High School Calculus and Computer Science Course Taking as Predictors of Success in Introductory College Computer Science

CHEN CHEN, Science Education Department, Center for Astrophysics I Harvard & Smithsonian
JANE M. KANG, Harvard Graduate School of Education
GERHARD SONNERT, Harvard College Observatory
PHILIP M. SADLER, Science Education Department, Center for Astrophysics I Harvard & Smithsonian

Success in an introductory college computer science (CS) course encourages students to major and pursue careers in computer science and many other STEM fields, whereas weak performance is often a powerful deterrent. This article examines the role of high school course taking (AP, regular, or none) in mathematics and in CS as predictors of later success in college introductory computer science courses, measured by students' final grades. Using a sample of 9,418 students from a stratified random sample of 118 U.S. colleges and universities, we found that the observed advantage of taking AP calculus over taking AP CS, seen in an uncontrolled model, was largely confounded by students' background characteristics. After applying multinomial propensity score weighting, we estimated that the effects of taking AP calculus and AP CS on college CS grades were similar. Interestingly, enrollment in both AP calculus and AP CS did not have any additional positive effect, suggesting that both AP calculus and AP CS strengthened similar skills that are important for long-term CS achievement. Taking regular CS did not have a significant effect; taking regular calculus had a positive effect, about half the size of taking AP calculus or AP CS. Thus, the study showed that simply exposing students to any kind of CS course before college does not appear to be sufficient for improving college CS performance; and that advanced CS and advanced calculus in high school may substitute for each other in the preparation of college CS.

CCS Concepts: • **Applied computing → Education**; • **Social and professional topics → K-12 education**; **CS1**;

Additional Key Words and Phrases: Computer science education, post-secondary education, secondary education, advanced placement Introduction

ACM Reference format:
Chen Chen, Jane M. Kang, Gerhard Sonnert, and Philip M. Sadler. 2020. High School Calculus and Computer Science Course Taking as Predictors of Success in Introductory College Computer Science. *ACM Trans. Comput. Educ.* 21, 1, Article 6 (December 2020), 21 pages.
https://doi.org/10.1145/3433169

Authors' addresses: C. Chen (corresponding author) and G. Sonnert , 60 Garden St., MS71, Cambridge, MA, 02138. emails: {chen.chen, gsonnert}@cfa.harvard.edu; J. Kang, Larsen 514, 14 Appian Way, Cambridge, MA, 02138; email: jmkang@g.harvard.edu; P. M. Sadler, 60 Garden St., MS71, Cambridge, MA, 02138; email: psadler@cfa.harvard.edu.

1946-6226/2020/12-ART6
https://doi.org/10.1145/3433169

1 INTRODUCTION

> *"Mathematics is an integral part of computer science and because of this, students must develop a good understanding of mathematics in order to be successful in computer science." [9].*
> *"Teaching a coding language [in high school] is often necessary for teaching computational thinking or algorithm design, and so it's a key part of most C.S. education." [47].*

Opportunities to learn computer science (CS) prior to college have been expanding in the U.S. through national and state initiatives (e.g., CS for All, Hour of Code) and course offerings at the high school level (e.g., AP Computer Science A, AP Computer Science Principles). The growth is spurred by the lure of an abundance of unfilled high-paying jobs as well as the critical need to broaden access and increase the participation of women and underrepresented minorities in the field. The focus on K–12 CS education is based on the idea that exposing students to CS earlier than in college will generate interest in the subject, bring computing ideas to a more diverse audience, and better prepare students to succeed in college CS courses.

Success in an introductory CS course in college is critical for encouraging students to major in CS [13]. Past studies have examined predictors of success in introductory college CS courses by examining factors such as high school performance, SAT scores, and gender [14, 58, 65]. These studies have generally found that mathematics preparation is a significant predictor of success in college CS but have found mixed results for the effect of prior CS preparation. Even the importance of mathematics preparation as a predictor, however, has been questioned by those who argue that both the content of the mathematics learned and the nature of the college introductory CS course affect this relationship [10, 49, 62].

The current study addresses the research question: How does high school CS course taking compare with mathematics course taking as a predictor of college introductory CS course grades? Understanding how high school mathematics preparation, on the one hand, and CS preparation, on the other, are related to success in college introductory CS is important. A major goal of the current expansion of K–12 CS education is to strengthen CS affinity at an early age, to stoke the pipeline for computing jobs, and to build computational thinking skills that are important to the performance and persistence in a wide range of learning platforms and careers [7, 18]. Understanding the extent to which high school mathematics and CS experiences are preparing students for their college CS courses can help educators be more thoughtful about what kinds of high school CS experiences they are providing and can potentially help high school students understand how current coursework may help them prepare for college.

2 LITERATURE REVIEW

2.1 Theoretical Framework

The overarching theoretical framework that can help us understand the potentially different effects of mathematics and CS course taking in high school on students' college CS performance in the long-term is the transfer of learning theory [43]. This theory tries to explain how individuals can apply their prior knowledge to perform (not always successfully) in a task that is contextualized differently from where they acquired the knowledge [27]. When the two contextual domains are close to each other, a learner needs to carry out near-transfer in which the learner may directly apply his/her prior knowledge. When the domains are distant from each other, the learner needs to achieve far-transfer in which he/she applies abstracted principles across domains. Typically, near-transfer is easier than far-transfer. The most commonly cited condition that predicts the success

in the transfer of learning is when the commonality or connections between the two tasks are highlighted and made explicit [36, 48].

The connections between mathematics and computer science run deep, and many consider the link between the two fields obvious. In fact, scholars are generally in agreement that the fields themselves are intertwined; that computer science theory is deeply mathematical; and that mathematical thinking has been advanced through computer science [34]. Historically, many CS departments in colleges and universities grew out of their mathematics departments, further linking these two fields. As of 2019, 47 states in the United States allowed computer science courses to count as mathematics courses [20].

Where scholars disagree is in what this link means for CS education. On one hand, some argue that a strong foundation in advanced mathematics is necessary for a rigorous study of CS, and that mathematics courses should be a part of any computing degree program [4]. Many skills learned in mathematics are important in computer programming, such as fluency with mathematical operations, an understanding of functions, and the use of variables [8]. More importantly, mathematics and CS use some similar ways of thinking. Studying mathematics is argued to enhance logical thinking and problem-solving skills essential to CS. Through mathematics, students also learn to be precise with symbolic notation and learn methods of proof and reasoning. Finally, the study of CS goes well beyond programming and progresses into theory, where connections to advanced mathematical ideas are explicit [8]. De-emphasizing advanced mathematics for CS majors potentially limits students' advancement in understanding the field and could do students a disservice [5].

On the other hand, opponents argue that typical advanced mathematics courses are not well-aligned with a CS program, and that mathematics requirements sometimes present an unnecessary barrier for students who might otherwise consider a major in CS. In particular, calculus is unconnected to concepts CS majors encounter in their beginning courses (and even subsequent ones), yet most of the top CS programs required students to take calculus [5]. Instead of a calculus requirement, many educators advocate for courses in discrete mathematics and point to a need for a greater emphasis on problem-solving [5, 49, 50].

By comparison, high school CS courses may shine in respect to contextualization. The concrete knowledge acquired in CS courses is expected to be directly applicable in coding and in the operation of computer systems, and the abstract skills are expected to be contextualized in the subject of CS. For example, training in computational thinking skills may build learner's problem-solving skills that are usually explicitly connected to computing [32]. Therefore, transfer of learning to college CS from high school CS should be considered nearer than the transfer from high school calculus. Furthermore, considering the increasing popularity of introductory CS courses and CS courses for non-majors, it is possible that high school mathematics is not as relevant for success in these courses as it was once thought to be.

It is noteworthy that cognitive skills (e.g., concrete or abstract knowledge) may not be the only channel for prior experience to support future performance. Researchers in self-efficacy theory have shown that the motivational system, such as self-efficacy beliefs, is equally, if not more, important in CS learning [12, 15]. An integral part of CS tasks is trial and error or debugging, which can quickly discourage a student who does not have an established self-efficacy belief in CS [43]. High school CS courses, especially the advanced-level courses, may help students develop the self-efficacy or at least desensitize them to the dreadful process of debugging. In the meantime, some college students may be intimidated by CS because it is mathematics-heavy. These students may benefit from high school mathematics to develop affinity and self-efficacy in mathematics.

2.2 Prior Findings about High School Preparation as Predictor of Success in College CS

Many studies have examined various predictors of performance in introductory college CS. The most commonly examined variables have been socioeconomic status [45], self-efficacy [11, 30], and cognitive skills such as abstraction ability, learning style, and spatial skills (e.g., References [10, 21, 64]). Almost every study that has examined predictors of success in college CS has used some measures of mathematical ability, some of which come from the high school years. These measures include standardized exam (i.e., SAT or ACT) scores in mathematics, number of high school mathematics courses, grades in high school mathematics courses, performance in other college mathematics courses, and diagnostic tests of algebra or problem-solving. Fewer studies have examined previous CS experiences, which include previous programming experience and CS courses taken in high school, as well as other high school coursework.

2.2.1 Mathematics Preparation. As mentioned above, most studies have found a positive relationship between mathematics preparation and success in introductory CS. For example, Reilly, Tomai, and Grabowski [52] examined records of 558 students over 4.5 years and found that students who have already taken calculus when they take introductory CS are more likely to pass the course and to receive A's and B's than those who have not completed the prerequisites for taking calculus. Fan and Li [57] surveyed 940 students from five schools in Taiwan and found that high school and college mathematics grades are positively associated with introductory college CS grades, although they also found that the relationship between mathematics entrance exam scores and CS grades is not statistically significant. White and Sivitanides [26] used records from 837 students over three years and found that college mathematics course performance is positively correlated with CS grades, even for a visual programming course.

Very few studies have found that mathematics is not a significant predictor of college introductory CS performance. Ventura [62] collected data from 498 students in an objects-first course. He found that the effect of mathematics is negligible, especially compared with factors such as effort (measured by lab participation) and comfort level. He notes that the number of high school mathematics courses was uncorrelated with CS course performance, and that SAT mathematics scores accounted for only 7% of the variance in course averages. Golding and McNamarah [28] also found that taking mathematics O and A level courses did not predict overall success among 96 students in a CS program (not just an introductory course) in Jamaica. They likewise cite a low R^2 value of 8%. Finally, Rauchas et al. [51] found from surveying 107 students in South Africa that, while mathematics is correlated with performance in introductory CS courses, English scores showed a higher correlation. While not denying the relevance of mathematics to learning CS, they suggest that language ability may be a better predictor.

2.2.2 CS Preparation. Several studies have specifically examined high school CS experiences, but the results here are more mixed. While they do not examine performance in college courses, Armoni and Gal-Ezer [3] found that students with more high school CS coursework are more likely to pursue computing in college than are students with fewer high school CS courses. In a study of 67 four-year higher education institutions, the College Board reported that "17.9 percent of AP computer science students majored in computer and information sciences, compared to 2.3 percent of the total sample" [22].

Numerous studies have found that prior computing experience contributed to success in college CS courses [26–30], especially for female students. Holden and Weeden [31] found for 159 college students that, while prior programming experience predicts performance in an introductory CS class, it fails to predict performance in subsequent CS classes. In a recent study, Chen et al. [16]

found that learning a textual language, such as Java (the primary programming language used in AP CS), before college is equally beneficial to college CS grades as is learning a graphic language, such as Scratch, that is not as coding heavy. Chen et al. [17] further showed that students who took a CS course in high school performed better in college CS than those who did not study any programming language, but did not perform better than students who self-learned CS out of school—known as "cowboy and cowgirl" programming, according to Kölling [42]. Numerous studies have shown a positive association between AP course taking and college grades in specific subjects [35, 37, 38]. No one, to the best of our knowledge, has examined the relationship between AP CS and college CS grades.

2.2.3 Comparing Mathematics and CS Preparation. Only a few studies compared mathematics and CS course taking. In general, these studies showed that mathematics preparation is the stronger predictor of college introductory CS performance. The studies differ, however, on whether CS preparation is a statistically significant predictor at all. Butcher and Muth [4] examined 269 first-semester freshmen at a single institution and found that the number of high school mathematics courses was associated with their performance in college, whereas being exposed to high school CS courses was not. Cantwell-Wilson and Shrock's [65] study of 105 students examined 12 factors, including previous programming experience and mathematics background. They found that the strongest predictors of success in introductory CS were comfort level in that course and high school mathematics background, measured by the number of high school courses taken. They also found that having prior CS experience in general was not a significant predictor, whereas taking a prior programming course had a positive and significant effect, which was, however, smaller than that of mathematics background. Schollmeyer [55] noted that high school CS courses can help but suggested, based on observation and student interviews, that, instead of focusing on programming, high school courses should emphasize problem-solving that makes use of mathematics skills. Beaubouef and Mason [9] argued that high school students are often encouraged to pursue a computer science degree in college because they did well using a word processor or web browser in high school computer courses, not realizing the amount of mathematics skills necessary for college computer science. To quote from Beaubouef [8] (p. 57), "despite of having taught computer science courses at all levels for several years, I am still amazed, and perhaps appalled, by comments and questions I get from students, questions such as 'Will we have to do any math on the test?' or 'I can't read the textbook. It's like reading a math book and everyone knows that you can't read a math book.' These students, unfortunately, do not remain computer science majors for long."

2.3 Missing Pieces in the Literature

Almost all the studies cited above were conducted with a relatively small number of students within a single institution, limiting their generalizability. Prior studies also adopt a correlational analysis method, ignoring the self-selection bias built into the high school course taking process. For example, the privileged segment of the population may have easier access to advanced courses and CS courses than may the unprivileged population. Some students, especially those from unprivileged population and who go to underfunded public schools may have limited or no options in this respect. Direct comparisons between those who took a course with those who did not take the course thus may be confounded by such a self-selection bias. In addition, many of the studies are also a few decades old, during which time the nature of CS education at both the K–12 and undergraduate levels has changed dramatically. Since AP computer science shifted to the Java language (from C++) in 2004, there has not been a study that compares this advanced high school computer science course against an advanced mathematics course, such as AP calculus. Moreover, we have been unable to find any study that examined the potential—additive or non-additive—effect

of enrollment in both advanced or regular mathematics and CS courses while in high school. We address some of these lacunae in the literature through this study.

3 RESEARCH QUESTION AND HYPOTHESES

Using data from a large-scale survey administered to about 10,000 CS students in 118 institutions across the U.S. in 2014, the current study contributes to the CS education literature by addressing the research question: How does high school CS course taking compare with calculus course taking as a long-term predictor of college introductory CS course grades?

Specifically, we examined (1) the effects of course taking in calculus and CS, differentiating between course types (AP, regular, and none); (2) the interaction effect between CS and calculus course taking (for potential additive or synergistic effects); (3) and the above effects before and after accounting for students' background information.

Based on the above-mentioned literature, we hypothesize that, before accounting for background information, taking AP courses will have a stronger effect than taking regular courses, which will have a stronger effect than taking no courses. The transfer of learning theory would expect that a high school CS course provides a nearer transfer than a high school calculus course onto college CS performance. Accordingly, we hypothesize that high school CS course taking will be a stronger predictor of success in college introductory CS than will high school calculus course taking. It is noteworthy, however, that our literature review showed some studies found that high school calculus was the primary determinant of college CS success. Therefore, if these findings hold true on the larger scale, we may find high school calculus to be more important than high school CS for students' performance in introductory college CS.

Our literature review does not suggest a hypothesis about course taking effects after accounting for background information. Regarding the additive effect of having taken both a calculus course and a CS course in high school, if they built different skills that contribute to long-term success in college computer science, we are likely to detect an additive effect. Otherwise, if the two courses built common skills, we are not likely to detect an additive effect, which should manifest as a negative interaction effect between course taking in high school calculus and CS.

When comparing between students enrolled in different high school courses, one should take into account that students in different courses usually have very different background characteristics. For example, as observed by Orban [44], some students took CS in high school just to avoid mathematics, especially since the sudden increase in 2014 of the number of states that allowed CS courses to count towards mathematics credit. This suggests that the common observation of the advantage of advanced mathematics preparation over CS preparation, as seen from the perspective of a college teacher, may be confounded by students' general academic or math aptitude or other background factors. This called for the application of techniques of matching or weighting for background information, which has commonly been used in the evaluation studies of AP programs [23, 35, 63].

4 METHODS

4.1 Survey

The survey, Factors Influencing College Success in Information Technology (FICSIT), consists of 50 questions that ask about students' demographic and background information, high school courses taken and grades received, earlier experiences with computers (such as when and how they were introduced to CS, the number of years of programming experience they have, and how often they used computers in high school), the organization and structure of their most advanced high school CS class (including types of classroom activities and how they would rate their teacher), personal

thoughts and feelings about CS, and future career plans. The retrospective survey was administered in the fall of 2014 to students early in their introductory-level university CS course. At the end of the semester, college course instructors reported the students' final grade. This study received IRB approval from the Committee on the Use of Human Subjects at Harvard University. The survey has been modified based on pilot testing, and its validity and test-retest reliability have been established.

4.2 Sampling Frame

We used a random selection procedure, stratified based on school type and school size. For our sample, the distinction between two-year and four-year institutions served as the first stratification criterion. Each of the two institutional types was further stratified by the size of the institution (small, medium, and large). We obtained 2012 enrollment numbers for American two- and four-year institutions from the *Integrated Postsecondary Education Data System (IPEDS)* website. The IPEDS dataset comprised 5,480 institutions, among them 2,340 two-year and 3,140 four-year schools. Subtracting schools that showed zero enrollment, we were left with 2,268 two-year and 2,747 four-year institutions. For an indicator of an institution's undergraduate enrollment size, we added its full-time undergraduates and its part-time undergraduates, weighted by 0.5. The weighting was chosen to reflect the lower overall participation rate in instruction of part-time students during a given semester. About two thirds (64.4%) of the total undergraduate enrollment was at four-year institutions; the remainder, at two-year institutions. To create the small, medium, and large bins within each institutional type, it was determined that roughly a third of the national undergraduate population at two-year institutions attended schools that had fewer than 4,600 undergraduates (these were termed "small" institutions), another third attended schools that had between 4,600 and 10,600 undergraduates ("medium"), and the final third attended schools with more than 10,600 undergraduates ("large"). For the four-year institutions, the corresponding cutoffs were 5,700 and 17,800. Finally, we dropped from the small bins 1,094 two-year institutions and 804 four-year institutions with an enrollment below 500. We thus ended up with six bins stratified by type and size that contained 816 small two-year schools, 253 medium two-year schools, and 107 large two-year schools, 1,506 small four-year schools, 314 medium four-year schools, and 121 large four-year schools.

After each of the six bins was randomized, we went down the six lists, reaching out to the instructors who taught introduction to computer sciences in the colleges. Instructors willing to participate received paper surveys for their students and administered them at the beginning of the fall 2014 semester during class, ensuring very high student participation rates. At the end of the semester, the instructor added the student's final grade to the questionnaire, tore off the cover sheet with the student's name to ensure anonymity, and returned the questionnaires to us for analysis. Table 1 presents, within each bin, the number of schools contacted, agreed to participate, and actually participated, and the number of instructors and student questionnaires returned. Of the 15 small two-year schools agreeing to participate, 8 did, with 8 instructors and 329 students. Of the 7 medium two-year schools agreeing to participate, all did, with 7 instructors and 325 students. The corresponding numbers for the large two-year schools were: 8 agreeing to participate, 8 participating, 24 instructors, and 776 students. Small four-year schools: 56 schools agreeing to participate, 49 schools participating, 49 instructors, 1,857 students; medium four-year schools: 36 agreeing to participate, 32 participating, 51 instructors, 4,781 students; large four-year schools: 16 agreeing to participate, 14 participating, 20 instructors, 2,131 students.

As a rough proxy for the CS enrollment in each of the bins, we calculated the number of BA and AA degrees awarded in "Computer and Information Sciences, General" (CIP 11.0101) in each

Table 1. The Sampling Frame Indicating the Number of Schools, Instructors, and Returned Student Questionnaires for Each Type of College

Bins (college types)	N of schools contacted	N of schools agreeing to participate	N of schools with actual returns	N of instructors in those schools	N of returned student questionnaires
2-year small	100	15	8	8	330
2-year middle	102	7	7	10	327
2-year large	77	8	8	24	776
4-year small	405	56	49	49	1,858
4-year middle	289	36	32	51	4,781
4-year large	107	16	14	20	2,131
Total	1,080	138	118	162	10,203

bin, from the IPEDS Completion Survey (accessed through WebCaspar). Comparing our sample to the target proportions in the bins, we found a relatively good match in terms of school type and school size. The one bin that was overrepresented in our sample was the medium four-year school bin, with all other bins being slightly underrepresented.

4.3 Sample

Starting with 10,197 responses, the following selection criteria were applied to obtain a final sample of 9,418 students. First, students were omitted if they provided no information on any of 20 questions asking about high school mathematics or computer course taking and also if they failed to respond to other key demographic questions throughout the survey, including information about gender, race, and whether the student attended high school in the U.S. The sample was further restricted according to the following two criteria: First, the sample was restricted to current undergraduate students, excluding any graduate students, current high school students, or other students in the college introductory CS course. Second, students were excluded if it had been more than five years since they had graduated from high school, with the rationale that their reporting of high school information is likely less reliable, and that their high school course taking pattern is likely to be less relevant to their performance in the college course. Table 2 shows background information of the students in the sample.

4.4 Outcome

Students' final grades in the college introductory computer science courses (college CS grade) were submitted by the instructors. Numerical grades on a 100-point system were used. If these grades were not available, letter grades were converted to numerical grades (A=96.3, A-=91.5, B+=88.5, B=85, …, F=40; Pass=83). The average grade was 84.9, with a standard deviation of 11.6.

4.5 Key Predictors

There were two key predictors in our models: the most advanced level of computer science courses (CS course) and the most advanced level of calculus courses (calculus course) that the participants reported to have taken in high school. Each predictor had three values: none, regular, and AP. If a participant did not take any course in a subject, the value was "none"; if a participant took regular, honors, or IB courses but did not take an AP course in the subject, the value was "regular" (thus, "regular" included regular, honors, and IB, and basically served as a residual category indicating

Table 2. Participants' Background Information

Variable	Percentage	Variable	Percentage
Gender		Race	
Male	73.1%	White	52.4%
Female	26.9%	Black	6.7%
CS course taken		Hispanic	13.2%
AP	9.1%	Asian	20.9%
Regular (incl. Honor and IB)	17.5%	Born in the U.S.	81.6%
None	73.4%	Parents' jobs relate to CS	26.1%
Calculus course taken		Parents support math learning[a]	66.7%
AP	38.5%	English is first language	75.6%
Regular (incl. Honor and IB)	14.4%		
None	47.1%		

[a]Parents-support-math-learning was a Likert scale with 0 = very unsupportive and 5 = very supportive. 66.7% reported 3 or above.

all "non-AP" courses); if a participant took an AP course in a subject, the value was "AP." If a participant took both regular and AP courses in a subject, the value was set to "AP," because AP was the most advanced-level course reported. Both AP Calculus AB and AP Calculus BC were included in our category of AP Calculus. Because the new AP Computer Science Principles course was not available to our cohort yet, our category of AP CS contains only AP Computer Science A. We also included interaction effects between CS and calculus courses to estimate the effect of taking both calculus and CS in high school.

Further, we included the grades that the participants received in each of the courses ("A+" = 4.33, "A" = 4, "A-" = 3.67, …, "Pass" = 2.8, "F" = 0). For those who took multiple regular courses in a subject (i.e., someone might take both regular and honors mathematics), we used the average score of the course grades. Participants who did not enroll in a course would, of course, not have any grade in the course. Therefore, the effect of course grade was only applicable to those who had taken a specific course. This data structure can be modeled by including an interaction effect between the course taken and the course grade (i.e., CS-AP × CS-AP-Grade) while excluding the main effect of course grade (see example in Reference [56]). In this specification, when a course was taken (i.e., CS-AP = 1), the interaction term became the grade (i.e., CS-AP × CS-AP-Grade = CS-AP-Grade); when a course was not taken (i.e., CS-AP = 0), the interaction term became zero and the parameter for grade would not be estimated.

4.6 Covariates

Our questionnaire collected a list of covariates, including gender (male, female), race/ethnicity (White, Black, Asian, Hispanic, or other race), parental education (did not finish high school, high school, some college, four years of college, graduate school), born in the U.S. (yes, no), first language was English (yes, no), public high school (yes, no), any parent's job related to computer science (yes, no), parents supported the learning of mathematics (Likert scale: 0 = very unsupportive; 5 = very supportive). According to the literature, these covariates might influence students' decisions about high school calculus and CS course enrollment. The gender, race, and SES disparity in CS affinity, persistence, and achievement has been well documented in the literature [40, 45, 54]. Prior research also showed that the decision-making process in the pursuit of STEM interest is very different for immigrant and non-immigrant students [2]. Students who went to public schools were less likely than those who went to private schools to have access to advanced CS courses [29]. Students

Table 3. Comparison of Background Information among High School Computer Science and Calculus Course Taking Groups

	No CS			Regular CS			AP CS		
	No Calc	Regular Calc	AP Calc	No Calc	Regular Calc	AP Calc	No Calc	Regular Calc	AP Calc
N	4,143	962	2,602	877	347	422	207	44	599
Male	75%	72%	68%	78%	68%	75%	88%	74%	75%
Born in the U.S.	75%	67%	81%	78%	54%	79%	79%	75%	74%
Parents' jobs relate to CS	19%	22%	26%	27%	24%	35%	31%	34%	34%
English is first language	70%	66%	72%	75%	56%	70%	69%	70%	67%
Went to public high school	78%	68%	78%	84%	69%	72%	78%	68%	76%
Asian	18%	28%	32%	15%	33%	35%	22%	23%	42%
Hispanic	15%	10%	12%	13%	7%	9%	17%	11%	6%
Black	11%	8%	5%	11%	11%	4%	9%	14%	5%
White	59%	54%	59%	62%	44%	55%	60%	59%	48%
Father's education[a]	2.22 (1.27)	2.68 (1.23)	2.88 (1.23)	2.25 (1.23)	2.43 (1.30)	2.92 (1.17)	2.42 (1.19)	2.67 (1.24)	3.13 (1.11)
Mother's education[a]	2.22 (1.20)	2.61 (1.16)	2.75 (1.16)	2.31 (1.17)	2.46 (1.21)	2.86 (1.08)	2.41 (1.10)	2.35 (1.31)	2.96 (1.09)
Parents support math learning[b]	2.90 (1.80)	2.96 (1.72)	3.09 (1.60)	3.32 (1.69)	3.37 (1.72)	3.65 (1.51)	3.78 (1.58)	4.00 (1.57)	3.87 (1.46)
High school English grade[c]	3.46 (0.72)	3.68 (0.56)	3.78 (0.49)	3.46 (0.69)	3.62 (0.63)	3.75 (0.48)	3.58 (0.58)	3.67 (0.55)	3.70 (0.59)

[a]Parental education predictors were coded as 0 = below high school, 1 = high school, 2 = 2-yr college or associate degree, 3 = 4-yr college, 4 = graduate school.
[b]Parents support math learning was a 0–5 Likert scale variable (0=very unsupportive, 5=very supportive).
[c]High school English grade was coded as A+ = 4.33, A = 4, A- = 3.67 … F = 0.

with stronger mathematics or CS affinity or self-efficacy might be more interested in enrolling in advanced calculus or CS courses in high school [6, 39, 46, 59]. However, to be able to argue that affinity or efficacy measures influenced students course selection in high school, these measures would have to refer to a time before the high school course selection, such as the beginning of high school. In this retrospective study, it would have been problematic to have students accurately recall their mindset three or more years ago. Instead, we asked if students' parents' jobs were related to computer science and if their parents supported their learning of mathematics, with the assumption that parental support in a specific subject may be strongly correlated to students' affinity and self-efficacy in the subject [1, 24, 60, 61]. We will discuss the limitation of this strategy further in the Limitations section of the article.

The grade students received in their last high school English course was also included as a covariate ("A+" = 4.33, "A" = 4, "A-" = 3.67, …, "Pass" = 2.8, "F" = 0). While not all students took advanced mathematics courses or computing courses in high school, all students took an English course, so this variable served as a proxy for general high school performance.

Table 3 shows the summary statistics for background variables by course taking groups. Most noticeably, students who took AP calculus consistently had higher levels of parental education and better high school English grades. Such differences between groups indicated a potential self-selection bias. This further motivates the use of regression and propensity techniques to

disentangle the effects of different levels of course taking from parental education and other confounding variables.

4.7 Analytic Models

We built a sequence of regression models to predict the final grade in college introductory CS courses, using our key predictors (high school CS and calculus courses and the interaction effects between these courses). We first built models (M1.1 and M1.2) without interaction terms that show the (non)additive effect of enrollment in both CS and calculus courses, without controlling for any covariates. Next, we built models that included the interaction effects, again without controlling for covariates (M2.1 and M2.2). The M1s and M2s presented the observed differences in final grade between the course taking groups, as they would appear, for instance, to college computer science professors taking a quick survey of their students, either considering or not considering the interaction effects.

Note that it is impossible to fully compensate for a potential self-selection bias, such as the one mentioned above, by controlling for background variables in a multiple regression. A more robust approach would be applying multinomial propensity score weighting (see example in Reference [19]). Hence, we used multinomial propensity weighting to balance the covariates between groups. Afterwards, we built multiple regression models M3.1 and M3.2. Only significant interaction effects were kept in the final models. In models M1.2, M2.2, and M3.2, we added the high school course grade, which was absent from the counterpart models (M1.1, M2.1, and M3.1). Because students were nested in universities, we ran multi-level regression models to account for variation between universities. However, the intraclass correlations were between 0.01 and 0.03, indicating that very little variation was explained by university clustering. In addition, the multilevel model yielded nearly identical parameter estimations for the key predictors. Therefore, we report flat models in this article.

5 RESULTS

The final models are presented in Table 4. We focus our report on four models: M1.1, M2.1, M3.1, and M3.2. Although M1.1 and M2.1 did not control for any covariates, we still deemed them valuable, because these models reflected the surface group differences, as immediately observed by college computer science teachers and other stakeholders. However, if one wishes to obtain a more precise estimate of the effect on college computer science performance of taking one of the high school courses, M3.1 should be used. Furthermore, we used M3.2 to equate each of the non-CS-AP groups with the CS-AP group at specific CS-AP grades.

According to M1.1, a surface model without interaction terms between high school CS and calculus course taking, AP CS and regular CS did not significantly predict college CS final grades. AP calculus and regular calculus were significant and positive predictors, and the effect size (on the 100-point scale) of AP calculus ($\beta = 4.96$) was 1.76 times that of regular calculus ($\beta = 2.81$) and 6.70 times that of AP CS ($\beta = 0.74$). A post hoc test showed that the difference between the effect of AP CS and AP calculus was statistically significant ($p < 0.001$).

Next, we consider M2.1, a surface model that included interaction terms for enrolling in both high school CS and calculus courses. Based on this model, high school AP CS, AP calculus, and regular calculus had significant and positive effects on college CS grade, whereas high school regular CS had no significant effect. In the absence of interaction effect (either CS course = 0 or calculus course = 0), the effect size (on the 100-point scale) of AP CS ($\beta = 2.60$) was similar to that of regular calculus ($\beta = 2.82$), but only about half the effect size of AP calculus ($\beta = 5.19$). A post hoc test showed that the difference between the effect of AP CS and AP calculus was statistically significant ($p < 0.001$).

Table 4. Models Predicting Final Grades in College Introductory to Computer Sciences

	M1.1[a]		M1.2		M2.1		M2.2		M3.1		M3.2	
	β	s.e.	β	s.e.	β	s.e.	β	s.e.	β	s.e.	β	s.e.
Intercept	82.14	0.21***	82.22	0.21***	82.06	0.21***	82.14	0.21***	81.57	0.74***	82.58	0.75***
CS courses (vs. no CS course)												
regular CS	0.32	0.39	0.70	0.40	0.33	0.39	0.71	0.40	0.50	0.32	0.90	0.33**
AP CS	0.74	0.52	0.73	0.53	2.60	0.91**	2.89	0.95**	3.12	0.54***	3.31	0.44***
Calculus courses (vs. no calculus course)												
Regular Calculus	2.81	0.43***	3.01	0.45***	2.82	0.43***	3.02	0.45***	1.64	0.76***	1.64	0.40***
AP Calculus	4.96	0.31***	4.71	0.32***	5.19	0.33***	4.95	0.33***	2.98	0.72***	2.87	0.39***
Interaction effect												
AP CS × AP Calculus					−2.76	1.11*	−3.17	1.15**	−3.44	0.72***	−4.35	0.75***
AP CS × Regular Calculus									−1.53	0.76*	−0.49	0.79
Control variables												
Gender (male vs. female)									−1.94	0.32***	−2.22	0.32**
Born in the U.S.									−1.11	0.42*	−0.27	0.43
Parents' jobs relate to CS									−0.65	0.33*	−0.04	0.33
Parents support math learning[b]									0.51	0.09***	0.26	0.09***
English is first language									1.12	0.42**	1.11	0.42**
Father education[c]									0.81	0.14***	0.66	0.15***
Mother education									−0.07	0.15	−0.21	0.15
Asian vs. White									0.37	0.37	0.24	0.37
Hispanic vs. White									−0.33	0.46	0.14	0.47
Black vs. White									−5.45	0.51***	−4.71	0.50***
Went to public high school									−0.32	0.34	−0.10	0.35
High school English grade[d]									2.40	0.16***	1.59	0.17***
Course grade (if the course was taken)[e]												
AP CS			1.40	0.50**			1.67	0.51**			2.87	0.23***
Regular CS			2.83	0.36***			2.81	0.36***			1.64	0.26***
AP Calculus			2.36	0.24***			2.35	0.24***			1.25	0.21***
Regular Calculus			3.35	0.38***			3.36	0.38***			1.91	0.25***

[a]M1s are models without any control variable or interaction effect; M2s include interaction effect based on M1s; M3s are models applied with multinomial propensity score weighting. *p < 0.05, **p < 0.01, ***p < 0.001 after Bonferroni adjustment for multiple comparison among courses.

[b]Parents support math learning was a Likert scale variable (0=very unsupportive, 5=very supportive); it was standardized and centered at zero.

[c]Parental education predictors were coded as 0 = below high school, 1 = high school, 2 = 2-yr college or associate degree, 3 = 4-yr college, 4 = graduate school.

[d]High school English grade was standardized and centered at zero.

[e]Course grades were standardized and centered at zero. Each course grade in this model was equivalent to interaction effect between a course and its grade.

There was an interaction effect between AP CS and AP calculus. The sign of the interaction effect was negative, and the magnitude of the effect was similar to the main effect of AP CS. This interaction effect showed that, if a student took both AP CS and AP calculus, the effects of the two courses did not add up; instead, the student would be expected to achieve a similar college CS grade to those who only took AP calculus.

A comparison of the effects of each of the course taking combinations, before controlling for covariates (according to M2.1) is shown in Figure 1. The x-axis is type of high school CS course; the colors indicate the various high school calculus course types. For each calculus course type, we see that Regular CS was not different from no-CS. We also see that AP CS was higher than Regular CS or no-CS, except for those who took AP calculus. Those who took AP calculus ranked in the top tier of expected college CS grades (between 86 and 88 grade points), regardless of their high school CS course enrollment. A combination of AP CS and regular calculus could also propel the predicted college CS grade to the first tier. However, the predicted college CS grade for those who took high school AP CS, but did not take any calculus, ranked in the second tier (between 84 and 86), about 2 points below the first tier.

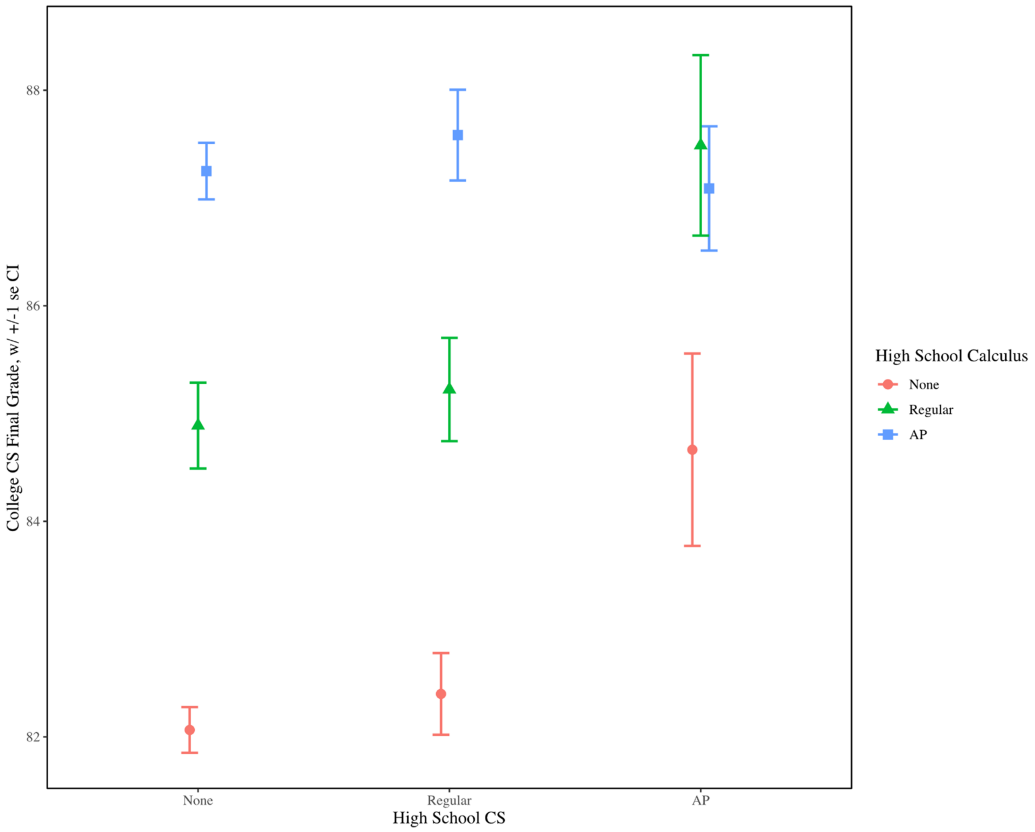

Fig. 1. Predicted college CS final grade by different high school course taking groups, based on M2.1.

After applying multinomial propensity score weighting and controlling for the covariates in M3.1, the effects of AP CS and AP calculus became similar to each other. According to M3.1, the effect size (on the 100-point scale) of AP CS ($\beta = 3.12$) was not significantly different from that of AP calculus ($\beta = 2.98$). M3.1 also found that regular CS did not have a significant effect, and that regular calculus had a significant positive effect, which was, however, smaller than those of AP CS and AP calculus ($p < 0.01$ in post hoc tests). In addition, M3.1 yielded a significant and negative interaction effect between AP CS and AP calculus. M3.1 also found a negative interaction effect between AP CS and regular calculus. These interaction effects removed the potential cumulative effects of dual enrollment. In Figure 2, we graphed college CS grade by each group. Different from Figure 1, Figure 2 shows that AP CS and AP calculus had nearly identical effects. Students who took either one of the two courses in high school were predicted to attain the first-tier grade in college CS. Similar to Figure 1, Figure 2 showed that students who took an additional course on top of AP CS or AP calculus were predicted to receive no additional benefits.

5.1 Course Grade

Including the course grade into the model allowed us to estimate the grade in a calculus course one needed to achieve to compensate the lack of AP CS, or conversely, the grade in an AP CS course one needed to achieve to compensate the lack of regular calculus or AP calculus.

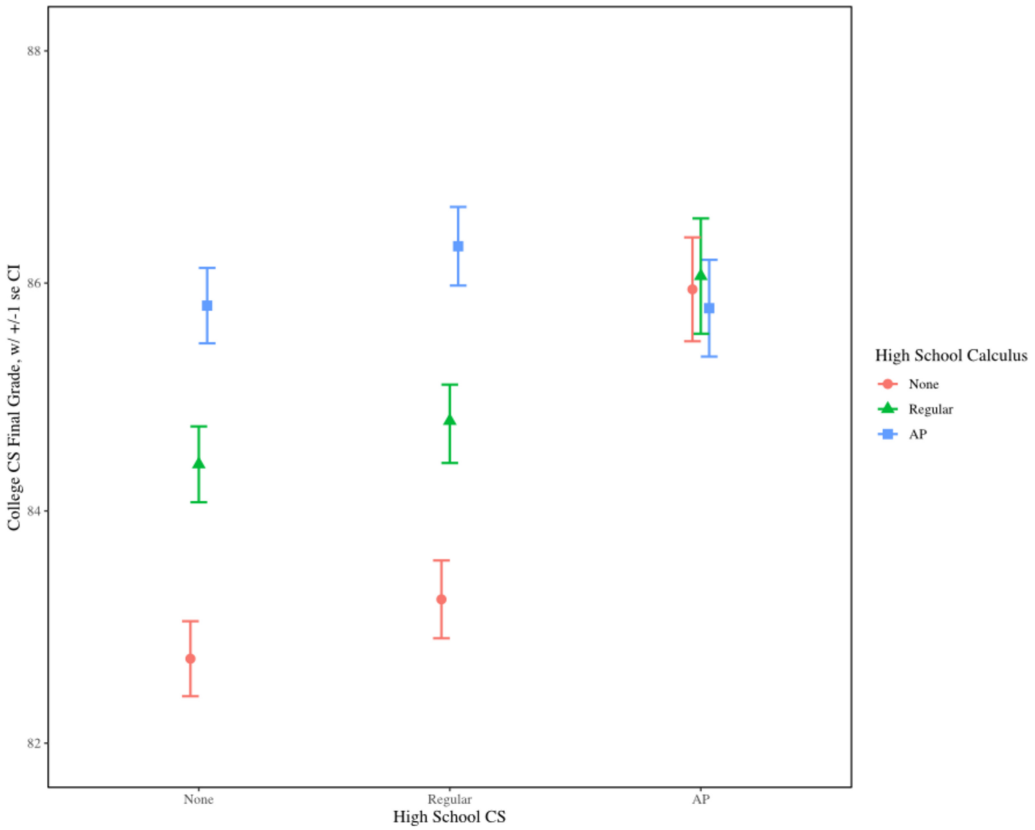

Fig. 2. Predicted college CS final grade by different high school course taking groups, based on M3.1.

First, focusing on M1.2 or M2.2 (the model without any controls), because taking regular calculus had nearly the same effect (higher by 0.13 points and not significantly different) as taking AP CS, students who took regular calculus (no CS) and achieved an average grade in that course were expected to receive a similar college CS grade as were students who took, and received an average grade in, high school AP CS. Because taking AP calculus had a larger effect (by 2.06 points) than taking AP CS, students who took AP calculus (no CS) and received grade B, 0.88 SD below the average, were still expected to achieve a similar college CS grade as students who took, and received an average grade (A-) in, high school AP CS. Conversely, students who took AP CS but did not take any calculus needed to achieve 1.23 SD (A+) above the average grade in AP CS to achieve the same college CS grade as students who took and received an average grade (A-) in, AP calculus. In short, this analysis suggested that students who took AP calculus had an advantage over those who took AP CS.

Second, focusing on M3.2 (the model with propensity score weighting), we found a result that was markedly different and even reverse from that of the model without any controls (M1.2). Students who took AP calculus (no CS) and achieved grade A, 0.8 SD above the average grade, in that course were expected to receive a similar college CS grade as students who took, and received only an average grade (A-) in high school AP CS. To reach similar college grades as the average former AP CS student, students who only took regular calculus needed, on average, to achieve even an A+, 1 SD above the average regular calculus grades.

Conversely, students who took AP CS but did not take any calculus could achieve A-/B+, 0.17 SD below the average grade, in AP CS and would still be expected to achieve the same college CS grade as students who took, and received an average grade (A-) in, AP calculus. Furthermore, they could earn grade B+, 0.6 SD below the average grade, in AP CS to achieve the same grade as those who took and received an average grade (A-) in, regular calculus. In short, the analysis after applying propensity score weighting suggested that students who took AP CS had a considerable advantage over students who only took regular calculus and a slight advantage over students who only took AP calculus.

Finally, it is noteworthy that, according to the model with propensity score weighting (M3.2), the final grade in college CS appeared to be more sensitive to AP CS grades than to other course grades. One standard deviation increase in AP CS was associated with 3.00 points increase in college CS grades, whereas the effect sizes for other courses ranged between 1.25 to 1.88 points (in all courses, 1 standard deviation was about 1 letter grade).

6 DISCUSSION

To summarize, we have five major findings:

Finding 1. When we conducted a raw comparison without accounting for background variables, AP calculus appeared to provide a considerable advantage over AP CS in terms of college CS achievement. Students who only took AP CS needed to achieve top grades in AP CS to compensate the lack of AP calculus.

Finding 2. However, this advantage was confounded by students' background characteristics—most noticeably parental education. After compensating for these, either by regression controls or propensity weighting, we found that the taking of AP CS and AP calculus had similar positive effects on students' college CS grades, on average.

Finding 3. Taking both AP CS and AP calculus did not provide an additional advantage.

Finding 4. All models showed that a regular CS course did not have a significant effect on college CS grades.

Finding 5. A regular calculus course had a significant positive effect on college CS grades. According to the propensity score weighting model, this effect was about half the size of the effect of AP CS. Students who only took regular calculus needed achieve top grades in that course to compensate the lack of AP CS.

Finding 1, based on a raw comparison, supports the observation from the perspective of college CS teachers and stakeholders that, in general, prior mathematics preparation is a stronger predictor of success in introductory college CS than is prior CS preparation (e.g., References [14, 65]). Based on this observation, a college CS teacher may over-simplistically advise a high school student to take AP calculus only, and that then it would not matter whether any CS course is taken, or not. If a high school student has his/her mind set on AP CS and he/she wanted to condense Finding 1 of this study to optimally prepare them for success in introductory college CS, then they would follow this advice: If you are able to, take AP calculus, and then it does not matter whether you take any CS course, or not. If you have your mind set on an AP CS course, you additionally should take at least regular calculus (or AP calculus).

However, Finding 2 revealed that the advantage of taking AP calculus over AP CS was largely confounded by background characteristics, such as parental education and general academic capability (using high school English grade as a proxy). After controlling or balancing these confounding variables, we estimated the effect of taking AP calculus to be not significantly different from that of taking AP CS. In fact, according to the model with propensity score weighting, the point estimate for those who took AP CS was higher than those who took AP calculus (not statistically

significant). After reducing the selection bias, more pertinent advice to a high school student who wishes to prepare for success in college CS is that taking either AP CS or AP calculus would be equally beneficial to college CS achievement.

Regarding educational policy, an increasing number of states in the United States have allowed CS courses to count as mathematics courses (10 states in 2010 and 47 states in 2019) [20]. This trend has raised concerns for different stakeholders. For example, the National Council of Teachers of Mathematics [41] reacted by demanding that only courses that explicitly teach mathematics should be counted as mathematics. The Council stated that "a computer science course should be considered as a substitute for a mathematics course graduation requirement only if the substitution does not interfere with a student's ability to complete core readiness requirements in mathematics" (p. 1). Our findings, in response to this debate, suggest that, if a student wishes to excel in college computer science, taking AP CS in place of mathematics classes may not be a bad idea. However, we do not know if by taking only AP CS they may miss the chance to learn other important skills that are only offered by traditional calculus (or other mathematics) classes. Conversely, for students who do not have access to AP CS courses in high school, taking AP calculus is expected to bring them to equivalent grades in college introduction to CS, compared with those who took AP CS.

Our data cannot fully explain the cause behind the effectiveness of AP CS and AP calculus. However, we can discuss how our results can be interpreted through the lens of the transfer of learning theory. According to the transfer of learning theory, if learners acquired similar transferable skills from AP CS and AP calculus, learners would have an equivalent single route to college CS success, regardless of which of the two high school courses they started off with. This single-route hypothesis posits that both AP CS and AP calculus introduce students to similar skills (or general STEM self-efficacy)—such as problem-solving, computational or algorithmic thinking skills—that are proven to be the cornerstones for long-term success in computer science [16, 18, 33]. Alternatively, if learners acquired distinct transferable skills from the two courses, they should be able to achieve college CS success on a dual route pathway. This dual-route hypothesis posits that AP CS and AP calculus provide unique routes to students' success in college CS. For example, AP CS may have emphasized technical coding skills or computational thinking skills, whereas AP calculus may have built a solid mathematics foundation. According to the dual-route model, the two distinct skills should then add up and bring about additional advantages to students who took both AP CS and AP calculus. However, this prediction was contradicted by our Finding 3—the two AP courses did not add up, and taking both did not provide any advantage over taking either one. Therefore, our findings were in favor of the single-route model. We should also note that, given expected final grades in the high 80s, the assumption of a general ceiling effect would not be able to explain the lack of a cumulative relationship.

To some stakeholders, Finding 3 and its implication for a single-route hypothesis may send an interesting and perhaps counterintuitive message that, for the long-term achievement in CS, enrollment in both AP calculus and AP CS appeared redundant. In 1975, mathematician Garrett Birkhoff [25] proclaimed in *American Scientist* that "it is the skillful combination of mathematics and computer science—or more properly of individuals who understand both and can coordinate them—that holds the greatest promise for the future." However, the lack of a cumulative effect of enrollment in AP CS and AP calculus suggested that the two AP courses are not yet coordinated in an optimal approach to produce a superior outcome (at least in terms of introductory CS) than that of single AP course enrollment.

Finding 4 showed consistently across different model specifications that taking regular CS in high school was not effective in enhancing college CS performance. A possible explanation for this finding is that regular CS may have emphasized factual knowledge (such as the historical context and development of CS) or point-and-click skills (such as text editing) that do not directly

prepare for the skills that are essential for college CS (such as problem-solving and computational/algorithmic thinking skills).

Finding 5 showed a relatively small but statistically significant positive effect of regular calculus. A possible explanation was that regular calculus prepared students with some foundation for computer science but not as solid or advanced as AP courses. Based on these findings, a piece of advice for high school students who wish to excel in college CS would be that regular CS was not effective, regular calculus was partially effective, and, on average, an AP course may still be the best bet to reach top-tier grades in college CS.

Our findings suggest several avenues for future research. First, as K–12 CS education expands, studies such as this one should continue to document how these new CS opportunities are supporting success in introductory college CS courses. It will be particularly interesting to see to what extent, if any, the new AP CS Principles course has different effects from those found for AP CS A. Moreover, future studies should investigate not only how to support success in introductory courses, but also how to encourage students to major and pursue jobs in the field.

Our findings also suggest policy implications for educators. First, simply exposing students to some kind of CS before college does not appear to be sufficient. Educators also need to pay attention to the quality of the courses and experiences they provide for students. Finally, high school students who are interested in pursuing CS in college should understand that calculus preparation will likely crosscut with CS preparation. Comprehensively supporting students to succeed in CS requires not only attending to K–12 CS education but continuing to strengthen calculus preparation as well.

7 LIMITATIONS

This study has several limitations. First, we need to make the traditional disclaimer for correlational studies: They cannot prove causality, even though our propensity weighting technique is sometimes called quasi-experimental. We used the term "effect" in this article as a synonym for statistically significant association after controlling for other variables in a regression model.

Second, another limitation is that the survey was given in an introductory college CS class. Therefore, we know only about students who enrolled in that kind of class and cannot generalize our findings to students who did not enroll in the first place. This leads us to be cautious in interpreting some findings of this study. For example, it could be that high school CS (or calculus) courses are doing such a good job in preparing students that the best-prepared students skip introductory-level courses in college and immediately enroll in higher-level courses.

Third, the outcome variable in this study was the final grade in introductory level CS courses in college. The between-class variation (e.g., some instructors might be stricter in grading than other instructors) can be adjusted by specifying a multi-level model. However, there is established literature in statistics [53], cautioning that controlling for post-treatment information (in our case, college enrollment was a post-treatment factor that occurred after the high school course enrollment) can only increase the bias in the estimation of the treatment effect. Accordingly, the flat model should be preferred, despite omitting the information about between-class variation. In any case, as mentioned above, we did specify a multi-level model, which yielded a very low intraclass correlation and produced nearly identical results, compared with the flat model.

Fourth, the propensity score weighting technique helped us adjust for the imbalance between groups in terms of students' background information that may contribute to the self-selection bias and confound the effect of course taking. We could only weight for a list of some of the "usual suspects" that may influence students' course selection in high school. As mentioned earlier, one key variable that we did not measure and weight for was students' computing affinity and self-efficacy at the beginning of high school, which may have influenced a student's preference in course enrollment. However, these efficacy indicators at the beginning of high school would have been difficult

to measure, because students would have had to recall their mindset of three or more than three years ago. We could have collected students' efficacy at the beginning of college, but that would have been information about what took place after the high school course enrollment, which could not have affected the self-selection process and should be considered post-treatment information. Our approach to address this challenge was to collect, and weight for, information about students' parents' profession, i.e., if their parents' jobs were related to computing, because such students might be expected to have more positive self-efficacy and affinity to computing [1, 24, 60, 61]. Despite this effort, we are aware that students' prior self-efficacy or affinity had not been fully accounted for in this study. Future studies should find alternative strategies to measure students' self-efficacy and affinity in computing at an early age to address this challenge. Relatedly, we did not have measures of potential mediating variables (that would be post-treatment variables) such as computational thinking skills, coding skills, and self-efficacy that can help us explore the routes and mechanisms that lead to college CS success from different high school calculus and CS courses.

Last but not least, we want to remind the readers that college course grade is an incomplete metric for one's academic success, especially in the field of computer science, which is primarily product oriented. Although introductory courses are often the gatekeeper courses for more advanced-level studies, grades measured in our study do not necessarily presage long-term success in CS learning or a CS career. Future research should collect information about students' performance and engagement in CS in the even longer-term (such as by the end of college) and should use more holistic measures (such as hands-on projects and affinity to computing) to investigate how long the boost from high school CS or mathematics course enrollment can last beyond the college introductory CS courses. Future study should also examine the impact of non-course or out-of-school experiences in computing and mathematics on college CS success and how these non-course experiences may interact with course taking experiences.

8 CONCLUSION

Using a large sample drawn from colleges and universities across the U.S., this study examined the common observation by college CS teachers that prior course taking in calculus seems to predict college CS achievement more strongly than does prior course taking in CS. This study partially validated this observation in that course taking in regular calculus had a positive effect and course taking in regular CS did not. However, we noticed that the observed advantage of taking AP calculus over taking AP CS was largely due to self-selection bias. For high school students who started off with similar background factors (e.g., parental education), we estimated that the effect of taking AP calculus was similar to that of AP CS. Interestingly, enrollment in both AP calculus and AP CS did not have any additional positive effect, suggesting that both AP calculus and AP CS prepared for common skills that are important for long-term CS achievement—representing alternative and fairly equivalent pathways to success in introductory college CS courses.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation (grant number 1339200). Any opinions, findings, and conclusions in this article are the authors' and do not necessarily reflect the views of the National Science Foundation.

Without the excellent contributions of many people, the FICSIT project would not have been possible. We thank the members of the FICSIT team: Wendy Berland, Hal Coyle, Zahra Hazari, Annette Trenga, and Bruce Ward. We would also like to thank several STEM educators and researchers who provided advice or counsel on this project as members of our Advisory Board: Hal Abelson; Lecia Barker, Chair of Advisory Board; Randy Battat; Joanne Cohoon; Maria Litvin; Clayton Lewis; Irene Porro; Kelly Powers; Lucy Sanders; Susanne Steiger–Escobar; and Jane

Stout. Last but not least, we are grateful to the many college computer science professors and their students who gave up a portion of a class to provide data.

REFERENCES

[1] Annette E. Alliman-Brissett and Sherri L. Turner. 2010. Racism, parent support, and math-based career interests, efficacy, and outcome expectations among African American adolescents. *J. Black Psychol.* 36, 2 (2010), 197–225. DOI: https://doi.org/10.1177/0095798409351830

[2] Shaljan Areepattamannil and Berinderjeet Kaur. 2013. Factors predicting science achievement of immigrant and non-immigrant students: A multilevel analysis. *Int. J. Sci. Math. Educ.* 11, 5 (2013), 1183–1207. DOI: https://doi.org/10.1007/s10763-012-9369-5

[3] Michal Armoni and Judith Gal-Ezer. 2014. High school computer science education paves the way for higher education: The Israeli case. *Comput. Sci. Educ.* 24, 2–3 (2014), 101–122. DOI: https://doi.org/10.1080/08993408.2014.936655

[4] Nana Asabere, Amevi Acakpovi, Wisdom Torgby, Edwin Brew, and Kwame Ampadu. 2016. Towards a perspective of the role of mathematics in Computer Science and Engineering (CSE) Education. *Int. J. Comput. Sci. Telecommun.* 7, 1 (2016), 5–9.

[5] Douglas Baldwin, Henry M. Walker, and Peter B. Henderson. 2013. The roles of mathematics in computer science. *ACM Inroads* 4, 4 (2013), 74–80. DOI: https://doi.org/10.1145/2537753.2537777

[6] Monica Banzato, and Paolo Tosato. 2017. An exploratory study of the impact of self-efficacy and learning engagement. *Int. J. E-Learn.* 16, 4 (2017), 349–369.

[7] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads* 2, 1 (2011), 48–54. DOI: https://doi.org/10.1145/1929887.1929905

[8] Theresa Beaubouef. 2002. Why computer science students need math. *ACM SIGCSE Bull.* 34, 4 (2002), 57. DOI: https://doi.org/10.1145/820127.820166

[9] Theresa Beaubouef and John Mason. 2005. Why the high attrition rate for computer science students. *ACM SIGSCE Bull.* 37, 2 (2005), 103–106. DOI: https://doi.org/10.1145/1083431.1083474

[10] Jens Bennedssen and Michael E. Caspersen. 2008. Abstraction ability as an indicator of success for learning computing science? In *Proceedings of the ACM Workshop on International Computing Education Research (ICER'08).* ACM Press, New York, NY, 15–25. DOI: https://doi.org/10.1145/1404520.1404523

[11] Susan Bergin and Ronan Reilly. 2006. Predicting introductory programming performance: A multi-institutional multivariate study. *Comput. Sci. Educ.* 16, 4 (2006), 303–323. DOI: https://doi.org/10.1080/08993400600997096

[12] Sylvia Beyer. 2014. Why are women underrepresented in computer science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Comput. Sci. Educ.* 24, 2–3 (2014), 153–192. DOI: https://doi.org/10.1080/08993408.2014.963363

[13] Matt Brown. 2013. CS0 as an indicator of student risk for failure to complete a degree in computing. *J. Comput. Sci. Coll.* 28, 5 (2013), 9–16.

[14] D. F. Butcher and W. A. Muth. 1985. Predicting performance in an introductory computer science course. *Commun. ACM* 28, 3 (1985), 263–268. DOI: https://doi.org/10.1145/3166.3167

[15] Vehbi Celik and Etem Yesilyurt. 2013. Attitudes to technology, perceived computer self-efficacy and computer anxiety as predictors of computer supported education. *Comput. Educ.* 60, 1 (2013), 148–158. DOI: https://doi.org/10.1016/j.compedu.2012.06.008

[16] Chen Chen, Paulina Haduong, Karen Brennan, Gerhard Sonnert, and Philip Sadler. 2018. The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages. *Comput. Sci. Educ.* 29, 1 (2018), 23–48. DOI: https://doi.org/10.1080/08993408.2018.1547564

[17] Chen Chen, Stuart Jeckel, Gerhard Sonnert, and Philip M. Sadler. 2019. "Cowboy" and "cowgirl" programming: The effects of precollege programming experiences on success in college computer science. *Int. J. Comput. Sci. Educ. Sch.* 2, 4 (2019), 22–40. DOI: https://doi.org/10.21585/ijcses.v2i4.34

[18] Chen Chen, Gerhard Sonnert, Philip M. Sadler, and David J. Malan. 2020. Computational thinking and assignment resubmission predict persistence in a computer science MOOC. *J. Comput. Assist. Learn.* 36, 5 (2020), 581–594. DOI: https://doi.org/10.1111/jcal.12427

[19] Si Chen, Chen Zhao, Yan Cao, Chen Chen, Catherine E. Snow, and Mai Lu. 2019. Long-term effects of China's one village one preschool program on elementary academic achievement. *Early Child Res. Quart.* 49, (2019), 218–228. DOI: https://doi.org/10.1016/j.ecresq.2019.06.010

[20] Code.org. 2019. Annual report: The state of K–12 computer science. Retrieved from https://code.org/files/annual-report-2018.pdf.

[21] Stephen Cooper, Karen Wang, Maya Israni, and Sheryl Sorby. 2015. Spatial skills training in introductory computing. In *Proceedings of the ACM Conference on International Computing Education Research (ICER'15).* ACM, 13–20. DOI: https://doi.org/10.1145/2787622.2787728

[22] Krista D. Mattern, Emily J. Shaw, and Maureen Ewing. 2011. *Advanced Placement® Exam Participation: Is AP® Exam Participation and Performance Related to Choice of College Major?* Research Report No. 2011-6. College Board.

[23] Daniel Murphy and Barbara Dodd. 2009. *A Comparison of College Performance of Matched AP® and Non-AP Student Groups*. Research Report No. 2009-6. The College Board, New York.

[24] Jeanne M. Friedel, Kai S. Cortina, Julianne C. Turner, and Carol Midgley. 2010. Changes in efficacy beliefs in mathematics across the transition to middle school: Examining the effects of perceived teacher and parent goal emphases. *J. Educ. Psychol.* 102, 1 (2010), 102–114. DOI: https://doi.org/10.1037/a0017590

[25] Garrett Birkhoff. 1975. Mathematics and computer science: Skillful combinations of mathematics and computer science hold great promise for the future. *Amer. Sci.* 63, 1 (1975), 83–91.

[26] Garry White and Marcos Sivitanides. 2003. An empirical investigation of the relationship between success in mathematics and visual programming courses. *J. Inf. Syst. Educ.* 14, 4 (2003), 409–416.

[27] Mary L. Gick and Keith J. Holyoak. 1987. The cognitive basis of knowledge transfer. In *Transfer of Learning*, S. M. Cornier and J. D. Hagman (Eds.). Academic, New York, 81–120. DOI: https://doi.org/10.1016/b978-0-12-188950-0.50008-4

[28] P. Golding and S. McNamarah. 2005. Predicting academic performance in the School of Computing & Information Technology (SCIT). In *Proceedings of the Frontiers in Education Conference*. DOI: https://doi.org/10.1109/fie.2005.1612248

[29] Joanna Goode. 2007. If you build teachers, will students come? The role of teachers in broadening computer science learning for urban youth. *J. Educ. Comput. Res.* 36, 1 (2007), 65–88. DOI: https://doi.org/10.2190/2102-5g77-ql77-5506

[30] Annagret Goold and Russell Rimmer. 2000. Factors affecting performance in first-year computing. *ACM SIGCSE Bull.* 32, 2 (2000), 39–43. DOI: https://doi.org/10.1145/355354.355369

[31] Edward Holden and Elissa Weeden. 2004. The experience factor in early programming education. In *Proceedings of the 5th Conference on Information Technology Education*. ACM, 211–218.

[32] Ting Chia Hsu, Shao Chen Chang, and Yu Ting Hung. 2018. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Comput. Educ.* 126, (2018), 296–310. DOI: https://doi.org/10.1016/j.compedu.2018.07.004

[33] Cagin Kazimoglu, Mary Kiernan, Liz Bacon, and Lachlan Mackinnon. 2012. A serious game for developing computational thinking and learning introductory computer programming. *Procedia Soc. Behav. Sci.* 47, (2012), 1991–1999. DOI: https://doi.org/10.1016/j.sbspro.2012.06.938

[34] Donald E. Knuth. 1974. Computer science and its relation to mathematics. *Amer. Math. Mon.* 81, 4 (1974), 323–343. DOI: https://doi.org/10.1080/00029890.1974.11993556

[35] Leslie Keng and Barbara G. Dodd. 2008. *A Comparison of College Performances of AP and Non-AP Student Groups in 10 Subject Areas*. The College Board, New York.

[36] Joanne Lobato. 2012. The actor-oriented transfer perspective and its contributions to educational research and practice. *Educ. Psychol.* 47, 3 (2012), 232–247. DOI: https://doi.org/10.1080/00461520.2012.693353

[37] Patricia Lund Casserly. 1986. *Advanced Placement Revisited*. Research Report No. 86-6. The College Board, New York.

[38] Philip M. Sadler, Gerhard Sonnert, Robert H. Tai, and Kristin Klopfenstein. 2010. *AP: A Critical Examination of the Advanced Placement Program*. Harvard Education Press, Cambridge, MA.

[39] Jonathan Mahadeo, Zahra Hazari, and Geoff Potvin. 2020. Developing a computing identity framework: Understanding computer science and information technology career choice. *ACM Trans. Comput. Educ.* 20, 1 (2020), 7–714. DOI: https://doi.org/10.1145/3365571

[40] Jane Margolis, Rachel Estrella, Joanna Goode, Jennifer Jellison Holme, and Kim Nao. 2017. *Stuck in the Shallow End: Education, Race, and Computing*. The MIT Press, Cambridge, MA.

[41] National Council of Teachers of Mathematics. 2016. Computer science and mathematics education, a position of the National Council of Teachers of Mathematics. Retrieved from https://www.nctm.org/uploadedFiles/Standards_and_Positions/Position_Statements/Computer%20science%20and%20math%20ed%20022416.pdf.

[42] Michael Kölling. 1999. The problem of teaching object-oriented programming, Part 1: Languages. *J. Obj.-orient. Prog.* 11, 8 (1999), 8–15.

[43] Yoshitaka Nakakoji and Rachel Wilson. 2020. Interdisciplinary learning in mathematics and science: Transfer of learning for 21st century problem solving at university. *J. Intell.* 8, 3 (2020), 32. DOI: https://doi.org/10.3390/jintelligence8030032

[44] Chris Orban. 2019. Computer science now counts as math credit in most states—Is this a good idea? Retrieved from https://theconversation.com/computer-science-now-counts-as-math-credit-in-most-states-is-this-a-good-idea-123424.

[45] Miranda C. Parker, Amber Solomon, Brianna Pritchett, David A. Illingworth, Lauren E. Margulieux, and Mark Guzdial. 2018. Socioeconomic status and computer science achievement: Spatial ability as a mediating variable in a novel model

of understanding. In *Proceedings of the ACM Conference on International Computing Education Research (ICER'18)*. ACM, 97–105. DOI: https://doi.org/10.1145/3230977.3230987

[46] Philip David Parker, Herbert W. Marsh, Joseph Ciarrochi, Sarah Marshall, and Adel Salah Abduljabbar. 2014. Juxtaposing math self-efficacy and self-concept as predictors of long-term achievement outcomes. *Educ. Psychol. UK* 34, 1 (2014), 29–48. DOI: https://doi.org/10.1080/01443410.2013.797339

[47] Hadi Partovi. 2017. Should computer science be a mandatory part of a high school curriculum? *Quora*. Retrieved from https://www.quora.com/Should-Computer-Science-be-a-mandatory-part-of-a-high-school-curriculum/answer/Hadi-Partovi.

[48] D. N. Perkins and Gavriel Salomon. 1989. Are cognitive skills context-bound? *Educ. Res.* 18, 1 (1989), 16–25. DOI: https://doi.org/10.3102/0013189x018001016

[49] Ayman Qahmash, Mike Joy, and Adam Boddison. 2015. To what extent mathematics correlates with programming: Statistical analysis. In *Proceedings of the International Conference on Computer Science Education Innovation Technology (CSEIT'15)*. Global Science Technology Forum Pte Ltd., 69–80. DOI: https://doi.org/10.5176/2251-2195_cseit15.18

[50] Anthony Ralston. 2005. Do we need ANY mathematics in computer science curricula? *ACM SIGCSE Bull.* 37, 2 (2005), 6–9. DOI: https://doi.org/10.1145/1083431.1083433

[51] Sarah Rauchas, Benjamin Rosman, George Konidaris, and Ian Sanders. 2006. Language performance at high school and success in first year computer science. *ACM SIGCSE Bull.* 38, 1 (2006), 398. DOI: https://doi.org/10.1145/1124706.1121467

[52] Christine Reilly, Emmett Tomai, and Laura M. Grabowski. 2015. An evaluation of how changes to the introductory computer science course sequence impact student success. In *Proceedings of the IEEE Frontiers in Education Conference*. 1–6. DOI: https://doi.org/10.1109/fie.2015.7344029

[53] Paul R. Rosenbaum. 1984. The consequences of adjustment for a concomitant variable that has been affected by the treatment. *J. Roy. Statist. Soc. Ser. Gen.* 147, 5 (1984), 656. DOI: https://doi.org/10.2307/2981697

[54] Linda J. Sax, Kathleen J. Lehman, Jerry A. Jacobs, M. Allison Kanny, Gloria Lim, Laura Monje-Paulson, and Hilary B. Zimmerman. 2017. Anatomy of an enduring gender gap: The evolution of women's participation in computer science. *J. High Educ.* 88, 2 (2017), 258–293. DOI: https://doi.org/10.1080/00221546.2016.1257306

[55] Martina Schollmeyer. 1996. Computer programming in high school vs. college. *ACM SIGCSE Bull.* 28, 1 (1996), 378–382. DOI: https://doi.org/10.1145/236462.236584

[56] Gerhard Sonnert and Philip M. Sadler. 2014. The impact of taking a college pre-calculus course on students' college calculus performance. *Int. J. Math. Educ. Sci. Technol.* 45, 8 (2014), 1188–1207. DOI: https://doi.org/10.1080/0020739x.2014.920532

[57] Fan Tai-sheng and Li Yi-ching. 2002. Is math ability beneficial to performance in college computer science programs. *J. Nat. Taipei Teach. Coll.* 15, 1 (2002), 69–98.

[58] Harriet G. Taylor and Luegina C. Mounfield. 1989. The effect of high school computer science, gender, and work on success in college computer science. In *Proceedings of the 20th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'89)*. ACM, New York, NY, 195–198. DOI: https://doi.org/10.1145/65293.65309

[59] Galen E. Turner, Eric D. Deemer, Heath E. Tims, Krystal Corbett, and Jeremy Mhire. 2014. *Cyber Value and Interest Development: Assessment of a STEM Career Intervention for High School Students*. Retrieved from http://ejse.southwestern.edu.

[60] Sherri L. Turner, Jason C. Steward, and Richard T. Lapan. 2004. Family factors associated with sixth-grade adolescents' math and science career interests. *Career Dev. Quart.* 53, 1 (2004), 41–52. DOI: https://doi.org/10.1002/j.2161-0045.2004.tb00654.x

[61] Ioanna Vekiri and Anna Chronaki. 2008. Gender issues in technology use: Perceived social support, computer self-efficacy and value beliefs, and computer use beyond school. *Comput. Educ.* 51, 3 (2008), 1392–1404. DOI: https://doi.org/10.1016/j.compedu.2008.01.003

[62] Philip R. Ventura. 2005. Identifying predictors of success for an objects-first CS1. *Comput. Sci. Educ.* 15, 3 (2005), 223–243. DOI: https://doi.org/10.1080/08993400500224419

[63] Russell T. Warne, Ross Larsen, Braydon Anderson, and Alyce J. Odasso. 2015. The impact of participation in the advanced placement program on students' college admissions test scores. *J. Educ. Res.* 108, 5 (2015), 400–416. DOI: https://doi.org/10.1080/00220671.2014.917253

[64] Christopher Watson, Frederick W. B. Li, and Jamie L. Godwin. 2014. No tests required: Comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE'14)*. 469–474. DOI: https://doi.org/10.1145/2538862.2538930

[65] Brenda Cantwell Wilson and Sharon Shrock. 2001. Contributing to success in an introductory computer science course: A study of twelve factors. *ACM SIGCSE Bull.* ACM, 184–188. DOI: https://doi.org/10.1145/366413.364581

Received June 2020; revised October 2020; accepted November 2020