Computer Science at Community Colleges: Attitudes and Trends

Trenton W. Dawson

Louisiana State University and Agricultural and Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_dissertations

Part of the Curriculum and Instruction Commons

Recommended Citation
https://digitalcommons.lsu.edu/gradschool_dissertations/5822

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Doctoral Dissertations by an authorized graduate school editor of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
COMPUTER SCIENCE AT COMMUNITY COLLEGES:
ATTITUDES AND TRENDS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The School of Education

by

Trenton W. Dawson
B.B.A., University of Oklahoma, 2002
M.Ed., Virginia Tech, 2017
May 2022
ACKNOWLEDGEMENTS

My graduate school journey started at Virginia Tech in Blacksburg, Virginia, in the winter of 2016. It was not only the first step toward an academic career education but symbolic of a positive new direction in my life. Since embarking on this path, there have been many challenges that, looking back, seem to have unwittingly shaped me into the rounded individual that I am today. The support and encouragement of professors, classmates, colleagues, family, and friends enabled me to successfully traverse the academic landscape to attain this once-in-a-lifetime education milestone.

I want to thank Dr. Kerri Tobin, my academic advisor, professor, mentor, and on occasion, counselor, whose advice ameliorated episodes of self-doubt that enabled me to persist through extraordinary challenges. I appreciate your eagle-eye during the editing phases of this dissertation and fearless feedback. You have been an excellent role model of scholastic brilliance paired with great virtues in kindness and compassion.

I want to thank Dr. David Kirshner for his outstanding mentorship and availability to offer me candid advice that compelled me to broaden my outlook and think in novel ways for solving challenging academic problems. Your years of supervision were an instrumental component of my educational development at every level.

I wish to express tremendous gratitude to Dr. Kim Skinner, whose course in sociocultural theory laid the foundation for more profound thought and reflection in learning theory. Your support and encouragement in my first year influenced the trajectory of my coursework that affected my choices for locating a theoretical underpinning for my dissertation.

Without the knowledge and proficiencies, I gained in your methods courses, Dr. Eugene Kennedy, my confidence to opt into a quantitative study would not have been imaginable. I want
to express gratitude to you for encouraging me to endure during an educationally overwhelming moment. I will fondly remember the sympathetic kindness you granted me.

I want to thank my incredible father, Gary Dawson, and stepmother, Carrie Dawson, for their unrelenting prayers and blessings and unwavering love and support. I also want to thank my amazing mother, Tammy Anderson, and stepfather, Tim Anderson, for their uplifting encouragement and never giving up on the idea that I would eventually find my way.

I want to thank my dearest friend, Jeff Mandon, whose thoughtful advice, spiritual guidance, and extensive intelligence helped steer me to resolutions throughout our incalculable hours of phone conversations. I could not have come within grasping distance of this momentous milestone without your loyalty, friendship, and continuous encouragement, and I hope to one day pass on to others what you so generously offered to me.

I want to thank my best friend, Trent Thibodeaux, for his friendship and for being an instrumental person in helping me acclimate to the vibrant Cajun culture. You have been a true friend and enormous supporter of my reaching an educational milestone. Zelda and I will always be grateful to you for your kindness and friendship while I worked to complete this degree. I also want to thank you for offering your amazingly well-lit space in New Iberia, Louisiana, for writing my dissertation.

Finally, I want to thank my siblings, Tennille Whiteman, Cory Anderson, and Tiffany Dawson, for their encouragement; and persistently pushing me to be the most definitive version of myself.
# TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................ iii

LIST OF TABLES ................................................................................................................ vii

LIST OF FIGURES .............................................................................................................. viii

ABSTRACT ........................................................................................................................... ix

CHAPTER ONE. STATEMENT OF THE PROBLEM ........................................................... 1
   Introduction ..................................................................................................................... 1
   Purpose for the Study .................................................................................................... 7
   Research Questions ....................................................................................................... 10
   Theoretical Framework ................................................................................................. 11
   Organization of the Dissertation .................................................................................. 14

CHAPTER TWO. A REVIEW OF THE LITERATURE ..................................................... 17
   Rationale for Broadening Access to Computer Science ............................................. 17
   Federal and State Funding of Computing Programs ................................................ 19
   Computer Science Education History ......................................................................... 19
   Block-based and Textual Programming Languages ................................................... 20
   Equity in Computing Education ................................................................................ 22
   The Computer Scientist Stereotype ............................................................................ 23
   Racial Justice in Computing Education ..................................................................... 24
   Instructional Strategies in Computing Education ....................................................... 25
   Systems of Assessment in Computing Education ..................................................... 27
   Community Colleges’ Role to Diversify the Computing Industry ............................. 32

CHAPTER THREE. METHODOLOGY ........................................................................ 35
   Introduction .................................................................................................................. 35
   Analogous Prior Work ................................................................................................. 35
   Research Questions ..................................................................................................... 38
   Research Methodology and Design ........................................................................... 38
   My Role as a Researcher .............................................................................................. 43
   Quantitative Statistical Methodology ....................................................................... 45
   Data Analysis .............................................................................................................. 48
   Conclusion ................................................................................................................... 54

CHAPTER FOUR. RESULTS ...................................................................................... 55
   Histograms .................................................................................................................... 56
   Analysis of Histogram Data ........................................................................................ 63
   Question One: How do previous programming courses correlate or predict a favorable identity and attitude toward computer science? ......................................................... 64
   Question Two: Do demographics like age, race, and gender predict favorable identity and attitude toward CS? ................................................................................... 70
   Age and Attitude Results ............................................................................................ 71
LIST OF TABLES

2.1. Empirical Studies of Assessment in Computational Thinking Literature .................. 28
2.3. Empirical Studies of Assessment in Computational Thinking Literature .................. 29
3.1. Effect Size Guidelines According to Cohen .......................................................... 53
4.1. Group Mean and SD (Attitude & Programming Experience) .................................... 65
4.2. T-Test (Attitude & Programming Experience) ...................................................... 66
4.3. Cohen’s $\delta$ / Effect Size (Attitude & Programming Experience) .......................... 66
4.4. One-way ANOVA (Attitude & Programming Experience) ..................................... 67
4.5. Descriptive Statistics (Attitude & Age) ................................................................. 72
4.6. One-way ANOVA (Attitude & Age) ................................................................. 73
4.7. Descriptive Statistics (Attitude & Race) ............................................................... 77
4.8. One-way ANOVA (Attitude & Race) ............................................................... 78
4.9. T-Test (Attitude & Gender) ................................................................................. 82
4.10. Group Mean and SD (Attitude & Gender) ............................................................ 82
4.11. Cohen’s $\delta$ / Effect Size (Attitude & Gender) .................................................... 83
4.12. One-way ANOVA (Attitude & Gender) ............................................................ 84
LIST OF FIGURES

3.1. Racial Distribution of Fall 2020 Enrollment in Community Colleges ........................................ 40
3.2. Age Distribution of Fall Enrollment in Community Colleges ....................................................... 40
3.3. Gender Distribution of Fall 2020 Enrollment in Community Colleges ........................................ 40
3.4. Demographic Distribution (Race) ............................................................................................... 47
3.5. Demographic Distribution (Gender) ........................................................................................... 47
3.6. Demographic Distribution (Age) ............................................................................................... 48
3.7. Demographic Distribution (Ethnicity) ....................................................................................... 48
4.1. Question One. Frequency Histogram ......................................................................................... 57
4.2. Question Two. Frequency Histogram ......................................................................................... 58
4.3. Question Three. Frequency Histogram ....................................................................................... 59
4.4. Question Four. Frequency Histogram ......................................................................................... 60
4.5. Question Five. Frequency Histogram ......................................................................................... 61
4.6. Question Six. Frequency Histogram ......................................................................................... 62
4.7. Mean Plot Diagram (Favorable Attitude & Programming Experience) ........................................ 68
4.8. Mean Plot Diagram (Unfavorable Attitude & Programming Experience) .................................... 69
4.9. Mean Plot Diagram (Favorable Attitude & Age) .......................................................................... 74
4.10. Mean Plot Diagram (Unfavorable Attitude & Age) .................................................................. 75
4.11. Mean Plot Diagram (Favorable Attitude & Race) ..................................................................... 80
4.12. Mean Plot Diagram (Unfavorable Attitude & Race) .................................................................. 80
4.13. Mean Plot Diagram (Favorable Attitude & Gender) .................................................................. 84
4.14. Mean Plot Diagram (Unfavorable Attitude & Gender) ............................................................. 85
ABSTRACT

This study aimed to understand the identity and attitude of students enrolled in computer science (CS) or programming-related course at community colleges nationwide. This study quantitatively evaluation data for estimating the relationships between students’ identity and attitudes toward computer science with prior programming experience and other demographic factors. I distributed the survey to community college faculty of computer science programs nationwide. Questions for this study were adapted from the Computing Attitude Survey developed by Weibe, Williams, Yang, & Miller (2003). Using two robust quantitative statistical methodologies, I investigated the correlations and predictability of previous programming experience, gender, race, and age with participants' attitudes toward computer science. This study drew its inspiration from prior works of Dorn and Tew (2015) and Chen, Haduong, Brennan, Sonnert, and Sadler (2018), whose studies looked at previous experiences in programming with a favorable attitude toward computer science. The primary independent variable was a students’ prior programming experience. Under evaluation, the dependent variables were students' programming experience and demographic characteristics such as race, gender, and age. This investigation showed a significant association between programming experience and attitude toward computer science. Among the demographic variables evaluated, students' racial identity was the only factor found highly correlated with attitudes toward computer science. Future work will consider the association between participants' accumulated college credit hours and specific programming language effects on computer science attitudes.
CHAPTER ONE. STATEMENT OF THE PROBLEM

Introduction

In the United States, according to the National Center for Educational Statistics (2019) report, there are 1,485 two-year colleges, commonly referred to as community or junior colleges, and 2,703 four-year colleges (para. 1). Of those roughly 1,500 two-year CCs in the US, two-thirds categorize themselves as public institutions. According to NCES Statistics (2019), CC institutions have declined since peaking at 1,738 in 2012. Today, 250 fewer CCs exist than a decade ago. Often located in the core of urban cities and surrounding suburbs, “CCs serve high-need populations in economically distressed regions of the country” (Myran & Sylvester, 2021, p. 591).

According to Kahlon, Boisvert, Lyon, Williamson, and Dubow (2019), “CCs enroll more Black, Indigenous, and People of Color (BIPOC) and women in larger numbers than four-year institutions” (p. 803). Many Hispanic students who earn bachelor’s degrees start at a CC level. In the U.S., 43% of CC students are non-white, and 42% are the first in their families to attend college (Jaggars, Fink, Fletcher, & Dundar, 2016). A stunning lack of diversity in CS programs nationwide is demonstrated in degrees awarded. The apparent inequity of awarded CS degrees among students of varying racial and gender identities cannot be disregarded. In 2018, “80% went to white males, 5% to Black students, 18% to Asian students, and 9% to Hispanic students” (Kahlon et al., 2019, p. 803). Lyon and Denner (2016) argue that CCs will become an important starting point with defined pathways for women and BIPOC to successfully transfer and complete their CS bachelor’s degrees at four-year universities.

Jaggars et al. (2016) addressed key findings and recommendations of CCs CS pathways to four-year CS degrees in their report. They stated that:
Navigating the CC pathway to a CS bachelor’s degree is complex and challenging, such that only students who are focused and fortunate are able to navigate the pathway successfully. We need to create more structured and supported pathways to help a larger number of underrepresented CC students attain a CS bachelor’s degree and a need for CCs and universities to partner more closely (p. 3).

Myran and Sylvester (2021) described that those students new to the CC environment are arriving at school with a skewed self-image and in high need of support. They “struggle with self-efficacy in taking command of their learning and career goals” (p. 591). Sylvester and Myran (2018) suggest that a plan for the success of high-needs students is the coordinated action of specific practices serving as career navigators for students in need of guidance from college administrators and instructors. For example, Mott CC utilizes a comprehensive student-centered curriculum with Career Pathway programs centered on career interest, prior employment, and education to connect them to career-relevant resources (Myran & Sylvester, 2021, p. 593).

Career readiness programs link students to services aligned explicitly with their talents and services to address their difficulties. The purpose of supportive services is to help boost students’ self-confidence and self-image, empowering them to take charge of their learning and attain their long-term professional goals.

As previously mentioned, CCs are public and private institutions of higher education, providing a range of general education courses, technical training, and other certification and credentialing programs. Typically completed in two years, CC programs are vitally important because they are accessible to regional residents. These crucial higher education institutions and workforce development resources offer highly focused programs comprising general education, prerequisites, and career-specific vocational courses. Bakley and Brodersen (2018) discuss how the open-access nature of these institutions and lower cost per credit hour compared to four-year
universities are contributing factors to students’ motivation to complete their general education courses, program-specific prerequisites, or preparatory classes at CCs.

According to Forbes, Song, Lyon, Maxwell, and Tucker (2019), students enrolled in a CC face economic challenges more often than peers at traditional four-year universities. Fourteen percent of bachelor’s degree-seeking CC students will attain a four-year degree within six years. Furthermore, research finds that many CC students have encountered university admission difficulties due to academic and financial barriers.

As stated by Lockard and Wolf (2012):

> The CS job sector is growing quickly and provides high wages, yet colleges have been unable to produce enough CS graduates to meet industry demand, with one analysis suggesting a shortage of nearly 100,000 CS graduates per year. The shortage is particularly severe among women and underrepresented minorities (p. 1).

Interestingly, traditional universities with competitive CS programs working to increase their student populations’ diversity often overlook CCs as a recruiting resource for students traditionally underrepresented in specialties. Indeed, the complexity of coding and the abstract programming languages in CS contributes to the small percentage of students expressing interest in CS. According to Guzdial (2015a), computing (e.g., CS, programming, coding) has generally been a field dominated by affluent Asian and white males. Approximately “80% of students majoring in CS fit that demographic; less than 20% are female, or of groups historically underrepresented in CS” (p. 8).

Margolis, Goode, and Ryoo (2014) for the democratization of computing programs to diversify the monocultural technology sector workforce. Historically, the pool of talent that continues to supply the software industry is far from a source of diverse talent. With a bachelor’s degree in CS, programming candidates can expect lucrative software developer positions,
earning an average salary of approximately $89,190 per year or $42.88 per hour (U.S. Department of Labor, 2020, para. 2). Fisher and Margolis (2002) claim CS conveys a reputation of a high-barrier fortress reserved for privileged males. With its difficult-to-grasp subject matter and highly selective programs at prestigious colleges, the stereotype reinforced is that CS is meant for the academically bright, male “geeky” type of white or Asian affluence. This firmly established stereotype of who is and who will become a programmer adversely affects outsiders’ view of themselves in the context of CS and programming (Margolis et al., 2014; Margolis, Goode, & Bernier, 2011; Margolis, 2008).

Some CCs, often located in urban areas, offer CS pathways for students to transfer their credits into CS programs at nearby four-year universities (Forbes et al., 2019, p. 964). These night and weekend class offerings cater to the full-time worker, whose nine-to-five schedule conflicts with regular university classes that meet in the daytime. An ongoing issue for underrepresented students who successfully achieve admission into university CS programs is their difficulty integrating smoothly into competitive academic environments concentrated on high achievement. Rodriguez and Kerrigan (2019) found that “those students considered nontraditional, can feel defeated and intimidated in a college environment given the high demands and expectations in and out of the classroom (p. 470). Although “computing occupations increasingly dominate the workforce, they fail to attract sufficient and diverse students (e.g., women, underrepresented minorities, and students living with disabilities) to meet workforce needs” (Chang, Cintron, Cohoon, & Tychonievich, 2018, p. 783).

Lyon and Denner (2016) describe hurdles students of underrepresented groups experience establishing peer relationships with others of similar backgrounds (p. 5). Administrators should develop effective strategic plans for addressing issues of inclusion in four-year CS degree
programs (p. 3). One strategy is to generate programs for supporting students with high needs and cultivating a collaborative learning environment where faculty and staff endorse and advocate for the diversity of students’ opinions and views (p. 7).

Economic demand for skilled programmers continues to rise steadily, and “demand for these skills is huge; not only are there perpetually hundreds of thousands of open CS positions, but computational skills are becoming a core competency for many modern jobs” (Lusa Krug, Bowman, Barnett, Pollock, & Shepherd, 2021, p. 397). The technology industry and university CS programs have fumbled to “captivate the imaginations of a wide array of urban youth of color from diverse places of origin” (p. 398). As large universities attempt to diversify CS programs using primitive techniques for maintaining the status quo, students at two-year colleges recommence to be an omitted resource of students of differing experiences of diverse milieus for confronting gender and race inequities in university CS programs (Dou, Bhutta, Ross, Kramer, & Thamotharan, 2020). Students of two-year CS programs at CCs preparing to transfer to four-year universities are an underutilized source of intelligence and talent for solving the increasing economic need for programmers that the tech industry has dubbed the problem the leaky pipeline (Blikstein, 2018).

Jaggars et al. (2016) reported that females and other underrepresented students thrive in an environment that is supportive and collaborative (p. 4). Highly competitive environments can have unintended consequences. Such toxic cultures allowed to flourish unhindered can hurt BIPOC students by “triggering stereotype threat, undermining confidence, or otherwise making them feel unwelcome and unfit, leading them to self-select out” (Google & Gallup, 2016; Margolis, 2008, p. 4).

As reported by Thompson (2021):
Weed out-courses vary by institutions, and these difficult courses, which are required for science, technology, engineering, and mathematics (STEM) majors, are often cited as a prime reason for leaving STEM, not only because many students fail these courses, but also because the competitive environment may feel unwelcoming (p. 961).

Women are affected harder after getting a low grade in an introductory STEM course and more likely than men to exit a STEM major after receiving a poor grade in an introductory STEM course. While men, on the other hand, are less affected by a low grade and only slightly less likely, than their male peers who received an "A" to persist in a STEM major after receiving a poor grade (Thompson, 2021, p. 956). Thus, when considering the impact of introductory courses, one must question who, exactly, is being weeded out. The educational system has cultivated secondary to post-secondary programs for advancing CS opportunities for BIPOC students and females with the legislature at the state, district, and local levels. For example, “a collaborative effort of the Code.org Advocacy Coalition launched a program to guide states seeking to implement policies for expanding access to CS curriculum in their most underserved K-12 school districts” (Google & Gallup, 2020, para. 3).

The Coalition’s nine policies guide participating states’ educational systems toward implementing robust CS programs at the state, district, and local school levels. Such programs like the nine policies allow students of underprivileged backgrounds to engage in a high-quality, authentic, computer-science curriculum in middle and high school. As reported by Y. Kafai and Margolis (2014), there is strong evidence that given a chance to learn to code, students’ attitudes toward computing will shift favorably (p. 1:2).

Chen et al. (2018), Dorn and Tew (2015), and Weintrop and Wilensky (2019) reported results from seminal research of a pre-and-post CS course attitude survey. After completing an
introductory CS course, the researchers identified statistically significant attitudinal changes in both males and females. Although both genders did experience positive shifts in attitude regarding CS after completing the course, female students showed significantly greater growth than male students (p. 23). In this chapter, the following topics are addressed:

1. The purpose of the study
2. The rationale for conducting this research
3. Defining the research questions
4. Theoretical framework

**Purpose for the Study**

Computer programmers write and test the code of software programs and other applications to ensure they are operating and functioning as intended. Sometimes, the programmer is only the implementer of software designs that software engineers initially developed. The programmer converts the software developer’s plan into programs that computers can understand and execute. Debugging code to identify an error is another common task in a computer programmer’s job. They scrutinize the code for errors, and in such cases where a bug is found in the code, programmers correct them. The search and find the process of fixing code is ubiquitously understood as code tracing and debugging.

Computer programmers write and test the code of software programs and other applications to ensure they are operating and functioning as intended. Sometimes, the programmer is only the implementer of software designs that software engineers initially developed. The programmer converts the software developer’s plan into programs that computers can understand and execute. Debugging code to identify an error is another common task in a computer programmer’s job. They scrutinize the code for errors, and in such cases
where a bug is found in the code, programmers correct them. The search and find process of fixing code is more ubiquitously understood as code tracing and debugging.

According to a report by the U.S. Department of Labor (2020), the typical computer programmer will

Update and expand existing programs; test programs for errors and fix the faulty lines of computer code; create and test code in an integrated development environment (IDE); and use code libraries, which are collections of independent lines of code, to simplify the writing (p. 1).

Experienced computer programmers can compose code in multiple languages (e.g., Java, and C++). According to the U.S. Department of Labor (2020), “most entry-level programmers earned a bachelor’s degree in CS or a related subject; however, some employers hire workers with an associate degree; and most programmers specialize in a few programming languages (para. 2). The high achieving, “geeky” male stereotype of the computer programmer permeates the field. Compounding the problem, when they get to college, many youths from privileged backgrounds have advanced programming skills before even taking their first computer programming course (Guzdial, 2015a).

Resource-deprived youth with no prior CS experience start programming coursework at two-year or four-year colleges. The preparation gap between students at either end of the economic spectrum places affluent learners ahead of other novices at the outset of college introductory programming courses. As Resnick and members of the research team who introduced youth from economically disadvantaged communities to after school computing programs, they created a strategy for introducing underserved youth to CS in middle school that gave the youth an “opportunity to become technologically fluent while creating projects that were meaningful to them and their communities” (Maloney et al., 2003, p. 1).
Novices of all ages struggle with learning to compose error-free code. The struggle of the newcomer is center stage in Computer Science Education (CSEd) literature, but its focus is on beginner programmers’ learning activities and experiences in pre-college educational contexts. This dissertation research study seeks to extend on previous works primarily concentrated on K-12 education. However, maintaining a keen focus on the beginner programmer, I strived to evaluate learners’ attitudinal shifts toward CS at CCs, an overlooked population who could play an important role in the CS education pipeline (Forbes et al., 2019). A bulk of CC students fall into at least one category historically underrepresented in CS.

Kahlon et al. (2019) suggest:

Given that CCs enroll minorities and women in larger numbers than four-year institutions, it is imperative for both CCs and four-year institutions to engage in collaborative efforts to broaden the participation of diverse peoples into computing and facilitate unobstructed pathways from the two-year to four-year educational programs. “While CCs are rich in diversity, their students face an inordinate number of obstacles transferring to a bachelor’s degree, particularly in CS” (p. 803).

Although these vulnerable student populations’ ages and grade levels vary, their shared financial, social, and personal challenges are the common denominators; thus, the CSEd literature seems most applicable for this research. I propose an examination of a neglected student population in CS and building context for future research. Decades of research in CSEd have, thus far, aimed at supporting beginner programmers in K-12 and four-year university CS programs through challenging intervals learners must overcome, yet many frequently quit or give up programming entirely (Sorva, 2013). Because the research on CSEd in the context of two-year college CS programs is scant, my research is a novel and essential addition to the existing CSEd literature and lays the groundwork for future studies of populations at CCs.
This study extended on prior works of Chen et al. (2018) and Dorn and Tew (2015), who focused on measuring attitudinal shifts of four-year university students after completing a college-level introductory CS course, while Weintrop and Wilensky (2019) quantitatively measured attitudinal shifts of high school students who completed an introduction to programming course. My work aimed at measuring attitudinal changes of students enrolled in CS programs at CCs with varying prior-programming experiences. Dorn and Tew (2015) purported that few instruments are available in CS for measuring attitudes toward CS and programming. Given the scarcity of validated and reliable attitudinal instruments in CS, the Computer Science Attitudes Survey was selected based on the works of Weibe et al. (2003). Weibe and associates modified the survey developed by Fennema and Sherman (1976, p. 2), whose Mathematics Attitudes Scale was the basis for creating the Computing Attitude Survey. No study to my knowledge has empirically investigated shifts in attitude toward computing at varying prior programming experiences students at CCs.

Research Questions

I proposed a quantitative study that aimed at capturing attitudinal changes of participants using a survey instrument derived from the Computing Attitudes Survey developed by Weibe et al. (2003). The population investigated was students enrolled in CS-related courses at CCs. My research aimed to contribute novel data to the CSEd field that has, over the decades, overlooked students’ learning experiences of CS at CCs.
This study answered two primary research questions: how did previous programming courses correlate with a favorable identity and attitude toward computer science, and did previous programming courses predict a favorable identity and attitude toward computer science? To understand the effect of background characteristics on attitude toward computer science, two sub-questions were developed:

1. Did demographics like age, race, and gender predict favorable identity and attitude toward CS?

2. What demographic factors, if any, correlated with CC students’ identities and attitudes toward CS, and how?

**Theoretical Framework**

**Introduction**

It is much less probable for women and BIPOC students to select a STEM major because there are much fewer role models in scientific fields that would help them imagine themselves as someone who could be successful in a scientific field. The dearth of women and BIPOC in principal STEM-related positions affects their identifying with STEM fields because role models are deficient (Starr, Anderson, & Green, 2019, p. 493). The systemic implicit and explicit bias of women and BIPOC students’ that question their abilities in in math, physical sciences, engineering, and CS fields perpetuates underrepresentation of them in some of the most lucrative careers in computing and technology (Starr et al., 2019, p. 494).

Quinn (2020, p. 388) claims that “teachers in diverse schools exhibit implicit and explicit stereotypical typecasting of women and BIPOC students as ones less competent than white students, specifically, white-male students, in STEM, but specifically, in CS learning domains (Margolis, 2008, p. 79). In contrast to obvious biases noticed in diverse schools, examination of predominantly Black schools showed no indication of racial or gender preference propensities of
According to Quinn (2020, p. 387), educators at racially diverse schools exhibited discernible bias suggesting “that Black students at diverse schools may be especially at risk of receiving biased evaluations.” According to Starr et al. (2019) elements of bias and stereotyping of teachers has contributed to fewer members of individuals from underrepresented in CS and other scientific domains

**Stereotype Threat Theory**

According to Stereotype Threat Theory, as defined by Steele (1997), individuals of subgroups that carry a negative stereotype in a learning domain and who are constantly having to prove their worthiness against their associated stereotype are most affected (p. 613). These individuals can ease their Stereotype Threat-related anxiety successfully traversing from one proving situation to another; however, this unending proving that underpins Stereotype Threat Theory can have negative consequences affecting identity and attitude toward a specific knowledge domain (e.g., CS) (Steele, 1997, p. 614).

A Stereotype Threat Theory, as defined by Steele (1997), individuals of subgroups that carry a negative stereotype in a learning domain and constantly must prove their worthiness against their associated stereotype are most affected (p. 613). These individuals can ease their Stereotype Threat-related anxiety successfully traversing from one skill-verifying situation to another; however, this unending proving that underpins Stereotype Threat Theory can have negative consequences affecting identity and attitude toward a specific knowledge domain. Steele (1997) defined Stereotype Threat Theory as:
The event of a negative stereotype about a group to which one belongs becoming self-relevant, usually as a plausible interpretation for something one is doing, for an experience one is having, or for a situation one is in, that has relevance to one’s self-definition and is cued by the mere recognition that a negative group stereotype could apply to oneself in a given situation (p. 614).

For example, underrepresented constituents who are functioning at advanced levels in their area of expertise “must disprove the stereotype’s relevance to their next, more advanced performance” (p. 618). Therefore, all prior efforts made by individuals that resulted in a debunked stereotype threat from that person are invalid at the outset of the next level of advancement and “must be rewon in each higher level of proving ground” (p. 618).

**Identity and Attitude Toward Computer Science**

Forming an identity, a sense of belonging, “is crucial to the success of college students — it mediates achievement goals, metacognition, and learning strategies employed” (O’Hara, 2020, p. 2). According to O’Hara (2020), a positive learning environment supports students’ motivation, self-efficacy, engagement in school, and a sense of belonging (p. 2). Identifying and belonging to a subgroup, in this case, CS, to which an individual seeks acceptance, “correlates to motivation, drive, and behavior while also playing a role in how they view themselves” (O’Hara, 2020; Maslow, 1970, p.1). O’Hara (2020) claims “the attitudes and behaviors of college faculty toward women and BIPOC students have the potential to create an inclusive or dismissive culture that promotes or discourages Stereotype Threat that can influence their performance, motivation, and identification with the learning domain (p. 1).

Steele (1997) Stereotype Threat can affect students’ sense of identity and favorable attitude toward a STEM learning domain because the constant proving of oneself to others can raise anxiety levels for these individuals that over the long term can begin to form negative
attitudes towards the field because of uncomfortable and high-pressure situations in which to perform. Steele (1997) alleges that Stereotype Threats can occur in group settings and at times when the person is alone. For example, Stereotype Threat happens when “a woman taking an important math test alone in a cubicle but under the threat of confirming a stereotyped limitation of ability” (Steele, 1997, p. 617). As mentioned previously, there is no everlasting remedy to cure Stereotype Threat. In the context of CS and programming, where the bar to solve novel problems is incessantly being raised, subgroups can disprove the stereotype and minimize their anxieties that can negate a strong identity and attitude toward CS at one level but the need to do so over and over, “can feel Sisyphean, that no amount of success up to that point can disprove the stereotype’s relevance to their next, more advanced performance” (p. 618).

**Organization of the Dissertation**

This dissertation includes five chapters. Chapter one acquainted readers with the CC role in developing diverse populations in a two-year CS degree program to either assume professional roles in the computing industry or transfer to a four-year university CS program. The introduction reviews known factors contributing to students’ motivation to complete CC coursework before transferring to a four-year university. Other discussion points include social and financial challenges students attending community often face in their attaining educational goals. Furthermore, the introduction addresses gender and racial inequalities in CS and the effect inequities have on long-standing CS stereotypes and growing achievement gaps.

Finally, legislative mandates, outreach programs, and research studies aiming to broaden access to high-quality CS education wraps up this section. The concluding sections of chapter one provided brief descriptions of applicable theoretical frameworks, which, as a graduate student, were introduced during my traversing the Curriculum Studies Program at Louisiana
State University. The discussion and presentation of Stereotype Threat Theory Steele (1997) as a framework to inform how students might form identities and attitudes toward CS, with a focus on groups historically underrepresented in CS.

Chapter two reviews of the literature on Computer Science Education focus on novice or beginner programmers in pre-college learning environments and computing pathways for earning two-year degrees in a CS-related discipline at CCs. Given that high school learners are close in age to first-year college students, I anticipate similar outcomes for this study compared to past studies of similar aims in K-12. Few studies, as mentioned, concentrated on learners’ experiences at CCs of students registered in CS-related courses. This dissertation presents a literature review of broad scope comprising the presentation of CSEd along a historical timeline, the pedagogical milestones, and groundbreaking instructional and assessment innovations that transpired over 60 years.

The literature review highlights important experimental research studies whose researchers focused on advancing equity with ingenuity and innovations to assist novice programmers in succeeding. Chapter two reviews the CC literature to highlight novel CS pathways and programming-related curricula developed by CS CC faculty to prepare their diverse cohorts of students for either two-year associate degrees or transfer to a four-year university. Chapter three describes the data and methods used for this study and explains the rationale for selecting sample populations and variables under evaluation. Furthermore, this chapter defines and categorizes each variable into a dependent or independent category and explains the appropriateness of this research as a quantitative-only study. Chapter four examines the results of this study with an analysis of the findings and answers to the initial research question proposed for this study.
The fifth chapter emphasizes how this research contributes significantly to the CSEd literature. This study adds new knowledge to the CSEd field by expanding the analysis of K-12 and four-year university students to overlooked CC students seeking two-year CS degrees or CS-related certifications. This research is novel in addressing an omitted population in computing education research. This study conceivably lays the groundwork for future works in computing education with a focus on CCs.
CHAPTER TWO. A REVIEW OF THE LITERATURE

This chapter reviews the literature on CS, focusing on pedagogical innovations, equity, and integration of high-quality CS and programming curriculum in financially distressed districts.

Rationale for Broadening Access to Computer Science

According to the United States Department of Labor (USDL), computer programmers need a four-year bachelor’s degree in CS or programming as a minimum requirement to enter the CS workforce (U.S. Department of Labor, 2020). However, technology companies employ non-degreed programmers with knowledge and skills in specific programming languages (U.S. Department of Labor, 2020). The fundamental difference between a programmer and a computer scientist is that a programmer is technically skilled in one or more specific programs while computer scientists earn broader knowledge CS in their earning a four-year degree. For example, a bachelor’s degree in CS exposes students to a broader knowledge of computing to impart an understanding of abstraction, data structures, algorithms, and machine learning that can be applied in the learning of most programming languages. For example, computer scientists might develop software programs as a function of their duties in the workplace, while a programmer’s tasks might be limited to updating, testing, and fixing existing software programs (U.S. Department of Labor, 2020, para. 1).

According to the USDL, programmers will perform the following tasks in the workplace (U.S. Department of Labor, 2020, para. 2):

1. Write programs in a variety of computer languages, such as C++ and Java
2. Update and expand existing programs
3. Test programs for errors and fix the faulty lines of computer code
4. Create and test code in an IDE

5. Use code libraries, which are collections of independent lines of code, to simplify the writing

Some programmers can earn certificates to become certified in a particular programming language or business-specific programming product (U.S. Department of Labor, 2020). Regardless of earning a CS four-year degree, a two-year degree in programming, or a programming-related certificate or license, computer scientists and programmers attend professional development seminars to learn new languages, maintain knowledge of updates to existing ones, and keep up with changing technology in general (U.S. Department of Labor, 2020, para. 3). Regardless of being self-taught or university-trained, or some combination of both, in programming or CS, they must keep updated and skilled in a field quickly evolving and constantly changing.

According to the USDL, the median annual pay for a computer programmer in May 2020 was $89,190. Software publishers earned the highest income with a median yearly income of $103,710, followed by finance and insurance at $92,000, manufacturing at $89,500, and computer systems design at $88,500 (U.S. Department of Labor, 2020, para. 1). The United States Bureau of Labor Statistics (USBLS) projects a ten percent decline in computer programming jobs between 2020 and 2030; however, despite the projected decline, a need to replace retired workers or workers who transfer to other companies will create 9,700 openings for computer programmers annually (Coalition, 2020).
Federal and State Funding of Computing Programs

In recent decades, the U.S. began to fund federal programs for expanding CS into K-12 schools. States began to pass mandates for high-caliber computing curriculum integration into schools. National efforts and not-for-profit programs aimed at teacher-training programs bridge the knowledge gap between the technical skills of the computer scientists and minimum knowledge of CS secondary school teachers should attain before teaching CS in secondary schools (Yadav, 2017).

Databases chock full of free educational tools are available as open-access repositories for secondary school CS teachers to browse through in their search for grade and learner-appropriate CS and programming materials for use in their classrooms (Kalelioglu, 2015). In forthcoming sections, I will explore how each contributed something valuable, through research or application, to CSEd and through which the combined efforts have advanced the prospering field to where it stands now.

Computer Science Education History

1960 to 1990

In his seminal book, Mindstorms: Children, Computers, and Powerful Ideas, Papert (1980), describes Logo as a programming language for youth to create objects using a graphical ‘Turtle’ that focuses learners’ attention on discovering, problem-solving, and ongoing activity for critical thinking while traversing a fun, game-like landscape. Logo’s graphical interface was novel, unlike any other learning technology, for engaging children in coding activities while simultaneously solving math problems. In the 1960s, Logo’s release offered a unique, novel, and
cutting-edge technology for youth to “conceptualize mathematics concepts in a fun, playful, yet logical way” (Feurzeig & Papert, 2011, pp. 478-488).

Nearly two decades after Logo launched, Boxer was released. Like Logo, Boxer had a graphical interface and encouraged its users to imagine the spatial relationships around a compound data structure. The visualization platform challenged students to consider a box as a metaphor for holding a value, such as a number or character (diSessa & Abelson, 1986). Logo and Boxer were the original learning systems that gained public use in schools during the mid-1980s (diSessa & Abelson, 1986, p. 867). As predicted by diSessa and Sherin (1998), computers would one day be integral to our daily life and powerful agents for social and educational change.

1990 to 2021

About twenty years after Boxer’s release, MIT researchers discovered a novel way to present a programming language in a graphical format that drastically differed from traditional text-based programming languages. To further explore the notion of a graphical-interface programming language for the modern era, in 2003, NSF awarded Resnick et al. (2009) a grant to develop a novel programming environment for youngsters to learn to code quickly. The researchers’ core mission was a simple concept that informed the design of their tool. They imagined it as a mode of programming appealing to people who might never have imagined themselves as a coder (Maloney et al., 2003).

**Block-based and Textual Programming Languages**

The group of researchers developed three core design principles in their tool development. First, the program must be tinkerable, second, meaningful, and lastly, an
environment that encourages interaction and social learning in the same way children would build objects with Lego Block-based Programming Environments.

Lego blocks inspired the design of Scratch for youth to “play, build, and organically create stories with colorful Block-based Programming Environments in the same way children build structures with physical Lego blocks” (p. 63). After several years of being piloted in after-school programs in financially deprived areas, Scratch was launched in May 2007 to the public for widespread use (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010, p. 63). Scratch designers made rich, engaging media environments inclusive, accessible, and exciting to grasp and sustain the interest of learners of diverse racial and cultural backgrounds. According to recent statistics published by The Importance of Being Earnest (TIOBE), Scratch is ranked first for block-based programming languages (TIOBE index, 2021). Scratch steadily increased in popularity and expanded worldwide. However, teachers’ use of Scratch in classrooms is productive in allowing students to engage with the software but falls short to some for its lack of cohesion among the range of different instructional materials.

To address frustrations growing of Scratch in the education community, MIT researchers developed a conceptual framework as stated by Brennan and Resnick (2012):

We developed a computational framework with key computational concepts, such as iteration, parallelism; computational practices, such as debugging projects or remixing others’ work; and computational perspectives designers could form about the world around them (p. 1).

The colorful design of drag-and-drop coding ubiquitous in block-based programming environments made coding simpler; in fact, novices no longer must grapple with syntactical errors common in text-based coding environments. Graphical and block-based programming environments like Scratch serve as starting points to engage newcomers to computing concepts.
However, unintended consequences of block-based environments blossom as learners attempt to transition from a block-based environment to traditional, text-based languages, like Java (Statter & Armoni, 2020). As reported by Weintrop, Hansen, Harlow, and Franklin (2018), “while block-based have excelled in informal spaces, their formal use in classrooms to prepare learners for future CS instruction is not without its challenges” (p. 149).

**Equity in Computing Education**

According to Wing (2006), digital literacy competencies and their accompanying frameworks help novices develop needed skills to prosper in an increasingly digitized world. As Resnick et al. (2009) declare, “those who interact with digital media all the time are numerous, but few can create games, animations, or simulations themselves; digital fluency means people can chat, browse, and interact but who can build and create media artifacts” (p. 62). There seems to be a widening gap in computing competencies along socioeconomic status lines between the affluent learners with aptitude in creating digital artifacts with coding skills while economically disadvantaged populations are more likely to consume other-created digital media (Grover, Pea, & Cooper, 2016).

As Grover et al. (2016) claims, “students with prior experiences actively creating high-value objects (e.g., code, digital media, or graphics) with computers performed better in programming post-tests than students with no prior experiences and who passively consume greater volumes of video and game media” (p. 556). There are several factors that influence the learning of code (Grover et al., 2016, p. 556). She declares main factors affecting rate of learning to code “include learners’ prior experience, interests, and attitudes concerning the subject being taught in addition to academic preparation, especially in foundational subjects such as mathematics” (Grover et al., 2016; Lewis & Shah, 2012, p. 558).
It is not surprising that abstraction is key to computing and a key skill taught in math. Given students' difficulties with loops and variables (which share a strong relationship with abstract and algebraic thinking) compared with conditionals, the link to prior math preparation needs further probing. This issue is also significant because students’ mathematics preparation historically co-varies with socioeconomic status and other indicators of diversity (Grover et al., 2016, p. 556).

Arpaci-Dusseau et al. (2013) conclude that without sound educational programs to develop a future generation of diverse programmers and computer scientists, the supply of trained programmers ready for work in the technology industry will fail to meet future economic demand for skilled labor. As Blikstein (2018) reported, “the economic productivity in the United States will be determined by the country’s capacity to churn out qualified programmers” (p. 9). Still, other experts are concerned that the prevailing educational pathways leading to careers in CS appeal principally to white males. Fisher and Margolis (2002) and others stated that those with requisite math and English skills and prior programming experience stand to be among the most prepared for navigating a technology-centric economy of careers (Denning, 2017).

**The Computer Scientist Stereotype**

Historically, the CS curriculum for novice learners has traditionally lacked diversity and culturally responsive pedagogy to capture the interest of females and students of underrepresented racial identities. The stereotype and bias teachers often hold of who will enjoy and succeed in CS courses are overwhelmingly the high-achiever white males in science and math (Dou et al., 2020). To address the curricular stereotype and explicit bias favoring males, researchers over the previous two decades have authored instructional approaches to expand
underrepresented students’ interests to include gender and culturally responsive CS pedagogies (Y. Kafai & Margolis, 2014; Sentance, Sinclair, Simmons, & Csizmadia, 2018).

Given the extensive CS resources available for free to teachers and students, educators and researchers found that early exposure to CS instruction helps attract the interest of students historically underrepresented in CS. For example, Grover et al. (2016) analyzed students’ prior experiences with computing and discovered a positive correlation between the richness of interactions with varying technologies, such as software programs for creating digital artifacts, and programming test performance. Diversity and inclusion in CS are foundational for the equity-of-participation rationale. Scholars of research focused on CSEd describe such inequities to access quality CS education and deficiencies of women and non-majority racial identities in CS education and the tech industry (Blikstein, 2018; Grover et al., 2016; Dorn & Tew, 2015).

**Racial Justice in Computing Education**

In 2021, several regional and national STEM and computing education conferences pivoted on themes based on racial justice and equity in CS education. For example, the Institute of Electrical and Electronics Engineer (IEEE)s Global Engineering and Education Conference: Women in Computing and The University of Texas’ 2021 UTeach Conference: Equity and Racial Social Justice Summit: Moving from Discussion to Action awarded preference to proposals focused on addressing racial injustice and gender inequity in computing. The drive to substantially affect changes in profound ways in CS is advancing through “strategic collaborations among universities, tech companies, and nonprofits whose works generated novel technology, culturally responsive instruction with a common goal of diversifying the tech industry” (Magerko et al., 2016, p. 14:1).
Instructional Strategies in Computing Education

Experts of CS, university faculty, and secondary and post-secondary educators contribute to develop new opportunities for socioeconomically underprivileged learners. As discussed in a previous section, the launch of Logo, followed by Scratch, were pivotal moments in computer-based learning aimed at beginner programmers to foster deep, meaningful, and productive learning of creative expression (Y. B. Kafai & Burke, 2013).

Blocks-to-Text-Based Programming Language Transition

However, significant obstacle is novices’ stunted progression toward more rigorous text-based programming. These issues arise as learners traverse through a transition from Block-based Programming Environments using Scratch or Alice or text-based programming environments such as Carnegie Mellon University Computer Science Academy. According to Weintrop et al. (2018), the ideal interlude for transitioning learners from block-based curriculum to textual environments should occur after five weeks from the outset of block instruction. Learners’ motivation peaks at five-week in block-based instruction (p. 5).

Given learners’ enthusiasm during this peak, Weintrop et al. (2018) recommend that students transition to textual languages during high motivation intervals and transform that enthusiasm into momentum to push past early textual difficulties before learners’ frustrations lead to wanting to give up. Experts of CS, university faculty, and secondary and post-secondary educators contribute to developing new opportunities for socioeconomically underprivileged learners. As discussed in a previous section, the launch of Logo, followed by Scratch, were pivotal moments in computer-based learning aimed at beginner programmers to foster deep, meaningful, and productive learning of creative expression.
Mental Models

Since the first introduction of visual programming environments, researchers have theorized about the beginners’ difficulties with an abstract conceptualization of code execution, program tracing, and debugging. Beginners often tussle mentally with a visualization known in CSEd as the mental model. To accurately conceive mental models, the learner conducts a mental tracking of the program’s steps before its runtime execution; then, the learner accurately explains the coded instructions and successfully predicts its outcomes (Guzdial, 2015b). A thoroughly developed mental model is a fundamental milestone in learning; knowing what code does, helps learners form correct mental models during CS skills development (Blikstein, 2018). Eventually, the student understands how computations of physical computers get processed and the acquisition of skills for controlling them (Fronza, Ioini, & Corral, 2017; Garcia-Penalvo & Mendes, 2017; Kurland & Pea, 1985; Zimmerman, Johnson, Wambsganss, & Fuentes, 2011).

An impasse often occurs when programming beginners unwittingly form incorrect mental models (e.g., abstraction, control flow, and variables) of the computer’s processes for executing program code. To address this common misunderstanding, in the 2000s, researchers of CS at Carnegie Mellon University, Cooper, Dann, and Pausch (2000, pp. 1 -2) found that “novices seriously grapple with visualizing the steps of a program that results in their confusion while tracing and debugging code.” Such groundbreaking research was foundational for developing an interactive 3D animation tool for novice programmers named Alice. The program scaffolded computing pedagogy for beginner coders in their interactions with a 3D world based on “graphic visualizations for helping users apprehend the mental model concept in which the learner understands their code and can predict the outcomes of each task’s role in solving the greater problem” (p. 3).
Systems of Assessment in Computing Education

Best practices for assessing novices or a beginner’s skills in computational thinking concepts are far from definable. As Grover (2017) asserted, there is a need for “systems of assessments that are complementary, encourage and reflect deeper learning, and contribute to a comprehensive picture of student learning” (p. 274). As Snow, Rutstein, Basu, Bienkowski, and Everson (2019) claimed, “many of the assessments that do exist largely focused on measuring fundamental CS conceptual knowledge and programming skills rather than the application of these skills in authentic computational problem-solving contexts” (p. 174). In Table 2.1 and Table 2.2, the graphic organizers present a comprehensive list of empirical studies that evaluated objective and subjective forms of assessment of students’ computational thinking skills in a K-12 learning environment.
<table>
<thead>
<tr>
<th>Paper author(s)</th>
<th>Curriculum / Pedagogy</th>
<th>Programming environment</th>
<th>Assessment tool/strategy</th>
<th>Sample</th>
<th>Length of study</th>
<th>CT skills assessed</th>
<th>CT framework / resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Franklin et al. (2013)</td>
<td>Interdisciplinary themes of computer science concepts</td>
<td>Scratch</td>
<td>Hairball Culminating project Digital storytelling</td>
<td>35</td>
<td>2 weeks</td>
<td>Events Initialization of state Message passing</td>
<td></td>
</tr>
<tr>
<td>Hoover et al. (2016)</td>
<td>Game design Climate change workshops</td>
<td>Scratch</td>
<td>Hairball Dr. Scratch</td>
<td>5</td>
<td>4 days</td>
<td>Flow control Data representation Abstraction Synchronization Parallelism Logic</td>
<td>Dr. Scratch 7 metrics (Moreno-León et al., 2015)</td>
</tr>
<tr>
<td>Zhong et al. (2016)</td>
<td>School-based &quot;learning storytelling by programming&quot; curriculum</td>
<td>Alice 2.4</td>
<td>TDIA (Three-dimensional Integrated Assessment)</td>
<td>144</td>
<td>18 weeks</td>
<td>Sequence Testing Debugging Loops Parallelism, Modularizing Creative expression Abstracting Reusing</td>
<td>3D CT framework (Brennan &amp; Resnick, 2012)</td>
</tr>
<tr>
<td>Chen et al. (2017)</td>
<td>Robotics curriculum Visual programming environment</td>
<td>Multiple-choice questions Programming artifact</td>
<td></td>
<td>121</td>
<td>6 months</td>
<td>(SDARE) Syntax Data Algorithms Representing Efficient/Effective</td>
<td>CSTA CT standards (csteachers.org)</td>
</tr>
<tr>
<td>Snow et al. (2017)</td>
<td>ECS (Exploring Computer Science) and EDC (Evidence-centered Design) domain analysis</td>
<td>ECS</td>
<td>EDC assessments</td>
<td>513</td>
<td>2 years</td>
<td>Human-computer interaction Problem solving Web design Introduction to programming</td>
<td>ECD (evidence-centered design) methodology) (Mislevy et al., 2017)</td>
</tr>
<tr>
<td>Funke, Geldreich, and Hubwieser (2017)</td>
<td>Unplugged activities Basic Scratch tasks</td>
<td>Scratch</td>
<td>Scratch artifact</td>
<td>58</td>
<td>3 days</td>
<td>Sequence Variables Events Synchronization</td>
<td>None</td>
</tr>
</tbody>
</table>
Table 2.2. Empirical Studies of Assessment in Computational Thinking Literature

<table>
<thead>
<tr>
<th>Study</th>
<th>Curriculum/Activities</th>
<th>Assessment Methodology</th>
<th>Time</th>
<th>Design</th>
<th>Framework/Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Witherpoon et al. (2017)</td>
<td>Robotics programming curriculum</td>
<td>Isomorphic multiple-choice questions</td>
<td>123</td>
<td>15 weeks</td>
<td>Algorithms iterations Boolean logic Program flow</td>
</tr>
<tr>
<td>Grover et al. (2019)</td>
<td>VELA (variables, expressions, looping, abstraction) activities and microworld. 4 digital activities 1 unplugged</td>
<td>Pre-assessment Interviews Focus groups Surveys Open-ended project Final Scratch project</td>
<td>71</td>
<td>2 pilots</td>
<td>Ubiquitous consistency Dynamic variation Graphical presentation incremental abstraction</td>
</tr>
<tr>
<td>Snow et al. (2019)</td>
<td>Four ECS (Exploring Computer Science) instructional units: equity and inquiry focus</td>
<td>Scratch &amp; Robots</td>
<td>Ongoing</td>
<td>Ongoing</td>
<td>Human computer interaction Problem solving Web design Programming Robotics Data analysis</td>
</tr>
<tr>
<td>Arastoopour et al. (2020)</td>
<td>CT-STEM biology unit</td>
<td>Pre- and posttest Question/response</td>
<td>41/1776</td>
<td>10 days</td>
<td>CT-STEM discourse elements Agents Agent actions Biological systems Experimentation Justifications Quantitative amount Temporal change Directional change Graphs</td>
</tr>
</tbody>
</table>

Quantitative Grading in Computer Science

Autograders are assessment tools that can quickly and efficiently evaluate code that returns a correct or incorrect response to the teacher or learner. Autograders are great for efficiency and taking the guesswork out of code analysis; however, the automatic process is quick, but the trade-off is the lack of capturing nuances and subjectivity of grading captured only with the time-consuming process of manually inspecting students’ work (Boe et al., 2013; Moreno-León, Robles & Roman-Gonzalez, 2015). An issue with Autograder is its objective approach to assessment. An objective-only approach to grading code favors the dominant race and gender; thus, adding a qualitative aspect to grading gives the teacher a more holistic and granular picture of each student’s learning progress. The benefits of qualitative grading strategies are their potential for balancing a gender bias found in objective-only grading practices (Moreno-Le´on et al., 2015, p. 18).
Assessments tools designed for block and graphic-based environments are program language-specific. For example, Scratch Autograders are specific to the Scratch program. For example, a students’ work created using Alice or another program cannot be graded by the autograder for Scratch. Autograders, as it now stands, do not generalize functionality to similarly designed programming languages (Weintrop & Wilensky, 2015). To address this common issue and frustration for some teachers, Weintrop and Wilensky (2015) designed The Commutative Assessment to gain insight into relationships among various programming languages that could be applied across other programming environments. The assessment questions comprising The Commutative Assessment cover five fundamental concepts of CS. The concepts of The Commutative Assessment were based on prior works and foundational questions in the FCS1 assessment (Tew & Guzdial, 2011). At completion, the Commutative Assessment contained twenty-eight multiple choices that covered five CS concepts: conditional logic, loops, functions, algorithms, and variables (Weintrop & Wilensky, 2015).

Regardless of the programming language, educators benefit from the research of the Commutative Assessment by offering them a resource for grading learners’ work-based programming concepts across varying programming languages and modalities (Weintrop & Wilensky, 2015). Witherspoon, Higashi, Schunn, Baehr, and Shoop (2017) evaluated other studies in which learners' knowledge was evaluated with quantitative strategies, whose visual programming curriculum was in virtual robotics. They developed 17 scenario-based isomorphic questions to assess learners, comprising the final assessment designed to” measure computing principles such as algorithm development, iteration, and Boolean logic” (Witherspoon et al., 2017, p. 4:8). In another example of quantitative evaluation, Chen et al. (2017) developed an assessment instrument for a robotics course that met once a week for one hour for six months.
Five critical components of computational thinking guided their framework’s design. Their assessment instrument contained ”23 items of which 15 were multiple-choice questions, scored dichotomously (e.g., correct=1; incorrect=0)” (p. 165).

**Qualitative methods of assessment**

Although features in efficiency and accuracy of Autograders might appeal to time-strapped teachers and instructors of CS courses, teachers might depend heavily on the efficiency of an Autograder. Teachers should be mindful of the limitations of grading beyond quantitative objectivity. While convenient, efficient, and expedient, dependence on an Autograder can conceivably avert teachers from qualitative inspection of learners’ coding projects for holistically identifying creativity and ingenuity found hidden as gradients of nuances (Boe et al., 2013). (Hoover et al., 2016) reported that for some Autograder assessment metrics, as a stand-alone assessment tool, it is insufficient at assessing a learners’ complete knowledge and found it difficult to compare the complexity of projects. They found that by pairing autograding with qualitative evaluation elements, they could better assess programming projects (pp. 178-179).

Tools for evaluating students’ work using qualitative techniques include creative design reports, digital storytelling, culminating projects, and portfolio assignments to offer learners an opportunity to apply their accumulated programming skills in various creative outlets for self-expression through code. Franklin et al. (2013) revealed through the analysis of participants’ digital stories that an Autograder assessment tool paired with the manual inspection of code for accuracy and originality provided a realistic assessment of learned knowledge.
Community Colleges’ Role to Diversify the Computing Industry

Given that the existing CS and CSEd literature has primarily focused on either the K-12 learner or students at four-year universities, the unique approach of this study is an investigation of CC students’ attitudes and beliefs regarding CS at varying levels of prior programming experience(s). More research must be conducted in the CC setting to empirically understand the relationship between the number of programming courses students have taken and their attitudes and beliefs concerning CS.

Two-year computer science degrees

Four co-variables were included in the analysis, gender identity, racial identity, ethnic identity, and age, paired with prior programming experience, and were used to form the basis of this study to identify predictors of attitude toward CS. CC CS programs are clear pathways with supportive resources to assist traditionally underrepresented students to succeed in CS. Also, community college two-year CS degrees and technical training in programming can be the overlooked solution to add more diverse talent to the qualified, skilled programmer talent pool. This study aimed to identify positive shifts in attitude concerning CS. This study expects to see an upward shift in a favorable perception of CS that could contribute to more CS learners of diverse backgrounds participating and choosing CS as a career. As mentioned, large CS programs inside a university setting are intimidating to some, mainly the transfer student from CCs entering a four-year institution for the first time.

Student support

CC preparation, coupled with supportive services, can further shape a more favorable self-image of newcomers to CS. After completing their two-year CS degree at a CC, their
success could contribute to a more favorable transfer to a four-year university experience, thus curbing adverse occurrences that have historically led some to self-select out (Lyon & Denner, 2016). The kinds of computing pathways created between four-year universities and CCs to broaden the participation of underrepresented peoples in CS.

The literature guides CCs two-year CS program developers in their building of robust CS pathways in the following ways:

- Strengthen the CS transfer pathway for students interested in obtaining a CS bachelor’s degree” (Google & Gallup, 2016).
- Local CCs should partner with four-year colleges and create CS programs that guarantee acceptance into the four-year university CS program if certain academic requirements are met (Tang, 2016).
- Provide supportive services that proactively monitor transfer students’ progress and establish a protocol for academic interventions, such as interviews, for struggling students (Anderson-Rowland, Rodriguez, & Grierson, 2011).
- Transfer students should be regarded with the same level of priority as major admission.
- Ensure the CCs’ courses meet universities’ strict admission requirements to ensure admission or disclose to CC students the limitations of the CS courses for transfer into a four-year CS program.
- Ensure the University CS program environment is welcoming to transfer students. Create a collaborative environment with friendly competition (Hyatt & Smith, 2020).

**Computer science programs leading to bachelor’s degrees**

The majority of CSEd research focuses on learners in K-12 classrooms, focusing on middle and high school level students. However, minimal empirical research has sought to understand CC students’ preexisting attitudes and beliefs respecting CS and programming at varying levels in their pathway to achieving a two-year CS degree at a CC. According to Forbes, Song, Lyon, Maxwell, and Tucker (2019) and colleagues, the traditional four-year university pathway for developing talent pools produces fewer graduates to meet the software developer
workforce demand. The CC systems will play a crucial role in diversifying the tech workforce and adding more trained workers to the CS career pipeline.

Although the demand for occupations in technology continues to increase, Chang, Cintron, Cohoon, and Tychonievich (2018) point out that the computing industry has failed to attract enough students traditionally underrepresented in CS. The work of this study will bring awareness to an overlooked student population enrolled in CCs seeking two-year degrees and technical training in programming and spark broader attention to the development of programs that prepare students in CC CS programs as workforce ready programmers or transfer readily to four-years bachelor’s degree CS programs.
CHAPTER THREE. METHODOLOGY

Introduction

This presents the research design, characteristics, and demographics of a practicable sample based on national statistics data of CCs, attributes, and components of Qualtrics survey instrument and quantitative methodology implemented for analyzing and interpreting participants’ data.

Analogous Prior Work

Previous studies by (Dorn & Tew, 2015) and Chen et al. (2018) had a significant influence on the design and development of my study. Each study, as well as one other, assessed the effects of a programming course on students’ attitudes toward computing at a four-year college or K-12 setting. I detected that no study in the CS education literature explored or examined students’ attitudes toward CS at public two-year CCs. This apparent gap in the literature at first was deeply disturbing. I wondered why this population of students was overlooked in the literature. This information helped form the basis for a concept to investigate this absent population’s prior experiences with CS and programming and their attitude toward CS and programming.

Computing Attitude Survey of Undergraduate CS Students

Dorn and Tew (2015) designed a computing attitude survey instrument using the “Colorado Learning Attitudes about Science Survey that measures novice-to-expert attitude shifts about the nature of knowledge and problem-solving in CS” for inspiration and guidance (p. 2). Chen et al. (2018, p. 28) created a computing attitude survey of their design with inspiration from the Computing Research Association Undergraduate Survey and the Scientific Attitude Inventory (Moore, Leigh, & Foy, 1997). Both studies evaluated undergraduate college students.
using a pre-and-post instructional intervention for mat at large public universities ranking high in science, technology, mathematics, CS, and engineering. My study, however, turns the attention from students in four-year CS programs to students at two-year colleges, enrolled in a programming-related course, or working toward earning a CS associate degree or CS certificate. Instead of distributing a pre-and-post instructional intervention survey, the participants in this study will complete one survey. The participants’ data will show their various levels of prior experience with computing, the chief independent variable under evaluation.

Gender disparity in computing was the central focus of the Dorn and Tew (2015) study, while Chen et al. (2018) captured broader demographic and socioeconomic data (independent variables) that included racial identity, gender identity, parents’ CS experience, accessibility to a home computer, and parents’ education level. Results of the Dorn and Tew (2015) study found that female-identified participants had statistically significant positive attitude shifts in CS; when they compared pre-course attitude survey results with post-course results, women showed significantly greater increases than men following an introductory computing course. Chen et al. (2018) “found that previous programming experiences, independent of language type (block-based or textual) have positive effects on students’ attitude in college introductory CS coursework (p. 37).

In a similar study of attitude toward CS in the context of K-12, Weintrop and Wilensky (2019) evaluated attitudinal shifts of high school students enrolled in an introductory programming course with an attitudinal survey that was “loosely based on questions from the Georgia Computes project” (Bruckman et al., 2009, p. 3:7). The researchers captured the attitude and interest of students toward CS using a 10-point Likert scale that asked students questions such as “I plan on taking more CS courses after this one” at the outset, midpoint, and conclusion
of the ten-week classes (Weintrop & Wilensky, 2019, p. 9). Results showed “very little quantitative change with no statistically significant difference between time periods within the group or at the same time period across groups” (p. 9). The researchers concluded that the modality of the environment (e.g., text-based or block-based) did not affect enjoyment in programming since learners received the same curriculum. In another study, Wilson, Sudol, Stephenson, and Stehlik (2010) support Weintrop and Wilensky's (2019) results by attributing enjoyment in programming to the curriculum used rather than the programming language or modality with which the CS instruction was delivered.

According to Dorn and Tew (2015) who elaborate on findings that showed a positive shift in attitude toward computing following a CS-related course:

> Researchers across various STEM fields have found evidence of a relationship between student perceptions and learning outcomes within an academic discipline which relates to a person’s emotional reaction to something, the behavioral component, which describes a person’s likely actions in response to a given situation (p. 2).

Of the three components described, the affective component is most intriguing to me for its relationship with the emotional response (p. 2); and, in this case, to CS. I wondered how attitude and identity in the CS domain change among learners at various intervals, such as the changes between no prior experience and one course or between one course and two courses. Therefore, instead of focusing on the subjects of previous investigations, undergraduate students in the context of a four-year university or high school students, I investigated attitudes and beliefs regarding CS with subjects enrolled at CCs, defined as enrolled in at least one CS course.

The survey questionnaire, of my own design, captured participants’ beliefs, opinions, and attitudes toward CS and compared their responses to their prior programming-related curricular experiences. Since this study is quantitative only, an online survey was appropriate for capturing
data. Given the abundance of caution related to COVID-19, I took extreme precautions to minimize face-to-face contact with participants; therefore, no qualitative research methods were used for data collection.

**Research Questions**

Students enrolled in computer science or programming-related courses at any level at a community college were the target population under evaluation. This study aimed to evaluate the findings of two guiding research questions: how did previous programming courses correlate with a favorable identity and attitude toward computer science; and did previous programming courses predict a favorable identity and attitude toward computer science?

To understand the effect of background characteristics on the identity and attitude toward CS, the following two sub questions examined participants’ demographic backgrounds and characteristics:

1. Did demographics like age, race, and gender predict favorable identity and attitude toward CS?

2. What demographic factors, if any, correlated with CC students’ identities and attitudes toward CS, and how?

**Research Methodology and Design**

The independent variables guided the primary research question. Did prior programming experiences, if any, correlate with attitude and identity of CS. Computer science coursework is demarcated, for this study as classroom-based computer science or programming-related instruction.

**Survey Design**

I anticipated gathering specific types of programming experiences of participants’ prior coursework with the questions on prior experience in the questionnaire. For example, the
questionnaire asks the participant about their prior experience with programming. Given
a yes response to this question, the participant was routed to the questionnaire that explicitly
asked the independent variable and students’ attitudes toward CS as the dependent variable in the
survey. Faculty at participating colleges received an email sent by me that included IRB-related
text and a link to the Qualtrics survey questionnaire. The participating faculty members or
instructor, at their discretion, distributed the link to the questionnaire survey to their students in
class or through electronic mail. The survey embedded some validation techniques in which the
participant responded to similar questions, asked inversely, that ensured consistency of responses
(Weibe et al., 2003).

**Population Characteristics**

According to the Community College Research Center (CCRC) at Teachers College
Columbia University, 7.7 million students were enrolled in public two-year degree programs
during the 2019-2020 academic year. The DOE Integrated Postsecondary Education Data System
(IPEDS) Fall 2020 shows the race composition of students enrolled at two public colleges in Fall
2021. The National Student Clearinghouse Research Center (Figure 3.2 and Figure 3.3) shows
gender and age demographics of enrolled students at CCs for that same year and semester.
I searched academic databases for literature in the computing education domain focused on CCs. Approximately twenty papers relevant to this study were downloaded and scanned. To
my knowledge, no other study fitting my search criteria had already empirically evaluated the affective component of students’ attitudes and identity toward CS or programming in the context of CCs. The appraisal of attitudes and identity of CS and programming students at CCs seemed to be a missing link in the CS education literature. The focus of this study was to bring awareness to the attitude and identity of an ignored student population in computer science education. As mentioned in the earlier text, at the present moment, earlier works of similar design focused on one of two learning contexts: K-12 or four-year university settings.

The former research context, led by colleagues, Weintrop and Wilensky (2019), explored the affective characteristic of students at a high school. The results of their study were mixed, but their findings diverged marginally from earlier investigations comparable in magnitude and framework. Others, whose research purposes were directed toward evaluating any negligible shifts in attitude resulting from a college-level computer science instructional intervention, were the empirical inquiries of investigators Dorn and Tew (2015) and Chen et al. (2018). Astonishingly, no study has endeavored an investigation into the academic experiences or sought awareness of the attitudes and identities of students in the context of computer science, programming, or a closely related field of students, at community colleges. Having discovered this wide-open gap in the CSEd literature was alarming given the considerable volume of research in neighboring educational milieus; however, having unearthed an under-researched context for my investigation brought to bear challenges, discussed in future sections, but also excitement. The results of this study are critical to the field of computer science education.

The benefit of scanning through the vastness of literature for similarly designed studies in peer-reviewed conference proceedings or empirical studies published in computing journals
permitted an acquiring of a more profound fluency with the current works of CS in general and the broader context of STEM education at CCs.

As I previously discussed in chapter two, the authors contributed to two-year college CS-related research. I sent email messages in November of 2021 to CS faculty at Portland CC, Houston CC, and Bluegrass Community and Technical College to gauge their interest in this study. I received a favorable response from each institution in which they indicated an interest in distributing my survey to their CS and programming students. The director of the CS program at Houston CC asked to review my survey before submitting an approval request to the department's dean to allow their students to participate in this study. The dean approved the faculty member’s request to allow their students to participate in the research study. A member of the CS faculty at Portland CC agreed to pass the survey on to the instructor of this spring’s programming course. This member of faculty at Portland CC also recommended distributing the survey through the special interest group (SIG) Association of Computing Machinery (ACM), ACM2Y.

Community College CS Faculty Access

I am acquainted with one special interest group for computer science educators working in community colleges. Through their membership channels, the Association of Computing Machinery (ACM) hosts a community college CS faculty interest group, ACM2Y (https://acm2y.acm.org), whose aim “advocates for a diverse group of computing students by building a targeted and resourceful community for faculty of two-year (CCs), higher-education programs” (para. 1).

During phases of early exploration of this study, I reached out to CS faculty in community colleges by email to gauge both support of such and any possible willingness they
might have for distributing my survey out to their students and, or the students of department colleagues. Luckily, my cold calling efforts eventually bore fruits of engagement with the chair of the ACM Committee for Computing Education in CCs. This individual, also a member of the faculty at Portland CC, expressed a friendly curiosity about the study and their support to assist me with distributing my questionnaire in their area.

Another contact at a community college in the Southeast US communicated their support for the distribution of my survey to colleges at their former institution. Although this professor had retired from CS, they offered to assist me in contacting current faculty at their former work establishment. Although these sympathetic connections helped boost my expectation for collecting a large data set, unfortunately, the students who decided to complete the survey. Although thrilled to receive strong interest from well-established CC faculty of CS in large cities, in the end, this study resulted in a small samples size. Later, I will examine what unexpected challenges were met and the root causes, in my view, affected the modest sample size outcome.

**My Role as a Researcher**

Prior to the live distribution of the questionnaire, I had encouraging email communication with faculty at community colleges in Houston, Portland, and Kentucky. The enthusiastic support of these faculty members, who were willing to circulate the survey through their institutions, was met with a tremendous sense of gratitude that also helped amplify a hope of a significant response rate. This assumption is perhaps a novice researcher gaffe that naïvely assumed students in large numbers would complete the survey. Unfortunately, my expectation to achieve a sample size of \( N = 1300 \) was abruptly realized after a slow uptake of the initial
survey distribution. Following week one of capturing survey responses, I reduced my $N$ goal to a new level thought to be more achievable.

Although support backing the distribution of my survey was explicitly indicated by CS faculty at CCs, these supportive individuals could not insist on or require the students to take the survey. The inclination to accept taking the survey was by students’ choice, and those who did take the survey did so without compulsory motive or incentivization. The student who submitted a questionnaire made a valuable contribution to empirical research, free from monetary inducement or academic recompence.

Given that these were just recently established contacts with CS faculty in CCs at a few sites, although encouraging, given their short duration existence, I reflect on this experience as naivety to expect a large response rate from the support of only a few faculty relationships in this field. My lack of contact with administrators, faculty, students, and others in community colleges could have contributed to unexpected barriers to accessing CS students. The layers, gatekeeping, if you will, whose filtering process was the lifeblood of this study, made access to the target subject population challenging. This “layered” dynamic could delineate or rationalize reasoning for a lackluster performance in responses.

In summary, the root cause of a low response rate could be attributed to circumstances out of the purview of the researcher. First, the sole dependence on CS faculty as my survey’s gatekeepers was effective, given the faculty supporting the research. However, a ripe opportunity for future research is designing a study that addresses these challenges of subject access by developing a strategy with access directly to the student population. The distribution strategy taken in this research adversely affected its response outcomes.
On the other hand, faculty were known to the student, and whose email that asked their students to complete the survey might boost the survey requests’ credibility, given the request originated from a familiar person and not from me, a student researcher unacquainted with them.

**Quantitative Statistical Methodology**

Based on the prior works of similar studies on the effects of a CS instructional intervention, I have determined the students’ attitudes and identities, taken together as Likert scale scores toward CS, and programming is the principal dependent variable. The critical independent predictor or correlation variable is the student’s prior programming experience. Other questions about quantities of credit hours previously completed were collected but excluded from this study. Perhaps those data will become pertinent for analysis in a subsequent study

**Independent variables**

The enumerated variables, categorized as nominal or categorical, are matched to the number or quantity of survey question(s) related to that variable.

1. Prior programming experience: one question
2. Racial identity: one question
3. Gender identity: one question
4. Age category: one question

**Dependent variable**

This study's principal dependent variable of interest is the output Likert scale score of questions listed below, which are interconnected to a student’s attitude and identity toward computer science and programming. These enumerated six survey questions were intentionally formulated for maximal measurement of one’s outlook, attitude, or opinion toward computer
science. These questions, extracted from the Weibe et al. (2003) Computing Attitude Survey, were slightly modified to learn more about students' stance, favorable or unfavorable, toward computer science and programming.

The *scores* from the following survey questions were used in this study:

1. I am sure that I can learn programming.
2. I have little self-confidence when it comes to a programming class.
3. I’m not the type to do well in computer programming.
4. Programming to solve problems has an appeal to me.
5. I think computer science and programming are boring.
6. Being regarded as smart in computer science would be a great thing.

Supplementary questions not directly related to the identity and attitude toward CS were included. Two questions asked the participant about the role of women in computer science. One question asked if the respondent believed both men and women should study computer science, while the other question asked the participant if they believed women to be as good as men at programming. In addition, two questions asked the participant to rate the likelihood of programming as an essential skill for future employment. While those questions are important, their data will not be used in this study but are potentially foundational for future evaluations.

**Sample Population Demographics**

The survey respondents had wide-ranging characteristics of gender, race, age, and prior experience with programming. Although the sample size is modest, the descriptive statistics charts and tables below provide a glance into the demographic characteristics of the sample.
Figure 3.4. Demographic Distribution (Race)

Figure 3.5. Demographic Distribution (Gender)
Dorn & Tew (2015) used factor analysis to “statistically identify underlying latent dimensions related to students’ attitude from their data-set” (p. 16). Using a different statistical model, Chen et al. (2018) implemented a Linear Regression Model to identify various predictor variables that affect novices’ attitudes toward CS. Given my familiarity with the regression model from my graduate coursework, and no experience using factor analysis of a data set, I

48
initially decided to implement a regression model for analyzing and evaluating participants’ data. However, those plans were modified after scrutinizing the data against the assumptions for each quantitative model. A regression model is ideal for establishing linearity relationships for ratio data types. Given this study’s variable types and small sample size, models of linearity were excluded.

There were many potential questions to consider before settling on six final questions/statements adapted or modified from the Computer Science Attitude Survey developed by Wiebe et al. (2003) and associates. Given the sharp focus on the identity and attitude of participants relative to their experience or inexperience with programming and computer science, an overabundance of similarly worded questions could confuse and mentally fatigued respondents; therefore, six plainly worded questions neatly framed in the context of computing and programming were selected. The comprehensive enumerated list of survey and demographic questions, directly exported from the Qualtrics survey tool, can be found in Appendix G.

Scores

The term *score* and *scores* are analogous for describing affect. In other words, *scores* are interchangeable with *identity* and *attitude*. Associating a high *score*, or high mean of *scores*, with high affect or a favorable attitude is not valid for all cases. There are varied occurrences in which inverse statements or question produces numerous low *score* (*score < 3*). While each question is unique, frequently, low scores indicate a favorable identity or attitude toward CS. The *score* is meant to describe the choice(s) on a scale of 1 - 5, on a 5-point Likert scale representing the participant’s response. The position of the point on the plot diagram is equal to the value of the *score*. 
Nominal variables

Independent variables, by default, are hypothesized to correlate or predict a relationship with attitude and identity toward CS. Most of the independent variables under evaluation for this research fall into the variable type known as categorical or nominal, meaning sequence of order or scale is not among a property of this variable type. Within SPSS, categorical variables are assigned as nominal (e.g., gender, race, and age).

Central Limit Theorem

Using properties of the Central Limit Theorem, Herzog, H., Francis, G., & Clarke, A. (2019) suggest including the median and mode to the mean of Likert scale responses. The median and mode can be used in conjunction with conducting an independent-samples -test and a one-way ANOVA (analysis of variance) to substantiate the median and mode scores to corroborate findings using the Central Limit Theorem that is explained later.

Median and Mode

Because ordinal and nominal data types are discontinuous, utilization of Pearson’s r, a model that assumes the dependent variables are of ratio type and continuous for estimating statistical significance between variables, is unsuitable for this study’s data types. The ordinal, dependent variables being evaluated were extracted from Likert scale responses originating along a “1” (strongly disagree) to “5” (strongly agree) interval scale. I converted Likert scale data, initially categorized as ordinal, to interval or scaled data, in SPSS to evaluate m (the mean/average), thus compatibility with the t-test and one-way ANOVA assumptions. According to Herzog and colleagues (2019) claim, the analysis of means through the conversion of ordinal variables (Likert scale scores) into an interval data type, then, after conversion, those data are run
through a series of -tests or ANOVA evaluations, can supplement other measures of central tendency (e.g., median, mode).

**Terminology**

Concepts and terminology in quantitative statistics frequently comprise multiple words or terms with synonymous meanings. For example, $t$-value, standard effect size, and Cohen’s $\delta$ are defined similarly according to Herzog et al. (2019, p. 47). Pertaining specifically to this document, the use of any of those terms can be assumed as having synonymous meaning.

**Unsuitable Measures of Central Tendency**

Pearson’s Correlation Coefficient or, just simply, Pearson’s $r$ is inappropriate for this study because an assumption of Pearson’s $r$ states that variables being evaluated must be of type ratio. The independent variables of this study consist mainly of nominal variables. As previously stated, Pearson’s $r$ is incompatible for evaluating categorical data.

The population means of the target population are unknown given the $N = 39$ sample size, sample size; therefore, for this study, the sample means are approximated and represented as $\bar{X}$ or $\bar{x}$. As specified, when the population mean standard deviation, represented as $\sigma_X$ cannot be determined given the $N = 39$ sample size, a sample estimate of the standard deviation, represented as $s_X$ can estimate the sample’s standard error when $\sigma_X$ is unknown (Herzog, 2019, p. 53).

**Suitable Measures of Association**

To measure levels of association between variables in this study, I used SPSS statistical software for measuring variances, if any, of means among variables or groups. I selected the independent-samples $t$-test and the one-way ANOVA to evaluate mean variances. Pearson’s Correlation Coefficient or Pearson’s $r$ was determined unsuitable for evaluation of
discontinuous, categorical variables, in conjunction with ordinal or interval variables, thus, the independent-samples \( t \)-tests and an ANOVA were most appropriate.

**Independent Samples \( t \)-test**

Each evaluation of variables is intended to estimate the association between variables. The following equation displays the hypothesis process for rejecting or not rejecting the null hypothesis based on an independent sample \( t \)-test. The null hypothesis is not rejected if means of group one and two are equal, thus, no variances between group means. The alternative hypothesis states groups mean are unequal.

The null hypothesis represented as: \( H_0: \mu_1 = \mu_2 \) where there is no variance between sample means (Herzog, 2019, p. 73).

The alternative hypothesis represented as \( H_a: \mu_1 \neq \mu_2 \) where sample means are unequal (Herzog, 2019, p. 73).

The critical value set at \((t_{cv} = +1.96)\) and \( p \) – value \((p < .05)\). A \( p \) – value whose result \((\text{Sig.}) = p < .05\) denote a statistically significant finding. The null hypothesis \( H_0 \) is rejected for the alternative hypothesis \( H_a \) in those cases.

**One-way ANOVA**

For this study, an independent-samples \( t \)-test, previously described, and the one-directional ANOVA are germane to this study and robust in SPSS for estimating the links and variability among variables, thus, an avoidance of any Type I or Type II error (Herzog, M. H., Francis, G., & Clarke, A., 2019, p. 47). Herzog et al. (2019) stated that, like a \( t \)-test hypothesis, ANOVA assumes:

The null hypothesis is represented as: \( H_0: \mu_1 = \mu_2 = \mu_3 \) where all sample means are equal (p. 115).
The *alternative hypotheses* is represented as: $H_a: \mu_1 = \mu_2 \neq \mu_3$ where at least one sample mean is unequal to another sample mean (p. 115).

To find the $F$-value using a one-way ANOVA, the variances between group means is divided by the variance within group means, shown in the following equation:

$$F = \frac{\text{Variance between group means}}{\text{Variance within group means}}$$

Herzog et al. (2019) also suggested observing the effect size, which can be represented as another element in evaluating significant results computed with a one-way ANOVA (p. 123). According to Herzog et al. (2019), “the effect size ratio tells you the proportion of total variability in the data explained by variability due to the treatment means” (p. 123). The effect size using Cohen’s $\delta$ informed relevant results, thus, revealing the degree of variability among means. Computing Cohen’s $\delta$ was determined with the following *effect size* equation:

$$\eta^2 = \frac{SS_{\text{between}}}{SS_{\text{total}}}$$

Table 3.1 presents effect size categories according to Cohen (p. 124). The data comprised in the table served for its referencing and interpreting use for determining effect size of output data of *t*-test data only (Herzog et al., 2019, p. 124).

<table>
<thead>
<tr>
<th>Cohen’s $\delta$</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Effect Size</strong></td>
<td>0.01</td>
<td>0.09</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 3.1. Effect Size Guidelines According to Cohen
Conclusion

I carefully considered various statistical models for computing and analyzing these data before choosing the two models most applicable for this study’s aim. On a side note, once the variable types were firmly established in SPSS, supported by the literature, and ran in SPSS without incident, a well-informed decision that justified the use of the $t$-test and the one-way ANOVA was achieved. These details are explained further in Chapter Four.
CHAPTER FOUR. RESULTS

The original design of this study called for regression analysis of the scores (independent variables) taken from the survey data in the form of Likert scale responses. Also, Pearson’s $r$ was initially assumed as most appropriate for analyzing independent and dependent variables before I ran a series of analyses in SPSS.

My original strategy summed the total score of each participants’ answers to all questions related to attitude and identity toward CS. From these total scores, I ran my first sequence of analysis in SPSS with Pearson’s $r$. Immediately, I noticed issues with the output. First, three of six questions in the survey were inversely asked or stated. These inversely asked questions were not necessarily a mirror reflection of an unfavorable attitude. Unwittingly, given the “1” or “2” value along the scale, inversely asked questions lowered total scores. Such as question one: “I think computer science and programming are boring.” From a Likert scale response perspective, one with a highly favorable attitude toward CS is likely to select “1” (strongly disagree) or at minimum “2” (somewhat disagree); whereas a participant whose attitude toward CS leans unfavorably is apt to choose the option “5” (strongly agree) or “4” (somewhat agree).

Three of the six attitude-identity-related questions were inversely asked or stated; thus, when I summed scores of all six questions, errors were realized. I assumed a high sum was indicative of a favorable attitude and a low score an unfavorable one. Problems soon emerged. To repair this error, questions were parsed and separated. Each, standing on its own as a dependent variable against which independent variables were compared via the t-test, one-way ANOVA, or a combination of both tests.

The demographic characteristics of the sample population were discussed and shown graphically in Chapter Three in the section titled Sample Population Demographics within
frequency distribution charts framed around the *prior programming experience* questions. The frequency figures show the demographic distribution of an \( N = 39 \) sample population:

- Figure 3.4. Demographic Distribution (Race)
- Figure 3.5. Demographic Distribution (Gender)
- Figure 3.6. Demographic Distribution (Age)
- Figure 3.7. Demographic Distribution (Ethnicity)

**Histograms**

Six histogram charts below are based on the frequency of participants’ responses to each question related to a favorable or unfavorable attitude or identity toward CS. The horizontal x-axis denotes the range of Likert scale responses to the question that is stated at the top of the chart. The vertical y-axis references the frequency of response, 1 – 5, for each survey question. As previously mentioned, the Likert scale is 5 points from 1 to 5 where “1” represents a strong disagreement to the question or statement; and, where “5” represents a strong agreement to it.

**Question One Frequency Histogram**

Figure 4.1 shows the number of responses to question one: “I am sure I can learn to program.” This question asks the students to self-assess confidence in learning programming with a rating along the 5-point scale. The sample of responses to this confidence-based or identity-based question produced a sample mean score of 4.44, about the midpoint between somewhat agree and strongly agree. The scores illustrated in Figure 4.1 are skewed right. The high mean and narrow 1.119 standard deviation is enough data of central tendency to claim this sample of students somewhat confident in learning programming.

For clarification, a participant whose scores high on confidence with programming is categorized as having a favorable attitude and identity toward CS and programming. On the
contrary, a low score is associated with low confidence in programming and thus is interpreted for this study’s purposes as a respondent whose attitude or identity toward CS is unfavorable. Although the mean scores for this question skew right to favorable, let it be known now that this graph was but one data point of the whole sample, excluding any predictor variable. While this mean data is interesting at a minimum, this study aimed to learn which independent and demographic variables, if any, were associated with attitude and identity toward CS and programming variables, if any, can be associated with attitude and identity toward CS and programming.

Figure 4.1. Question One. Frequency Histogram

**Question Two Frequency Histogram**

The mean, mode, and standard deviations for question two are displayed in Figure 4.2. These responses are akin to the previous question; however, the degree of similarity is inversely related. The main distinction between the previous question that asked the subject to rate their
self-confidence in programming and the second question is the positioning of self-confidence in
the question. Where question one asked the subject to agree or disagree with the statement
themed around confidence, this question asks the student to rate their agreement or disagreement
with having little self-confidence concerning a programming class.

![Frequency Histogram](image)

**Figure 4.2. Question Two. Frequency Histogram**

This inverse question/statement produced a sample mean of 2.28; between option “2”
(somewhat disagree) than “3” neither agree nor disagree. As revealed in Figure 4.2, the mode of
“1” signals that most disagreed with the question’s statement of little confidence in
programming. The 1.38 standard deviation, while still somewhat slim but not as narrow as
question one’s standard deviation, does shows more variance around the mean responses.
Another addendum to note is the spike at option “4” whose answers somewhat agreed to the
statement of one with little self-confidence in programming or for interpretation purposes, an
unfavorable identity toward programming.
Question Three Frequency Histogram

The first identity-related question presented in the survey to students asked the respondent to rate their agreement or disagreement with the statement: “I am not the type to do well in computer programming.” An inverse-themed question shows that most disagree with this statement, as displayed in Figure 4.3. Not only did most answers fall left of the neutral median, but the answers also skewed strongly left. A snug .922 standard deviation around the 1.69 sample mean conferred the degree to which students disagreed with the statement “I am not the type to do well in computing” falls somewhere between strongly and somewhat disagree. Note, in contrast, at least two responses somewhat agreed with the question.

Figure 4.3. Question Three. Frequency Histogram

For one, the sample of students who took this survey elected enrollment in a programming course, and second, they were motivated enough to set aside time for completing
this questionnaire. Given those two scenarios, having at least two respondents who agreed with being *not the programming type* is a thought-provoking discovery.

**Question Four Frequency Histogram**

Figure 4.4 represents responses toward having a positive *affect* toward CS and programming. This “appeal with programming” question produced the greatest sample mean of 4.56 and a minor standard deviation of .821. Although a few responses fell left of the median, indicative of neutral or apathy towards CS, the mode of 5, high mean, and strongly skewed right histogram, explains those sampled agreed beyond somewhat that CS and programming are appealing.

![Frequency Histogram](image)

**Figure 4.4. Question Four. Frequency Histogram**

**Question Five and Six Frequency Histograms**

Figure 4.5 unveils a mode of “1” and a mean of 1.69. These data points confirm that most students in this sample disagree with the survey stating that “CS and programming are boring.”
A mean of 1.59 and a small .91 standard deviation help strengthen an interpretation of this histogram that most in this sample perceive CS and programming as the opposite of boring. A point worth noting, only one respondent agreed with the statement that CS and programming are boring. Given these data points, I can conclude that most students surveyed have a positive affect toward CS and do not agree that CS or programming is boring.

![Frequency Histogram](image)

**Figure 4.5. Question Five. Frequency Histogram**

The final question themed on attitude and identity toward CS asks the student to rate their agreement or disagreement with a question related to being regarded as *smart* in CS as a great thing. The mode, mean, and SD are nearly identical to question four which asked students to rate the appeal of CS. The 4.44 mean and 0.94 standard deviations are confirming evidence pointing in the direction of most students agreeing that being regarded as someone smart in CS would be a great thing.
Figure 4.6. Question Six. Frequency Histogram

Being regarded as smart in computer science would be a great thing.
Analysis of Histogram Data

Reviewing survey responses for each attitude and identity-themed question in histogram format yielded fresh previews of sample-wide data relative to the six survey questions or statements. The next logical task is a deeper dive into data outputs taken from sequences of testing performed in SPSS, then looking at those data for sense-making results as the degree to which these statistical findings impart insight for answering research questions one, two, and three.

Mentioned multiple times in the previous text, two questions asked the respondent to rate their agreement or disagreement with CS-identity-related questions or statements. Questions of attitude with identity toward CS offer a broader perspective of students’ identification with the CS and programming learning domain. Questions related only to attitude and appeal of CS and programming omit an important factor of identification with the subject matter. The survey questions were chosen, and later, the parsing of questions as stand-alone dependent variable scores afforded a broad landscape of opportunity for measuring associations of both identity and attitude with prior programming experience and demographic factors.

In summary, the research design aimed at detecting students whose attitude toward computing is favorable, yet because of inexperience with programming or a cultural background characteristic, the student lacks connections with the subject, thus an unfavorable identification to CS context. Without both attitude and identity questions taken together, it could be assumed my results found could be either inaccurate, worst-case scenario, or incomplete, at best. The rationale for each statistical methodology has been established; equations illustrated their relevance for supplementing results shown, the process for hypothesis testing, and at which level critical values were set for analyzing these data. Given that the groundwork has been laid to
inform methodology, the focus shifts now to SPSS-computed results and interpretations of findings.

**Question One: How do previous programming courses correlate or predict a favorable identity and attitude toward computer science?**

The review of histogram data only provided measures of central tendency for frequency, mode, mean, and standard deviation. Although Likert scale responses can be used for measures of central tendency of the whole group, when factoring independent, nominal variables such as students' prior experience and demographic characteristics, other analytical measures of association become particularly advantageous for parsing and fine-tuning data for sense-making and drawing inferences. The aim is to determine the degree to which the independent variable(s) are associated with *scores* Output data relative to the relationship between or among variables is the empirical validation process essential for substantiating the claim(s) I make by answering my three research questions.

**Critical Values & Confidence Intervals**

Thus, the methodology examined in chapter three described how determining the independent samples *t*-test and one-way ANOVA statistical models were most applicable. As noted in chapter three, the alpha level was set to 0.05 (*α* = 0.05). Given a level, assume each confidence interval was set at (95%); thus, a *significant* result is equivalent to (*p* < .05)

The first statistical analysis of these data was used *t*-tests for comparing two groups. These analyses are comparing differences in means and significance of variance, if any. Two groups of independent variables were categorized as either having prior programming experience (group one) or no prior programming experience (group two).
Only one question of the six discussed in subsequent sections is considered for $t$-test evaluation for its most direct affect-related tone. The full text of the survey question is exhibited in Table 4.1 on the far-left in the grey-shaded cell: “Programming to solve problems has an appeal to me.” The second column denotes answers to the survey question: ($1 = $yes$)$ and ($2 = $no$)$. Given the proximity of group means, 4.67 and 4.40, respectively, non-significant results between variables, as shown in Table 4.2, are unsurprising. The question asked the student to rate their programming appeal for solving problems. The responses to this question evaluated produced a low variance between means results (orange bordered cells are highlighted to draw attention to enclosed data and their annotations denote *exciting* but *not* significant results). The standard deviations of each group, as well as the standard error of the mean of both groups, further justify the similarities of group means.

While interesting, the shown examples, Table 4.1 and Table 4.2, attempt to correlate prior experience to only one survey question. In succeeding sections, all six questions are evaluated against variables to illustrate a grander picture of any relationships between or among variables for predicting a favorable attitude and identity toward CS and programming.

<table>
<thead>
<tr>
<th>Group Statistics</th>
<th>I have prior experience with computer programming?</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming to solve problems has an appeal to me</td>
<td>1</td>
<td>24</td>
<td>4.67</td>
<td>.565</td>
<td>.115</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>15</td>
<td>4.40</td>
<td>1.121</td>
<td>.289</td>
</tr>
</tbody>
</table>

Levene’s test for equality of variance (Table 4.2) shows no statistically significant findings in either the $F$-value or $P$-value; therefore, mean variances were similar, thus an insignificant result.
Table 4.2. T-Test (Attitude & Programming Experience)

<table>
<thead>
<tr>
<th></th>
<th>Levene's Test for Equality of Variances</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>F</td>
<td>Sig.</td>
<td>t</td>
<td>df</td>
</tr>
<tr>
<td>Programming to solve problems has an appeal to me.</td>
<td>Equal variances assumed</td>
<td>3.676</td>
<td>.062</td>
<td>.987</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>Equal variances not assumed</td>
<td></td>
<td></td>
<td>.856</td>
<td>18.507</td>
</tr>
</tbody>
</table>

Table 4.3. Cohen’s δ / Effect Size (Attitude & Programming Experience)

<table>
<thead>
<tr>
<th></th>
<th>Standardizer</th>
<th>Point Estimate</th>
<th>95% Confidence Interval</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming to solve problems has an appeal to me.</td>
<td>Cohen's d</td>
<td>.821</td>
<td>.325</td>
<td>-327</td>
<td>972</td>
</tr>
<tr>
<td></td>
<td>Hedges' correction</td>
<td>.836</td>
<td>.318</td>
<td>-329</td>
<td>952</td>
</tr>
<tr>
<td></td>
<td>Glass's delta</td>
<td>1.121</td>
<td>.238</td>
<td>-417</td>
<td>865</td>
</tr>
</tbody>
</table>

a. The denominator used in estimating the effect sizes.
Cohen’s d uses the pooled standard deviation.
Hedges’ correction uses the pooled standard deviation, plus a correction factor.
Glass's delta uses the sample standard deviation of the control group.

The examples (Table 4.1, Table 4.2, & Table 4.3) were discussed as an introduction to future analysis between a dependent variable analyzed with two independent variables (group one and group two). Throughout future analyses, the one-way ANOVA has greater capacity in computing results of multiple dependent variables simultaneously. For example, the six questions, from which an estimated association of the relationships between variables, such as prior programming experience; or age, race, and gender, are computed. In summary, the aim of analyzing the results was to identify significant variable correlations.

One-Way ANOVA

The initial one-way ANOVA analysis examines the relationship between prior programming experience (independent variable) with the six attitude-related questions.
(dependent variables). Table 4.4 reveals two significant relationships between the independent and dependent variables (yellow bordered cells are highlighted to draw attention to enclosed data – and their annotations denote a significant result). Specifically, the fourth and fifth questions, shown along the left side of the table, resulted in a significant difference between mean values.

The mean difference between the scores of the fourth question related to self-confidence with programming and prior experience were statistically significant: \( F(1, 37) = 6.214, p = 0.017 \). A result of 0.033 less than \( \alpha = 0.05 \). Prior programming experience is a significant predictor for identifying as not the type to do well in programming. The strength of the relationship between the two variables is substantiated by a significant result found in the fourth question: \( F(1, 37) = 16.625, = p < 0.001 \).

Table 4.4. One-way ANOVA (Attitude & Programming Experience)

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I am sure that I can learn programming.</td>
<td>Between Groups</td>
<td>3.323</td>
<td>1</td>
<td>3.323</td>
<td>2.778</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>44.267</td>
<td>37</td>
<td>1.196</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>47.590</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming to solve problems has an appeal to me.</td>
<td>Between Groups</td>
<td>.666</td>
<td>1</td>
<td>.656</td>
<td>.974</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>24.933</td>
<td>37</td>
<td>.674</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>25.590</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Being regarded as smart in computer science would be a great thing.</td>
<td>Between Groups</td>
<td>.256</td>
<td>1</td>
<td>.256</td>
<td>.285</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>33.333</td>
<td>37</td>
<td>.901</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>33.590</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have little self-confidence when it comes to a programming class.</td>
<td>Between Groups</td>
<td>10.339</td>
<td>1</td>
<td>10.339</td>
<td>6.214</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>61.558</td>
<td>37</td>
<td>1.654</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>71.897</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I'm not the type to do well in computer programming.</td>
<td>Between Groups</td>
<td>10.016</td>
<td>1</td>
<td>10.016</td>
<td>16.025</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>22.292</td>
<td>37</td>
<td>.602</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>32.308</td>
<td>38</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I think computer science and programming are boring.</td>
<td>Between Groups</td>
<td>.503</td>
<td>1</td>
<td>.503</td>
<td>.601</td>
</tr>
<tr>
<td></td>
<td>Within Groups</td>
<td>30.933</td>
<td>37</td>
<td>.836</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>31.436</td>
<td>38</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Two mean plot diagrams shown in Figure 4.7 and Figure 4.8 illustrate the interactions between students’ prior programming experience and their attitude toward CS with similarly worded questions asked inversely.

![Plot Diagram](image)

**Figure 4.7. Mean Plot Diagram (Favorable Attitude & Programming Experience)**

The results of a $t$-test and one-way ANOVA demonstrate no significant association between questions two and four and prior programming experience, the illustration of means in a plot graph display thought-provoking findings. For example, the question asked the student to rate their appeal with CS, while question two asked the student to rate CS as “boring.” The diagrams corroborate findings given that those whose background did not include prior programming experience scored high toward an agreement that CS is boring and low in agreement that CS is appealing.
The findings become more interesting looking at the mean scores of students whose background included having prior programming experience. The mean score of students with prior programming experience is higher for the question that CS is appealing and lower for the question that CS is boring. These findings add further evidence supporting my hypothesis (e.g., programming experience = more favorable attitude toward CS. Also, ubiquitously found in the CS education literature are conclusions that purport any exposure to programming pedagogy correlates with having a more favorable attitude toward CS and programming than those unexposed to CS curriculum.

**Question One Summary**

Two of the six questions correlated as statistically significant for predicting a favorable attitude towards CS. While the remaining questions showed no significant relationship with prior programming experience based on their $P$-value, the mean plot diagrams illustrate that prior programming experience influences responses, albeit tiny.

Figure 4.8. Mean Plot Diagram (Unfavorable Attitude & Programming Experience)
Further exploration of the findings around the questions of CS as “appealing” and “boring” is of interest and given larger sample size, perhaps the $P$-value of these results will lean in favor of a wider variance between means of the dependent variables when analyzed with prior programming experience. In summary, the analysis of these data confirms that prior programming experience has a noteworthy relationship with a favorable attitude and identity toward CS and programming.

**Question Two: Do demographics like age, race, and gender predict favorable identity and attitude toward CS?**

I performed analyses to approximate any association of significance between attitude and identity toward CS and demographics like age, race, and gender. Given the categorical variable types of the demographic variables, as discussed in detail in Chapter three, two statistical models were selected for their relevance to this study. Since the categorical options for age and race exceed two, $t$-tests were not performed; they exceeded a maximum of two groups. The only demographic variable fitting for $t$-test in research question number two is gender as binary (two groups maximum). To ensure a nominal variable like gender can be analyzed with $t$-tests, dependent variables must meet $t$-test assumptions/requirements for analyses.

**$T$-test assumptions**

The only assumption of the $t$-test that required deeper analyses to confirm no violation of the test occurs, the Likert scale scores must be structured and categorized appropriately. According to Herzog et al. (2019) “the dependent variables must be on a ratio scale of measurement” (p. 97), thus, calculating the mean of nominal and ordinal is inappropriate for these variable types. Herzog suggests much debate surrounds Likert scale data and how to properly categorize it. For purposes of this study, however, the Likert scale scores collected has
been assigned as interval scale data. Ideally, ratio scale data is best for \( t \)-testing, however, Herzog suggests “in many cases the \( t \)-test behaves rather reasonably for interval data” (p. 97). Given the authors’ endorsement of interval data for use in \( t \)-tests, it seems other assumption get violated at the running of \( t \)-tests for estimating the relationship between attitude toward CS and scores. The \( t \)-test results of gender only add additional, confirmatory, data to the one-way ANOVA findings. The \( t \)-test findings on gender are discussed further in a succeeding section.

**Age and Attitude Results**

**Descriptive Statistics**

Over fifty percent \((n = 22)\) of the students surveyed fell into the youngest of five age categories (18 to 24 years of age). The remaining forty-four percent of students \((n = 17)\) were interspersed among the four other (older) age categories demarcated in 5-year increments shown in the second-most left column of Table 4.5. Survey scores of the six questions related to attitude and identity toward CS were evaluated with age categories reported by the respondents.

Three interesting findings illustrated in Table 4.5 are annotated with highlighted borders to connect two data points (purple, green, and turquoise bordered cells are highlighted to draw attention to enclosed data – these data are *not* significant). First, in turquoise, students in the 25 to 29 age categories show the second-highest mean relative to the first question about confidence with programming; yet, that result could be contradicted with the highest of any other group for the third question about having *little* confidence in programming. Annotations in purple and green illustrate other interesting contradictory findings worthy of further exploration.
Table 4.5. Descriptive Statistics (Attitude & Age)

<table>
<thead>
<tr>
<th>Attitude &amp; Age</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std Error</th>
<th>95% Confidence Interval for Mean</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>I am sure that I can learn programming.</td>
<td>18-24</td>
<td>22</td>
<td>4.59</td>
<td>5.18</td>
<td>3.24</td>
<td>4.94</td>
<td>2</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>25-29</td>
<td>5</td>
<td>4.60</td>
<td>5.49</td>
<td>3.92</td>
<td>5.28</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-34</td>
<td>6</td>
<td>4.33</td>
<td>1.63</td>
<td>6.67</td>
<td>3.22</td>
<td>6.05</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>35-44</td>
<td>2</td>
<td>5.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.00</td>
<td>5.00</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>45+</td>
<td>4</td>
<td>3.35</td>
<td>2.62</td>
<td>1.03</td>
<td>1.33</td>
<td>3.83</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>39</td>
<td>4.44</td>
<td>1.11</td>
<td>0.79</td>
<td>4.07</td>
<td>4.80</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

| I am programming to solve problems has an appeal to me. | 18-24 | 22  | 4.50 | 8.73 | 4.20 | 4.80 | 3 | 5 |
| 25-29 | 5  | 5.00 | 0.00 | 0.00 | 5.00 | 5.00 | 5 | 5 |
| 30-34 | 6  | 4.67 | 5.10 | 2.11 | 4.12 | 5.21 | 4 | 5 |
| 35-44 | 2  | 5.00 | 0.00 | 0.00 | 5.00 | 5.00 | 5 | 5 |
| 45+ | 4  | 4.00 | 2.00 | 1.00 | 0.82 | 7.18 | 1 | 5 |
| Total | 39 | 4.56 | 8.21 | 1.31 | 4.30 | 4.83 | 1 | 5 |

| I have little self-confidence when it comes to a programming class. | 18-24 | 22  | 4.68 | 5.88 | 4.43 | 4.93 | 3 | 5 |
| 25-29 | 5  | 4.40 | 8.54 | 3.29 | 5.61 | 3 | 5 |
| 30-34 | 6  | 4.00 | 1.35 | 0.51 | 2.67 | 3.33 | 2 | 5 |
| 35-44 | 2  | 5.50 | 0.70 | 2.50 | 1.85 | 10.85 | 4 | 5 |
| 45+ | 4  | 3.75 | 1.83 | 0.94 | 0.74 | 6.76 | 1 | 5 |
| Total | 39 | 4.44 | 9.40 | 1.51 | 4.13 | 4.74 | 1 | 5 |

| I'm not the type to do well in computer programming. | 18-24 | 22  | 3.40 | 1.47 | 1.73 | 5.07 | 2 | 5 |
| 25-29 | 5  | 3.40 | 1.34 | 0.60 | 1.73 | 5.07 | 2 | 5 |
| 30-34 | 6  | 1.50 | 5.49 | 0.24 | 0.93 | 2.07 | 1 | 2 |
| 35-44 | 2  | 1.50 | 7.07 | 5.00 | -4.85 | 7.85 | 1 | 2 |
| 45+ | 4  | 1.25 | 6.00 | 2.50 | 0.45 | 2.05 | 1 | 2 |
| Total | 39 | 2.26 | 1.37 | 0.20 | 1.84 | 2.73 | 1 | 5 |

| I think computer science and programming are boring. | 18-24 | 22  | 1.77 | 1.06 | 1.30 | 2.25 | 1 | 4 |
| 25-29 | 5  | 1.77 | 1.06 | 0.22 | 1.30 | 2.25 | 1 | 4 |
| 30-34 | 6  | 1.67 | 8.16 | 0.33 | 0.81 | 2.52 | 1 | 3 |
| 35-44 | 2  | 1.00 | 0.00 | 0.00 | 1.00 | 1.00 | 1 | 1 |
| 45+ | 4  | 1.00 | 0.00 | 0.00 | 1.00 | 1.00 | 1 | 1 |
| Total | 39 | 1.69 | 0.22 | 0.14 | 1.39 | 1.99 | 1 | 4 |

One-Way ANOVA

The results revealed from analyzing Table 4.5 descriptive statistics data offered slight glimpse into the appearance of a weak relationship between six questions on attitude and identity toward CS and categories of age. I tested the same variables with a one-way ANOVA model for
comparing variances of means among the groups. In chapter three, the null hypothesis assumes group means are equal; therefore, “any observed differences in the sample means comes from the variance” (Herzog et al., 2019, p. 112). To reject the null hypothesis that assumes all group means are equal, a $P$-value or $p < 0.05$ must result. Discussed earlier, but restated again for emphasis, the alpha level ($\alpha = 0.05$) determines where to fix the confidence interval. The confidence interval is calculated. In this case, the following $(1 - \alpha = 0.95)$ represents the process for setting the confidence interval. In SPSS, we convert the decimal 0.95 to its percentage equivalent (e.g., 95%). This critical value is the basis for evaluating the statistical significance of the $P$-value of ($p < 0.05$).

<table>
<thead>
<tr>
<th>Table 4.6. One-way ANOVA (Attitude &amp; Age)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANOVA</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Sum of Squares</td>
</tr>
<tr>
<td>--------------------------------------</td>
</tr>
<tr>
<td>I am sure that I can learn</td>
</tr>
<tr>
<td>programming.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>Programming to solve</td>
</tr>
<tr>
<td>problems has an appeal</td>
</tr>
<tr>
<td>to me.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>Being regarded as smart</td>
</tr>
<tr>
<td>in computer science</td>
</tr>
<tr>
<td>would be a great thing.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>I have little self-</td>
</tr>
<tr>
<td>confidence when it</td>
</tr>
<tr>
<td>comes to a programming</td>
</tr>
<tr>
<td>class.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>I’m not the type to do</td>
</tr>
<tr>
<td>well in computer</td>
</tr>
<tr>
<td>programming.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
<tr>
<td>I think computer science</td>
</tr>
<tr>
<td>and programming are</td>
</tr>
<tr>
<td>boring.</td>
</tr>
<tr>
<td>Between Groups</td>
</tr>
<tr>
<td>Within Groups</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>
The results revealed in Table 4.5 shows computation using the one-way ANOVA model reported no statistically significant findings. Question four: “I have little self-confidence when it comes to a programming class” resulted in a nearly significant result: $F(4,34) = 2.52, p = .059$, $P$-value 0.009 greater than the set critical value ($\alpha = 0.05$).

The mean plot diagrams shown in Figure 4.9 and Figure 4.10 confirm the low variability of means among groups. The combination of descriptive statistics, statistics reported in the one-way ANOVA, and mean plot diagrams verify no statistically significant relationship between age and favorable attitude and identity toward CS and programming for this sample population.

![Figure 4.9. Mean Plot Diagram (Favorable Attitude & Age)](image-url)
Gender and Attitude Summary

Though the results of the analyses rendered zero statistically significant $P$-values, this could be attributed to a small sample size ($N = 39$) divvied up among five groups; moreover, a single age group contained over fifty percent of the sample responses. Further research of age factors and their implication on attitude and identity toward CS could be explained better given a substantially larger sample size ($N$). A sample size of at least $N = 200$ where groups are dispersed equally across age categories might accurately approximate the relationship between age and attitude toward CS and programming. The findings of this analysis do offer some insight into that relationship; however, an expanded sample could substantiate or refute these conclusions.

Race and Attitude Results

As shown in Table 4.7, the column designates the number of respondents for each category related to race. Race is a nominal variable type with categories; therefore, options for
evaluating the associations between variable is limited to descriptive statistics and a one-way ANOVA. The one-way ANOVA model using scaled Likert scores of participants, as mentioned in the previous section on age, is appropriate for evaluating these relationships between race and attitude toward CS.

Table 4.7 illustrates the mean, standard deviation, and standard error of the nominal variable (race) with the scaled interval variable (scores). Two findings of interest are in the first question, “I am sure I can learn to program.” Black or African American students surveyed reported the highest agreement (high score) with a 0.05 standard deviation. Question five, “I am not the type to do well in programming,” resulted in a high disagreement (low score) of Black or African American students. These results are taken together to validate a favorable attitude and identity toward CS for Black and African American students whose representation in CS programs is lower than their Asian and white peers. These results point to promising news of an increasingly favorable perception in attitude and identity toward CS for Black or African American CS students.
Race is a significant predictor of attitude toward CS and programming based on findings from the one-way ANOVA model. Based on these findings, race is the greatest predictor of affect toward CS and programming. More research of students in CS community college programs should be conducted to validate these outcomes. The ($n=4$) of Black and African American respondents not a considerable size; therefore, while this is a remarkable result, more research is needed to make grand inferences to the greater population. Based on the results, I
rejected the null hypothesis that all means are equal; specifically, for questions one, three, and six.

Table 4.8 shows the scores from three of six questions resulted in statistically significant $P$-values. The scores of the first question revealed a statistically significant result of $(F(4,33) = 3.74, p = 0.013)$ that demonstrates a relationship between race and attitude toward CS. The third and sixth question showed a similar pattern of statistically significant results of $(F(4,33) = 2.80, p = 0.041)$ and $(F(4,33) = 2.91, p = 0.036)$ respectively.

Based on these findings, race is the greatest predictor of affect toward CS and programming. More research of students in CS community college programs should be conducted to validate these outcomes. The $(n=4)$ of Black and African American respondents not a considerable size; therefore, while this is a remarkable result, more research is needed to make grand inferences to the greater population.\(^1\) Based on the results, I rejected the null hypothesis that all means are equal; specifically, for questions one, three, and six.

\(^1\) Additional analyses were conducted with combined racial categories, but results did not significantly differ.
The mean plot diagrams shown in Figure 4.11 and Figure 4.12 uncover an interesting finding. I might expect white and Asian races to have the highest mean appeal of CS and the lowest mean of dissatisfaction toward CS, given the stereotype commonly referenced in the literature. However, Black, or African American students rated their appeal toward solving programming problems as a factor they strongly agree with. The second mean plot confirms this with Black or African American students’ disagreeing with the statement that programming is boring in greater magnitude than their Asian and white peers.
Race and Attitude Results Summary

Race is estimated as having the greatest correlation and predictive relationship among demographic variables with a favorable attitude toward CS and programming. These results
indicate shifts towards a favorable attitude and identity of CS in groups historically underrepresented in CS domains. Further exploration with a larger sample size of students at community colleges enrolled in programming-related courses is needed to confirm these preliminary results.

**Gender and Attitude Results**

In the same way as the prior programming experience variable, the gender variable can be evaluated with a corresponding Likert score to estimate any associations between the nominal gender variable and the interval score variable. One respondent selected transgender male as their gender identity; therefore, they were assigned to the male group to adhere to the two-groups maximum for t-testing.

The interactions between variables of attitude and gender indicate no statistically significant relationship between the variables. Although a third of the respondents identified as female (n=13) and two-thirds identified as male (n=26), no signals of significance between sample means are seen. As illustrated in Table 4.9 and graphically displayed as mean plot diagrams of Figure 4.13 and Figure 4.14, the difference of means between groups is μ < 0.20. Both groups’ standard deviations are worth noting. Although the male group shows a slightly higher mean than the female group, both groups’ mean land approximately at the midpoint, between “4” somewhat agree, and “5” strongly agree with the statement “programming to solve problems has an appeal to me.”
Table 4.9. T-Test (Attitude & Gender)

<table>
<thead>
<tr>
<th>Group Statistics</th>
<th>N</th>
<th>Mean</th>
<th>Std. Deviation</th>
<th>Std. Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming to solve problems has an appeal to me.</td>
<td>Female</td>
<td>13</td>
<td>4.46</td>
<td>1.127</td>
</tr>
<tr>
<td></td>
<td>Male</td>
<td>26</td>
<td>4.62</td>
<td>6.37</td>
</tr>
</tbody>
</table>

Table 4.10 reveals no statistically significant difference in variances between attitude toward CS and gender, further confirming findings of the descriptive group statistics that gender is not a strong predictor of attitude toward CS.

Table 4.10. Group Mean and SD (Attitude & Gender)

<table>
<thead>
<tr>
<th>Independent Samples Test</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Levene's Test for Equality of Variances</td>
<td>F</td>
<td>Sig.</td>
<td>df</td>
<td>Sig. (2-tailed)</td>
</tr>
<tr>
<td>Programming to solve problems has an appeal to me.</td>
<td>Equal variances assumed</td>
<td>1.349</td>
<td>.253</td>
<td>-.547</td>
</tr>
<tr>
<td>Equal variances not assumed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Another confirmatory data point worth noting is the effect size the gender variable in relation to a question about the appeal of programming. The effect size of the independent samples shows a Cohen’s $\delta$ point estimate = -.186, a medium effect size can be concluded based on Cohen’s $\delta$ guidelines shown in Table 3.1 of Chapter Three.
Having run a simple one-question analysis of attitude toward CS with the gender variable, I was perplexed these data showed no statistically significant difference between means of the two gender groups. Consequently, I wanted to look at the variances of all questions related to attitude and identity toward CS to hopefully find at least one question that points to a significant difference. I could not reject the null hypotheses for the six questions related to attitude or identity toward CS and gender.

**Gender and Attitude Results Summary**

After running a series of one-way ANOVA calculations for each question with gender, the results shown in Table 4.12 reveal no P-value fell below the critical level (p = < 0.05), for a significant outcome; therefore, the null hypothesis of equal means could not be rejected. These results were surprising and worthy of further examination given a larger sample size.

<table>
<thead>
<tr>
<th>Programming to solve problems has an appeal to me.</th>
<th>Cohen’s d</th>
<th>Hedges’ correction</th>
<th>Glass’s delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standardizer&lt;sup&gt;a&lt;/sup&gt;</td>
<td>.828</td>
<td>-.186</td>
<td>-.852 -.483</td>
</tr>
<tr>
<td>95% Confidence Interval</td>
<td>Lower</td>
<td>Upper</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-.834 -.473</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.11. Cohen’s δ / Effect Size (Attitude & Gender)**

<table>
<thead>
<tr>
<th></th>
<th>Standardizer&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Point Estimate</th>
<th>95% Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming to solve</td>
<td>.828</td>
<td>-.186</td>
<td>-.852 -.483</td>
</tr>
<tr>
<td>problems has an appeal</td>
<td>Hedges’ correction</td>
<td>-.182</td>
<td>-.834 -.473</td>
</tr>
<tr>
<td>to me.</td>
<td>Glass’s delta</td>
<td>-.241</td>
<td>-.908 -.430</td>
</tr>
</tbody>
</table>

<sup>a</sup>The denominator used in estimating the effect sizes.
Cohen’s d uses the pooled standard deviation.
Hedges’ correction uses the pooled standard deviation, plus a correction factor.
Glass’s delta uses the sample standard deviation of the control group.
Table 4.12. One-way ANOVA (Attitude & Gender)

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I am sure that I can learn programming.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.321</td>
<td>1</td>
<td>.321</td>
<td>.251</td>
<td>.519</td>
</tr>
<tr>
<td>Within Groups</td>
<td>47.269</td>
<td>37</td>
<td>1.278</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>47.590</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Programming to solve problems has an appeal to me</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>205</td>
<td>1</td>
<td>.205</td>
<td>.299</td>
<td>.598</td>
</tr>
<tr>
<td>Within Groups</td>
<td>25.385</td>
<td>37</td>
<td>.666</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>25.590</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Being regarded as smart in computer science would be a great thing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.013</td>
<td>1</td>
<td>.013</td>
<td>.14</td>
<td>.906</td>
</tr>
<tr>
<td>Within Groups</td>
<td>33.577</td>
<td>37</td>
<td>.907</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>33.590</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have little self-confidence when it comes to a programming class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>205</td>
<td>1</td>
<td>.205</td>
<td>.106</td>
<td>.747</td>
</tr>
<tr>
<td>Within Groups</td>
<td>71.692</td>
<td>37</td>
<td>1.936</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>71.897</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I'm not the type to do well in computer programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.115</td>
<td>1</td>
<td>.115</td>
<td>.133</td>
<td>.718</td>
</tr>
<tr>
<td>Within Groups</td>
<td>32.192</td>
<td>37</td>
<td>.870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>32.308</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I think computer science and programming are boring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Between Groups</td>
<td>.051</td>
<td>1</td>
<td>.051</td>
<td>.060</td>
<td>.807</td>
</tr>
<tr>
<td>Within Groups</td>
<td>31.365</td>
<td>37</td>
<td>.846</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>31.416</td>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.13. Mean Plot Diagram (Favorable Attitude & Gender)
Figure 4.14. Mean Plot Diagram (Unfavorable Attitude & Gender)
Question Three: What demographic factors, if any, correlate with CC students’ identities and attitudes toward CS, and how?

The final research question asked if any of the findings from the quantitative analyses of the survey data were significant enough to correlate between an independent demographic variable with dependent variable scores related to attitude and identities toward CS. Based on the findings of quantitative methodology deemed appropriate for each type of independent variable, only one demographic factor is worthy of a correlation with attitude and identity toward CS. As shown and described in, earlier sections, race identity was revealed to correlate to attitude and identity toward CS strongly. Although the results were based on a small sample size, it was not surprising that racial identity was a strong predictor of attitude toward CS. To authenticate the claim that racial identity was a factor that correlates significantly to students’ identity and attitude, I ran statistical models that analyzed the survey data of questions specifically asking respondents to rate their agreement or disagreement along a 5-point Likert scale. For each question about attitude or identity toward CS, the demographic of the race was the only factor whose results were significant in half of the questions.

Although the demographic factor of age only resulted in one statistically significant finding, age cannot be strongly associated with the scores. Surprisingly, gender did not result in any statistically significant findings.
CHAPTER FIVE. CONCLUSIONS AND IMPLICATIONS

The aim of my dissertation serves two significant undertakings. First, awareness through research was brought to an overlooked population segment in computer science education. This dissertation’s theme is empirical, unseen up to this moment in community college CS programs. Scholarly focus in community college contexts needs innovative research for an area desperately lacking representation in the computer science education literature. This dissertation study has unearthed a new educational context ripe with opportunities for more robust, expansive studies. Second, my dissertation’s research could inform future investigations of similar designs and aims. The summary of findings, their ties to learning theory, my plans for future work, and what new questions emerged because of this empirical research could have genuine implications for moving the field of computer science education forward.

Summary of Findings

Independent variables of prior programming experiences and participants' demographics underwent rigorous scrutiny using statistical methodology appropriate for these variable types. Prior programming experience was found to have a moderate predictive effect on attitude and identity of computer science and programming among community college students in this study. These conclusions are significant for the following two reasons. First, these findings echo the results of earlier studies (Dorn & Tew, 2015; Chen et al., 2017). Second, because my participants were enrolled when taking the survey in a computer science-related course, a favorable attitude bias could affect the results. The self-selecting enrollment into a programming course, characteristic of the sample, has the capacity for weakening the significance of my conclusions. A future study might consider a control group of non-CS enrolled students into their study. A control group variable has the potential for revealing the true nature between attitude and identity
toward CS and no prior programming experience. Of the demographic factors evaluated, the
demographic characteristics, race showed the strongest correlation with attitude and identity
toward CS.

**Contribution to the Field of CS Education**

Accessing a career in computer science and programming commands well-paid salaries
for those equipped with the competencies and skills recognized as economically in high demand.
According to the USDL (2020), salaries in computer science and programming can range from
$80,000 to $100,000 per year. Career pathways to CS careers can begin at community colleges.
Having reviewed the community college literature on computer science in-depth, I see
community colleges as a prolific and emerging pathway for many seeking CS-related careers or
to achieve a four-year CS bachelor’s degree.

Jaggars et al. (2016) claimed that community colleges should be regarded as feasible
pathways that can lead to sustaining CS employment; however, given the non-traditional nature
of the community college demographic already discussed in Chapter One, this diverse segment
might need more structures of support in larger numbers than their four-year university peers.
These essential institutions are becoming a practicable educational environment for non-
traditional and traditional students seeking a straightforward pathway to CS.

Prior CS education research has focused on either K-12 or four-year university CS
programs (Sorva, 2013). Robust research on community college CS programs has been
essentially ignored for unknown reasons. The results of this study make substantial contributions
to CS Education in two profound ways. First, this study breaks new ground for its empirical
investigation of students’ attitudes toward computer science at community colleges. Second,
while this research is thematically parallel to prior works of Dorn & Tew (2015) and Chen et al.
(2018), the national distribution of a computing attitude survey, broader than a single campus or classroom, adds a new dimension to empirical research to the field of Computer Science Education. A fertile ground with many growth opportunities for computing-related research lies in the rich context of CS programs at community colleges.

**Stereotype Threat Theory**

The literature documents that “women are underrepresented in math, physical sciences, engineering, and computer science fields” and are missing some of the highest-paid positions (Malcom & Feder, 2016, p. 493). According to the Stereotype Threat Theory, first coined by Steele (1997), students of underrepresented groups must often prove their aptitudes in education and career domains. Stereotype Threat Theory claims those whose background fits into a negative stereotype are under increased and continuous scrutiny from peer groups with gender or race associated positively with the subject matter (p. 614). *Intersectionality* is an expression that describes where one’s identity intersects with a system of oppression (Delgado & Stefancic, 2017). Such terminology helps us understand various forms of oppression in scientific fields (Delgado & Stefancic, 2017). For example, a prior study found that visualizing yourself as successful in a job within a particular field can be correlated to an optimistic outlook (Meevissen et al., 2011). Positive attitudes contribute to one’s belief of succeeding, specifically in a field dominated by others whose backgrounds differ in gender and race (Peters et al. 2010).

Although this study commenced with an interest in attitude toward CS, a deeper investigation into theoretical frameworks located *identity with CS* as a critical factor found to be nearly synonymous with attitude. As the study proceeded, an emerging interest in capturing the identity and attitude toward CS became an important application in the CS domain context (O’Hara, 2020).
First, I noticed questions adapted for this study, derived from Wiebe et al. (2003) Computing Attitude Survey, were questions or statements of identity (e.g., “I am not the type to do well in programming” or “being regarded as a computer scientist would be a great thing”). This study's incipient questions of identity correlated with attitude; one informs the other. Steele (1997) suggests a tenant of identity is the sense of belonging, a vital ingredient in the achievement of college students (O’Hara, 2020, p. 2). According to Maslow (1970, p. 1), a sense of acceptance and belonging are connected to students’ motivation and drive to succeed. By asking questions of identity through the Stereotype Threat Theory lens, new dimensions for interpretation in this study became conceivable.

As noted by Steele (1997), “Stereotype threat, then, as a situational pressure "in the air" so to speak, affects only a sub-portion of the stereotyped group and, in schooling, probably affects confident students more than unconfident ones” (p. 617). Unbiased admission policies for assigning students of color and women to computing courses have become increasingly important. Administrators must communicate explicitly to teachers that “all students belong in STEM programs, not only the stereotypical few” (O’Hara, 2020, p. 4). Starting from the top-down, school administrators and teachers” have the potential to create a culture that influences student learning, motivation, and belonging” (O’Hara, 2020, p. 1).

Margolis et al. (2000; 2008) talk of the lack of representation of BIPOC and women in CS perpetuated further from intermittent proving of oneself as defined by Stereotype Threat Theory. This proving ground produces more anxiety in those whose skills are brought into question regularly. This continuing effort to prove becomes exhausting for many, thus, leading to an unfavorable attitude and identity with CS that can lead to an early departure out of the STEM domain (Steele, 1997, p. 614).
Currently, no known “remedy” exists to irradicate Stereotype Threat entirely. These negative stereotypes could persist until students of unrepresented groups become more ubiquitous in CS spaces. My dissertation's findings showed students' positive identity and attitudes of underrepresented groups, and more findings on similar themes could augment Stereotype Threat. However, more research on community college CS programs is necessary to substantiate these conclusions.

**Implications**

There is longstanding empirical evidence to substantiate the limited access to high-quality CS education for youth in economically disadvantaged areas (Margolis, 2008; Brennan and Resnick, 2013; Maloney et al., 2003). Opportunities of access to CS Education must start in middle school or high school to give students, specifically BIPOC, a chance to engage with computational thinking and programming pedagogy for forming an early identity with programming. While not all students' attitudes and identities will be favorable following early intervention with programming, at minimum, each student is given an equitable opportunity to engage with and form an opinion about CS in their youth.

**Sample size**

The size of the sample in this study makes generalizing these findings to the overall population of students in community colleges enrolled in CS courses challenging. Although this sample size was humble, a third of those surveyed identified as female. In addition, each race category was represented. Future studies of similar aims should consider a benchmark of nearly equivalent distribution of responses for each demographic category. Evenly distributed respondents in gender and race of a larger sample size could strengthen claims in this study about relationships between variables.
Data collection

The strategy implemented for collecting participants' data was challenged by the fact that distribution of the survey to students was mediated by CS faculty at community colleges. Pre-distribution relationships were not established between me, the researcher, and CC CS faculty. My dependence upon generous faculty to circulate the survey to their programming students was an undertaking. This could explain the sample size. The reliance upon faculty to distribute this survey to their students was perhaps a naive assumption that most would participate; however, that was not the outcome, and lessons were learned.

Likert Scale Data

Selecting a quantitative statistical model fitting for 5-point Likert scale data presented challenges already reviewed in Chapters Three and Four. In a future study, determining a way for collecting data of a ratio type could yield use of linearity model to establish correlations and predictors among variables with increased precision.²

Conclusion

The works of this study lay the groundwork for more research on students at community colleges enrolled in programming courses. While this study adds vital data to the literature on CS education, it was limited in size and scope. Given that this research charts new territory, as a first empirical evaluate CS students at community colleges, there is enormous personal gratification in noticing the gap within the literature. There were statistically significant findings related to

² Non-parametric analyses are sometimes considered more appropriate for Likert-type responses. Future research will consider these models of analyses into consideration (Mircioiu & Atkinson, 2017).
prior experience and race. I am encouraged and motivated to expand upon this research in future endeavors, and hopeful my findings spark an interest in conducting more empirical research in the context of computer science in community colleges.
APPENDIX B. MEAN PLOT DIAGRAMS (GENDER)
APPENDIX C. MEAN PLOT DIAGRAM (AGE)
APPENDIX D. IRB FORMS

TO: Kerri Tobin  
LSUAM | Col of HSE | Education

FROM: Alex Cohen  
Chairman, Institutional Review Board

DATE: 23-Sep-2021

RE: IRBAM-21-0906

TITLE: Do high school students' perceptions and attitudes towards computer science and programming change following a text-based computer science course intervention?

SUBMISSION TYPE: Initial Application

Review Type: Expedited Review

Risk Factor: Minimal

Review Date: 22-Sep-2021

Status: Approved

Approval Date: 22-Sep-2021

Approval Expiration Date: 21-Sep-2022

Expedited Categories: 07

Requesting Waiver of Informed Consent: Yes

Re-review frequency: Annually

Number of subjects approved: 1300

LSU Proposal Number:

By: Alex Cohen, Chairman

Continuing approval is CONDITIONAL on:

1. Adherence to the approved protocol, familiarity with, and adherence to the ethical standards of the Belmont Report, and LSU's Assurance of Compliance with DHHS regulations for the protection of human subjects*
2. Prior approval of a change in protocol, including revision of the consent documents or an increase in the number of subjects over that approved.
3. Obtaining renewed approval (or submittal of a termination report), prior to the approval expiration date, upon request by the IRB office (irrespective of when the project actually begins); notification of project termination.
4. Retention of documentation of informed consent and study records for at least 3 years after the
study ends.
5. Continuing attention to the physical and psychological well-being and informed consent of the individual participants, including notification of new information that might affect consent.
6. A prompt report to the IRB of any adverse event affecting a participant potentially arising from the study.
8. SPECIAL NOTE: When emailing more than one recipient, make sure you use bcc. Approvals will automatically be closed by the IRB on the expiration date unless the PI requests a continuation.

* All investigators and support staff have access to copies of the Belmont Report, LSU's Assurance with DHHS, DHHS (45 CFR 46) and FDA regulations governing use of human subjects, and other relevant documents in print in this office or on our World Wide Web site at http://www.lsu.edu/research

Louisiana State University  
131 David Boyd Hall  
Baton Rouge, LA 70803

O 225-578-5833  
F 225-578-5983  
http://www.lsu.edu/research
TO: Kerri Tobin  
LSUAM | Col of HSE | Education

FROM: Alex Cohen  
Chairman, Institutional Review Board

DATE: 26-Oct-2021

RE: IRBAM-21-0906

TITLE: Does the completion of a community college introduction to programming/computer science course have an effect on a student's attitude and perception of computer science and programming?

New Protocol/Amendment/Continuation: Amendment

1. Title Change to reflect the new population under evaluation: Does the completion of a community college introduction to programming/computer science course have an effect on a student's attitude and perception of computer science and programming? (note: The new title is not reflected in the title of the application)

2. The population under evaluation was K-12 students. The new population under evaluation are non-minors (18+ years of age) enrolled in a community college computer science introduction course. I am seeking approximately 30 participants for this study.

Brief Amendment Description:

3. The survey added questions about race, ethnicity, age, and gender (including options for nonbinary/nonconforming, trans, and other).

4. One question was added that asks the student how important it is to them that an instructor or teacher has a background similar to theirs (race, age, gender, ethnicity).
5. One question asks about the number of college-level credit hours they have completed.

6. Another question asks if they have had any prior experience programming. If no, the survey skips the student past the following question. If yes, the participant is asked a question about their prior experience in programming in a block or text-based programming language.

<table>
<thead>
<tr>
<th>Review Type:</th>
<th>Expedited Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Risk Factor:</td>
<td>Minimal</td>
</tr>
<tr>
<td>Review Date:</td>
<td>25-Oct-2021</td>
</tr>
<tr>
<td>Status:</td>
<td>Approved</td>
</tr>
<tr>
<td>Approval Date:</td>
<td>25-Oct-2021</td>
</tr>
<tr>
<td>Approval Expiration Date:</td>
<td>21-Sep-2022</td>
</tr>
<tr>
<td>Re-review frequency:</td>
<td>(annual unless otherwise stated)</td>
</tr>
<tr>
<td>Number of subjects approved:</td>
<td>30</td>
</tr>
</tbody>
</table>

By: Alex Cohen, Chairman

Continuing approval is CONDITIONAL on:

1. Adherence to the approved protocol, familiarity with, and adherence to the ethical standards of the Belmont Report, and LSU's Assurance of Compliance with DHHS regulations for the protection of human subjects*
2. Prior approval of a change in protocol, including revision of the consent documents or an increase in the number of subjects over that approved.
3. Obtaining renewed approval (or submittal of a termination report), prior to the approval expiration date, upon request by the IRB office (irrespective of when the project actually begins); notification of project termination.
4. Retention of documentation of informed consent and study records for at least 3 years after the study ends.
5. Continuing attention to the physical and psychological well-being and informed consent of the individual participants, including notification of new information that might affect consent.
6. A prompt report to the IRB of any adverse event affecting a participant potentially arising from the study.
8. SPECIAL NOTE: When emailing more than one recipient, make sure you use bcc.

* All investigators and support staff have access to copies of the Belmont Report, LSU's Assurance with DHHS, DHHS (45 CFR #6) and FDA regulations governing use of human subjects, and other relevant documents in print in this office or on our World Wide Web site at [http://www.lsu.edu/research](http://www.lsu.edu/research)
APPENDIX E. INFORMED CONSENT FORM

Study Title: Does prior programming/computer science experience affect students’ attitude and beliefs about computer science and programming?

Study purpose: This research aims to determine changes in students’ attitudes and identities about computer science/programming following a community college intro to computer science course.

The following information will be collected for each student:
Scaled Attitudes Toward Computer Science Score Age 
Gender 
Ethnicity Origin / Racial Identity 
Number of completed college-level credits 
Previous programming experience (yes/no) 
Previous programming language/environment experience (block, text, robotics).

Inclusion criteria: A enrolled student in a community college introductory to programming/computer science course AND participant age is 18 years or above.

Exclusion criteria: Not enrolled in a community college introductory computer science/programming course AND participant’s age is less than 18 years old.

Risks: There are no known risks.

Investigators: The following investigators are available for questions about this study, M-F, 8:00 a.m. – 3:00 p.m., Trent Dawson, School of Education, (225) 348-7617/ dtrent2@lsu.edu and Dr. Kerri Tobin, School of Education, LSU, (570) 852-3685/ ktobin@lsu.edu.

Right to Refuse: Participation in the study is voluntary, and students may change their minds and withdraw from the study at any time without penalty.

Benefits: The benefit contributes to the research of broadening access to computing education and improving equitable access to computing education.

Financial information: There are no costs, nor is there any compensation to the student for participation.

Informed consent: There are no risks involved in participating in the study. Subjects may choose not to participate or to withdraw from the study at any time without penalty or loss of any benefit to which they might otherwise be entitled. Results of the study may be published, but no names or identifying information will be included in the publication.

Approval: This study has been approved by the LSU IRB. For questions concerning participant rights, please contact Alex Cohen, Chairman, Institutional Review Board at (225) 578-8692,
irb@lsu.edu, or www.lsu.edu/research.

By to this survey, you are to participate in this research study.

Link to consent form and survey: http://lsu.qualtrics.com/jfe/form/SV_2lzw7Kv8M60gHf8
Hi Student!

My name is Trent Dawson. I am a Ph.D. student in the LSU School of Education. I am conducting a study of attitudes toward computer science of students enrolled in a community college introductory computer science or programming course. I am contacting you to request your participation in this study by completing a brief survey. The survey is anonymous, and your participation will not be recorded or documented. Also, participation in this survey will in no way impact your status or grade in any class you are taking. Your participation is requested, however, as the results of this study will provide valuable information about the impacts on attitudes and perceptions of computer science after completing a community college-level introduction to computer science course.

The LSU Institutional Review Board has approved this study. If you have questions about the IRB office, contact Alex Cohen, Chairman, Institutional Review Board, (225) 578-8692, irb@lsu.edu, or www.lsu.edu/research. If you have questions about the survey, please feel free to contact me: Trent Dawson via email at dtrent2@lsu.edu

If you have already completed the survey, please disregard this email.

If you agree to participate in the survey and are a student at LSU and at least 18 years of age, please use the link below to:

(1) complete the study consent form and
(2) answer the short survey.

http://lsu.qualtrics.com/jfe/form/SV_2lzw7Kv8M60gHf8

Thank you for your time!

Trent Dawson, M Ed.
Ph.D. Candidate | (A.B.D.)
Computer Science Education Researcher
Department of Curriculum and Instruction, Curriculum Studies Program Louisiana State University, School of Education
APPENDIX G. QUESTIONNAIRE/SURVEY

I am sure that I can learn to program.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I have little self-confidence when it comes to a programming class.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I'm not the type to do well in computer programming.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

Programming to solve problems has an appeal to me.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I think computer science and programming are boring.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I'd be happy to get a good grade in this programming class.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree
I'd be happy to get a good grade in this programming class.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

Being regarded as smart in computer science would be a great thing.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I believe both males and females in our school should study computer science.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

Females are as good as males at programming.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I think I will need programming for a future job.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree

I will use programming in many ways throughout my life.
  o Strongly disagree
  o Somewhat disagree
  o Neither agree nor disagree
  o Somewhat agree
  o Strongly agree
I think it is important to have at least one computer science instructor or professor with a background similar to mine (e.g., race, gender, age, or ethnicity).
- Strongly disagree
- Somewhat disagree
- Neither agree nor disagree
- Somewhat agree
- Strongly agree

Number of college-level credit hours, in any field or major, you have completed (not including this semester’s courses).
- 0 – 6 credit-hours
- 7 – 12 credit-hours
- 12 – 18 credit hours
- 18 or more credit hours

I have prior experience with computer programming.

☐ Yes
☐ No

The number of programming course(s) previously taken in middle school and high school? (e.g., programming, computational thinking, computer science, or robotics)
- 1 course
- 2 courses
- 3 courses or more

The number of programming/coding course(s) previously completed in college? (e.g., Python, Java, C++)
- 1 course
- 2 courses
- 3 courses or more

I have prior programming experience in which programming environments (s)? (Multiple answers allowed).

☐ Block-based, drag and drop (e.g., Scratch, Snap!, Robotics, Micro:bit, Alice)
☐ Text-based (e.g., Python, Java, C++, Haskell, or JavaScript)
The gender with which I most closely identify.
- Female
- Male
- Transgender Female
- Transgender Male
- Gender Variant/Nonconforming

My age is between?
- 18 - 24
- 25 - 29
- 30 - 34
- 35 - 44
- 45 years of age or above

The race(s) with which I most closely identify
- American Indian or Alaska Native
- Asian
- Black or African American
- Native Hawaiian or Other Pacific Islander
- White
- Other

Ethnic Origin: Are you Hispanic or Latino?
- Hispanic or Latino
- Non-Hispanic or Non-Latino
REFERENCES


VITA

Trenton W. Dawson was born in Oklahoma, where he attended the University of Oklahoma as an undergraduate student majoring in business marketing. He has worked in various industries that include consumer products marketing, pharmaceutical sales, environmental consulting, and educational technology. In the winter of 2016, Trent launched into master’s degree coursework in instructional design and technology at Virginia Tech, thus, commencing his career trajectory in educational technology. At Virginia Tech, he worked in the educational technology department, interned at NASA/JPL with two additional graduate students under the guidance of an instructional design and technology professor, and was fortunate to work as a graduate assistant with APEX Center for Entrepreneurs.

Since graduating with his master’s degree, he has taught adult education courses at South Louisiana Community College and Baton Rouge Community College. He assisted LSU faculty in various graduate assistant roles in computer science education. Finally, he briefly experienced research in computer education as a member of the Carnegie Mellon University Computer Science Academy project team based in Pittsburgh, Pennsylvania.