includes transfer credit information, whether students transferred credit for a course from another institution, or from an AP exam.

Table 3.1 displays percentages of Female, First-Generation, and URM students in three different populations: those enrolled in the College of Engineering (COE), those who ever enroll in CS1, and those Enroll in CS as a major.

Students who earn a grade in CS1 may not necessarily intend to major in CS. In fact, CS1

Table 3.1: Population characteristics

demographic	COE (%) N ∼ 10k-20k	CS1 (%) N ∼ 3k-4k	Enrolled in CS (%) N ∼ 1k - 2k
Female	20.39	20.91	16.08
First Generation	15.17	15.72	15.69
URM	9.83	10.93	9.13

is a requirement for another computationally-focused major, CFM, and an optional elective for MATH. Table 3.2 shows CS1 enrollment by major during the same term. GE is General Engineering, and this is the most common combination as most students who intent to major in CS enroll in CS1 before they have declared the CS major. The major UND is a designation to indicate "Undecided" students who are not enrolled in the college of engineering. UND students could be enrolling in CS1 because they intend to major in CS and hope to apply to the college of engineering at a later date, or for different reasons. The pass rate, defined as earning a C or better since this is the requirement to count CS1 towards a degree, varies by major. Unsurprisingly, students who have already declared their major as CS at the time they take CS1 have the highest pass rate, while UND and CFM students have the lowest.

Table 3.2: CS1 pass rate (C or better) by major

Major	N	% Pass
GE	1679	77.31
UND	352	56.53
CFM	301	56.81
MATH	243	62.14
ME	213	80.75
CS	152	92.76

SAT Scores. Previous studies suggest high school performance, as measured by SAT scores, will be associated with performance in introductory college courses (Suresh 2006). Figure 3.2 visualizes average SAT Math and Verbal scores by demographic.

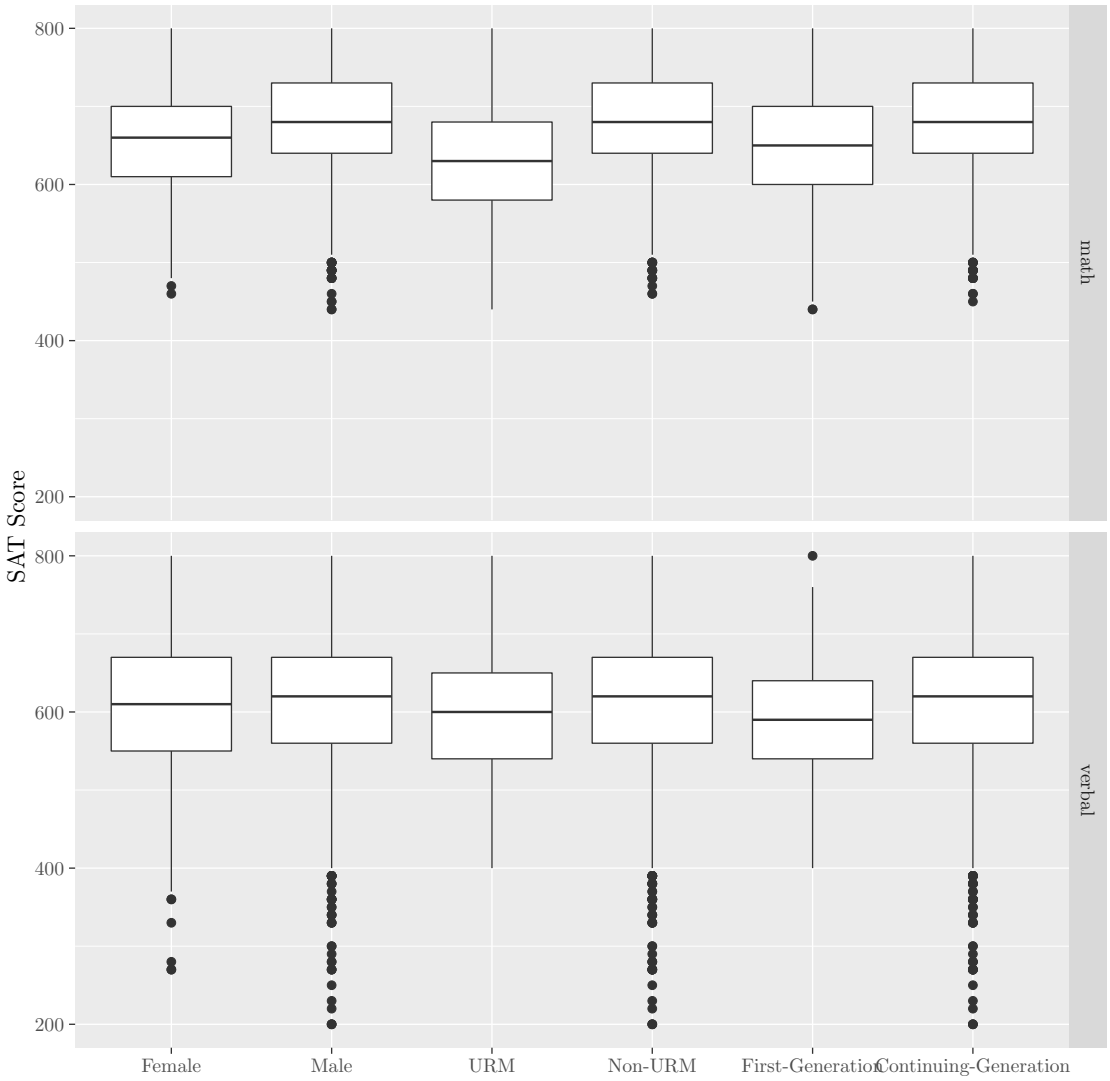

Figure 3.2: Average SAT Math and Verbal scores for all students taking CS1. Lower and upper edges of each box represent 25% and 75% of the scores. Horizontal bars represent the mean score, and dots indicate outliers.

Female, URM, and first-generation students have lower average Math SAT scores than their

peers.

Figure 3.3 compares the pathways underserved CS degree earners take to get to the field, compared to their peers.

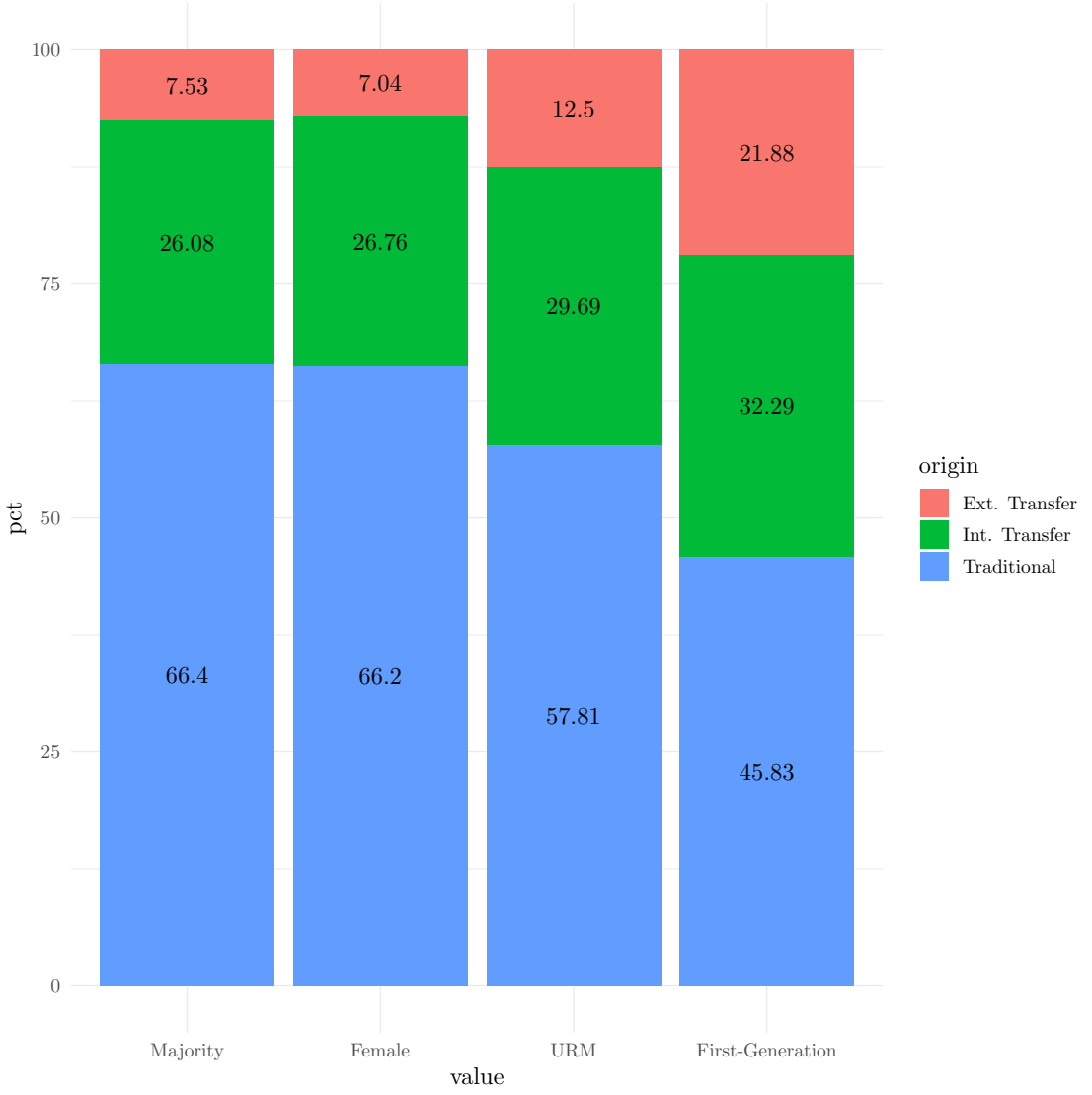

Figure 3.3: Proportions of represented and underrepresented populations arriving at CS from different pathways. Majority students are male, Non-URM, *and* continuing-generation. The External Transfer pathway contains students transferring from community colleges or other 4-year institutions, while the Internal Transfer pathway contains students already at LRI transferring in from outside the College of Engineering.

As compared to the majority population entering CS, more URM and First-Generation students are entering the major from external and internal transfers. The largest difference is seen in the first generation population in which over 50% of those entering CS are starting elsewhere. The difference in incoming pathways for URM and first-generation students in particular will affect how we interpret results for these populations. For example, because CS1 is not a requirement exclusive to CS, I define intent to major in CS by being enrolled in CS1 while a general engineering (GE) major. This definition will unfortunately disproportionately exclude URM and first-generation students from that part of the analysis.

3.4.4 Data Preparation

This study focuses on the performance in CS1 and CS2 as both outcome variables (RQ1) and as predictor variables (RQ2-3). I characterize performance as a combination of number of attempts at a course, as well as the final grade earned. To simplify analysis and reporting, I adopt a similar grouping and naming scheme to Suresh (2006) and define four performance groups:

Sailers are students who do not repeat a course and earn a grade in the A-B range

Plodders are students who earn a grade in the C range with only one attempt, or students who repeat the course and earn a grade in the A-B range on their last attempt

Struggler are students who earn a grade in the D-F range on their first attempt and don't retake the course, or students who repeat the course at least once and earn a grade in the C range on their last attempt. Course withdrawls as also included in this grouping.

Alternative students are those who do not enroll in CS1 or CS2 but meet the requirement through other means. This is much more common for CS1 and is often due to transferring in course credit from another university, using CS AP credit to meet the

requirement, or by taking an alternative course at LRI that counts towards the CS1 requirement.

To support the analysis of matriculation, persistence, and graduation, I computed 'event' and 'time to event' variables for each student and the following events:

- matriculating into CS – a student begins a CS program. They may declare CS as a major after some time as a general engineering, change majors into CS from outside the college of engineering, or transfer from a different school directly into the CS program.
- leaving CS – a student who has been enrolled in CS leaves the program without a degree. This includes changing majors to another engineering major, to a major outside engineering, or leaving the institution. The time of this event is the first term a previous CS student is no longer recorded as a CS major, and they did not earn a CS degree.
- earning a CS degree – a student earns a CS degree within 6 years of their first enrolled term.

In addition to these events, I use a designation of "In progress" to indicate students who leave the data set, but have not experienced an event of interest. A large number of these cases are students with enrollment terms less than 6 years before the end of the data collection period, so we would not have expected them to have graduated yet.

3.4.5 Analysis

Multinomial Logistic Regression (RQ1). To explore relationships between prior academic performance and performance in CS1 I ran a multinomial logistic regression. This analysis determines the likelihood of several mutually exclusive outcomes based on a set of control

and predictor variables. In my analysis, high performing CS1 students (Sailers) serve as the outcome reference group.

Logistic Regression (RQ2). To relate performance in CS1 to likelihood of matriculation into CS I ran a logistic regression analysis. This analysis is a simpler version of generalized multinomial logistic regression except instead of several possible outcomes logistic regression determines the likelihood of a single binary outcome: whether or not a CS1 student matriculates into CS.

Event History Analysis (RQ3). Finally, to address RQ3 and investigate how CS2 performance may affect time-to-graduate, I used event history analysis (EHA). EHA has the capability of including all students in the analysis, even those who's enrollment start date is such that we would not expect them to have experienced an event such as leaving CS or graduating CS yet. In the analysis, these students are considered "censored", and correspond to those labeled as "In Progress".

Another important concept in EHA is that of the *risk* set. People are considered *at risk* or in the risk set if they are still in the sample pool but have not yet experienced an event of interest by a given time. A key quantity in EHA is the hazard function $h[k,t]$ which is the probability that event k occurs at time period t, given that no other event of interest has occurred prior to time period t. The maximum likelihood estimator for this quantity is

$$\hat{h}[k,t] = \frac{\text{number of subjects who experience event } k \text{ during period } t}{\text{number of subjects at risk during period } t} \tag{3.1}$$

.

We often also are interested in the probability that an event occurs by a given time, rather than at a given time (e.g., the probability of earning a degree within 8 terms). This value

is obtained from the cumulative incident function, which is defined in terms of the hazard function to be the accumulated probability of and event occurring by a particular time period.

Modeling time can be done in several ways, choosing an appropriate method depends on the research question and the nature of the data themselves (Singer and Willett 2003). I count Fall and Spring terms as 1 unit of time each, and inter-session terms as .33 units of time. At any given term a student's enrolled time is the cumulative sum of the time units from their first term to the current. The implication of this is that students who enroll in Summer or Winter terms will accumulate enrolled time faster than their peers that only enroll in Fall and Spring terms.

3.5 Results

3.5.1 Performance in CS1 (RQ1)

Studies have shown that performance in an introductory programming course can have a significant impact on students' self-perception of their ability to be successful in Computer Science. At LRI, the CS1 course is typically taken during a student's second semester in the college, still prior to declaring an engineering major and so it serves as an important experience that likely informs a student's decision to persist in computer science or not.

A Chi-squared (χ^2) test of proportions revealed differences in CS1 performance across different demographic groups. There was significant effect between a student's gender and their CS1 performance group ($\chi^2(3) = 39.06, p < .001$) as well as their URM status and performance in CS1 ($\chi^2(3) = 40.67, p < .01$).

This supports the results of the multinomial logistic regression displayed in Table 3.3. Two models were fit with multinomial logistic regression: Model 1 contains just the control vari-

ables of gender, URM, and first-generation status as predictors while Model 2 contains the control variables as well as High School performance predictors.

		Model 1		Model 2	
	Variable	Plodder	Struggler	Plodder	Struggler
Control Variables	Female	1.40*	1.77*	1.32*	1.56*
	URM	1.46*	1.84*	1.27	1.34*
	First-Generation	1.18	1.32*	1.03	0.96
HS Performance	Math SAT[a]			0.90*	0.79*
	Verbal SAT[a]			0.94*	0.93*
	CS AP			0.79	0.31*
Model Fit	McFadden Adj. R^2	0.096		0.11	
	BIC	7292		7223	

Table 3.3: Multinomial Logistic Regression Results for Performance in CS1.
[a] SAT scores are in units of 30

Odds ratios for each demographic group indicates the ratio of the likelihood of that group being a member of the performance group to the reference demographic. Reference demographics are Male, Non-URM, and continuing-generation, respectively. Odds ratios greater than 1 indicate an increased likelihood of being in either the Plodder or Struggler group rather than the Sailers group, while an odds ratio less than 1 indicates a decreased likelihood of being in either the Plodder or Struggler groups relative to Sailers.

Model 2 indicates that gender, URM, SAT scores, and whether students took the CS AP exam have a significant effect on CS1 performance. As an example to interpret the results, Female students are 32% more likely to be Plodders than Sailers, and 56% more likely to be Strugglers than Sailers.

McFadden's Adjusted R^2 is a measure of model fit that includes a penalty for number of predictors. The increase in McFadden's Adjusted R^2 from Model 1 to Model 2 indicates Model 2 is a better fit for the data, as does the decrease in the BIC value (Dziak et al. 2019) from Model 1 to Model 2. Despite the improved model fit parameters for Model

2, values between .20 and .40 of McFadden's Adjusted R^2 are considered to indicate good model. Since the value for Model 2 is below this threshold, while we may still gain some insight by interpreting the results, there are likely other significant factors that influence CS1 performance that are not captured by either model.

Both Math and Verbal SAT scores were found to be significant predictors of performance in CS1, however a test for multicolinearity indicates that the two SAT scores do not have an independent affect on performance (Field, Miles, and Field 2012). Having CS AP credit also had a significant effect on performance, students with AP credit are 69% less likely (Odds ratio 0.31) to be Strugglers than Sailers.

The results of a χ^2 test of proportions indicate a significant relationship between taking the CS AP exam and gender ($\chi^2(1) = 4.97, p < .05$), URM ($\chi^2(1) = 4.22, p < 0.05$), and first-generation status ($\chi^2(1) = 22.76, p < 0.001$). Fewer women, URM students, and first-generation students came in with AP CS credit than expected. One possible explanation is a lack of access to AP CS for these groups, or lack of encouragement by educators to pursue AP CS. Another could be that LRI is not attracting women, URM, and first-generation students that are taking CS AP.

3.5.2 Matriculation Events (CS2)

Because CS1 is generally taken in a student's second semester, and before officially declaring a major, students in the first-year general engineering program who also enroll in CS1 are likely intending to major in CS. It is important to note that restricting this analysis to only those CS1 students who are general engineers excludes internal transfers, those students who change majors to computer science from outside the college of engineering, from the analysis. This will impact interpretation of results since disproportionately more URM and first-generation students are entering computer science through this alternative pathway. It

is also important to recall that enrollment management practices at LRI dictate that only students who earn a GPA of 3.0 during their first year are guaranteed their first choice of major. This means that a student who selects CS as their first choice of major, does well in CS1, but has an overall GPA lower than 3.0 may not matriculate to CS.

To understand how performance in CS1 affects likelihood to matriculate into CS, I conducted a logistic regression using matriculation into CS as the outcome variable, and CS1 performance groups as predictor variables. Results indicate a significant effect of CS1 performance on likelihood to matriculate.

Focusing on just those students who register for CS1, Figure 3.4 shows the results of a logistic regression testing the effects of CS1 grade and demographics on the probability of matriculating into CS within one year of the last attempt at CS1.

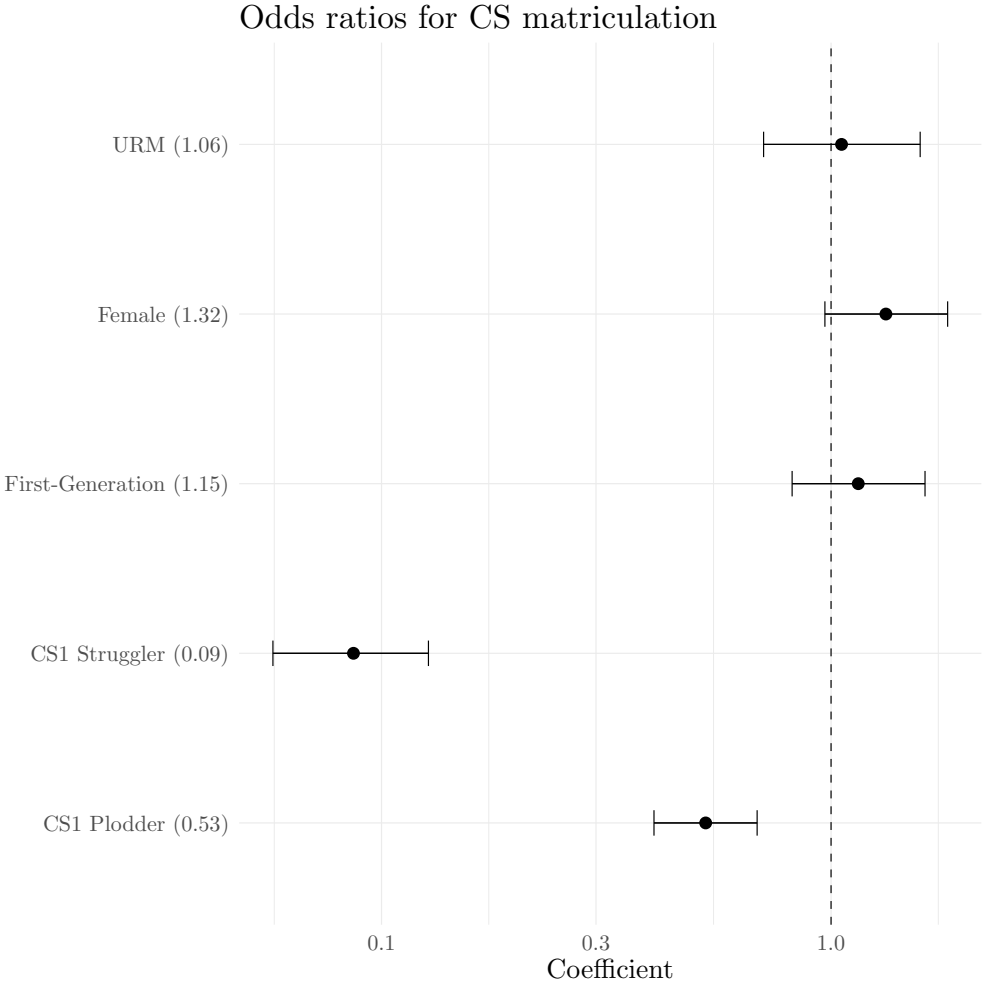

Figure 3.4: Matriculation of CS1 students into CS

The model containing CS1 performance as a predictor increased the model fit parameter (McFadden's Adjusted $R^2 = 0.10$) compared to a model with only demographic variables (McFadden's Adjusted $R^2 = 0.0031$). While the value for the full model fit falls below the .2 threshold used to define an acceptable fit, it is large enough to suggest these covariates do have meaningful influence on likelihood to matriculate into CS, but there are likely other important factors that this model does not capture, such as enrollment management practices described above.

Like the odds ratios that were the result of the multinomial logistic regression, values greater than one mean a higher probability of matriculation compared to the reference group. As with the previous models, the demographic reference groups are Male, Non-URM, and Continuing Generation. For the performance groups Struggler and Plodder, the reference group is Sailer.

The results show that Plodders are 47% less likely (odds ratio 0.53) to matriculate, and Stugglers are 91% less likely (odds ratio 0.09) to matriculate compared to Sailers. While none of the demographic variables are statistically significant, it is interesting to note that Female students are 32% more likely to matriculate than Male and first-generation 15% more likely to matriculate than continuing-generation. The effect of gender is unexpected, as it may appear contrary to the knowledge that Women are under-represented in computer science on average.

Sailer and Plodder Matriculation. Since we know that performance in CS1 is a strong indicator of matriculation into CS, we could conclude that CS1 Sailers that do not matriculate are making that choice based on information other than course performance. Figure 3.5 shows the percentage of Female, URM, and first-generation CS1 Sailer and Plodders matriculating into CS, compared to the percentage for the whole performance group.

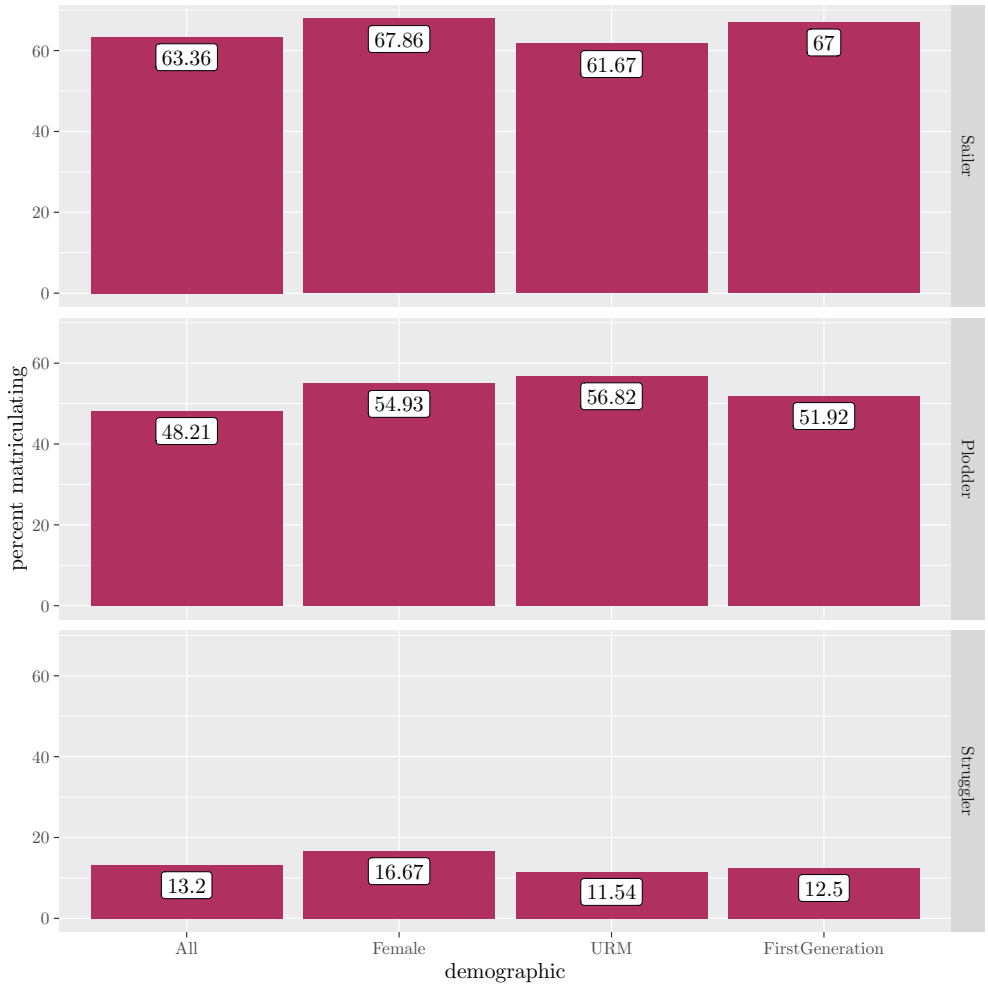

Figure 3.5: Percentage of students in each CS1 performance group who matriculation into CS

Looking first at the Sailer subgroup, 63% go on to matriculate to computer science. Given this baseline, we can see that disproportionately more Female and first-generation high performing CS1 students go on to matriculate in to CS, while slightly fewer high performing URM students matriculate.

In the Plodder subgroup, Female students, URM, and first-generation students all have higher matriculation rates than the group average. In the case of URM, this is compatible with a finding by Sax et al. (2018) that URM students across 15 large research universities

reported slightly higher measures of belonging than their peers, and so we might expect this to compensate for lower performance when deciding to matriculate into CS. The increase in representation of first-generation students matriculating into CS is consistent with a previous study (chapter 2) that found first-generation students were more likely than their peers to enter a CS program and eventually graduate with a CS degree.

3.5.3 Exit Events (RQ3)

Exit events are those that occur to students who enrolled in CS at some point. A student can either leaving by earning a CS degree, by changing majors, or by leaving the institution all together. Differences in timing of exit events across demographic groups could be an indicator that certain experiences are disproportionately affecting different groups of students. Understanding this could aid in efforts to increase persistence in groups that have been underrepresented in computer science, such as women and URM students. For this analysis, I include CS2 performance instead of CS1 since it is taken after CS1 and when students have generally already matriculated into CS.

Figure 3.6 depicts the probability of leaving CS or earning a degree by term, by CS2 performance groups. Plodders and Strugglers have a lower probability to graduate, and higher probability of leaving. While these results are not surprising, we can also observe a difference in the risk of leaving CS between Sailers and Plodders at any given term. Strugglers are much more likely to leave earlier than plodders and Sailers. This suggests they may be leaving as a direct result of their poor performace in CS2, while those who leave from the Plodder and Sailer groups may be leaving for different reasons.

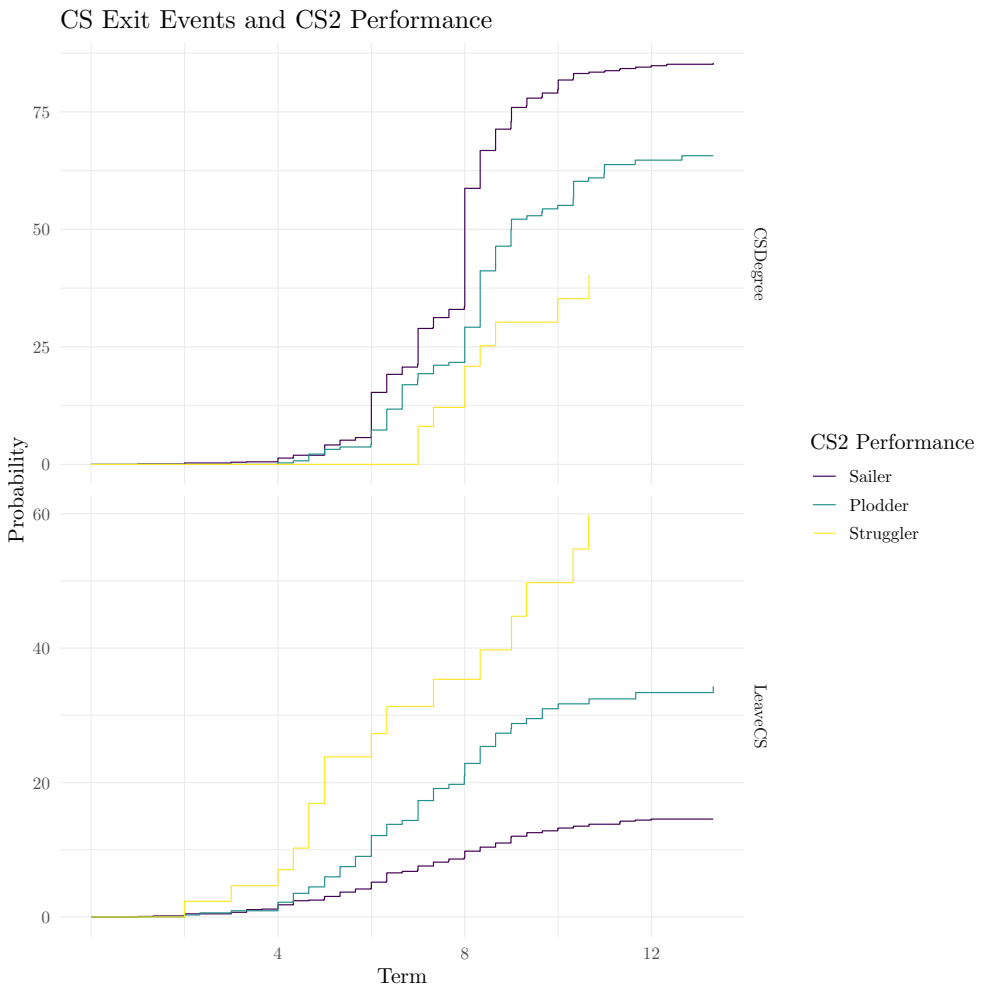

Figure 3.6: CS Exit Events Grouped by CS2 performance.

To explore the effects of CS1 and CS2 performance on the likelihood for CS matriculates to earn a CS degree I conducted a logistic regression in which earning a CS degree within 6 years of entering LRI was the outcome variable.

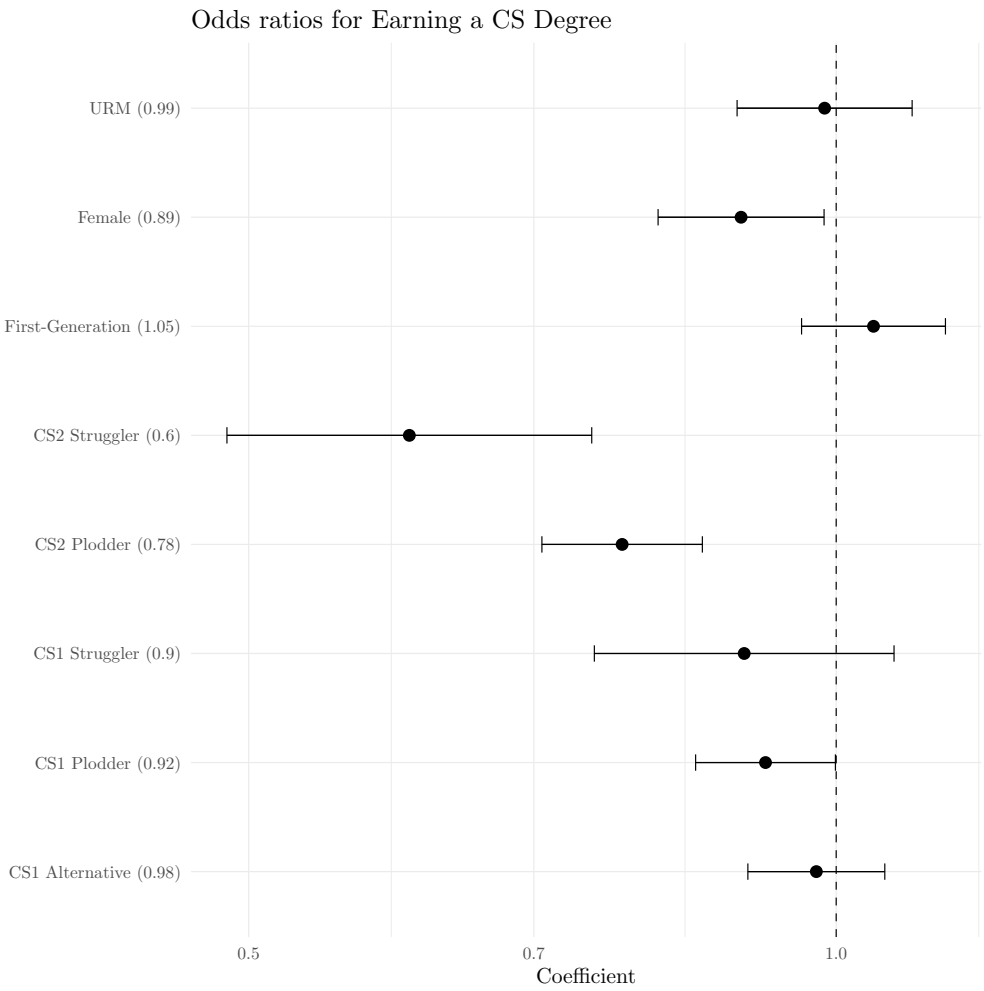

Figure 3.7: Logistic regression (Odds ratio) for earning a CS degree

As before, I first fit a model with only demographic variables (McFadden's Adjusted $R^2 = 1.46 \times 10^{-3}$), then another with the CS1 and CS2 performance predictors included (McFadden's Adjusted $R^2 = 0.149$). The increase in McFadden's Adjust R^2 suggests that the second model is a better fit for the data. Results show that lower performance in both CS1 and CS2 both contribute to a lower likelihood to earn a CS degree, and Female CS students are 11% less likely (odds ratio 0.89) to earn a CS degree than Male CS students.

3.6 Discussion

The results of this analysis, and relatively low goodness-of-fit measures in particular, indicate that while past academic performance may explain future academic performance in computer science, there are likely other factors at play that are not included in these data. The model developed by Suresh (2006) suggests factors such as study habits, perceived culture of support, and motivation to succeed can all impact performance in gatekeeper courses, in addition to past academic performance. The lack of these factors in the analysis could explain the relatively low measure of fit of the multinomial logistic regression model of CS1 performance. Regarding the low measure of fit for the model predicting matriculation based on performance in CS1, enrollment management practices may be a significant factor of this relationship.

3.6.1 First-generation and URM Students

With the limitation of the model in mind, it does indicate that performance in CS1 may be better explained by high school performance than demographics alone, and variation observed across demographics may in fact be a result of variation in SAT scores across demographics. However, when taking into account SAT scores, first generation students were less likely to struggle with CS1 compared to continuing generation students, and the higher matriculation rates for first-generation are consistent with the study in Chapter 2 which found first-generation students had higher persistence and yield rates in computer science.

One possible explanation may be that first-generation students are less likely to change majors (Engle and Tinto 2008), and so those who develop an intent to major in CS during the FYE program follow through and matriculate into CS. This behavior is consistent with the event history analysis which shows first-generation CS students are less likely to leave

CS than their peers, show in Figure 3.8.

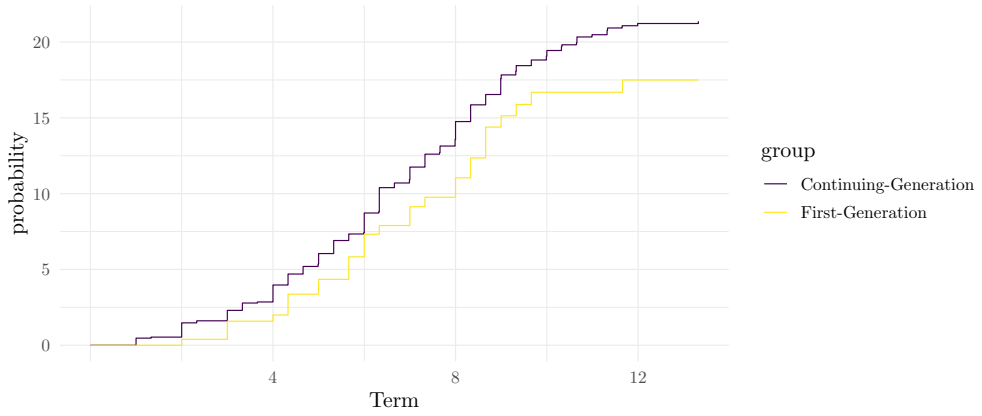

Figure 3.8: CS Exit Events Grouped by Generation Status.

The gap between the probability of continuing-generation and first-generation leaving CS does not widen noticeably after the 4th term. This suggests that after that time first-generation and continuing-generation students are similarly likely to leave, it is in the first 2 years that first-generation students are more likely to persist than their peers.

Given that first-generation students in all CS1 performance groups matriculated to CS at higher rates then their peers, there are likely factors beyond academic performance at play. Lohfink and Paulsen (2005) found several notable differences between factors that influence the persistence of first-generation and continuing-generation students including the ability to live at home and satisfaction with social life (Lohfink and Paulsen 2005). It is possible that first-generation students are finding a satisfying social life in the Computer Science department, or that the possibility of a future programming job that allows them to stay close to home may make computer science especially attractive to them.

It is also important to recall that the method used to define intent to major in CS excludes students who enter the CS pathway from some other other than the GE program. Because relatively more first-generation students are entering through these alternative pathways,

the matriculation results concerning this demographic are incomplete.

3.6.2 Matriculation, and Graduation of CS1 Women

Women's slightly higher likelihood of matriculating into computer science seem to disagree with other studies that indicate women are more likely than men to leave because of perceptions of grades (Biggers, Brauer, and Yilmaz 2008). There are several possible explanations for this observation. First, it is possible that the assumption that enrollment in CS1 is an indicator of intent to major in CS is not consistent across demographics. It could be that more men than women are taking CS1 for reasons other than to major in CS. Another possibility is that while perceptions of grades do inform decisions to major, other factors may be contributing as well. For example, Barker, McDowell, and Kalahar (2009) found that student-student interactions were a powerful predictor of decisions to major.

Men are also more likely to matriculate into computer science with an alternative to CS1. Of the 283 people who matriculated to CS within a year after recording alternative CS1 credit, 243 (85%) were male. This could be a reflection of men being more likely to have alternative CS1 credit than women, or that men were more likely to have a perception that CS1 would be boring and not useful to them and so being more likely than women to use their credit to opt out of CS1.

Despite a slightly higher likelihood that women in CS1 to matriculate than men, women make up an underrepresented 15.8% of the population enrolled in CS1 while registered as a GE major. It is possible that the relatively fewer women that enrolled in the course were more motivated to pursue CS than men. There is evidence to suggest that because women CS majors are more similar in important regards to male CS majors than to female non-majors (Beyer and Haller 2006). Additionally, female students are likely aware of the male-dominated culture of computer science (Papastergiou 2008) and so it is possible that

Female CS students are self-selecting into CS1 more so than Male CS1 students. In other words, it is could be that the women who take CS1 are already more motivated to continue into CS than the men.

The low representation of women enrolled in CS1 could be due to factors other than interest in CS. As discussed earlier, the timing of CS1 on the published timetable is seemingly at odds with students using the resources of the FYE program to make an informed major choice. Students who discover CS as possible interest after their first semester, and thus do not enroll in CS1 their second, may assume that taking CS1 later than the second semester may delay their graduation timeline. While this is not necessarily the case, students may not know that unless they talk to an academic advisor. Differences in who seek out advise from their academic advisors could lead to differences in who enrolls in CS1 or chooses an alternate path.

Finally, the results of the logistic regression indicate that women who do matriculate into CS are less likely than men to complete a CS degree within 6 years. Thus, while the experience women have in CS1 may be encouraging them to matriculate to CS, they are still less likely to complete a CS degree. This would suggest they are leaving at higher rates than men after matriculating. The probability curves shown in Figure 3.9 support this, showing women have a higher probably than men to leave computer science, with the gap widening with time.

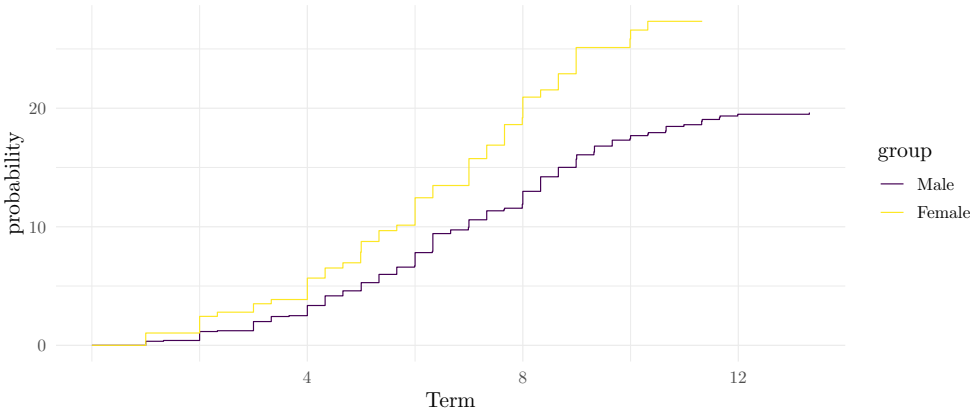

Figure 3.9: CS Exit Events Grouped by Gender.

Together these results seem to suggest that the experiences in CS1 itself are not reducing the likelihood of women, URM, or first-generation students to matriculate into computer science, but other factors, components of the FYE program or features external to it, may be contributing to a lower representation of women even enrolling in CS1 as GE students. The reduced likelihood that women who matriculate to CS graduate, compared to men, could be due to factors related to the CS department itself, or to the FYE program. In a study conducted by Biggers, Brauer, and Yilmaz (2008) one of the most common reasons for students to leave computer science was a loss of interest in the field, or a change of perception of what a future as a computer science would look like. It is possible that the CS1 course at LRI provided positive experiences for women, but later courses in the curriculum shifted their perception of computer science leading them to leave.

3.7 Implications

While the representation of women entering CS1 is low, that many do matriculate into CS but later leave without earning a degree in CS suggests experiences after CS1 may be an important area to target retention efforts. Courses during these middle-years of engineering tend to focus on technical skills with limited connections made to practice (Lord and Chen

2014), which could be contributing to an increased attrition rate in women. Practitioners should look to this middle-year curriculum for opportunities to expand coursework to include opportunities for students to have authentic learning experiences.

That CS1 is a required course for majors other than CS is also a point for reflection. Pass rates in CS1 vary a great deal by enrolled major at the time taking the course. Given the importance of creating authentic learning experiences (Magana et al. 2016), it is likely the current curriculum is more aligned with the needs and expectations of CS majors, rather than others. It may be appropriate to either create additional versions of CS1 targeted to different audiences, such as different experiences levels (Cohoon and Tychonievich 2011) or different content focus (Guzdial and Forte 2005). Since there *are* other first-year level CS programming courses available–including one intended for non-majors, it is possible one of these other options would be a better fit for certain majors. If that is the case, academic advisors may need to help direct students who need the introductory programming for a non-CS requirement to an alternative option better suited to their needs.

3.8 Limitations and Future Work

Much of this analysis was framed around the assumption that earning a grade in CS1 is an indicator of intent to major in computer science. Unfortunately, interpreting a student's reasons for enrolling in CS1 is not this simple as the course serves as a required course and as an elective for several majors outside of the college of engineering. Additionally, by only including students who were enrolled in GE at the time they took CS1 excludes disproportional more first-generation and URM students. To address this, a different measure of intent to major in CS is needed.

The relative likelihood of first-generation students to matriculate and persist in computer science demonstrates a need for future studies that explore this relationship in more depth. As

discussed in the previous chapter, there may be important variations within first-demographic students, such as SES, that may help to explain these patterns.

The plodders group in both CS1 and CS2 deserve more detailed analysis. This is an interesting group as they are likely to be on the cusp of persisting or leaving the major, and so better understanding other factors that go into this decision is important to support these students.

One important factor that was not part of this analysis was the role of faculty representation. There is evidence that students who see faculty as a potential role-model in ways congruent with their gender or racial identity, may persist in STEM at higher rates than students who do not have positive role-models in their fields (Bettinger and Long 2005; Price 2010). Relevant to this possibility is that the representation of women in the CS faculty increased during the years data were collected (See Table 1.1). Future work is needed to determine if this change had a measurable affect on women's persistence in CS. However, the observed high rates of persistence of URM CS students despite the lack of a diverse CS faculty should not be construed as evidence that faculty representation is not an important component of student success.

The effects of enrollment management were also part of this analysis. Future work should further analyze the population of students who take CS1 and do not matriculate into CS. Given that CS is a competitive major at LRI, it is possible that some students who intend to major in CS, and who are high performers in CS1, still are not permitted to matriculate to the major due to a low overall GPA.

Finally, while it is expected that students take CS1 while they are still in the FYE program and CS2 once they have matriculated into CS, this is not always the case. If students discover an interest in CS during their second term, while still in the FYE program, they may seek

help with an advisor to design a course plan that allows them to take CS1 in their third term, after having matriculated into the CS. Thus, there may be additional factors that impact CS1 performance that may be different for students who take CS1 while still in the FYE program and those taking it after. Further analysis is needed to explore the effects of the FYE on CS1 performance and subsequent persistence in CS.

3.9 Conclusion

These results indicate a need to better understand factors other than academic performance to explain representation within computer science. In particular, exploring non-traditional pathways may better capture the variation in matriculation and persistence in URM and first-generation students. Taking time into account when inspecting events such as leaving CS suggests there is variability in a likelihood to leave CS at different points in the curriculum.

References

AIR (2012). *Broadening Participation in STEM: A Call to Action*. American Institutes for Research (cit. on p. 45).

Barker, Lecia J., Charlie McDowell, and Kimberly Kalahar (2009). "Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major." In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA). SIGCSE '09. New York, NY, USA: ACM, pp. 153–157. DOI: 10.1145/1508865.1508923 (cit. on p. 70).

Basken, Paul (2018). *Engineering and Computer Science: Time to Separate?* URL: http://www.asee-prism.org/engineering-and-computer-science-time-to-separate/ (visited on 2019) (cit. on p. 45).

Benbow, Ross J. and Erika Vivyan (2016). *Gender and Belonging in Undergraduate Computer Science: A Comparative Case Study of Student Experiences in Gateway Courses. WCER Working Paper No. 2016-2*. Wisconsin Center for Education Research (cit. on p. 48).

Bennedsen, Jens and Michael E. Caspersen (2007). "Failure Rates in Introductory Programming." In: *ACM SIGCSE Bulletin* 39.2, pp. 32–36 (cit. on pp. 46, 48).

Bettinger, Eric P. and Bridget Terry Long (2005). "Do Faculty Serve as Role Models? The Impact of Instructor Gender on Female Students." In: *American Economic Review* 95.2, pp. 152–157 (cit. on p. 74).

Beyer, Sylvia and Susan Haller (2006). "Gender Differences and Intragender Differences in Computer Science Students: Are Female Cs Majors More Similar to Male Cs Majors or Female Nonmajors?" In: *Journal of Women and Minorities in Science and Engineering* 12.4. DOI: 10.1615/JWomenMinorScienEng.v12.i4.50 (cit. on p. 70).

Biggers, Maureen, Anne Brauer, and Tuba Yilmaz (2008). "Student Perceptions of Computer Science: A Retention Study Comparing Graduating Seniors with Cs Leavers." In: *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. Vol. 40. SIGCSE '08. New York, NY, USA: ACM, pp. 402–406. DOI: 10.1145/1352135.1352274 (cit. on pp. 70, 72).

Cohoon, James P. and Luther A. Tychonievich (2011). "Analysis of a CS1 Approach for Attracting Diverse and Inexperienced Students to Computing Majors." In: *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*. SIGCSE '11. New York, NY, USA: ACM, pp. 165–170. DOI: 10.1145/1953163.1953217 (cit. on pp. 47, 73).

CRA (2015). *CRA Taulbee Survey*. URL: https://cra.org/resources/taulbee-survey/ (visited on 2018) (cit. on p. 50).

Dziak, John J et al. (2019). "Sensitivity and Specificity of Information Criteria." In: p. 13 (cit. on p. 59).

Engle, Jennifer and Vincent Tinto (2008). "Moving Beyond Access: College Success for Low-Income, First-Generation Students." In: *Pell Institute for the Study of Opportunity in Higher Education* (cit. on p. 68).

Field, Andy, Jeremy Miles, and Zoë Field (2012). *Discovering Statistics Using R*. Sage publications (cit. on p. 60).

Guzdial, Mark and Andrea Forte (2005). "Design Process for a Non-Majors Computing Course." In: *ACM SIGCSE Bulletin*. Vol. 37. ACM, pp. 361–365 (cit. on pp. 47, 73).

Kinnunen, Päivi and Beth Simon (2011). "CS Majors' Self-Efficacy Perceptions in CS1: Results in Light of Social Cognitive Theory." In: *Proceedings of the Seventh International Workshop on Computing Education Research*. ICER '11. New York, NY, USA: ACM, pp. 19–26. DOI: 10.1145/2016911.2016917 (cit. on p. 47).

Lewis, Colleen M., Ken Yasuhara, and Ruth E. Anderson (2011). "Deciding to Major in Computer Science: A Grounded Theory of Students' Self-Assessment of Ability." In: *Proceedings of the Seventh International Workshop on Computing Education Research*. ICER '11. New York, NY, USA: ACM, pp. 3–10. DOI: 10.1145/2016911.2016915 (cit. on pp. 46, 47).

Lohfink, Mandy Martin and Michael B. Paulsen (2005). "Comparing the Determinants of Persistence for First-Generation and Continuing-Generation Students." In: *Journal of College Student Development* 46.4, pp. 409–428. DOI: 10.1353/csd.2005.0040 (cit. on p. 69).

Lord, Susan M. and John C. Chen (2014). "Curriculum Design in the Middle Years." In: *Cambridge Handbook of Engineering Education Research*. New York, NY: Cambridge University Press, pp. 181–195 (cit. on p. 72).

Magana, Alejandra J. et al. (2016). "A Case Study of Undergraduate Engineering Students' Computational Literacy and Self-Beliefs about Computing in the Context of Authentic Practices." In: *Computers in Human Behavior* 61, pp. 427–442 (cit. on p. 73).

Margolis, Jane and Allan Fisher (2002a). "Middle and High School: A Room of His Own." In: *Unlocking the Clubhouse: Women in Computing*. MIT press, pp. 33–48 (cit. on p. 46).

— (2002b). *Unlocking the Clubhouse: Women in Computing*. MIT press (cit. on p. 46).

McKinney, Dawn and Leo F. Denton (2004). "Houston, We Have a Problem: There's a Leak in the CS1 Affective Oxygen Tank." In: *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science*

Education. SIGCSE '04. New York, NY, USA: ACM, pp. 236–239. DOI: 10.1145/971300.971386 (cit. on p. 47).

NAE, National Academy of Engineering (2018). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. National Academies Press. DOI: 10.17226/24926 (cit. on p. 45).

National Academies of Sciences, Engineering (2017). *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments*. DOI: 10.17226/24926 (cit. on p. 50).

Papastergiou, Marina (2008). "Are Computer Science and Information Technology Still Masculine Fields? High School Students' Perceptions and Career Choices." In: *Computers & Education* 51.2, pp. 594–608. DOI: 10.1016/j.compedu.2007.06.009 (cit. on p. 70).

Price, J. (2010). "The Effect of Instructor Race and Gender on Student Persistence in STEM Fields." In: *Economics of Education Review* 29.6. 901, pp. 901–910. DOI: 10.1016/j.econedurev.2010.07.009 (cit. on p. 74).

Roberts, E. (2016). "A History of Capacity Challenges in Computer Science." In: *Retrieved June* 1, p. 2016 (cit. on p. 46).

Sax, Linda J. et al. (2018). "Sense of Belonging in Computing: The Role of Introductory Courses for Women and Underrepresented Minority Students." In: *Social Sciences* 7.8, p. 122. DOI: 10.3390/socsci7080122 (cit. on pp. 48, 49, 64).

Singer, Judith D. and John B. Willett (2003). *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford University Press (cit. on p. 58).

Suresh, Radhika (2006). "The Relationship between Barrier Courses and Persistence in Engineering." In: *Journal of College Student Retention: Research, Theory & Practice* 8.2, pp. 215–239. DOI: 10.2190/3QTU-6EEL-HQHF-XYF0 (cit. on pp. 46, 48, 49, 53, 55, 68).

Chapter 4: Manuscript 3

MATLAB Instruction in a First-Year Engineering Course

Target Journal: International Journal of Engineering Education

Abstract

The purpose of this study is to gain a better understanding of the varied ways students in the first year engineering program experience MATLAB instruction. As a common and significant component of first-year engineering programs, MATLAB is also one of the few or only ways first-year engineering students are introduced to the concept of programming and data processing. The focus on variation of experience mediated by MATLAB will help educators make informed choices about how to provide instruction and assignment scoping when using MATLAB to ensure more students are able experiencing MATLAB instructions in ways that align with first-year learning objectives.

4.1 Introduction

There is growing national interest in developing more computer programmers. A significant driver of this effort is the observation that "computer science (CS) is a 'new basic' skill necessary for economic opportunity and social mobility" (CSForAll Consortium 2017). Efforts such as the CS For All movement seek to provide access to a basic CS education to every child in the United States as one way to ensure more people acquire this skill. Within engineering, it is well established that computing is an integral part of engineering research and design (Hankley 2004). Related to this, many first-year engineering programs include some form of computer programming instruction, either through dedicated courses or integrated into other topics (Brannan and Wankat 2005; Reid and Reeping 2014).

Teaching programming to first-year engineering is challenging due in part to the challenge of teaching programming in general (Sleeman 1986), coupled with the varied levels of experience and motivations to learn programming in a first-year engineering course (Azemi and Pauley 2008). Different engineering disciplines leverage programming in different ways and in a general engineering course, students may not understand how or why programming is relevant to their desired discipline.

The fact that some form of programming is taught to general engineers, however, presents an opportunity to increase interest in computer science. If students who otherwise wouldn't have had access to early programming experiences gain these experiences during the course of a common first-year program while they are still in the process of selecting a major, they may become motivated to pursue computer science. However, the use of MATLAB as a programming environment complicates this process. MATLAB is not designed to be a general programming environment (Azemi and Pauley 2008), and is not a tool associated with the computer science discipline. Thus, if students do have a positive experience with MATLAB, they may not necessarily associate this with programming as a skill, and may still not consider computer science as a prospective major.

In this study, I discuss how first-year engineering students in a general engineering (GE) program at a large public research institution describe MATLAB instruction. Engineering students at this institution, Large Research Institution (LRI) are required to complete a common first-year engineering (FYE) program prior to matriculating into a disciplinary major such as computer science. In the common coursework of this FYE program, students are instructed in the use of MATLAB for data analysis as well as for logical thinking. This work provides understanding of how the challenges we would expect to see in such a context may interact with other contextual factors, such as instructional strategies, and student's perception of the role of programming in their future, to direct students' interpretation of

their experiences and shape future action.

The results of this study will help inform MATLAB instruction in the context of first-year engineering courses. In particular, the results suggest implications for instruction to promote positive experiences for most students.

4.2 Literature Review

Many studies have been conducted to better understand how people learn programming and how best to teach programming, see Pears et al. (2007) or Robins, Rountree, and Rountree (2003) for two seminal literature reviews of the topic. The context for the vast majority of these studies are introductory programming courses within computer science departments. Significantly, even in these specialized fields that are dedicated to the learning and teaching of programming, there are concerns of high failure rates (McCracken et al. 2001) in achieving desired learning outcomes. Attempts to address this problem have largely consisted of 'trial and error' approach to pedagogical and curricular changes (Bruce and McMahon 2002). Teaching programming in an engineering context shares some of the challenges described in theses sources, but introduces additional considerations as well.

4.2.1 Programming for Engineers

A salient challenge of teaching programming to engineers is the fact that different engineering disciplines have different expectations and requirements regarding programming skills (Azemi and Pauley 2008). Literature pertaining to introductory programming to engineers often focuses on the needs of specific disciplines such as Material Science and Engineering (Magana, Falk, and Reese Jr 2013) and Mechanical and Aerospace Engineering (Mascaro and Mascaro 2015; Mobasher et al. 2002; Rosca 2006). Each of these studies provides useful insight regarding the specific needs associated with individual engineering majors, for example,

teaching curve fitting functions as a means to model electrical circuits from measured voltage and current data. While some of these insights may transfer to a general engineering context, there are still additional considerations when deploying a shared curriculum to students with a range of disciplinary interests.

Many reports on common introductory programming instruction for general engineers (Hrynuk et al. 2008; Hsu, Amirtharajah, and Knoesen 2013; Lehr and Grant 2007; M 2009; Morrell 2007; Nagurney 2001, e.g.) focus on curricular features and high-level student feedback. No studies have been found that investigate the deeper experiences that engineers in particular have during introductory programming instruction in the way that has been done with computing-specific disciplines.

Many courses incorporate a hands-on component with hardware, such as Arduino boards (Mascaro and Mascaro 2015) or simple robotics (Azemi and Pauley 2008; Hamrick and Hensel 2013). These hardware components help situate and motivate programming as a tool necessary to collecting and interpreting data from the external world and in turn controlling aspects of the external world. Like the previously studies though, these reports provide no in-depth analysis of how students are experiencing the interaction of hardware and software in their learning.

Given the importance of a disciplinary-focused introduction to programming (Magana, Falk, and Reese Jr 2013), there is a notable lack of literature surrounding the unique challenges of teaching programming in an general engineering context.

4.2.2 Varied Experiences Programming

Several studies investigate how computer science students *experience* learning to program. The theoretical basis for this line inquiry is that students experience the learning process differently, and to provide a learning environment which allows all students to thrive we must

first understand the variety of ways that students learn. Within this category of studies, some look broadly at learning to program in general (e.g. Bruce, Buckingham, et al. 2004; Thuné and Eckerdal 2009) while others look at learning specific concepts within computer programming such as learning the concepts of objects and classes in Java (Eckerdal and Thuné 2005) or the concept of recursion (Booth 1992).

These studies are phenomenographical in nature. Phenomenographic research is a qualitative method used to explore variations in how people experience a given phenomenon. A foundational assumption of phenomenographic inquiry is that while each individual experiences a phenomenon differently, there are a finite number of qualitatively different ways that a population experiences a given phenomenon. The result of phenomenographic analysis is a list of categories of the ways people experience a phenomenon with descriptions of each category. By categorizing the varying ways people experience learning, the results of a phenomenography are well suited to inform pedagogical practice, or curricular design (Case and Light 2011).

The phenomenographic method is designed to consider individual experiences in relationship to the complete set. Interview transcripts are viewed together as a *pool of meaning* from which the researcher follows an iterative process to form categories of experience (Åkerlind 2005a). The result of a phenomenography is an outcome space in which a finite number of experiences are positioned in relation to one-another in some logical way, often hierarchical (Lönngren 2017).

The outcome space of both Bruce, Buckingham, et al. (2004) and Thuné and Eckerdal (2009) contain five categories of experiences ordered from simple to complex. Both provide general recommendations to create learning environments that offer opportunities for students to experience programming in more varied ways. For example, in the most complex category, Bruce, Buckingham, et al. (2004) finds that learning to program is seen as becoming a

member of a larger community. Without intentional changes in the curriculum or instruction, this experience will likely only be available to students who already have prior experience and are aware that a larger programming community exists. Thus, an instructor may integrate work that specifically connects to the larger community and discuss how the community can be used as a resource.

4.3 Need for this Research

Despite the ubiquity of programming instruction in early engineering education, there is a lack of research investigating how engineering students experience their programming education. This gap is in contrast to the research of teaching and learning programming in computing-specific domains, such as computer science. While it is tempting to assume that research of teaching and learning programming may transfer from domain-specific areas to general engineering, there are critical differences in the motivation and structure of introductory programming within engineering compared to computing disciplines. First, students enrolled in computing disciplines likely already believe programming to be an important skill for their discipline of choice. In contrast, in a general first-year curriculum students lack the same motivation to learn programming that students who choose computing majors implicitly have (Forte and Guzdial 2005). Second, the reasons why programming is an important skill differs between computing and non-computing disciplines. Within computing fields programming is a central focus of inquiry and research, whereas in non-computing engineering disciplines programming is essential for its association with modeling, abstraction, and numerical analysis (Hankley 2004). Further, it is important that programming instruction be conducted in disciplinary-specific ways (Magana, Falk, and Reese Jr 2013), a recommendation that poses particular challenges to common first-year engineering programs.

Understanding how programming is experienced by first-year engineers is important for im-

proving the educational experience in general, as well as for identifying potential barriers to full participation. Introducing computing concepts and programming to non-majors without connecting to their diverse interests may discourage some from pursuing further learning (Forte and Guzdial 2005). Understanding these experiences is especially important because in the programming experience in the first-year curriculum at LRI is often the only formal introduction to a skill non-CS majors will have. Thus, while those with pre-college programming experience may have already decided to major in CS and their experience with MATLAB in the first year will likely not affect their decision, for those coming in not considering CS as a major, but who have potential to excel in the major, this experience could determine whether they consider CS as a viable choice. In other words, the programming experience obtained in the first-year program could present an opportunity to expand the pool of students interested in CS to include those who had arrived thinking it would never be an option for them. This could help mitigate some of the negative consequences of the varied access students have to pre-college programming experiences that could be contributing to the underrepresentation of women and minorities in CS. Results from Chapters 2 and 3 show that URM and first-generation students are developing interest in computer science during the course of the FYE program, but the results do not indicate if this is a result of an aspect of the FYE program itself, or something external to it.

With this in mind, there is an opportunity within the first-year program to adopt practices that work to broaden participation within computing. By understanding the variety of ways first-year engineering students experience programming, we are better poised to align instruction and assessment with learning goals. To that end, this study addresses the research question:

How do first-year engineering students describe experiencing MATLAB instruction?

4.4 Theoretical Framework

The key components of this proposed research are the students' experience learning programming, and their experience using MATLAB as a tool to learn programming. While phenomenography is rooted in the framework of variation theory that addresses the first component, the focus on experiences as recalled by participants backgrounds the context in which those experiences were had. In this case, the material context of the MATLAB software is of critical interest which motivates a small but significant extension of phenomenography to better capture the varied ways the tool itself may mediate student experience.

An extension of phenomenography in the context of learning about computing is not without precedent. Berglund (2005) reaches a similar conclusion in their study of how students learn about networking protocols situated in a milieu of tools. They observe that under the "classical" definition of phenomenography described by Marton and Booth (1997), participant's experience of a phenomenon is inseparable from the context in which the phenomenon takes place. While Berglund (2005) was primarily concerned with social context–the other people in the learning environment have a significant impact on how an individual experiences learning–the argument applies to material context as well. Students learning programming with MATLAB will have a different experience than if they had instead used Visual Studio for example.

4.4.1 Sociomateriality and Experience

Theorizing about the interconnections between the social and material context in learning environments is not new. Recognizing that technology use in the classroom was severely "under-theorized", Johri (2011) proposed sociomateriality as a theoretical lens that engineering education researchers might use. The central idea of sociomateriality is that the social world (people, our thoughts and actions) and the material world (both physical objects like

computers and non-physical like software) cannot be inspected in isolation of one another. Using a sociomaterial perspective, the unit of analysis becomes the social-material system as a whole and the relationships between components of the system.

One advantage of sociomateriality in this context is the focus it puts on the material aspects of the tools we use. This is especially relevant to the field of engineering education because a common aspect of engineering that we teach is design, specifically the design of material objects that others will interact with. Theoretical frameworks that do not stress the importance of materiel context in shaping experience risk losing sight of how that context itself contributes to varied experiences in ways that may be significant to educators. In other words, as educators we would like to know whether a tool such as MATLAB creates barriers to having a positive learning experience for some students so that we might work to dismantle them.

4.5 Methods

4.5.1 The Site

At Large Research Institution, all first-year engineering students complete a common two semester sequence, FYE1 and FYE2, introducing them to general engineering concepts and skills. Embedded within this course is a MATLAB curriculum designed to teach basic programming and data processing skills. Since lack of prior experience is a significant barrier to entry into a computing field (Lewis, Yasuhara, and Anderson 2011), this required programming module embedded in a first-year engineering program presents an opportunity to provide positive early experiences with programming. Because the way students interpret their experiences shapes future action (Oyserman et al. 2017), a positive programming experience during the first year could provide motivation for some students who otherwise would not have considered a degree in computing to pursue that interest. Alternatively, a poor

experience with programming during the first year could discourage otherwise promising students from pursuing a computing degree. This is especially relevant for students with little or no awareness of the field, as this will be their one and therefore defining experience with computing. In light of the evidence presented in Chapter 2 that many URM and first-generation students are choosing computer science after their first year, this early experience could be a particularly important component to support broadening participation in computer science.

4.5.2 Participants

The sampling frame for this study consisted of all students enrolled in the first-year engineering program (FYE1 or FYE2) at Large Research Institution during the Spring 2018-Fall 2018 time period. Because the objective of phenomenography is to understand how experiences differ, it is important to use a sampling strategy that maximizes the potential variation in experience (Case and Light 2011). Literature suggests that both gender and prior programming experience both mediate how students perceive introductory programming courses. While I did have access to gender information, there was no direct indicator of prior programming experience. Instead, I was able to use responses to a survey administered to all engineering students prior to their first semester that included a question about intended engineering major. Based on literature that suggests decisions to major in computer science are driven in part by prior programming experience (Lewis, Yasuhara, and Anderson 2011), I used this response as a proxy for prior programming experience during recruitment.

Using this information, I recruited in several batches to build a statistically non-representative stratified sample (Sandelowski 1995) along gender and intent to major in CS. I successfully recruited 16 participants, categorized across the four sampling strata as shown in Table 4.1.

Interestingly, the sampling for intent to major in CS did not result in a balanced sample of

Table 4.1: Characteristics of recruited participants

	Male	Female
CS Intent	5	3
No CS Intent	4	4

participants with and without prior experience: 12 of the 16 participants in the resulting sample described prior experiences with programming (5 female, 7 male). Additionally, of the two participants that described taking the CS AP exam in high school, one took it without any intent to major in CS. This suggests that many participants in this sample have an interest in programming, even if that interest does not lead to a desire to pursue computer science as a major.

4.5.3 Interview protocol

The data source for this study are a set of semi-structured interviews conducted over the course of the spring and fall semesters. A semi-structured interview allowed for more of a conversational flow while still providing reasonable assurance that I visited a consistent set of topics across participants(Patton 2002). The protocol shown in Protocol 1 was designed to collect information across four broad categories: prior experience, perceptions of MATLAB and programming instruction in general, perceptions of assignments and in-class activities, and perceptions of instruction of the MATLAB interface itself.

Patton (2002) says that a purpose of interviewing is to allow the researcher to "enter into the other persons' perspective" (p.341). Phenomenographic interviews are no different, but they have a specific goal of eliciting the ways participants think about particular experiences (Åkerlind 2005a). Phenomenographic interviews often center around concrete examples of the phenomenon recalled by the participant, but while question may start with "what", as in "what did you do?" the goal of follow-up prompts is to move the participant to addressing

"why", as in "why did you do it that way?" (Åkerlind 2005a). Participants for this study were asked to bring several examples of completed MATLAB work to serve as concrete examples to center the conversation around. To compensate for their time, participants received a $25 gift card.

Protocol 1: This semi-structured intake interview is designed to collect information about participant's programming experience prior to engaging in the MATLAB content in their course.

1. Can you start by telling me about any programming experience you had before college? If you did have prior experience:
   (a) can you describe how it may have influenced your perception and experience in FYE1/FYE2?
   (b) how did this prior experience compare with your experience in the first-year engineering course?
2. Describe how your instructor introduced MATLAB and programming as a topic?
   (a) How did your instructor talk about the relationship between programming and MATLAB? Did they say they were teaching you programming? MATLAB? Programming with MATLAB?
   (b) What did your instructor do that was helpful in being successful with the module? What resources did they provide that were helpful? What could they have done that would have been more helpful?
   (c) Did your instructor teach any strategies for completing the work, such as strategies for starting a new project, or for figuring out what to do when an

error occurred?

(d) Did you use any of these strategies while working on assignments? Were there strategies that made sense when explained but did not seem to apply to the work you did?

3. Tell me about the programming assignments that you brought to share. Why did you select these assignments? What stood out about them?

   (a) Which assignments were more interesting to you? Why?

   (b) Sometimes what is interesting does not always feel useful, were there assignments that you thought were useful even if they were not necessarily interesting, or interesting but not useful? What assignments were hard or easy?

   (c) What made the easy assignments easy and hard ones hard? How did you approach these assignments? your other homework? What strategies did you used?

4. How did your instructor describe MATLAB itself as a programming environments?

   (a) What did you like about MATLAB as a programming environment? Was there anything in particular about the interface that made working on assignments feel easy, enjoyable, or interesting?

   (b) What did you dislike about MATLAB as a programming environment? Was there anything about the interface that made working on assignments difficult, frustrating, or tedious?

5. Is there anything else you would like to add about your understanding of computer programming or how it relates to engineering work?

The influence of sociomateriality as a framework for emphasizing the context of an experience is seen in the interview question about the MATLAB interface itself.

4.5.4 The Phenomeographic Method and Analysis

Phenomenographic analysis begins by combining all transcripts into a pool of meaning. There is some variation in how transcripts are divided for analysis across different phenomenographers– some view each transcript as a whole representation of an experience, while others extract multiple excerpts from each transcript that each describe an experience (Åkerlind 2005a). Since many participants had completed both semesters of the first-year course, and there is variation in activities within each course, participants described several different experiences throughout an interview. For this reason, I started analysis with an excerpt-level approach beginning by dividing each transcript into a collection of excerpts and then grouping the excerpts together based on similarity.

During the first phase of the analysis, I interpreted each excerpt in the context of the entire transcript and the pool of meaning, and grouped excerpts describing similar experiences together. This process is iterative, and across several iterations I solicited feedback from colleagues trained in social science research. During the feedback sessions I described my current understanding of the set of experiences and how I saw them in relation to one another. This is a method of establishing trustworthiness that is well suited for phenomenographic research (Collier-Reed, Ingerman, and Berglund 2009).

After several iterations with these colleagues, we discovered that limitations in the context and the participant sample were contributing to challenges with developing a coherent outcome space. Specifically, because of the variation in instructor, and the fact that I had no way to intentionally sample across instructors, there were several excerpts that, while read as meaningful and interesting experiences, did not seem to group with others. This was

likely due to the fact that there were one or more isolated participant-instructor pairs and so there were not enough participants for each instructor to develop noticeable patterns across participants.

The work towards developing an outcome space from the excerpts did suggest two high-level dimensions along which experiences fell: perceived level of scaffolding, and an affective dimensions that captured whether a student interpreted their experience as positive or negative. This motivated a second phase of analysis in which I went back to the whole transcripts and identified each participant-instructor experience and where they fell on these two dimensions. For participants who took both courses, this approach yielded two separate experiences for each participant.

Trustworthiness. Establishing trustworthiness in phenomenographic research helps strengthen the outcome and impact (Collier-Reed, Ingerman, and Berglund 2009). The feedback from colleagues described above helps to establish trustworthiness through *dialogic reliability checks* (Åkerlind 2005b) by generating agreement across researchers through iterative discussion (Collier-Reed, Ingerman, and Berglund 2009). Using this process, I refined my groupings and descriptions until I was satisfied that they captured all the ways in which my participants described their experiences.

Another concern in phenomenographic research is the impact of research bias on the interpretation of student experiences (Ashworth and Lucas 2000). One method to mitigate some of the effects of researcher bias is *bracketing*, a process by which a researcher makes their own presuppositions known in an effort to consciously mitigate their influence on participant interpretations (Tufford and Newman 2010).

I have taught several semesters of the first-year engineering course that is the context for this study, so I necessarily have my own ideas of what MATLAB instruction looks like in

that course. I have also taught several semesters of a programming course in an Electrical and Computer Engineering department and so have my own beliefs about what programming instruction is and should be. To help mitigate my own beliefs about MATLAB and programming instruction, I designed the interview protocol to allow participants to describe their experience in detail, with prompts to ask for clarification and more description. During analysis, I was careful to set aside my own expectations of what MATLAB instruction was, and group excerpts only based on what other participants were saying, and not to the extent what I was reading aligned with my own expectations.

4.6 How Students Experience MATLAB Instruction

The result of this two-phased phenomenography is a two dimensional outcome space shown in Figure 4.1.

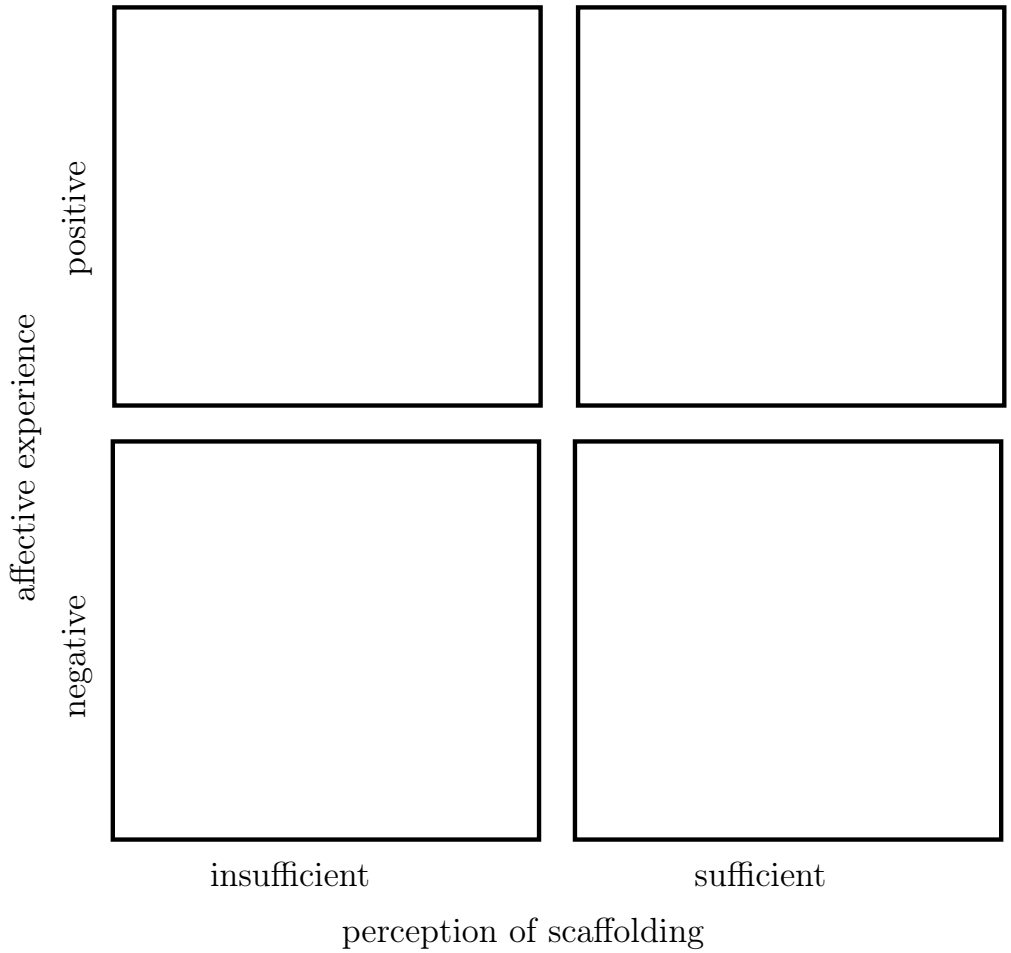

Figure 4.1: Outcome space containing the different experiences of MATLAB instruction.

Perception of scaffolding, shown as the horizontal dimension relates to how well the instruction participants experienced prepared them to be successful in the course. This ranges from insufficient on the left end – characterized by descriptions of struggling to complete the work or understand the material given the resources provided by the instructor, to sufficient on the right – characterized by descriptions of an ability to complete the work given the provided information.

On the vertical access the second dimension of affect ranges from negative experiences on the low side characterized by little to no enjoyment in either the instruction of MATLAB, or the

corresponding work. Positive experiences were characterized by high levels of enjoyment, or interest in the programming instruction itself, or the associated work.

4.6.1 Positive and Sufficient

Participants in this group, listed in Table 4.2, described positive experiences and sufficient instructor scaffolding.

Table 4.2: Participant-experiences that were positive with perceived sufficient scaffolding

	Gender	Race/Ethn.	Prior Exp.	Description
Participant 2	female	white	none	second semester
Participant 3	male	Asian	multiple	first semester content
Participant 4	white	Asian	some	first semester
Participant 5	female	white	block	second semester
Participant 6	male	mixed	multiple	Arduino-based activities
Participant 9	male	non-white	none	second semester project
Participant 11	female	white	multiple	first semester, flowcharts
Participant 15	female	white	some	project work, hands-on activities

Participants who had positive experiences and a perception of sufficient scaffolding described a learning environment in which the instructor provided clear instructions, ample resources to complete the work, and was available and approachable to answer questions. They shared a favorable opinion of MATLAB, or programming as it was used in their first-year engineering course.

> It's really not that hard to learn the basics, but you have to teach it. It has to be taught. You can't just learn it yourself, so she went through all the basics in the

> beginning, and that was really helpful, cause she started from actual scratch, but just went really fast so that we could get into the deeper material in MATLAB. – Participant 2, second semester

In this excerpt Participant 2, who had no prior programming experience, is contrasting her positive experience with sufficient scaffolding in the second semester of the course with the instruction she experienced first semester. In the first semester she struggled completing the work because her instructor did not teach "the basics", but her second semester instructor did. The quick pace of this did not bother her, as it allowed the class to "get into the deeper material". Participant 2 went on to say that many of "the basics" that she learned from her second semester instructor was a repetition of what was taught first semester but she never learned.

Positive experiences were associated with enjoyment in the process of solving problems or coding itself, or with the understanding of how programming or MATLAB might

> He was very good at teaching it. He showed the reasoning and the thought process behind programming structures, such as loops and if-statements and nested if-statements, and all those complicated things. He taught them in a very engineering-oriented way where there was a problem that had to be solved using a program and what was the most efficient and easiest way to do it? – Participant 4, first semester

This participant recognized a connection between how the instructor was teaching MATLAB, and engineering problem solving. It was common for participants to have positive experiences when they noticed a connection between programming instruction and engineering or real-world applications. For example, Participant 6 described an experience with two robotic projects as relevant to engineering:

> So both of those I think were cool because they gave you a sense of combining like several fields. Just the mathematics behind it, the computer science and programming it, and like real-world experimentation portion. So I think that was nice. – Participant 6

The response was to a question about how instructors positioned MATLAB in the context of engineering. While their instructor did not explicitly talk about this, they perceived the examples that used, including these two robotic projects, as situating MATLAB as a tool that has real-world applications.

Sufficient scaffolding wasn't always explicitly associated with the instructor. For example, Participant 15 describes a positive group experience:

> I'd say the most fun one was definitely the line following robot. I think I was most successful in that one. That was we were put in groups, and then for homework assignment we had to submit a MATLAB file with a line following robot and then the next day in class we were told to test those out and see whose worked and everything. We'd use whoever's worked on the next part of the project in class. I liked that one a lot. – Participant 15

She and her group were able to get the robot working after fixing a "fidgety" wheel. While there is not explicit mention of scaffolding, the ease with which she discussed completing the homework and comparing results with her group suggests she had the resources needed to be successful.

4.6.2 Positive and Insufficient

Participants in this group, listed in Table 4.3 described positive experiences despite also describing insufficient scaffolding from their instructor.

Table 4.3: Participant-experiences that were positive with perceived insufficient scaffolding

	Gender	Race/Ethn.	Prior Exp.	Description
Participant 10	male	white	some	second semester content
Participant 11	female	white	multiple	first semester
Participant 13	male	mixed	some	first semester

These participants describe insufficient scaffolding, such as not being provided enough background information or examples to complete work, but maintained a positive view of the work. All three participants had prior experience with programming, and two said they took the CS AP exam in high school. Their descriptions of positive experiences usually related to an enjoyment of programming in general, which allowed them to see past the instructional challenges they faced.

> I like to learn new things every day so when I'm working with a new language that I don't know and I accomplish something that I can accomplish in Java, I get really excited because it's new, and it's foreign and the syntax is different. If I accomplish the same thing I'm doing in Java, I find it very interesting and intriguing that two different languages I can do the same thing now. Participant 11

In some cases the positive view was maintained with an assumption that the lack of instruction was part of the experience, for example Participant 13 claimed the instructor did not provide any strategies for completing work, and when asked how they proceeded when they encountered a problem they said

> I just sat there and analyzed further the problem. But I would also look things

> up online if there's a function that can speed this up, so I didn't have to program the function itself. – Participant 13

For them, this was what was expected of engineers, or computer scientists, and was perceived to be good instruction:

> I would say it was pretty well taught, but that's from my own perspective. Because generally you want to experiment basically. That's why I think they call it computer science, but it goes beyond that. So you'd experiment a lot of times... see what works, see what doesn't. – Participant 13

One participant described insufficient scaffolding not in terms of lacking what they needed to be successful with the course, but lacking what they needed to meet their personal goal of understanding:

> The one thing I didn't like is I had no idea what a lot of this was doing. ... I prefer a blank slate, and I do everything myself. That way I understand everything. But I felt like I was shut off from a lot of this. I knew what I needed to know to get the robot to work, but I didn't know why it worked. Or how it was working. – Participant 10

This participant was describing the line following robot activity in which a large chunk of code was provided to them. While they had a positive experience with the project, saying it "was really cool to see some hands-on thing", they were unable to understand how everything worked together because the instructor did not explain the code that was provided them.

4.6.3 Negative and Insufficient

Participants in this group, listed in Table 4.4, described negative experiences and insufficient instructor scaffolding.

Table 4.4: Participant-experiences that were negative with perceived insufficient scaffolding

	Gender	Race/Ethn.	Prior Exp.	Description
Participant 2	female	white	none	first semester
Participant 3	male	Asian	multiple	second semester drone project
Participant 4	white	Asian	some	second semester
Participant 5	female	white	block	first semester
Participant 9	male	non-white	none	second semester M4
Participant 10	male	white	some	second semester evaluation criteria
Participant 15	female	white	some	exam, assignments without flowchart

This category consisted of perceptions of instruction that did not provide adequate resources to complete and understand the assigned work. This was accompanied by experiences of frustration and struggle associated with the lack of sufficient scaffolding.

> I had a really, really hard time my first semester with MATLAB. I honestly couldn't do anything. Every single assignment, I would get and be like, "Well, I have no idea what I'm doing. – Participant 2, first semester

In this excerpt, Participant 2 describes the struggle she experienced during her first semester. In contrast to her positive experience described above, this instructor did not teach "the basics", and without that she had difficulty understanding how to complete the work. This

experience was further exacerbated by stress induced by fast, in-class grading on new material:

> Yeah. I mean, he did [go over examples in class], but they were very fast, and it was definitely a more stressful ... Cause he would do in-class grading on it. He would ask a question, and be like, "Multiple choice, what should we output for this?" It was just stressful. It wasn't like, "Oh, I wanna learn." It's just like, "How do I get the answer? Who has the answer around me?" You know what I mean? – Participant 2, first semester

It is unclear if these exercises were actually graded, or if the stress was due to the pressure to get the right answer. Either way, it is clear this experience caused anxiety, and given that Participant 5 "honestly couldn't do anything" with each assignment, was not helpful in providing her the resources she need to be successful.

In several cases the lack of sufficient scaffolding was described in terms of integrating mathematical concepts into the assigned programs, as with this excerpt:

> This was, "Prepare a script file to fit that," I don't even know what the word is, exponential model? I don't know how to do that now, I can't even explain that. I don't know how to do that right now, let alone through MATLAB, and I feel like that was really challenging, was that it was harder for you to figure out the math. Even if you're the best programmer in the world, if you don't know how to do addition, I don't know. It was just really hard to even know where to start, I guess. – Participant 5, first semester

In this excerpt, the particpant's expression of frustration is not about the programming itself, but rather understanding what is being asked mathematically. They say "I don't know how

to do that right now, let alone through MATLAB" recognizing that they would first need to understand the underlying concept before learning to implement the concept in a program.

4.6.4 Negative and Sufficient

While the participant in this group, listed in Table 4.5, describes sufficient scaffolding to complete the work, they also had a negative experience related to the expectations of the work, how it was presented, or how it was evaluated.

Table 4.5: Participant-experiences that were negative and supportive

	Gender	Race/Ethn.	Prior Exp.	Description
Participant 3	male	Asian	multiple	first semester evaluation criteria
Participant 6	male	mixed	multiple	highly structured assignments, math

In some cases, participants described sufficient support to complete the assignments, but were unhappy with how they were designed for evaluated. For example:

> The point of this project was pretty much to, given three sets of data, of the locations of two boats, we were supposed to figure out how many times the boats are within, say, 27 miles of each other. We were supposed to use functions, but I didn't because it was simple enough that you don't have to do it. You just have to do the distance formula and then see if they were within distance of each other. – Participant 3

In the excerpt above the participant describes an assignment in which they received a poor grade, even though they believe they solved the problem that was asked. Though the assignment required the use of a function, they did not view this as part of the problem they were

being asked to solve, instead focusing on what was being asked of the data. They were aware of the required use of a function, but chose not to use one "because it was simple enough that you don't have to do it." They seemed less concerned with the fact they received a lower grade than with the problem setup in the first place:

> It doesn't really bother me much, but what bothered me was that it was such a simple program that I didn't think the need of functions was necessary. – Participant 3

A negative experience was not always associated disagreement with assignment requirements, sometimes it was associated with a confusion about why they were asked to do something a certain way, as with this experience about using a math formula provided by the instructor:

> One thing that was kind of confusing would be, in [FYE2] some of the equations they had us use for, I think even the final, was how to calculate power. And it was really vague. They wanted us to do some super simple calculation, and so some of my friends and even myself, we had taken physics. So when they said, oh, to convert this to power, just divide it by sixty. And we said, isn't that a bit weird? I mean this isn't one of the equations to really convert to power. But they'd ensured that that's all you needed to do. So it was kind hand weighty. Because they just kind of made sure that everything worked out. But I don't really know what you could do to clarify that any more. Participant 6

In this case, it isn't using the math itself that is challenging–they were given what was needed to complete the work–it is a perceived disconnect between the math they are asked to use with what they are familiar with from their physics class that leads to a negative experience.

4.7 Discussion

Looking across experiences, it becomes clear that similar instructional strategies may lead to quite different student experiences. Open ended problems can either be perceived as a lack of sufficient scaffolding by students struggling to translating the problem text into program requirements, or sufficient scaffolding by students who believe they are more realistic. Highly specific rubrics can elicit positive feelings because they helped understand "what was necessary" and "makes sure you don't forget anything, because there can be little things that you miss out on" (Participant 5), or contribute to confusion and negative experiences when students do not understand, or disagree with the criteria.

4.7.1 A Theoretical Framework for Understanding Experiences

While the use of sociomateriality provided a useful theoretical base for planning the study and informing the interview protocol, the analysis naturally progressed in a direction that took the focus off of the material interactions with MATLAB. Rather, prior knowledge, in the form of assumptions about engineering, or cues from the instructor, proved to be salient features that shaped how students interpreted their experiences. Because of this, identity-based motivation (IBM) is a useful framework for understanding this process. Under IBM, environmental cues interact with past experiences to influence how people perceive new experiences (Oyserman et al. 2017). In particular, this process model suggests that environmental cues affect the dynamic interpretation of experiences in ways that impact decisions to adopt either productive or unproductive strategies towards one's goals. This provides a helpful framework for understanding the experiences described above, in particular to help understand how the same instructional practices can lead to productive experiences for some students and unproductive for others. Because these interpretations of productive and unproductive experiences contribute to future behavior, it is particularly important to

understand factors that often lead to unproductive interpretations and develop strategies for minimizing them.

Viewed through this lens, the experiences above can be seen as either productive or unproductive. Some students found relatively straight forward assignments that were similar to one another a productive experience because they saw them as a way to practice basic skills that they would later need to draw on without thinking. Others had an unproductive interpretation of similar assignments as being tedious and a waste of time.

4.7.2 Productive Experiences

Many productive experiences were associated with perceptions of work that was applicable to engineering or the real-world. Magana, Falk, Vieira, et al. (2016) found that the strategy of embedding computing education in a meaningful engineering context, enabled students to better use their newly learned computing tools to support their disciplinary learning. It is important to note that while hands-on robotics projects tended to provide this experience, this can also be achieved without any additional equipment. Several participants described similar experiences of recognizing the utility of programming as a tool for engineering when they were asked to work with large data sets.

Of course, we must be cautious to provide sufficient scaffolding for loading the data sets, as one participant described a potentially unproductive experience associated with confusion about how to load data into MATLAB:

> It was kind of difficult, just because I wasn't sure how to use [the provided data file]. And there were so many different files that were saved and I just felt like it wasn't clear. It was kind of complicated and I got help a lot. It saved as a CSV file, which I know nothing about. So I wasn't sure how to use this on MATLAB.

– Participant 16

While this participant may have struggled with loading data into MATLAB, she was able to leverage the skills taught in class to help solve a problem that was directly relevant: running out of money on her meal plan. She described first thinking about how to come up with a budget by hand, but that she would have to do a lot of manual calculations each day, and then made the connection to MATLAB:

> And I was like, "Man, that's gonna take so much time. How am I going to do that?" And I didn't think any of the functions that I would need would be complicated. So then I was like, "Well there's MATLAB. That could do it for me." – Participant 16

This quote is an example of how productive experiences can lead a participant to pursue strategies that develop a deeper interest in the material. By choosing to leverage the skills learned in class, this participant acquired more experience with MATLAB as a programming environment, and with programming as a tool to solve practical problems. It is likely that the success she experienced with her meal plan budget program will lead her to view programming as a potential solution to future problems as well.

4.7.3 Unproductive Experiences

All of the experiences that fell in the negative and insufficient grouping were likely interpreted as unproductive by the participants. Struggling to complete assignments, not knowing where to start, or not having the necessary scaffolding to successfully use a new mathematical concept simultaneously while learning a new technical skill will likely lead students to question their ability to succeed in engineering.

One approach to avoid this is to ensure that any new concept is always demonstrated in class, with an example provided, before assigning it as homework. For example, Participant 5 described a successful experience using MATLAB to find the mean and standard deviation in a homework assignment because there was an in-class activity that demonstrated this before the assignment was started. In contrast, the same participant also described an unproductive experience attempting an assignment that required the fitting of an exponential model, something she hadn't seen before.

Other participants had unproductive experiences that may be attributed to a mismatch between the instructors expectation and student's expectation of the goal of the assigned work. For example, Participant 10 talked at length about a perceived over-emphasis on output formatting that cost him some points on an exam:

> I clearly got all the logic part right. But I didn't display it right, and I was just a little annoyed that that was the reason why I got it wrong. So I distinctly remember sitting in the test for 25 minutes, to 30 minutes, just thinking, "I know what the answer is, but how can I show that I ... how can I show it in the exact format they want?" And I couldn't figure it out. And I just thought that was really annoying. – Participant 10

To him, "the answer" was producing a value, and formatting of that value was not part of "the logic part". He perceived programming as primarily a logical activity, and himself as someone who understood the logic. In this and other instances, Participant 10 talked about output formatting as something external to programming, and something he did not value. The instance with the exam described in the above excerpt was not an isolated incident, Participant 10 also described struggling with output formatting on earlier assignments. It would seem that the earlier experiences were unproductive: Participant 10 did not adjust his

strategies to become successful in future assignments. From an IBM perspective, Participant 10 did not integrate the ideal that output formatting was an important component of writing good programs into his own identity as a programmer. Thus, even if he had a goal of becoming a successful programmer, he was not adjusting his actions to eventually meet this goal.

It is possible that this participant would have been able to integrate the concept of output formatting into his understanding of what programming was, and his sense of self as a programmer, had the instructor provided more of an explanation as to the importance of output formatting. For example, the same participant described understanding why data plotting was a major focus of the MATLAB work by saying "we're engineers, so plotting is a big thing that lots of different fields of engineering will involve." This believe about what is part of engineering came from prior knowledge; the participant stated that the instructor did not explain *why* plotting was a large component of the MATLAB work.

4.8 Implications for Practice

In the context of a common FYE program, how students interpret their early experiences with the general engineering coursework can have significant impact on which discipline they decide to pursue. While it is not feasible to provide unique experiences associated with each discipline, it is important to help students experience success at activities that are congruent with their sense of self, and their future goals (Oyserman 2013). An important consideration, however, is that students must be empowered to make connections between the activity and their own sense of self and goals, rather than be told by the instructor how the experiences aught to fit in with their identity (Elmore et al. 2016). For many participants in this study, rich, hands-on activities provided sufficient flexibility in content to allow individuals to focus on aspects that were salient to them. First-year instructors should design curriculum around

activities such as these, and avoid activities that are disconnected from meaningful real-world connections. Areas that participants described difficulty identifying how activities aligned with their own goals as engineers included times they were struggling to synthesize across different areas of domain knowledge to complete a task, or described a lack of go-to strategies for resolving errors in their programs. A focus on supporting students in these areas may allow them to see connections between the work and their own goals, rather than being preoccupied with not knowing how to complete given work.

4.8.1 Provide Sufficient Scaffolding for Domain Knowledge

Across participants, the integration of mathematics and other domain knowledge into programming was largely described in unsupportive terms. One explanation for this is that instructors simply overestimated student's prior knowledge or assumed they were picking up the math concepts from prerequisite or corequisite courses. Another explanation comes from cognitive load theory (Paas, Renkl, and Sweller 2003) which would suggest that even if students understood the math in isolation, or in their math/physics classes, because programming was a new skill and learning it took a certain amount of cognitive load, it was a struggle to then integrate additional knowledge into this work.

> we were given handwritten notes as to how to do drone performance calculations. A lot of the units didn't align with each other, so it was very complicated to make a program that did work. When we ran it, we would get the general shape using the values without even bothering about units in the program, but we had no idea if it actually worked. So in the end, when we made the drone, we didn't really base it on any predictions – Participant 3

In some cases the cognitive load wasn't associated with other engineering or science concepts, but simply the technical aspect of working with MATLAB:

> I didn't know how to run my code, and they never taught that. Yeah, it's very obvious, but I didn't even know running a code was a thing. That's not a requirement coming into the school of engineering. I didn't know what this window was. I didn't know what a comment was. Yeah, we learned what a comment was, but it was more the syntax we didn't really ... I didn't know what these were, up here. The whole program was very confusing. – Participant 2

In this excerpt, the participant expresses a frustration with not understanding how to use the MATLAB interface to run their code. The program interface is an often overlooked source of cognitive load, and different interfaces have been shown to have different levels of cognitive load associated with them (Uysal 2015). While an intent of this study was to direct focus to this material interface, the analysis unfolded in a different direction, focusing instead on participant interpretation of experience. The above excerpt demonstrates that there is indeed rich data available in the transcripts that would motivate the use of sociomateriality as an analytic lens, and this is a suggested direction for future work.

It is important that instructors scaffold potential domain knowledge to help students focus their efforts on just understanding the programming aspect. One way to do this in programming education is to break a complex problem into a number of sub-goals, each with worked examples (Margulieux, Catrambone, and Guzdial 2016). Participant 5 describes how an in-class activity that demonstrated how to find the standard deviation in MATLAB helped her complete a MATLAB assignment:

> We had an assignment we had to do where we had to find the standard deviation of the speed amongst all the boats, and in class we did an assignment where you had to find the standard deviation of, I think it was Olympic medals won between countries. I thought that was really cool because it showed you a really, really

> simple way of how to do the math behind it, so when you did the assignment on your own that was harder, you at least knew how do the math. I really like that. I think it really made it a lot easier to write, and a lot easier to understand. – Participant 5

In this example, knowing she already had a demonstration of how to find the standard deviation using MATLAB, Participant 5 perceived the homework assignment as a productive experience. Because she knew how to "do the math", she was able to focus her efforts on the content of the activity, analyzing speeds of boats.

4.8.2 Articulate and Model Problem Solving Strategies

There was a general lack of instruction surrounding how to check a program for correctness, or debugging strategies to use when it was determined a program was not working as expected. This is also an area in need of future work as often times participants would both claim there was no instruction in debugging, but then later describe some strategies that sounded like, or could be used as debugging strategies. In some cases, students were able to make headway by seeking assistance from their peers:

> My professor helped by just saying that it was too complicated and that I could make it simpler. That didn't really help me figure out what to do, but I just knew that I had to change it. So then I asked my classmates and they were like, I did this and I used this while-loop where I did an if statement and I was like, "I guess that makes sense." – Participant 16

In this excerpt the participant describes their professor "helping" but not knowing how to proceed. Instructors should spend more time helping students develop solid strategies for identifying problems with their code and fixing them, or as in this case, understanding

what is meant by "complicated" and strategies for simplifying code. One method that could aid in this is to reorient the instruction to focus on *patterns* instead of *syntax* (Muller, Haberman, and Averbuch 2004). In this method, instruction focuses on identifying patterns of solutions that can be applied to different types of problems. These patterns could be code or algorithm patterns themselves, but they may also be patterns for diagnosing problems, such as using trace code to identify intermediate values generated by a program, or patterns for testing multiple inputs against generated outputs. There is evidence that by helping students to more readily identify patterns, and apply solutions, they will be better equip to view programming as a tool that can be applied to new problems they have yet to see (Eckerdal, Thuné, and Berglund 2005).

4.8.3 Curriculum Design

There was quite a lot of variation across instructors and classes, especially with regard to how programming concepts are connected to the practice of solving problems. This is compounded by the variation induced by the fact that MATLAB is taught both in the context of data analysis and understanding, as well as general purpose programming or algorithmic thinking. None of the participants articulated a conceptual difference between the two, suggesting they are experiencing it all as "MATLAB" without necessarily forming different strategies for the different domains and problems. Course designers should reflect on what the learning objectives are for the course, and evaluate whether each of the activities aligns with those objectives.

4.9 Limitations and Future Work

While it was unclear whether students who had not intended on majoring in computer science would be swayed by any of the experiences described by participants in this study, participants who took the introductory CS programming course were able to compare and contrast

their experiences in the two courses. While they did not necessarily describe similar applications of programming, they did appear to talk about the two programming environments– Java with one of several possible IDEs used in the CS course, and MATLAB used in the introductory engineering course as comparable tools. One participant even commented that "MATLAB's custom IDE is surprisingly faster than Greenfoot". What participants did not express was any evidence that they viewed Java and MATLAB as two different tools, each designed to tackle different types of problems. Future studies should explore this phenomenon further, as well as collect data about instructors beliefs and intents regarding MATLAB as a tool.

There is also need to better understand the types of programming activities students can succeed at after completing the course. While all participants brought examples of assignments they had completed to the interview, some struggled to articulate how they had completed it, or what the focus of the assignment was. That is, it is unclear what skills, if any, students retained from their experiences and are able to apply to new situations. Future work could invite students who recently completed the course to be observed while working on appropriately scoped MATLAB problems. Identifying differences between what students were doing, how they talked about what they were doing, and instructors perceptions of what they were teaching would be important information to help adjust instruction to better match learning goals.

It is clear from many of the excerpts that there is more nuance in participants' experiences than can be expressed in the four-quadrant outcome space in Figure 4.1. Indeed, the interview data collected for this study contained much rich description which the chosen method of phenomenography tended to background in the process of grouping experiences into a small number of categories. While it is limited in its granularity, the outcome space described here does provide a convenient scaffold for conducting additional analysis. One possibility is us-

ing thematic analysis around the dimensions of perceived instructor scaffolding and affective experience.

Related to the last point, another dimension that was evident in the data but was obfuscated by the chosen analysis technique was the social interaction between students in the course, as well other students in the FYE program. In particular, perceptions and interaction between students with different levels of programming experience and interest likely play a significant role in how students interpret their experiences as this has been shown to be a factor in disciplinary programming courses (Lewis, Yasuhara, and Anderson 2011). The use of IBM as a framework could be expanded to analyze the data with a focus on how references to peers may impact perceptions of the experiences participants describe.

References

Åkerlind, G. (2005a). "Learning about Phenomenography: Interviewing, Data Analysis, and the Qualitative Research Program." In: *Doing Developmental Phenomenography*. Ed. by John A. Bowden and Pam Green. Vol. 19. Melbourne: RMIT University Press, pp. 63–73 (cit. on pp. 83, 89, 90, 92).

Åkerlind, Gerlese S. (2005b). "Variation and Commonality in Phenomenographic Research Methods." In: *Higher Education Research & Development* 24.4, pp. 321–334 (cit. on p. 93).

Ashworth, Peter and Ursula Lucas (2000). "Achieving Empathy and Engagement: A Practical Approach to the Design, Conduct and Reporting of Phenomenographic Research." In: *Studies in Higher Education* 25.3, pp. 295–308. DOI: 10.1080/713696153 (cit. on p. 93).

Azemi, Asad and Laura L. Pauley (2008). "Teaching the Introductory Computer Programming Course for Engineers Using Matlab." In: *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*. IEEE, T3B–1 (cit. on pp. 80–82).

Berglund, Anders (2005). "Learning Computer Systems in a Distributed Project Course: The What, Why, How and Where." Dissertation. Acta Universitatis Upsaliensis (cit. on p. 86).

Booth, Shirley (1992). *Learning to Program: A Phenomenographic Perspective* (cit. on p. 83).

Brannan, Kenneth and Phillip Wankat (2005). "Survey Of First Year Programs." In: 2005 Annual Conference, pp. 10.1188.1–10.1188.23 (cit. on p. 79).

Bruce, Christine S. and Camille A. McMahon (2002). "Contemporary Developments in Teaching and Learning Introductory Programming: Towards a Research Proposal." In: *Faculty of Information Technology, QUT Teaching & Learning Report* 2 (cit. on p. 81).

Bruce, Christine, Lawrence Buckingham, et al. (2004). "Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University." In: *Journal of Information Technology Education: Research* 3, pp. 143–160 (cit. on p. 83).

Case, Jennifer M. and Gregory Light (2011). "Emerging Methodologies in Engineering Education Research." In: *Journal of Engineering Education* 100.1, p. 186 (cit. on pp. 83, 88).

Collier-Reed, Brandon I., Åke Ingerman, and Anders Berglund (2009). "Reflections on Trustworthiness in Phenomenographic Research: Recognising Purpose, Context and Change in the Process of Research." In: *Education as change* 13.2, pp. 339–355 (cit. on pp. 92, 93).

CSForAll Consortium, ConCCon (2017). *CSforAll Consortium*. URL: http://www.csforall.org/about/ (visited on 2017) (cit. on p. 79).

Eckerdal, Anna and Michael Thuné (2005). "Novice Java Programmers' Conceptions of Object and Class, and Variation Theory." In: *ACM SIGCSE Bulletin*. Vol. 37. ACM, pp. 89–93 (cit. on p. 83).

Eckerdal, Anna, Michael Thuné, and Anders Berglund (2005). "What Does It Take to Learn 'Programming Thinking'?" In: *Proceedings of the First International Workshop on Computing Education Research*. ICER '05. New York, NY, USA: ACM, pp. 135–142. DOI: 10.1145/1089786.1089799 (cit. on p. 113).

Elmore, Kristen et al. (2016). "When the Going Gets Tough: Implications of Reactance for Interpretations of Experienced Difficulty in the Classroom." In: *AERA Open* 2.3, p. 233285841666471. DOI: 10.1177/2332858416664714 (cit. on p. 109).

Forte, Andrea and Mark Guzdial (2005). "Motivation and Nonmajors in Computer Science: Identifying Discrete Audiences for Introductory Courses." In: *Education, IEEE Transactions on* 48.2, pp. 248–253 (cit. on pp. 84, 85).

Hamrick, Todd R. and Robin A. M. Hensel (2013). "Putting the Fun in Programming Fundamentals - Robots Make Programs Tangible." In: 2013 ASEE Annual Conference & Exposition, pp. 23.1012.1–23.1012.8 (cit. on p. 82).

Hankley, William (2004). "Software Engineering Emphasis For Computing Courses." In: 2004 Annual Conference, pp. 9.1105.1–9.1105.7 (cit. on pp. 79, 84).

Hrynuk, J. et al. (2008). "Freshman Engineering: An Introductory Computer Course Teaching MATLAB and LABVIEW." In: *Proceedings of the 2008 American Society for Engineering Education Annual Conference and Exposition* (cit. on p. 82).

Hsu, Stanley W., Rajeevan Amirtharajah, and Andre Knoesen (2013). "Lab and Team Project Development for Engineering Problem Solving Using MATLAB, with Emphasis on Solar Power and Engineering for Sustainability." In: 2013 ASEE Annual Conference & Exposition, pp. 23.841.1–23.841.14 (cit. on p. 82).

Johri, Aditya (2011). "The Socio-Materiality of Learning Practices and Implications for the Field of Learning Technology." In: *Research in Learning Technology* 19.3, pp. 207–217. DOI: 10.1080/21567069.2011.624169 (cit. on p. 86).

Lehr, Steven and Christopher Grant (2007). "Utilizing Programming Projects In A Freshmen Programming Course." In: 2007 Annual Conference & Exposition, pp. 12.1579.1–12.1579.13 (cit. on p. 82).

Lewis, Colleen M., Ken Yasuhara, and Ruth E. Anderson (2011). "Deciding to Major in Computer Science: A Grounded Theory of Students' Self-Assessment of Ability." In: *Proceedings of the Seventh International*

Workshop on Computing Education Research. ICER '11. New York, NY, USA: ACM, pp. 3–10. DOI: 10.1145/2016911.2016915 (cit. on pp. 87, 88, 115).

Lönngren, Johanna (2017). "Wicked Problems in Engineering Education: Preparing Future Engineers to Work for Sustainability." Doctoral thesis. Chalmers University of Technology (cit. on p. 83).

M, G. M. Sarria (2009). "Introduction to Programming for Engineers Following the Parachute Paradigm." In: *2009 39th IEEE Frontiers in Education Conference*. 2009 39th IEEE Frontiers in Education Conference, pp. 1–4. DOI: 10.1109/FIE.2009.5350472 (cit. on p. 82).

Magana, Alejandra J., Michael L. Falk, and Michael J. Reese Jr (2013). "Introducing Discipline-Based Computing in Undergraduate Engineering Education." In: *ACM Transactions on Computing Education (TOCE)* 13.4, p. 16 (cit. on pp. 81, 82, 84).

Magana, Alejandra J., Michael L. Falk, Camilo Vieira, et al. (2016). "A Case Study of Undergraduate Engineering Students' Computational Literacy and Self-Beliefs about Computing in the Context of Authentic Practices." In: *Computers in Human Behavior* 61, pp. 427–442 (cit. on p. 106).

Margulieux, Lauren E., Richard Catrambone, and Mark Guzdial (2016). "Employing Subgoals in Computer Programming Education." In: *Computer Science Education* 26.1, pp. 44–67. DOI: 10.1080/08993408.2016.1144429 (cit. on p. 111).

Marton, Ference and Shirley A. Booth (1997). *Learning and Awareness*. Psychology Press (cit. on p. 86).

Mascaro, Debra J. and Stephen Mascaro (2015). "An Integrated Project-Driven Course in Computer Programming for Mechanical Engineering Students." In: *age* 26, p. 1 (cit. on pp. 81, 82).

McCracken, Michael et al. (2001). "A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students." In: *SIGCSE Bull.* 33.4, pp. 125–180. DOI: 10.1145/572139.572181 (cit. on p. 81).

Mobasher, Amir et al. (2002). "Incorporating Matlab In Mechanical Engineering Courses at Alabama A&M University." In: *Proceedings of the American Society for Engineering Education Annual Conference & Exposition*. 2002 Annual Conference, pp. 7.655.1–7.655.9 (cit. on p. 81).

Morrell, Darryl (2007). "Design Of An Introductory Matlab Course For Freshman Engineering Students." In: 2007 Annual Conference & Exposition, pp. 12.458.1–12.458.7 (cit. on p. 82).

Muller, Orna, Bruria Haberman, and Haim Averbuch (2004). "(An Almost) Pedagogical Pattern for Pattern-Based Problem-Solving Instruction." In: *ACM SIGCSE Bulletin*. Vol. 36. ACM, pp. 102–106 (cit. on p. 113).

Nagurney, L. S. (2001). "Teaching Introductory Programming for Engineers in an Interactive Classroom." In: *31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193)*. 31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education. Conference Proceedings (Cat. No.01CH37193). Vol. 3, S2C-1–5 vol.3. DOI: 10.1109/FIE.2001.964019 (cit. on p. 82).

Oyserman, Daphna (2013). "Not Just Any Path: Implications of Identity-Based Motivation for Disparities in School Outcomes." In: *Economics of Education Review*. Assets and Educational Attainment: Theory and Evidence 33, pp. 179–190. DOI: 10.1016/j.econedurev.2012.09.002 (cit. on p. 109).

Oyserman, Daphna et al. (2017). "An Identity-Based Motivation Framework for Self-Regulation." In: *Psychological Inquiry* 28.2-3, pp. 139–147. DOI: 10.1080/1047840X.2017.1337406 (cit. on pp. 87, 105).

Paas, Fred, Alexander Renkl, and John Sweller (2003). "Cognitive Load Theory and Instructional Design: Recent Developments." In: *Educational psychologist* 38.1, pp. 1–4 (cit. on p. 110).

Patton, Michael Q. (2002). "Qualitative Interviewing." In: *Qualitative Research and Evaluation Methods*. Thousand Oaks, Calif: Sage Publications (cit. on p. 89).

Pears, Arnold et al. (2007). "A Survey of Literature on the Teaching of Introductory Programming." In: *ACM SIGCSE Bulletin* 39.4, pp. 204–223 (cit. on p. 81).

Reid, Kenneth and David Reeping (2014). "A Classification Scheme for "Introduction to Engineering" Courses: Defining First-Year Courses Based on Descriptions, Outcomes, and Assessment." In: *American Society for Engineering Education Annual Conference & Exposition. Indianapolis, IN (1-11). Washington DC: American Society for Engineering Education* (cit. on p. 79).

Robins, Anthony, Janet Rountree, and Nathan Rountree (2003). "Learning and Teaching Programming: A Review and Discussion." In: *Computer Science Education* 13.2, pp. 137–172 (cit. on p. 81).

Rosca, Raluca (2006). "Learning Matlab Just In Time Or Freshman Year?" In: 2006 Annual Conference & Exposition, pp. 11.876.1–11.876.12 (cit. on p. 81).

Sandelowski, Margarete (1995). "Sample Size in Qualitative Research." In: *Research in Nursing & Health* 18.2, pp. 179–183. DOI: 10.1002/nur.4770180211 (cit. on p. 88).

Sleeman, Derek (1986). "The Challenges of Teaching Computer Programming." In: *Commun. ACM* 29.9, pp. 840–841. DOI: 10.1145/6592.214913 (cit. on p. 80).

Thuné, Michael and Anna Eckerdal (2009). "Variation Theory Applied to Students' Conceptions of Computer Programming." In: *European Journal of Engineering Education* 34.4, pp. 339–347. DOI: 10.1080/03043790902989374 (cit. on p. 83).

Tufford, Lea and Peter Newman (2010). "Bracketing in Qualitative Research." In: *Qualitative Social Work* 11.1, pp. 80–96 (cit. on p. 93).

Uysal, Murat Pasa (2015). "Evaluation of Learning Environments for Object-Oriented Programming: Measuring Cognitive Load with a Novel Measurement Technique." In: *Interactive Learning Environments* 0.0, pp. 1–20. DOI: 10.1080/10494820.2015.1041400 (cit. on p. 111).

Chapter 5: Discussion

The purpose of this work was to understand the how a general first-year engineering program may affect decisions to major in computer science, with a particular focus on under-served populations. The work identified and explored possible leverage points that may be used to direct efforts to broaden participation in computer science. Using both quantitative and qualitative techniques, I conducted three studies that examined different aspects of the curriculum related to computer science at a single large research institution in the United States with a common first-year engineering program. Manuscript 1 and 2 utilized a common data set, population data for the college of engineering at the institution over the years 2009-2018. Manuscript 3 used data from interviews conducted with students of the first-year engineering program at this institution.

Manuscript 1 explored patterns of representation in computer science from before the start of the first-year, through graduation. This provides a high-level view of how different populations progress through the different decision points: who matriculates, who persists, and who graduates with a CS degree. Results indicate that representation changes for women, URM, and first-generation students in CS after the common FYE program: the representation of women declines after the FYE program, while representation of URM and first-generation students increases during this time. Limitations of the data available preclude making causal claims regarding these changes, however, and future work is needed to determine if and how the FYE program is driving these changes.

Manuscript 2 focuses on the introductory programming courses CS1 and CS2 and how performance in those courses is correlated with demographics and CS degree attainment. While a premise of this study was that CS1 and CS2 may be acting as barrier courses, results raise questions of this claim. To the extent that CS1 *may* be acting as a barrier course, results

indicate it is not any more a barrier to women, URM, or first-generation students. In fact women who complete CS1 are more likely to matriculate into CS than men who complete CS1. URM and first-generation students who take CS1 also have a slightly elevated likelihood of matriculating into CS compared to their peers. An important factor in this study that was not explored is the impact of enrollment management policies in the FYE program at LRI. Because only students who earn at least a 3.0 GPA are guaranteed their first choice of engineering major, the competitiveness of the CS major and this GPA policy may have a significant impact on who matriculates, and could explain why many students who perform well in CS1, regardless of demographics, are not matriculating into CS.

Manuscript 3 turns the focus to the first-year general engineering course which all engineering students take, and the experiences students have of the programming instruction in that course. The common curriculum of the FYE program offers an opportunity to provide early positive programming experiences to students who otherwise would not have them, potentially opening a path to computer science for historically under-served populations. Results suggests that these varied experiences may be productive for some students and unproductive for others. The use of MATLAB complicates this opportunity, as that platform is not associated with computer science and so it is up to individual students' to not only separate out their experience with programming from the tool itself, but also to make connections between the programming experience as a possible link to an interest in computer science.

5.1 Implications

5.1.1 Implications for Practice

With regard to women in computer science, the results of Manuscript 1 are consistent with national data that indicates women are underrepresented in the field (NCSE 2018), and that the representation of women in computer science is well bellow the college average.

Further, the representation of women declined at each successive decision point, from intent-to-major, to matriculation into the major, to graduating. Time series-analysis suggests that the risk of women leaving computer science increases with the time in the program more-so than men. This would support the explanation that women are leaving for non-academic reasons, such as perceptions of poor fit (Lewis, Anderson, and Yasuhara 2016). One way to focus efforts to adjust the culture of computer science at Large Research Institution is to emphasize the role computer science plays in the world, such as by including more real-world examples in coursework (Benbow and Vivyan 2016), and provide more opportunities for student-student interaction (Barker, McDowell, and Kalahar 2009). This is particularly important in the middle years, as this tends to be a time in the curriculum in which courses are highly technical and disconnected from one another (Lord and Chen 2014).

While it may be tempting to view the timing of the drop in persistence after the FYE program as placing any explanation for the pattern on the role of the CS department or curriculum itself, effects of the FYE program on this decline cannot be ruled out. One reason many women leave computer science is losing interest in the field (Biggers, Brauer, and Yilmaz 2008). While loss of interest could be due to a number of factors, one is a possible misunderstanding of what computer science is, or the type of work it entails, prior to beginning the major. Since one of the rules of the FYE program is to provide information about each engineering major, a loss of interest in CS after making the decision to matriculate could be due to a misrepresentation of the field by the FYE program. While more work is needed to determine if the explanation does lie with the FYE program, a practical implication is that administrators and advisors of the FYE program should review material students have access to learn about the different engineering majors and corroborate them with disciplinary representatives to determine how well they align with what students are likely to experience once they leave the FYE program for a disciplinary major.

If instead the explanation for the drop in persistence of women lies in the discipline itself, then one place to begin efforts to improve retention are in CS1 and CS2. While none of the predictors modeled were strong indicators of course performance, there was still evidence that women are more likely to struggle with these courses than men. The model validated by Suresh (2006) indicates several factors that influence performance in barrier courses, such as CS1 and CS2:

1. Student behaviors such a study habits,
2. Perceived Faculty Behaviors such as the promotion of a "weed out" culture,
3. Perceived culture of support, such as easy access to academic advisors, and knowing where to turn for help when struggling with coursework

Ultimately, initiatives to improve course climate must be tailored to the local environment (Benbow and Vivyan 2016), so work must first be done to understand that particular context at LRI. One important contextual component is how disciplines are represented to students, and how students are advised during the FYE program.

With respect to the FYE program itself, the same factors of student behaviors, perceived faculty behaviors, and perceived culture of support are likely at play. During the FYE program however there is an additional complexity in understanding these factors because they may interact with the factors present in the disciplinary majors. For example, a student may perceive their FYE faculty as being supportive and encouraging of their motivation to pursue computer science, but upon entering the discipline may develop quite different perceptions of the disciplinary faculty.

In the context of early programming experience in the FYE program, student perceptions of struggling with the work often centered around integrating unfamiliar concepts simul-

taneously with learning programming concepts. This suggests efforts to reduce and focus the number of different topics in the course are taken, and more scaffolding is provided for these topics. Both of these strategies are considered good pedagogical practices in general (Halpern and Hakel 2003) and have particular importance in students' early experiences as they are weighing their options regarding disciplinary major.

There was evidence that many students *are* perceiving sufficient scaffolding to both integrate different topics and learning the programming, but a challenge with this course is to provide a consistent experience across many different instructors, and students with varied prior experience with programming. The use of hands-on projects and opportunities to work with personally meaningful data, in other word "authentic learning" (Magana et al. 2016), were generally seen as productive experiences by students. It is important that these opportunities exist across all course sections, and if they do not, then there is a need to understand why that is. It is certainly the case that supporting hands-on activities and allowing students to work with their own data requires different instructional strategies than a more lecture style, and by-the-book curriculum, and so it is possible instructors do not have the resources or confidence they need to effectively support these activities.

Additionally, the variability in which students perceived sufficient scaffolding associated with MATLAB itself was concerning. Some participants described sufficient instruction regarding the MATLAB interface, how to load and save files, and how to make use of the major components of the environment, while other participants described a lack of instruction about what a script file was, or even how to run code that they had written. Understanding and using the tool itself is a source of cognitive load in addition to learning new programming concepts (Uysal 2015). It is important to manage cognitive load to minimize the load that is extraneous to the learning objectives of the course (Paas, Renkl, and Sweller 2003). How much scaffolding, and how much of the environment should be introduced remains an open

question and there is likely no one-size-fits-all answer. For example, for most students new to programming, knowing about breakpoints and how to use them is likely beyond the scope of the general engineering program, but at least one participant who was familiar with the use of breakpoints from other experiences expressed dissatisfaction with MATLAB because he believed it did not support breakpoints, when in fact it does. One approach may be to plan instruction around a scoped set of tools that would be sufficient to complete the work in the course, but provide a written guide "MATLAB for X Users"–where "X" is the editing environment used in CS1 or CS2, and make it available to those students who want it.

Finally, while several participants described connections between the programming work they completed in the FYE program and the field of engineering as a whole, only one mentioned connections to computer science specifically. Another participant described how a particular instructor would talk about different applications of programming in various engineering disciplines each class, but not computer science. The use of MATLAB does not lend itself to making connections to computer science as it is not a tool typically used in that discipline. While it would not make sense to foreground computer science applications more than others in a general engineering course, the course coordinators may want to review which disciplines are gaining exposure and which are not, and possibly make adjustments. While it would be impossible to provide a full authentic experience for each of the 15 engineering majors at LRI, this is an area that could benefit from flexibility in datasets students are asked to analyze, or even banks of assignments and examples targeted at different disciplines that students could choose from.

Programming has long been included as part of engineering programs (Azemi and Pauley 2008; Kramer 1994; Reid and Reeping 2014), but little work exists that explore the challenges in adopting good discipline-based approach to teaching programming that would work in a common first-year program and also provide an experience that attracts a more diverse group

to computer science. What role does the common first-year curriculum play? What are the implications of requiring CS1 in a student's second semester, and does the experience in the common first-year have a role to play in preparing women who do choose CS to persist and ultimately graduate? While the results discussed provide a foundation for understanding how a FYE program may impact decisions to major in CS, more work is needed to fully answer these questions.

5.1.2 Implications for Research

The findings in this work support other calls for more discipline-focused work to understand representation challenges rather than taking STEM to be a monolithic culture (Benbow and Vivyan 2016). The first-generation and URM CS students in this study exhibited patterns that did not align with the broader patterns that have been observed for these groups in STEM as a whole. Some factors that are known to impact these groups are levels of family involvement, financial concerns, and a strong social network. The and other factors are likely to interact in complex ways with the variety of disciplinary and departmental cultures across within the college of engineering. For example, perhaps the cultural mismatch that is theorized to affect first-generation college students in engineering (Stephens et al. 2012) is less of a factor in computer science. Additionally, race, gender, and SES will likely interact with first-generation status in different ways (Dennis, Phinney, and Chuateco 2005), so researchers should seek out large, multi-institutional datasets that allow for exploring interactions of these variables as the sample size is often too small when restricted to a single institution and degree program.

Data-sets such as the Multiple-Institution Database for Investigating Engineering Longitudinal Development (MIDFIELD) could be leveraged for some of this work, though MIDFIELD does not contain first-generation information. Given the results concerning first-generation

students found in this work, it is important that continued efforts to collect and curate data for engineering education research contains this information if possible.

Finally, while methods such as logistic regression and multinomial regression are becoming more widespread in education research, event history analysis remains somewhat limited. Part of this may be due to the relative lack of longitudinal time-series data that would benefit from this analysis, but as access to longitudinal data increases, EHA is a useful tool at researchers disposal. While the use of EHA in this work was fairly limited, at most comparing event probability curves between groups, the technique has the capacity for answering more complex time-based questions such as the effects of working some terms and not others, or gaps in enrollment might have on dropout and time-to-degree events (Scott and Kennedy 2005). This analysis would be particularly important when considering the trajectories of low-SES first-generation students as these populations may be more likely to work while enrolled (Fernandez et al. 2008; Ishitani 2006).

5.2 Future Work

The results of these three studies invite opportunities for future work:

- Conduct similar analysis on a multi-institutional data-set to determine if the patterns for first-generation and URM students are specific to LRI or exist in other computer science environments. This would then inform possible explanations for this pattern compared to first-generation and URM students in engineering as a whole.

- Investigate additional factors that may impact performance in CS1 and CS2, such as classroom climate, student behaviors, and perceptions of faculty support. The role of faculty representation is of particular interest since representation of women in the CS faculty increased during the time of the study.

- Investigate other components of the FYE program, and their interaction. In particular, event programming, first-year advising and enrollment management all play an important role in guiding students' major choices, but data regarding these components were not included in the current analysis.

- Conduct interviews with first-generation students in the general engineering program to better understand how this population may be experiencing their first year of college in a common first-year program.

- Many participants had a difficult time recalling and articulating particular strategies they used to complete their programming work. The work to understand perceptions of instruction could be complemented by conducting observations of participants working on programming homework assigned for the general engineering course to better understand what they are doing to complete the work. Differences between what they are doing, what they say they are doing, and what they say instructors are teaching would be valuable information to help guide improvement to instruction in the first-year program.

References

Azemi, Asad and Laura L. Pauley (2008). "Teaching the Introductory Computer Programming Course for Engineers Using Matlab." In: *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*. IEEE, T3B–1 (cit. on p. 126).

Barker, Lecia J., Charlie McDowell, and Kimberly Kalahar (2009). "Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major." In: *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (Chattanooga, TN, USA). SIGCSE '09. New York, NY, USA: ACM, pp. 153–157. DOI: 10.1145/1508865.1508923 (cit. on p. 123).

Benbow, Ross J. and Erika Vivyan (2016). *Gender and Belonging in Undergraduate Computer Science: A Comparative Case Study of Student Experiences in Gateway Courses. WCER Working Paper No. 2016-2*. Wisconsin Center for Education Research (cit. on pp. 123, 124, 127).

Biggers, Maureen, Anne Brauer, and Tuba Yilmaz (2008). "Student Perceptions of Computer Science: A Retention Study Comparing Graduating Seniors with Cs Leavers." In: *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. Vol. 40. SIGCSE '08. New York, NY, USA: ACM, pp. 402–406. DOI: 10.1145/1352135.1352274 (cit. on p. 123).

Dennis, Jessica M., Jean S. Phinney, and Lizette Ivy Chuateco (2005). "The Role of Motivation, Parental Support, and Peer Support in the Academic Success of Ethnic Minority First-Generation College Students." In: *Journal of college student development* 46.3, pp. 223–236 (cit. on p. 127).

Fernandez, Michael J. et al. (2008). "First Generation College Students in Engineering: A Qualitative Investigation of Barriers to Academic Plans." In: *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*. IEEE, T4D–1 (cit. on p. 128).

Halpern, Diane F. and Milton D. Hakel (2003). "Applying the Science of Learning to the University and beyond: Teaching for Long-Term Retention and Transfer." In: *Change: The Magazine of Higher Learning* 35.4, pp. 36–41 (cit. on p. 125).

Ishitani, Terry T. (2006). "Studying Attrition and Degree Completion Behavior among First-Generation College Students in the United States." In: *The Journal of Higher Education* 77.5, pp. 861–885 (cit. on p. 128).

Kramer, K. A. (1994). "Using C Programming as a Vehicle to Overcome Barriers." In: *Proceedings of 1994 IEEE Frontiers in Education Conference - FIE '94*. Proceedings of 1994 IEEE Frontiers in Education Conference - FIE '94, pp. 30–33. DOI: 10.1109/FIE.1994.580463 (cit. on p. 126).

Lewis, Colleen M., Ruth E. Anderson, and Ken Yasuhara (2016). ""I Don'T Code All Day": Fitting in Computer Science When the Stereotypes Don'T Fit." In: *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ICER '16. New York, NY, USA: ACM, pp. 23–32. DOI: 10.1145/2960310.2960332 (cit. on p. 123).

Lord, Susan M. and John C. Chen (2014). "Curriculum Design in the Middle Years." In: *Cambridge Handbook of Engineering Education Research*. New York, NY: Cambridge University Press, pp. 181–195 (cit. on p. 123).

Magana, Alejandra J. et al. (2016). "A Case Study of Undergraduate Engineering Students' Computational Literacy and Self-Beliefs about Computing in the Context of Authentic Practices." In: *Computers in Human Behavior* 61, pp. 427–442 (cit. on p. 125).

NCSE (2018). *Women, Minorities, and Persons with Disabilities in Science and Engineering*. National Center for Science and Engineering Statistics, NSF (cit. on p. 122).

Paas, Fred, Alexander Renkl, and John Sweller (2003). "Cognitive Load Theory and Instructional Design: Recent Developments." In: *Educational psychologist* 38.1, pp. 1–4 (cit. on p. 125).

Reid, Kenneth and David Reeping (2014). "A Classification Scheme for "Introduction to Engineering" Courses: Defining First-Year Courses Based on Descriptions, Outcomes, and Assessment." In: *American Society for Engineering Education Annual Conference & Exposition. Indianapolis, IN (1-11). Washington DC: American Society for Engineering Education* (cit. on p. 126).

Scott, Marc A. and Benjamin B. Kennedy (2005). "Pitfalls in Pathways: Some Perspectives on Competing Risks Event History Analysis in Education Research." In: *Journal of Educational and Behavioral Statistics* 30.4, pp. 413–442 (cit. on p. 128).

Stephens, Nicole M. et al. (2012). "Unseen Disadvantage: How American Universities' Focus on Independence Undermines the Academic Performance of First-Generation College Students." In: *Journal of personality and social psychology* 102.6, p. 1178 (cit. on p. 127).

Suresh, Radhika (2006). "The Relationship between Barrier Courses and Persistence in Engineering." In: *Journal of College Student Retention: Research, Theory & Practice* 8.2, pp. 215–239. DOI: 10.2190/3QTU-6EEL-HQHF-XYF0 (cit. on p. 124).

Uysal, Murat Pasa (2015). "Evaluation of Learning Environments for Object-Oriented Programming: Measuring Cognitive Load with a Novel Measurement Technique." In: *Interactive Learning Environments* 0.0, pp. 1–20. DOI: 10.1080/10494820.2015.1041400 (cit. on p. 125).

Appendices

Appendix A: Site Demographics

A.1 College of Engineering Demographics

Tables A.1, A.2, and A.3 show demographic information for all students entering the COE between Fall 2009 and Fall 2018.

Table A.1: URM and First-Generation Student Representation in the COE

		Continuing-Gen.	First-Gen.	Total
Non-URM	N	15063	2301	17364
	Row %	86.7%	13.3%	90.2%
	Column %	92.0%	79.6%	
	Table %	78.2%	12.0%	
URM	N	1302	588	1890
	Row %	68.9%	31.1%	9.8%
	Column %	8.0%	20.4%	
	Table %	6.8%	3.1%	
Total	N	16365	2889	19254
	%	85%	15%	

Table A.2: URM Student Representation by Gender in the COE

		Male	Female	Total
Non-URM	N	13847	3517	17364
	Row %	79.7%	20.3%	90.2%
	Column %	90.5%	89.1%	
	Table %	71.9%	18.3%	
URM	N	1460	430	1890
	Row %	77.2%	22.8%	9.8%
	Column %	9.5%	10.9%	
	Table %	7.6%	2.2%	
Total	N	15307	3947	19254
	%	79.5%	20.5%	

Table A.3: First-Generation Student Representation by Gender in the COE

		Male	Female	Total
Continuing-Generation	N	12949	3416	16365
	Row %	79.1%	20.9%	85.0%
	Column %	84.6%	86.5%	
	Table %	67.3%	17.7%	
First-Generation	N	2358	531	2889
	Row %	81.6%	18.4%	15.0%
	Column %	15.4%	13.5%	
	Table %	12.2%	2.8%	
Total	N	15307	3947	19254
	%	79.5%	20.5%	

A.2 Computer Science Demographics

Tables A.4, A.5, and A.6 show demographic information for all students entering the COE between Fall 2009 and Fall 2018 and who's first matriculation major was CS.

Table A.4: URM and First-Generation Student Representation in CS

		Continuing-Gen.	First-Gen.	Total
Non-URM	N	1361	233	1594
	Row %	85.4%	14.6%	90.8%
	Column %	92.3%	82.9%	
	Table %	77.5%	13.3%	
URM	N	113	48	161
	Row %	70.2%	29.8%	9.2%
	Column %	7.7%	17.1%	
	Table %	6.4%	2.7%	
Total	N	1474	281	1755
	%	84%	16%	

Table A.5: URM Student Representation by Gender in CS

		Male	Female	Total
Non-URM	N	1346	248	1594
	Row %	84.4%	15.6%	90.8%
	Column %	91.1%	89.5%	
	Table %	76.7%	14.1%	
URM	N	132	29	161
	Row %	82.0%	18.0%	9.2%
	Column %	8.9%	10.5%	
	Table %	7.5%	1.7%	
Total	N	1478	277	1755
	%	84.2%	15.8%	

Table A.6: First-Generation Student Representation by Gender in CS

		Male	Female	Total
Continuing-Gen.	N	1238	236	1474
	Row %	84.0%	16.0%	84.0%
	Column %	83.8%	85.2%	
	Table %	70.5%	13.4%	
First-Gen.	N	240	41	281
	Row %	85.4%	14.6%	16.0%
	Column %	16.2%	14.8%	
	Table %	13.7%	2.3%	
Total	N	1478	277	1755
	%	84.2%	15.8%	

Appendix B: Patterns of Participation in Computer Science

Table B.1: Representation of female, URM, and first-generation students in Computer Science compared with the College of Engineering.

variable	program	Incoming	Declared	Degree
Female	COE	1911 (22.04%)	1890 (21.92%)	953 (20.68%)
Female	CS	95 (18.13%)	201 (16.83%)	127 (13.42%)
First-Generation	COE	1231 (14.2%)	1222 (14.17%)	707 (15.34%)
First-Generation	CS	58 (11.07%)	187 (15.66%)	152 (16.07%)
URM	COE	877 (10.11%)	874 (10.14%)	462 (10.03%)
URM	CS	39 (7.44%)	107 (8.96%)	89 (9.41%)