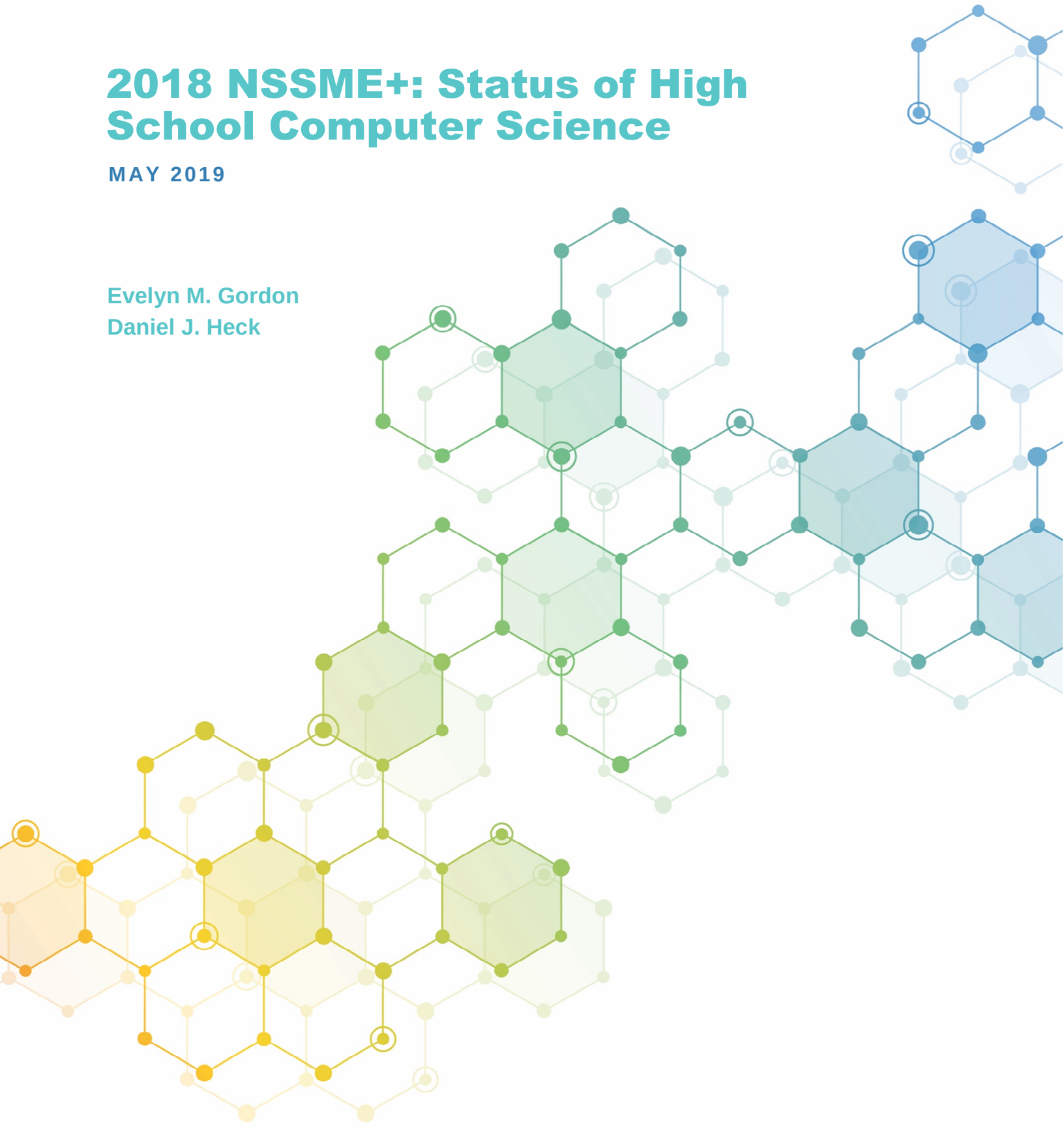
NSSME | THE NATIONAL SURVEY OF SCIENCE & MATHEMATICS EDUCATION

2018 NSSME+: Status of High School Computer Science

MAY 2019

Evelyn M. Gordon
Daniel J. Heck



Disclaimer

The *2018 NSSME+: Status of High School Computer Science* was prepared with support from the National Science Foundation under grant number DGE-1642413. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Suggested Citation

Gordon, E. M. & Heck, D. J. (2019). *2018 NSSME+: Status of high school computer science*. Chapel Hill, NC: Horizon Research, Inc.

Additional Information

More details and products from the 2018 NSSME+, as well as previous iterations of the study, can be found at: http://horizon-research.com/NSSME/

TABLE OF CONTENTS

LIST OF TABLES

High School Computer Science Courses Offered

High School Computer Science Instruction

Resources Available for High School Computer Science

Factors Affecting High School Computer Science Instruction

Introduction

In 2018, the National Science Foundation supported the sixth in a series of surveys through a grant to Horizon Research, Inc. The first survey was conducted in 1977 as part of a major assessment of science and mathematics education and consisted of a comprehensive review of the literature; case studies of 11 districts throughout the United States; and a national survey of teachers, principals, and district and state personnel. A second survey of teachers and principals was conducted in 1985–86 to identify trends since 1977. A third survey was conducted in 1993, a fourth in 2000, and a fifth in 2012. This series of studies has been known as the National Survey of Science and Mathematics Education (NSSME).

The 2018 iteration of the study included an emphasis on computer science, particularly at the high school level, which is increasingly prominent in discussions about K–12 STEM education and college and career readiness. The 2018 NSSME+ (the plus symbol reflecting the additional focus) was designed to provide up-to-date information and to identify trends in the areas of teacher background and experience, curriculum and instruction, and the availability and use of instructional resources. The research questions addressed by the study are:

1. To what extent do computer science, mathematics, and science instruction reflect what is known about effective teaching?
2. What are the characteristics of the computer science/mathematics/science teaching force in terms of race, gender, age, content background, beliefs about teaching and learning, and perceptions of preparedness?
3. What are the most commonly used textbooks/programs, and how are they used?
4. What influences teachers' decisions about content and pedagogy?
5. What formal and informal opportunities do computer science/mathematics/science teachers have for ongoing development of their knowledge and skills?
6. How are resources for computer science/mathematics/science education, including well-prepared teachers and course offerings, distributed among schools in different types of communities and different socioeconomic levels?

The 2018 NSSME+ is based on a national probability sample of schools and computer science, mathematics, and science teachers in grades K–12 in the 50 states and the District of Columbia. The sample was designed to yield national estimates of course offerings and enrollment, teacher background preparation, textbook usage, instructional techniques, and availability and use of facilities and equipment. Every eligible school and teacher in the target population had a known, positive probability of being sampled. A total of 7,600 computer science, mathematics, and science teachers in 1,273 schools across the United States participated in this study, a response rate of 78 percent.

This report describes the status of high school (grades 9–12) computer science instruction based on the responses of 289 computer science teachers.[1] Details on the survey sample design, data collection and analysis procedures, and creation of composite variables[2] are included in the *Report of the 2018 NSSME+*.[3] Occasionally, comparisons to high school science and mathematics teachers are made in this report; detailed results for these groups can be found in the main 2018 NSSME+ report. The standard errors for the estimates presented in this report are included in parentheses in the tables. The narrative sections of the report generally point out only those differences that are substantial as well as statistically significant at the 0.05 level.

This status report of high school computer science teaching is organized into major topical areas:

- Characteristics of the computer science teaching force;
- Professional development of computer science teachers;
- Computer science courses offered;
- Computer science instruction, in terms of time spent, objectives, and activities;
- Resources available for computer science instruction; and
- Factors affecting computer science instruction.

In addition, each section contains a set of analyses examining the distribution of key outcomes across schools and classes of different demographic characteristics. For these analyses, data are examined by four school-level factors and two class-level factors:

School level

1. Percentage of students in the school eligible for free/reduced-price lunch,
2. School size,
3. Community type, and
4. Region of the country.

Class level

1. Prior achievement level of students, and
2. Percentage of students in the class from race/ethnicity groups historically underrepresented in STEM fields.[4]

Additional information about these factors is included in Appendix D of the *Report of the 2018 NSSME+*. Although the specific equity factors displayed in the body of this report vary by

[1] A computer science teacher is defined as someone who teaches at least one class of computer science that includes programming or requires programming as a prerequisite.

[2] Factor analysis was used to create several composite variables related to key constructs measured on the questionnaires. Composite variables, which are more reliable than individual survey items, were computed to have a minimum possible value of 0 and a maximum possible value of 100.

[3] Banilower, E. R., Smith, P. S., Malzahn, K. A., Plumley, C. L., Gordon, E. M., & Hayes, M. L. (2018). *Report of the 2018 NSSME+*. Chapel Hill, NC: Horizon Research, Inc.

[4] Includes students identified as American Indian or Alaskan Native, Black or African American, Hispanic or Latino, or Native Hawaiian or Other Pacific Islander.

outcome, tables showing each examined outcome by all relevant equity factors are included in Appendix E of the *Report of the 2018 NSSME+*.

High School Computer Science Teachers' Backgrounds and Beliefs

A well-prepared teaching force is essential for an effective education system. This section provides data about the nation's high school computer science teachers, including demographic data, teaching experience, college degree and coursework, beliefs about teaching and learning, and perceptions of preparedness.

Teacher Characteristics

As can be seen in Table 1, nearly all high school computer science teachers characterize themselves as white. The majority of high school computer science teachers are male, unlike the high school science and mathematics teaching force. Although nearly half have more than 10 years of experience teaching at the K–12 level, many are novice teachers of computer science, with about a third having two or fewer, and two-thirds having five or fewer, years of experience teaching the subject.

Table 1
Characteristics of the High School Computer Science Teaching Force

	PERCENT OF TEACHERS
Sex	
Female	40 (3.6)
Male	60 (3.6)
Other	0 ---†
Hispanic or Latino	
Yes	8 (2.2)
No	92 (2.2)
Race	
White	94 (1.7)
Asian	4 (1.4)
Black or African American	3 (1.3)
American Indian or Alaskan Native	2 (0.5)
Native Hawaiian or Other Pacific Islander	1 (0.6)
Age	
≤ 30	12 (2.9)
31–40	31 (3.8)
41–50	25 (3.3)
51–60	21 (2.8)
61 +	11 (2.8)
Experience Teaching any Subject at the K–12 Level	
0–2 years	10 (2.2)
3–5 years	19 (3.2)
6–10 years	23 (3.0)
11–20 years	32 (3.4)
≥ 21 years	15 (2.6)
Experience Teaching Computer Science at the K–12 Level	
0–2 years	35 (3.8)
3–5 years	28 (2.8)
6–10 years	16 (2.7)
11–20 years	18 (2.6)
≥ 21 years	3 (1.2)
Full Time Job in Computer Science Prior to Teaching	
Yes	35 (4.3)
No	65 (4.3)

† No high school computer science teachers in the sample selected this response option. Thus, it is not possible to calculate the standard error of this estimate.

Equity analyses were conducted to examine how teachers are distributed—for example, whether teachers with the least experience are concentrated in high-poverty schools (i.e., schools with high proportions of students eligible for free/reduced-price lunch). Table 2 shows the percentage of classes taught by teachers with varying experience teaching computer science in schools with different proportions of students eligible for free/reduced-price lunch. A majority of computer science classes in the highest quartile (i.e., schools with the most students eligible for free/reduced-price lunch) are taught by teachers with only 0–2 years of experience teaching the

subject. In contrast, only about a quarter of classes in schools with the fewest eligible students are taught by teachers with such limited experience.

Table 2

Equity Analysis of High School Computer Science Classes Taught by Teachers With Varying Experience, by Proportion of Students Eligible for Free/Reduced-Price Lunch

	PERCENT OF CLASSES			
	LOWEST QUARTILE	SECOND QUARTILE	THIRD QUARTILE	HIGHEST QUARTILE
Experience Teaching Computer Science				
0–2 years	28 (5.0)	31 (8.3)	23 (8.2)	56 (9.8)
3–5 years	30 (5.3)	29 (7.1)	36 (12.1)	12 (6.7)
6–10 years	16 (3.6)	17 (5.9)	8 (3.5)	21 (5.3)
11–20 years	24 (4.9)	22 (6.5)	33 (11.4)	3 (2.8)
≥ 21 years	2 (1.4)	2 (1.9)	1 (0.7)	8 (4.9)

Table 3 shows the percentage of high school computer science classes taught by teachers from race/ethnicity groups historically underrepresented in STEM by the proportion of students in the class from these groups. Classes with the greatest proportion of students from these groups (i.e., the highest quartile) are more likely than classes with the smallest proportion of students from these groups (i.e., the lowest quartile) to be taught by teachers from these historically underrepresented groups.

Table 3

Equity Analysis of High School Computer Science Classes Taught by Teachers From Race/Ethnicity Groups Historically Underrepresented in STEM

	PERCENT OF CLASSES
Percent of Historically Underrepresented Students in Class	
Lowest Quartile	5 (3.0)
Second Quartile	7 (3.6)
Third Quartile	3 (2.3)
Highest Quartile	47 (11.1)

Teachers were also asked about their path to certification. As can be seen in Table 4, 38 percent of high school computer science teachers have earned a teaching credential through an undergraduate program leading to a bachelor's degree, and 24 percent through a post-baccalaureate credentialing program that did not include a master's degree. Interestingly, 16 percent of computer science teachers have not earned a teaching credential, a notably larger proportion than the 7 percent of uncertified high school science and mathematics teachers. Data from the NSSME+ do not explain this difference, but it may reflect a greater tendency among schools to offer computer science classes taught by adjunct or part-time teachers with no credential, or college-level faculty providing dual credit opportunities.

Table 4
High School Computer Science Teachers' Paths to Certification

	PERCENT OF TEACHERS
An undergraduate program leading to a bachelor's degree and a teaching credential	38 (3.7)
A post-baccalaureate credentialing program (no master's degree awarded)	24 (3.2)
A master's program that also led to a teaching credential	22 (2.8)
Has not earned a teaching credential	16 (2.7)

Table 5 shows the content areas high school computer science teachers are certified to teach (i.e., have a credential, endorsement, or license in that area). Only 44 percent have a computer science certification, perhaps because many states have only recently begun creating this certification. These data, along with the teaching experience data, also indicate that many computer science teachers started as teachers of others subjects, as 34 percent are certified to teach mathematics and 28 percent are certified to teach business.

Table 5
High School Computer Science Teachers' Areas of Certification

	PERCENT OF TEACHERS
Certified in One or More Areas	**84 (2.7)**
Computer Science	44 (3.6)
Mathematics	34 (3.4)
Business	28 (2.4)
Engineering	10 (2.4)
Science	9 (2.3)
Not Certified	**16 (2.7)**

College Degrees and Coursework

In order to help students learn, teachers must themselves have a firm grasp of important ideas in the discipline they are teaching. Because direct measures of teachers' content knowledge were not feasible in this study, the survey used a number of proxy measures, including teachers' major areas of study and courses completed.

As can be seen in Table 6, only 1 in 4 computer science teachers have a college or graduate degree in computer engineering, computer science, or information science. Very few have a degree in computer science education, though almost half have an education degree.

Table 6
High School Computer Science Teacher Degrees

	PERCENT OF TEACHERS
Computer Engineering, Computer Science, Information Science, or Computer Science Education	**25 (3.2)**
Computer Engineering, Computer Science, or Information Science	24 (3.3)
Computer Science Education	4 (2.1)
Non-Computer-Science-Related Field	**89 (2.7)**
Education (general or subject specific, other than computer science education)	46 (3.9)
Mathematics	27 (3.2)
Business	23 (2.8)
Natural sciences (e.g., Biology, Chemistry, Physics, Earth Sciences)	10 (2.5)
Electrical engineering	5 (2.0)
Other engineering	5 (1.9)
Other subject	26 (3.7)

Table 7 shows the percentage of high school computer science teachers with coursework in each of a number of areas. A large majority of computer science teachers have taken an introduction to programming or an introduction to computer science course. Substantially fewer have taken other, more specific, courses related to computer science such as algorithms, computer networks, or artificial intelligence. However, a large majority of computer science teachers also have taken mathematics coursework in topics often used in computer science, either in statistics or linear algebra.

Table 7
High School Computer Science Teachers Completing Various College Courses

	PERCENT OF TEACHERS
Computer Science/Engineering	
Introduction to computer science/programming	84 (2.5)
Algorithms (e.g., sorting; search trees, heaps, and hashing; divide-and-conquer)	50 (3.8)
Operating systems/computer systems	45 (3.5)
Database systems (e.g., the relational model, relational algebra, SQL)	38 (3.7)
Software design/engineering	35 (3.1)
Computer networks (e.g., application layer protocols, Internet protocols, network interfaces)	32 (3.7)
Computer graphics (e.g., ray tracing, the graphics pipeline, transformations, texture mapping)	22 (3.6)
Computer engineering	19 (2.9)
Electrical/electronics engineering	19 (3.3)
Human-computer interaction (e.g., human information processing subsystems; libraries of standard graphical user interface objects; methodologies to measure the usability of software)	17 (3.2)
Artificial intelligence (e.g., machine learning, robotics, computer vision)	14 (2.7)
Other upper division computer science	39 (3.9)
Other types of engineering courses	23 (3.6)
Mathematics	
Statistics	84 (2.7)
Linear algebra	72 (3.0)
Probability	59 (3.3)
Discrete mathematics (e.g., combinatorics, graph theory, game theory)	44 (4.1)
Number theory (e.g., divisibility theorems, properties of prime numbers)	44 (3.6)

The Computer Science Teachers Association (CSTA) has published recommendations for computer science teacher certification,[5] and the International Society for Technology in Education (ISTE) has published standards for computer science educators.[6] Although there is not perfect agreement between these lists from CSTA and ISTE, they are reasonably consistent. Taken together, they suggest computer science teachers have coursework in the following four content areas: programming, algorithms, data structures, and some element of computer systems or networks. As can be seen in Table 8, only 1 in 4 computer science teachers have taken courses in all four recommended areas. Nearly half of computer science teachers have completed coursework in at least 3 of the 4 recommended areas.

Table 8
High School Computer Science Teachers' Coursework Related to CSTA/ISTE Preparation Standards

	PERCENT OF TEACHERS
Courses in algorithms, computer systems/networks, data structures, and programming	25 (3.3)
Courses in 3 of the 4 areas	21 (3.2)
Courses in 2 of the 4 areas	20 (2.7)
Course in 1 of the 4 areas	21 (2.6)
Courses in 0 of the 4 areas	13 (2.1)

Perceptions of Preparedness

The survey also asked teachers three series of items focused on their preparedness for computer science instruction. The first asked how well prepared they feel to teach different topics in computer science. The second and third addressed their pedagogical preparedness.

As can be seen in Table 9, about a half of high school computer science teachers consider themselves very well prepared to teach about algorithms and programming. About one-third view themselves as very well prepared to teach about impacts of computing and computing systems, and only about one-fourth to teach about data and analysis and networks and the Internet.

Table 9
High School Computer Science Teachers' Perceptions of Their Preparedness to Teach Each of a Number of Topics

	PERCENT OF TEACHERS			
	NOT ADEQUATELY PREPARED	SOMEWHAT PREPARED	FAIRLY WELL PREPARED	VERY WELL PREPARED
Algorithms and programming	5 (1.6)	14 (2.0)	34 (4.1)	47 (4.0)
Impacts of computing	6 (1.7)	19 (2.5)	40 (3.7)	35 (3.4)
Computing systems	7 (1.4)	28 (3.2)	35 (3.5)	31 (3.9)
Data and analysis	9 (1.9)	24 (2.5)	39 (3.6)	27 (4.1)
Networks and the Internet	11 (2.1)	35 (4.1)	31 (3.6)	23 (3.4)

[5] Ericson, B., Armoni, M., Gal-Ezer, J., Seehorn, D., Stephenson, C., & Trees, F. (2008). *Ensuring exemplary teaching in an essential discipline. Addressing the crisis in computer science teacher certification.* New York: Association for Computing Machinery.

[6] International Society for Technology in Education. (2011). *Standards for computer science educators.* Retrieved from https://www.iste.org/standards.

The survey also asked teachers two series of items focused on their preparedness for a number of tasks associated with instruction. First, teachers were asked how well prepared they feel to use various student-centered pedagogies, including developing students' understanding and abilities, encouraging participation of students, and differentiating their instruction to meet learners' needs. Second, teachers were asked about how well prepared they feel to carry out a number of tasks related to teaching in a specific computer science unit, including monitoring and addressing student understanding.

Roughly half of high school computer science teachers feel very well prepared to encourage students' interest in computer science, develop students' ability to do computer science, and encourage participation of all students in computer science (see Table 10). Fewer than one-quarter feel very well prepared to differentiate computer science instruction to meet the needs of diverse learners or to incorporate students' cultural backgrounds into computer science instruction.

Table 10
High School Computer Science Teachers Considering Themselves Very Well Prepared for Each of a Number of Tasks

	PERCENT OF TEACHERS
Encourage students' interest in computer science	49 (3.6)
Develop students' abilities to do computer science (e.g., breaking problems into smaller parts, considering the needs of a user, creating computational artifacts)	48 (3.7)
Encourage participation of all students in computer science	45 (3.8)
Develop students' conceptual understanding	42 (3.6)
Develop students' awareness of STEM careers	36 (4.2)
Use formative assessment to monitor student learning	35 (3.4)
Provide computer science instruction that is based on students' ideas	28 (3.9)
Differentiate computer science instruction to meet the needs of diverse learners	21 (3.3)
Incorporate students' cultural backgrounds into computer science instruction	16 (3.1)

As can be seen in Table 11, computer science teachers tend to feel less well prepared to find out what students think or already know about key ideas and anticipate difficulties they may have than they do to monitor understanding during or assess understanding at the end of a unit.

Table 11
High School Computer Science Classes in Which Teachers Feel Very Well Prepared for Each of a Number of Tasks in the Most Recent Unit in a Designated Class

	PERCENT OF CLASSES
Monitor student understanding during this unit	43 (4.6)
Assess student understanding at the conclusion of this unit	41 (4.0)
Implement the instructional materials to be used during this unit	41 (4.2)
Find out what students thought or already knew about the key computer science ideas	29 (4.6)
Anticipate difficulties that students may have with particular computer science ideas and procedures in this unit	26 (3.9)

Each set of items addressing teachers' perceptions of preparedness was combined into a composite variable measuring high school computer science teachers' perceptions of content preparedness, pedagogical preparedness, and preparedness to implement instruction in a

particular unit (see Table 12). Regarding content preparedness, high school computer science teachers perceive themselves to be far less prepared to teach their respective content than high school science teachers and mathematics teachers (mean scores of 64 vs. 88 and 82, respectively). Similarly, high school computer science teachers view themselves as less prepared for topic-specific pedagogies (when asked about the most recent unit) than high school science and mathematics teachers (mean scores of 71 vs. 80 and 83, respectively). However, high school computer science teachers' perception of their pedagogical preparedness is quite similar to that of high school science teachers, and slightly lower than mathematics teachers' perceptions. Among computer science teachers, there were no statistically significant differences when the three preparedness composites were examined by equity factors.

Table 12
Mean Scores for High School Computer Science Teachers' Perceptions of Preparedness Composites

	MEAN SCORE
Content Preparedness	64 (1.5)
Pedagogical Preparedness	68 (1.7)
Preparedness to Implement Instruction in Particular Unit	71 (1.6)

Pedagogical Beliefs

Teachers were asked about their beliefs regarding effective teaching and learning. Table 13 shows the percentage of high school computer science teachers agreeing with each of the statements. At least 90 percent agree that students should learn computer science by doing computer science, students learn best when instruction is connected to their everyday lives, teachers should ask students to justify their solutions, and most class periods should provide opportunities for students to share their thinking and reasoning.

However, many computer science teachers also have views inconsistent with what is known about effective instruction. For example, 71 percent agree that hands-on/manipulatives/ programming activities should be used primarily to reinforce a computer science idea that the students have already learned. And despite recommendations that students develop understanding of concepts first and learn terminology later, 3 out of 4 high school computer science teachers agree that students should be provided with definitions for new vocabulary at the beginning of instruction on a computer science idea.

Table 13
High School Computer Science Teachers Agreeing† With Various Statements About Teaching and Learning

	PERCENT OF TEACHERS
Reform-Oriented Beliefs	
Students should learn computer science by doing computer science (e.g., breaking problems into smaller parts, considering the needs of a user, creating computational artifacts).	97 (1.2)
Teachers should ask students to justify their solutions to a computational problem.	92 (1.6)
Most class periods should provide opportunities for students to share their thinking and reasoning.	91 (2.5)
Students learn best when instruction is connected to their everyday lives.	90 (2.0)
Most class periods should provide opportunities for students to apply computer science ideas to real-world contexts.	79 (3.1)
It is better for computer science instruction to focus on ideas in depth, even if that means covering fewer topics.	58 (3.9)
Traditional Beliefs	
At the beginning of instruction on a computer science idea, students should be provided with definitions for new vocabulary that will be used.	75 (2.7)
Hands-on/manipulatives/programming activities should be used primarily to reinforce a computer science idea that the students have already learned.	71 (3.5)
Students learn computer science best in classes with students of similar abilities.	51 (3.3)

† Includes high school computer science teachers indicating "strongly agree" or "agree" on a five-point scale ranging from 1 "strongly disagree" to 5 "strongly agree."

These items were combined into two composite variables: Traditional Teaching Beliefs and Reform-Oriented Teaching Beliefs. As can be seen in Table 14, although computer science teachers hold relatively strong traditional beliefs about instruction, they agreed more strongly with the reform-oriented statements.

Table 14
Mean Scores for High School Computer Science Teachers' Beliefs About Teaching and Learning Composites

	MEAN SCORE
Reform-Oriented Beliefs	82 (0.9)
Traditional Beliefs	67 (1.4)

Looking at these composites by the equity factors, there is no statistically significant relationship between computer science teachers' beliefs and the proportion of students in the class from race/ethnicity groups historically underrepresented in STEM (see Table 15). Classes in schools with higher proportions of students eligible for free/reduced-price lunch are slightly more likely than those in lower-poverty schools to be taught by teachers with stronger reform-oriented beliefs.

Table 15
Equity Analyses of Class Mean Scores for High School Computer Science Teachers' Beliefs About Teaching and Learning Composites

	MEAN SCORE	
	TRADITIONAL BELIEFS	REFORM-ORIENTED BELIEFS
Percent of Historically Underrepresented Students in Class		
Lowest Quartile	65 (2.1)	80 (1.7)
Second Quartile	72 (4.1)	82 (2.5)
Third Quartile	61 (1.8)	85 (1.8)
Highest Quartile	66 (4.5)	84 (1.8)
Percent of Students in School Eligible for FRL		
Lowest Quartile	65 (1.7)	80 (1.4)
Second Quartile	67 (3.5)	82 (1.6)
Third Quartile	69 (5.2)	86 (2.4)
Highest Quartile	61 (2.8)	85 (2.3)

Leadership Roles and Responsibilities

In addition to asking teachers about their educational background, beliefs, and preparedness, the survey asked teachers whether they have served in various leadership roles in the profession in the last three years. As can be seen in Table 16, over a third of high school computer science teachers have (1) served on a school computer science committee, (2) been a lead teacher or department chair, and (3) taught a computer science lesson for other teachers to observe. In high schools that offer computer science, many have only one computer science teacher. Thus, it is not surprising that few computer science teachers have observed another teacher's instruction to provide feedback or served as a mentor or coach for another computer science teacher.

Table 16
High School Computer Science Teachers Having Various Leadership Responsibilities Within the Last Three Years

	PERCENT OF TEACHERS
Served on a school or district/diocese-wide computer science committee	39 (4.0)
Served as a lead teacher or department chair	36 (3.6)
Taught a computer science lesson for other teachers to observe	36 (3.7)
Led or co-led a workshop or professional learning community for other teachers focused on computer science or computer science teaching	22 (3.1)
Observed another teacher's computer science lesson for the purpose of giving them feedback	17 (2.7)
Supervised a student teacher in their classroom	15 (2.6)
Served as a formal mentor or coach for a computer science teacher	10 (2.2)

Professional Development of High School Computer Science Teachers

Computer science teachers, like all professionals, need opportunities to keep up with advances in their field, including both disciplinary content and how to help their students learn important computer science content. The 2018 NSSME+ collected data on teachers' participation in

professional development, as well as characteristics of the professional development. These data are discussed in this section.

One important measure of teachers' continuing education is how long it has been since they participated in professional development. Roughly 80 percent of high school computer science teachers have participated in discipline-focused professional development (i.e., focused on computer science content or the teaching of computer science) within the last three years (see Table 17).

Table 17
High School Computer Science Teachers' Most Recent Participation in Computer Science-Focused Professional Development

	PERCENT OF TEACHERS
In the last 12 months	64 (3.8)
1–3 years ago	18 (2.7)
4–6 years ago	4 (1.2)
7–10 years ago	2 (1.4)
More than 10 years ago	1 (0.6)
Never	11 (2.7)

Although some involvement in professional development may be better than none, a brief exposure of a few hours over several years is not likely to be sufficient to enhance teachers' knowledge and skills in meaningful ways. Accordingly, teachers were asked about the total amount of time they have spent on discipline-focused professional development in the last three years; results are shown in Table 18. Over half of high school computer science teachers have participated in 36 hours or more of professional development related to computer science or computer science teaching.

Table 18
Time Spent by High School Computer Science Teachers on Computer Science-Focused Professional Development in the Last Three Years

	PERCENT OF TEACHERS
None	18 (2.9)
Less than 6 hours	3 (1.1)
6–15 hours	8 (2.0)
16–35 hours	17 (2.3)
36–80 hours	24 (3.2)
More than 80 hours	30 (3.0)

Teachers who had recently participated in professional development were asked about the nature of those activities (see Table 19). Workshops are the most prevalent activity, with 88 percent of high school computer science teachers who have had professional development indicating they have attended a program/workshop related to their discipline. Participation in professional learning communities and completing an online course/webinar are the next most prevalent activities (62 and 59 percent of teachers, respectively).

Table 19
High School Computer Science Teachers Participating in Various Computer Science-Focused Professional Development Activities in Last Three Years

	PERCENT OF TEACHERS†
Attended a professional development program/workshop	88 (2.4)
Participated in a professional learning community/lesson study/teacher study group	62 (3.8)
Completed an online course/webinar	59 (4.7)
Attended a national, state, or regional computer science teacher association meeting	35 (3.9)
Received assistance or feedback from a formally designated coach/mentor	29 (3.7)
Took a formal course for college credit	20 (3.1)

† Only high school computer science teachers indicating that they participated in computer science-focused professional development in the last three years are included in these analyses.

It is widely agreed that teachers need opportunities to work with colleagues who face similar challenges, including other teachers from their school and those who have similar teaching assignments. Other recommendations include providing opportunities for teachers to engage in investigations, both to learn disciplinary content and to experience inquiry-oriented learning; examine student work and other classroom artifacts for evidence of what students do and do not understand; and apply what they have learned in their classrooms and subsequently discuss how it went.[7] Accordingly, teachers who had participated in professional development in the last three years were asked a series of additional questions about the nature of those experiences.

As shown in Table 20, about three-fourths of teachers who participated in professional development have had opportunities to engage in activities to learn computer science in the last three years. Another common characteristic is experiencing lessons as students would from the textbooks/units used in the classroom (62 percent). Further, about half of computer science teachers attending professional development have had substantial opportunities to work closely with other computer science teachers who taught the same grade and/or subject, whether or not they were from their school, and to examine classroom artifacts. High school computer science teachers rarely have had substantial opportunities to rehearse instructional practices during professional development.

[7] Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. *Educational Researcher, 38*(3), 181–199.

Elmore, R. F. (2002). *Bridging the gap between standards and achievement: The imperative for professional development in education.* Washington, DC: Albert Shanker Institute.

Garet, M. S., Porter, A. C., Desimone, L., Birman, B. F., and Yoon, K. S. (2001). What makes professional development effective? Results from a national sample of teachers. *American Educational Research Journal, 38*(4), 915–945.

Table 20
High School Computer Science Teachers Whose Professional Development in the Last Three Years Had Each of a Number of Characteristics to a Substantial Extent†

	PERCENT OF TEACHERS‡
Had opportunities to engage in activities to learn computer science content	76 (3.6)
Had opportunities to experience lessons, as their students would, from the textbook/units they use in their classroom	62 (3.7)
Worked closely with other teachers who taught the same grade and/or subject whether or not they were from their school	51 (4.0)
Had opportunities to examine classroom artifacts (e.g., student work samples, e-portfolios, videos of classroom instruction)	46 (3.9)
Had opportunities to apply what they learned to their classroom and then come back and talk about it as part of the professional development	39 (3.5)
Had opportunities to rehearse instructional practices during the professional development (i.e., try out, receive feedback, and reflect on those practices)	31 (3.8)
Worked closely with other teachers from their school	26 (3.9)

† Includes high school computer science teachers indicating 4 or 5 on a five-point scale ranging from 1 "not at all" to 5 "to a great extent."

‡ Only high school computer science teachers indicating that they participated in computer science-focused professional development in the last three years are included in these analyses.

Another series of items asked about the focus of professional development opportunities teachers have had in the last three years. As can be seen in Table 21, the most common emphases related to understanding and doing computer science: deepening their computer science content knowledge, including programming (70 percent); learning how to use programming activities that require a computer (64 percent); and deepening understanding of how computer science is done (63 percent). Half of computer science teachers' professional development has had a substantial focus on implementing the computer science textbook/online course to be used in their classroom. Only about a quarter have attended professional development that emphasized differentiating computer science instruction to meet the needs of diverse learners or incorporating students' cultural backgrounds into computer science instruction, two areas that likely will need greater emphasis to help ensure students from all backgrounds have opportunities in this field.

Table 21
High School Computer Science Teachers Reporting That Their Professional Development in the Last Three Years Gave Heavy Emphasis† to Various Areas

	PERCENT OF TEACHERS‡
Deepening their own computer science content knowledge, including programming	70 (3.6)
Learning how to use programming activities that require a computer	64 (4.1)
Deepening their understanding of how computer science is done (e.g., breaking problems into smaller parts, considering the needs of a user, creating computational artifacts)	63 (3.6)
Implementing the computer science textbook/online course to be used in their classroom	50 (4.0)
Learning about difficulties that students may have with particular computer science ideas and/or practices	48 (4.2)
Monitoring student understanding during computer science instruction	40 (3.6)
Learning how to provide computer science instruction that integrates engineering, mathematics, and/or science	36 (3.7)
Differentiating computer science instruction to meet the needs of diverse learners	29 (3.4)
Incorporating students' cultural backgrounds into computer science instruction	25 (3.4)

† Includes high school computer science teachers indicating 4 or 5 on a five-point scale ranging from 1 "not at all" to 5 "to a great extent."

‡ Only high school computer science teachers indicating that they participated in computer science-focused professional development in the last three years are included in these analyses.

Responses to the seven items describing the characteristics of professional development experiences were combined into a single composite variable called Extent Professional Development Aligns with Elements of Effective Professional Development. Several items related to a focus on student-centered instruction in recent teacher professional development were combined into a composite variable called Extent Professional Development Supports Student-Centered Instruction. Taken together, the mean scores on these composite items suggest there is considerable room for improvement in high school computer science teacher professional development (see Table 22).

Table 22
High School Computer Science Teacher
Mean Scores for Professional Development Composites

	MEAN SCORE
Extent Professional Development Aligns With Elements of Effective Professional Development	56 (1.6)
Extent Professional Development Supports Student-Centered Instruction	58 (1.8)

When looking at scores on these composites by equity factors, only one difference is significant (see Table 23). High school computer science classes with the largest proportion of students from race/ethnicity groups historically underrepresented in STEM are more likely to be taught by teachers who have experienced aspects of effective professional development than classes with the smallest proportion of students from these groups (mean scores of 64 and 51, respectively). However, it is important to note that for computer science, even the highest quartile contains relatively few students from these groups.

Table 23
Equity Analyses of High School Computer Science
Class Mean Scores for Professional Development Composites

	MEAN SCORE	
	EXTENT PROFESSIONAL DEVELOPMENT ALIGNS WITH ELEMENTS OF EFFECTIVE PROFESSIONAL DEVELOPMENT	EXTENT PROFESSIONAL DEVELOPMENT SUPPORTS STUDENT-CENTERED INSTRUCTION
Percent of Historically Underrepresented Students in Class		
Lowest Quartile	51 (3.2)	54 (3.5)
Second Quartile	59 (3.8)	62 (5.5)
Third Quartile	56 (2.6)	60 (3.4)
Highest Quartile	64 (3.3)	61 (4.2)

High School Computer Science Courses Offered

The 2018 NSSME+ collected data on computer science course offerings in the nation's schools. Teachers provided information about titles of secondary computer science courses; class sizes; gender and racial/ethnic composition; and prior achievement levels. These data are presented in this section.

About half of high schools offer one or more computer science courses, meaning courses they include programming or require programming as a prerequisite (see Table 24). The data also show that although a majority of high school students in the nation have access to a computer science course, nearly a third do not.

Table 24
Access to Computer Science Instruction, by Schools and Students

	PERCENT OF SCHOOLS OFFERING	PERCENT OF STUDENTS WITH ACCESS
Any Level Computer Science	53 (2.9)	70 (1.9)

Table 25 shows the percentage of high schools that offer computer science instruction by equity factors. Larger schools are more likely to offer computer science than smaller schools, and rural schools are less likely to offer it than suburban or urban schools. There are also regional differences, with schools in the Northeast more likely to offer computer science than schools in the South and Midwest. However, the apparent difference between high-poverty schools and low-poverty schools is not statistically significant.

Table 25
Equity Analyses of High Schools Offering Computer Science Instruction

	PERCENT OF SCHOOLS
Percent of Students in School Eligible for FRL	
Lowest Quartile	52 (4.9)
Second Quartile	54 (5.5)
Third Quartile	36 (4.8)
Highest Quartile	39 (5.8)
School Size	
Smallest Schools	20 (5.1)
Second Group	38 (6.0)
Third Group	48 (4.4)
Largest Schools	73 (3.9)
Community Type	
Rural	26 (4.4)
Suburban	53 (3.8)
Urban	61 (6.0)
Region	
Midwest	41 (5.1)
Northeast	65 (6.3)
South	40 (4.1)
West	47 (7.6)

The percentages of high schools offering different types of computer science and computer technology courses are shown in Table 26. Almost half of schools offer computer technology courses that do not include programming. Introductory high school computer science courses and computer science courses that might qualify for college credit are each offered at about a third of high schools. Specialized computer science courses that require programming are offered at only about 1 in 5 high schools.

Table 26
High Schools Offering Various Computer Science and Technology Courses

	PERCENT OF SCHOOLS
Computer technology courses that do not include programming (e.g., Computer Literacy, Keyboarding, Computer Applications, Web Design)	47 (2.4)
Introductory high school computer science courses that include programming but do not qualify for college credit (e.g., Computer Science Discoveries, Computer Science Essentials)	36 (2.4)
Specialized/elective computer science courses with programming as a prerequisite that do not qualify for college credit (e.g., game or mobile app development, robotics)	21 (1.7)
Courses that might qualify for college credit (e.g., AP Computer Science A)	35 (2.1)

Almost four-fifths of high schools do not offer any AP computer science course (see Table 27). AP Computer Science A and AP Computer Science Principles are each offered in about 1 in 6 high schools, and about 1 in 10 offer both courses. As for computer science courses overall, the percentage of grades 9–12 students with access to each course is substantially greater than the percentage of schools offering it.

Table 27
Access to AP Computer Science Courses, by Schools and Students

	PERCENT OF HIGH SCHOOLS OFFERING	PERCENT OF HIGH SCHOOL STUDENTS WITH ACCESS
No AP Computer Science Courses Offered	79 (1.6)	59 (2.2)
AP Computer Science A	16 (1.4)	34 (2.3)
AP Computer Science Principles	14 (1.5)	28 (2.2)
Both AP Computer Science Courses	9 (1.1)	21 (2.2)

Table 28 shows the percentage of schools that offer AP computer science course by the equity factors. Not surprisingly, large schools are more likely to offer AP computer science courses than small schools. Rural schools are less likely than suburban or urban schools, and high-poverty schools are less likely than low-poverty schools, to offer AP computer science. There are also regional differences, with schools in the Northeast more likely to offer computer science than schools in the West and Midwest. Schools in the South are also more likely to offer computer science than schools in the Midwest, but other regional differences are not statistically significant.

Table 28
Equity Analyses of High Schools Offering at Least One AP Computer Science Course

	PERCENT OF SCHOOLS
Percent of Students in School Eligible for FRL	
Lowest Quartile	33 (4.6)
Second Quartile	22 (3.4)
Third Quartile	15 (3.3)
Highest Quartile	15 (3.8)
School Size	
Smallest Schools	7 (3.7)
Second Group	14 (3.9)
Third Group	21 (3.1)
Largest Schools	41 (3.5)
Community Type	
Rural	8 (2.2)
Suburban	28 (2.8)
Urban	30 (4.8)
Region	
Midwest	12 (2.7)
Northeast	35 (5.4)
South	24 (3.4)
West	18 (2.8)

In addition to gathering school-level information about course offerings, the survey asked each teacher for the course type of a randomly selected class, which allows for an estimate of the percentage of courses of each type in schools. As can be seen in Table 29, introductory courses account for almost half of all computer science courses that include programming or have programming as a prerequisite. Just over a third of classes might qualify for college credit; only 16 percent of classes are specialized or elective computer science courses.

Table 29
Most Commonly Offered High School Computer Science Courses

	PERCENT OF CLASSES
Introductory high school computer science courses that include programming (e.g., Computer Science Discoveries, Computer Science Essentials)	48 (4.0)
Specialized/elective computer science courses with programming as a prerequisite (e.g., Robotics, Game or Mobile App Development)	16 (2.8)
Courses that might qualify for college credit (e.g., AP Computer Science A)	36 (4.2)

The typical high school computer science class has approximately 17 students; two-thirds of classes have between 8 and 26 students. Overall, about half of all K–12 students in the nation are female, and about half are from race/ethnicity groups historically underrepresented in STEM. However, both groups make up fewer than one-third of students in high school computer science courses (see Table 30).

Table 30
Demographics of Students in High School Computer Science Courses

	PERCENT OF STUDENTS	
	FEMALE	HISTORICALLY UNDERREPRESENTED
Introductory high school computer science courses that include programming	30 (3.7)	30 (3.3)
Specialized/elective computer science courses with programming as a prerequisite	27 (5.7)	30 (9.2)
Computer science courses that might qualify for college credit	25 (2.5)	23 (5.8)

Teachers were also asked to indicate the prior achievement level of students in the randomly selected class relative to other students in the school. As can be seen in Table 31, almost no computer science classes are composed of mostly low prior-achieving students. Roughly a third to half of computer science courses are heterogeneous in terms of prior achievement level. Twenty-four percent of introductory computer science classes, 41 percent of specialized/elective classes, and 49 percent of classes that might qualify for college credit consist of mostly high prior-achieving students.

Table 31
Prior Achievement Grouping in High School Computer Science Classes

	PERCENT OF CLASSES			
	MOSTLY LOW ACHIEVERS	MOSTLY AVERAGE ACHIEVERS	MOSTLY HIGH ACHIEVERS	A MIXTURE OF LEVELS
Introductory high school computer science courses that include programming	1 (0.8)	30 (5.1)	24 (5.6)	45 (5.8)
Specialized/elective computer science courses with programming as a prerequisite	0 ---†	13 (4.8)	41 (9.8)	46 (10.6)
Computer science courses that might qualify for college credit	0 ---†	17 (5.7)	49 (7.1)	34 (6.4)

† No high school computer science teachers in the sample selected this response option. Thus, it is not possible to calculate the standard error of this estimate.

High School Computer Science Instruction

The 2018 NSSME+ collected data about teachers' perceptions of their autonomy in making curricular and instructional decisions. Questions also focused on teachers' instructional objectives, class activities they use in accomplishing these objectives, and how student performance is assessed in a particular, randomly selected class. These data are discussed in this section.

Teachers' Perceptions of Their Decision-Making Autonomy

Many in education believe that classroom teachers are in the best position to know their students' needs and interests and, therefore, should be the ones making decisions about tailoring instruction to a particular group of students. Teachers were asked the extent to which they had control over a number of curricular and instructional decisions for their classes.

As can be seen in Table 32, high school computer science teachers tend to perceive themselves as having strong control over pedagogical decisions, such as determining the amount of homework to be assigned (77 percent of classes) and choosing criteria for grading student

performance (71 percent). In contrast, teachers in fewer than half of classes perceive themselves as having strong control over selecting programming languages to use.

Table 32
High School Computer Science Classes in Which Teachers Report Having Strong Control Over Various Curricular and Instructional Decisions

	PERCENT OF CLASSES
Determining the amount of homework to be assigned	77 (3.6)
Choosing criteria for grading student performance	71 (4.1)
Selecting teaching techniques	68 (4.5)
Determining the amount of instructional time to spend on each topic	63 (4.4)
Selecting the sequence in which topics are covered	63 (4.2)
Selecting curriculum materials (e.g., textbooks/online courses)	58 (4.7)
Determining course goals and objectives	57 (4.3)
Selecting content, topics, and skills to be taught	53 (4.2)
Selecting programming languages to use	49 (4.3)

These items were combined into two composite variables—Curriculum Control and Pedagogy Control. Curriculum Control consists of the following items:

- Determining course goals and objectives;
- Selecting curriculum materials;
- Selecting content, topics, and skills to be taught;
- Selecting the sequence in which topics are covered; and
- Selecting programming languages to use.

For Pedagogy Control, the items are:

- Selecting teaching techniques;
- Determining the amount of homework to be assigned; and
- Choosing criteria for grading student performance.

The mean composite scores displayed in Table 33 indicate that teachers perceive more control of decisions related to pedagogy than curriculum, though scores on both composites are high. In addition, there are no significant differences by school or class equity factors.

Table 33
High School Computer Science Class Mean Scores for Curriculum Control and Pedagogy Control Composites

	MEAN SCORE
Curriculum Control	78 (1.7)
Pedagogy Control	89 (1.4)

Instructional Objectives

The survey provided a list of possible objectives of instruction and asked teachers how much emphasis each would receive in an entire course of a particular, randomly selected class. Table 34 shows the percentage of high school computer science classes with a heavy emphasis on each objective. Learning how to do computer science, understanding computer science concepts,

developing students' confidence that they can successfully pursue computer science careers, and increasing student interest receive a heavy emphasis in a majority of classes. Learning vocabulary and/or the syntax of a particular programming language receives a heavy emphasis in only a third of classes.

Table 34
High School Computer Science Classes With Heavy Emphasis on Various Instructional Objectives

	PERCENT OF CLASSES
Learning how to do computer science (e.g., breaking problems into smaller parts, considering the needs of a user, creating computational artifacts)	60 (3.5)
Understanding computer science concepts	55 (3.6)
Developing students' confidence that they can successfully pursue careers in computer science	52 (3.9)
Increasing students' interest in computer science	50 (3.6)
Learning how to develop computational solutions	43 (4.1)
Learning about real-life applications of computer science	39 (4.3)
Learning computer science vocabulary and/or program syntax	33 (3.9)

The objectives related to reform-oriented instruction (understanding computer science concepts, learning how to develop computational solutions, learning how to do computer science, learning about real-life applications of computer science, increasing students' interest in computer science, and developing students' confidence that they can successfully pursue careers in computer science) were combined into a composite variable. Overall, scores on this composite are fairly high, indicating that high school computer science classes are likely to emphasize reform-oriented instructional objectives (see Table 35). Interestingly, classes with a higher proportion of students from race/ethnicity groups historically underrepresented in STEM fields are more likely than classes with a lower proportion of these students to emphasize reform-oriented objectives. Similarly, classes in schools with a higher proportion of students eligible for free/reduced-price lunch are more likely than classes in schools with fewer of these students to emphasize reform-oriented objectives.

Table 35
High School Computer Science Class Mean Scores for the Reform-Oriented Instructional Objectives Composite

	MEAN SCORE
Overall	**81 (1.0)**
Percent of Historically Underrepresented Students in Class	
Lowest Quartile	75 (1.9)
Second Quartile	80 (2.1)
Third Quartile	81 (1.7)
Highest Quartile	86 (2.2)
Percent of Students in School Eligible for FRL	
Lowest Quartile	78 (1.4)
Second Quartile	80 (1.8)
Third Quartile	82 (2.7)
Highest Quartile	85 (2.9)

Class Activities

Teachers were asked several items about their instruction in the randomly selected class. One item asked how often they use different pedagogies (e.g., explaining ideas to students, small group work). Another asked how often they engage students in practices associated with the discipline of computer science. Response options for both of these sets of items were: never, rarely (e.g., a few times a year), sometimes (e.g., once or twice a month), often (e.g., once or twice a week), and all or almost all computer science lessons.

Table 36 shows the percentage of high school computer science classes in which teachers use various activities at least once a week. Having students work on programming activities using a computer (97 percent) and the teacher explaining ideas to the class (84 percent) are by far the most common modes of instruction in high school computer science classes. Whole class discussions, students working in small groups, and students explaining and justifying their method for solving a problem occur at least weekly in about two-thirds of high school computer science classes.

Table 36
High School Computer Science Classes in Which Teachers Report Using Various Activities at Least Once a Week

	PERCENT OF CLASSES
Have students work on programming activities using a computer	97 (1.4)
Explain computer science ideas to the whole class	84 (2.9)
Engage the whole class in discussions	71 (3.3)
Have students work in small groups	66 (3.6)
Have students explain and justify their method for solving a problem	63 (3.4)
Have students compare and contrast different methods for solving a problem	41 (3.8)
Have students present their solution strategies to the rest of the class	35 (4.0)
Have students write their reflections (e.g., in their journals, on exit tickets) in class or for homework	32 (4.4)
Have students read from a textbook/online course in class, either aloud or to themselves	31 (4.1)
Use flipped instruction (have students watch lectures/demonstrations outside of class to prepare for in-class activities)	24 (3.2)
Focus on literacy skills (e.g., informational reading or writing strategies)	21 (3.3)
Have students do hands-on/manipulative programming activities that do not require a computer	21 (3.6)

Teachers were asked how often they engage students in the practices of computer science described in the CSTA's *K–12 Computer Science Standards*.[8] These practices include developing and using abstractions, recognizing and defining computational problems, testing and refining computational artifacts, communicating about computing, and fostering an inclusive computing culture. As can be seen in Table 37, activities related to creating, testing, and refining computational artifacts occur most frequently. For example, creating computational artifacts, writing comments within code, considering how to break a program into modules/procedures/objects, and adapting existing code to a new problem occur at least once a week in 60 percent or more of classes. Aspects of computer science related to end users are less often emphasized. For example, only 30 percent of classes have students create instructions for an end-user

[8] Computer Science Teachers Association (2017). *CSTA K–12 Computer Science Standards.* Retrieved from http://www.csteachers.org/standards.

explaining a computational artifact on a weekly basis. Similarly, fewer than a quarter of high school computer science classes have students create a computational artifact to be used by someone else or get input on computational products from people with different perspectives at least once a week.

Table 37
High School Computer Science Classes in Which Teachers Report Students Engaging in Various Aspects of Computer Science Practices at Least Once a Week

	PERCENT OF CLASSES
Create computational artifacts (e.g., programs, simulations, visualizations, digital animations, robotic systems, or apps)	75 (2.8)
Write comments within code to document purposes or features	72 (2.8)
Consider how a program they are creating can be separated into modules/procedures/objects	62 (3.1)
Identify and adapt existing code to solve a new computational problem	60 (3.6)
Provide feedback on other students' computational products or designs	47 (4.1)
Systematically use test cases to verify program performance and/or identify problems	46 (4.2)
Identify real-world problems that might be solved computationally	45 (4.3)
Use computational methods to simulate events or processes (e.g., rolling dice, supply and demand)	45 (3.6)
Explain computational solution strategies verbally or in writing	42 (3.6)
Create instructions for an end-user explaining how to use a computational artifact	30 (3.6)
Compare and contrast the strengths and limitations of different representations such as flow charts, tables, code, or pictures	22 (3.3)
Create a computational artifact designed to be used by someone outside the class or other students	22 (3.6)
Get input on computational products or designs from people with different perspectives	21 (3.2)
Analyze datasets using a computer to detect patterns	20 (3.3)

These items were combined into a composite variable; mean scores on this composite, overall and by equity factors, are shown in Table 38. The overall score of 56 indicates that, on average, students are engaged in this set of activities once or twice a month. Equity analyses revealed no statistically significant differences by these factors.

Table 38
High School Computer Science Class Mean Scores for Engaging Students in Practices of Computer Science Composite

	MEAN SCORE
Overall	**56 (1.3)**
Prior Achievement Level of Class	
Mostly High	55 (1.7)
Average/Mixed	56 (1.7)
Percent of Historically Underrepresented Students in Class	
Lowest Quartile	53 (2.0)
Second Quartile	54 (4.1)
Third Quartile	57 (3.0)
Highest Quartile	59 (2.9)
Percent of Students in School Eligible for FRL	
Lowest Quartile	54 (1.9)
Second Quartile	57 (2.4)
Third Quartile	54 (3.4)
Highest Quartile	60 (4.1)

High school computer science teachers were also asked which activities took place in their most recent lesson. As can be seen in Table 39, 84 percent of lessons include students working on programming tasks using a computer, and 70 percent include the teacher explaining ideas to the whole class. About half include small group work, whole class discussion, or students watching a demonstration.

Table 39
High School Computer Science Classes Participating in Various Activities in Most Recent Lesson

	PERCENT OF CLASSES
Students working on programming tasks using a computer	84 (2.8)
Teacher explaining a computer science idea to the whole class	70 (3.7)
Students working in small groups	57 (4.2)
Whole class discussion	49 (4.1)
Teacher conducting a demonstration while students watched	46 (3.6)
Students reading about computer science	20 (2.8)
Students doing hands-on/manipulative programming activities not using a computer	19 (2.9)
Students completing textbook/worksheet problems	16 (3.0)
Students writing about computer science	13 (3.0)
Test or quiz	9 (1.6)

On average, 40 percent of time in high school computer science classes is spent with students working individually (see Table 40). Whole class activities and small group work take up 29 and 22 percent of class time, respectively.

Table 40
Average Percentage of Time Spent on Different Activities in the Most Recent High School Computer Science Lesson

	AVERAGE PERCENT OF CLASS TIME
Students working individually (e.g., reading textbooks, programming, taking a test or quiz)	40 (2.1)
Whole class activities (e.g., lectures, explanations, discussions)	29 (2.3)
Small group work	22 (2.1)
Non-instructional activities (e.g., attendance taking, interruptions)	9 (0.5)

Homework and Assessment Practices

Teachers were asked about the amount of homework assigned per week in the randomly selected class. In about half of high school computer science classes, students are given 30 minutes or less of homework a week, with 16 percent having no homework (see Table 41). At the other end of the spectrum, 1 in 5 classes are assigned more than 60 minutes of homework per week.

Table 41
Amount of Homework Assigned in High School Computer Science Classes Per Week

	PERCENT OF CLASSES
None	16 (2.6)
1–15 minutes per week	13 (2.9)
16–30 minutes per week	22 (4.4)
31–60 minutes per week	29 (3.9)
61–90 minutes per week	12 (2.5)
91–120 minutes per week	4 (1.0)
More than 2 hours per week	4 (1.2)

Resources Available for High School Computer Science

The quality and availability of instructional resources are major factors affecting computer science teaching. The 2018 NSSME+ included a series of items on instructional materials—which ones teachers use and how teachers use them. Teachers were also asked about the availability and use of a number of other instructional resources, including various types of computing devices and Internet capabilities.

Instructional Materials

The survey collected data on the use of various instructional materials, including commercially published textbooks or programs. Of particular interest is how much latitude teachers have in selecting instructional resources. Table 42 shows that only about a quarter of high school computer science classes have materials designated for them by the school or district. Among these classes, free, web-based resources and commercially published textbooks are the most common types of designated materials.

Table 42
High School Computer Science Classes for Which Various Types of Instructional Materials Are Designated

	PERCENT OF CLASSES
District Designates Instructional Materials	
No	74 (3.7)
Yes	26 (3.7)
Types of Designated Instructional Materials†	
Lessons or resources from websites that are free (e.g., Khan Academy, code.org)	59 (9.8)
Commercially published textbooks (printed or electronic), including the supplementary materials (e.g., worksheets) that accompany the textbooks	54 (11.3)
Lessons or resources from websites that have a subscription fee or per lesson cost (e.g., BrainPOP, Discovery Ed, Teachers Pay Teachers)	33 (10.1)
Online units or courses that students work through at their own pace (e.g., MOOCs, EdX, IMACS)	16 (4.6)
State, county, district, or diocese-developed units or lessons	10 (3.9)

† Only high school computer science classes for which instructional materials are designated by the state, district, or diocese are included in these analyses.

Regardless of whether instructional materials had been designated for their class, teachers were asked how often instruction was based on various types of materials. In high school computer science, about two-thirds of classes are based on teacher-created lessons at least once a week (see Table 43). Lessons from free websites are a distant second, serving as the basis for instruction at least once a week in 43 percent of classes.

Table 43
High School Computer Science Classes Basing Instruction on Various Instructional Resources at Least Once a Week

	PERCENT OF CLASSES
Units or lessons you created (either by yourself or with others)	64 (3.9)
Lessons or resources from websites that are free (e.g., Khan Academy, code.org)	43 (4.0)
Online units or courses that students work through at their own pace (e.g., MOOCs, EdX, IMACS)	32 (4.6)
Units or lessons you collected from any other source (e.g., conferences, journals, colleagues, university or museum partners)	28 (3.6)
Commercially published textbooks (printed or electronic), including the supplementary materials (e.g., worksheets) that accompany the textbooks	26 (3.4)
Lessons or resources from websites that have a subscription fee or per lesson cost (e.g., BrainPOP, Discovery Ed, Teachers Pay Teachers)	9 (2.2)
State, county, district, or diocese-developed units or lessons	7 (2.8)

Only one textbook is used by more than 10 percent of high school computer science classes using a textbook: *HTML and CSS*, by Pearson. If computer science teachers reported that their class was sometimes based on lessons from free or fee-based websites, they were asked to list up to three online sources of lessons or activities they use most frequently. Only one online source—code.org—is used in more than 10 percent of these classes.

Table 44 shows the publication year of high school computer science instructional materials. Given the growing presence of computer science classes, it is surprising that a third of classes that use textbooks are using materials published in 2009 or earlier.

Table 44
Publication Year of Textbooks/Programs Used in High School Computer Science Classes†

	PERCENT OF CLASSES
2009 or earlier	33 (7.3)
2010–12	26 (5.9)
2013–15	24 (6.5)
2016–18	17 (5.1)

† Only high school computer science classes using commercially published textbooks/programs are included in these analyses.

High school computer science teachers were also asked whether the most recent unit in their randomly selected class was based primarily on a commercially published textbook, commercially published online courses, or materials developed by the state or district. About two-thirds of teachers indicated their most recently completed unit was based on one of these types of materials (see Table 45). These teachers were then asked how they used the material. Two important findings emerge from these data. First, in these classes, the materials heavily influence instruction. Teachers in 84 percent of classes use the materials to guide the structure and content emphasis of the unit. Second, it is clear that teachers modify these materials substantially when designing instruction. In 70 percent of these classes, teacher supplement with activities from other sources. Further, teachers modify activities and skip activities they did not see as important in about half of the classes using these types of materials.

Table 45
High School Computer Science Teachers' Use of Instructional Materials in Most Recent Unit

	PERCENT OF CLASSES
Most Recent Unit Based on Commercially Published or State/District-Developed Material	
No	37 (5.4)
Yes	63 (5.4)
Ways Textbook is Substantially† Used‡	
I used these materials to guide the structure and content emphasis of the unit.	84 (3.6)
I incorporated activities (e.g., problems, investigations, readings) from other sources to supplement what these materials were lacking.	70 (5.2)
I modified activities from these materials.	56 (6.4)
I picked what is important from these materials and skipped the rest.	49 (7.3)

† Includes high school computer science teachers indicating 4 or 5 on a 5-point scale ranging from 1 "not at all" to 5 "to a great extent."

‡ Only high school computer science classes in which the most recent unit was based on commercially published or state/district-developed materials are included in these analyses.

Teachers were asked why they skip parts of their materials. As can be seen in Table 46, teachers in a majority of these classes skip activities because: (1) they have another activity that works better than the one skipped, (2) they do not have enough instructional time, and (3) the activities skipped being too difficult for the students.

Table 46
Reasons Why Parts of High School Computer Science Materials Are Skipped

	PERCENT OF CLASSES†
I have different activities for those computer science ideas that work better than the ones I skipped.	68 (5.6)
I did not have enough instructional time for the activities I skipped.	60 (5.8)
The activities I skipped were too difficult for my students.	51 (7.2)
The computer science ideas addressed in the activities I skipped are not included in my pacing guide/standards.	49 (6.7)
My students already knew the computer science ideas or were able to learn them without the activities I skipped.	44 (6.2)
I did not have the knowledge needed to implement the activities I skipped.	35 (7.5)
I did not have the materials needed to implement the activities I skipped.	28 (7.0)

† Only high school computer science classes in which (1) the most recent unit was based on commercially published or state/district-developed materials and (2) teachers reported skipping some activities are included in these analyses.

As can be seen in Table 47, in classes where teachers supplement materials, the most frequent reasons are: (1) teachers having additional activities that they like, (2) providing students with additional practice, and (3) differentiating instruction for students at different achievement levels.

Table 47
Reasons Why High School Computer Science Materials Are Supplemented

	PERCENT OF CLASSES†
I had additional activities that I liked.	79 (5.7)
Supplemental activities were needed to provide students with additional practice.	79 (5.0)
Supplemental activities were needed so students at different levels of achievement could increase their understanding of the ideas targeted in each activity.	73 (5.6)
Supplemental activities were needed to prepare students for standardized tests.	52 (6.9)
My pacing guide indicated that I should use supplemental activities.	34 (6.3)

† Only high school computer science classes in which (1) the most recent unit was based on commercially published or state/district-developed materials and (2) teachers reported supplementing some activities are included in these analyses.

Finally, when teachers reported that they modified their materials, they rated each of several factors that may have contributed to their decision (see Table 48). Teachers not having enough time to implement the activities as designed (54 percent of classes) and the activities being too difficult for students (43 percent) are the most common reasons.

Table 48
Reasons Why High School Computer Science Materials Are Modified

	PERCENT OF CLASSES†
I did not have enough instructional time to implement the activities as designed.	54 (6.5)
The original activities were too difficult conceptually for my students.	43 (6.5)
The original activities were not structured enough for my students.	37 (7.3)
The original activities were too easy conceptually for my students.	33 (6.3)
I did not have the necessary materials/supplies for the original activities.	32 (7.1)
The original activities were too structured for my students.	31 (6.6)

† Only high school computer science classes in which (1) the most recent unit was based on commercially published or state/district-developed materials and (2) teachers reported modifying some activities are included in these analyses.

Other High School Computer Science Instructional Resources

High school computer science teachers were asked about school policies related to the provision of instructional resources in their randomly selected class. Typically, if a particular technology is required, the school provides it for students (see Table 49). It is somewhat surprising that any classes require students to provide their own computers or mobile computing devices, but a small percentage do. Even data storage devices (which 13 percent of high school computer science classes require students to provide) can present a financial obstacle to students.

Table 49
Provision of Technologies in High School Computer Science Classes

	PERCENT OF CLASSES		
	COMPUTERS	MOBILE COMPUTING DEVICES	DATA STORAGE DEVICES
Not required for this class	n/a	57 (4.2)	46 (3.3)
Provided by the school, and students are not allowed to use their own	35 (4.5)	9 (2.2)	9 (2.8)
Provided by the school, but students are allowed to use their own	58 (4.5)	15 (2.3)	26 (3.4)
Students are expected to provide their own, but the school has some available for use	2 (0.7)	10 (2.9)	7 (2.2)
Students are required to provide their own	5 (1.6)	8 (3.4)	13 (2.4)

Computer science teachers were presented with a list of more general instructional technologies as indicators of whether classes have access to resources for computer science instruction and asked about availability (either in the classroom or upon request) in their randomly selected class. Almost all high school computer science classes have access to projection devices (e.g., Smartboard, document camera, LCD projector), and more than half have access to robotics equipment (see Table 50).

Table 50
Availability† of Instructional Technologies in High School Computer Science Classes

	PERCENT OF CLASSES
Projection devices (e.g., Smartboard, document camera, LCD projector)	99 (0.5)
Robotics equipment	57 (3.3)
Probes for collecting data (e.g., motion sensors, temperature probes)	40 (3.9)

† Includes high school computer science teachers indicating the resource is always available in their classroom or available upon request.

The survey also asked teachers how great a problem each of several factors presents in their instruction. Given the extent to which high school computer science classes rely on web-based instructional materials, it is perhaps not surprising that school restrictions on Internet content is a problem in 37 percent of classes (see Table 51). Lack of support to maintain technology is a problem in 34 percent of classes, and lack of functioning computing devices in 27 percent of classes. Given the ubiquity of Internet in schools, it is surprising that teachers in almost 1 in 5 high school computer science classes see the lack of reliable Internet access as a problem.

Table 51
Factors Perceived as Problems† in High School Computer Science Classes

	PERCENT OF CLASSES
School restrictions on Internet content that is allowed	37 (4.3)
Lack of support to maintain technology (e.g., repair broken devices, install software)	34 (4.4)
Lack of functioning computing devices (e.g., desktop computers, laptop computers, tablets, smartphones)	27 (4.5)
Lack of reliable access to the Internet	19 (4.4)
Insufficient power sources for devices (e.g., electrical outlets, charging stations)	14 (3.1)

† Includes high school computer science teachers indicating "somewhat of a problem" or "serious problem" on a three-point scale from 1 "not a significant problem" to 3 "serious problem."

Factors Affecting High School Computer Science Instruction

High school computer science teachers were asked about various factors that affect instruction in their randomly selected class. As can be seen in Table 52, principal support; students' motivation, interest, and effort; time to plan; and the amount of time for professional development are all seen as promoters of effective instruction in two-thirds or more of classes. A number of factors are seen as having a neutral or mixed effect on computer science instruction in approximately half of classes: college entrance requirements, teacher evaluation policies, parent/guardian expectations and involvement, current state standards, and textbook selection policies.

Table 52
Effect† of Various Factors on Instruction in High School Computer Science Classes

	PERCENT OF CLASSES		
	INHIBITS	NEUTRAL	PROMOTES
Principal support	3 (1.1)	18 (2.7)	79 (2.9)
Students' motivation, interest, and effort in computer science	10 (2.6)	14 (3.3)	76 (4.0)
Amount of time for you to plan, individually and with colleagues	11 (2.1)	19 (3.6)	70 (3.8)
Amount of time available for your professional development	12 (2.3)	21 (3.5)	67 (3.8)
Students' prior knowledge and skills	15 (3.1)	25 (3.5)	60 (4.0)
College entrance requirements	5 (1.3)	49 (4.7)	47 (4.9)
Teacher evaluation policies	9 (2.0)	46 (4.9)	45 (5.0)
Parent/guardian expectations and involvement	9 (2.1)	48 (3.9)	43 (4.1)
Current state standards	11 (2.6)	49 (4.5)	40 (4.7)
Textbook selection policies	13 (2.5)	60 (4.9)	27 (4.5)

† High school computer science teachers rated the effect of each factor on a five-point scale ranging from 1 "inhibits effective instruction" to 5 "promotes effective instruction." The "inhibits" column includes those indicating 1 or 2. The "promotes" column includes those indicating 4 or 5.

Three composites from these questionnaire items were created to summarize the extent to which various factors support effective instruction: (1) Extent to Which School Support Promotes Effective Instruction (i.e., amount of time for professional development, and amount of planning time); (2) Extent to Which the Policy Environment Promotes Effective Instruction (i.e., textbook selection, teacher evaluation, and current state standards); and (3) Extent to Which Stakeholders

Promote Effective Instruction (i.e., students' motivation and interest, students' prior knowledge, parent/guardian expectations and involvement). As shown in Table 53, school and stakeholder support are generally high (mean scores of 74 and 70, respectively) compared with the policy environment (mean score of 59). No statistically significant patterns were found in the means for these composites when they were examined by equity factors.

Table 53
High School Computer Science Class Mean Scores for Factors Affecting Instruction Composites

	MEAN SCORE
Extent to Which School Support Promotes Effective Instruction	74 (1.9)
Extent to Which Stakeholders Promote Effective Instruction	70 (1.7)
Extent to Which the Policy Environment Promotes Effective Instruction	59 (2.1)

Summary

Nearly all high school computer science teachers characterize themselves as white, and the majority of them are male. Although only a small percentage of teachers are from race/ethnicity groups historically underrepresented in STEM, classes with the highest proportion of students from these groups are more likely to be taught by teachers from these groups than classes with the lowest proportion of students from these groups.

In terms of teaching experience, many high school computer science teachers are new to teaching computer science, with a third having 0–2 years of experience teaching the subject—including over half of computer science teachers in high-poverty high schools. However, fewer are new to K–12 teaching in general, and nearly half of high school computer science teachers have more than 10 years of teaching experience. Their most common areas of teaching certification are computer science, mathematics, and business; 16 percent have not earned a teaching credential. Only a fourth have a degree in computer engineering, computer science, information science, or computer science education, although nearly half have taken coursework in at least 3 of 4 computer science content areas recommended by CSTA and ISTE. Thus it is not surprising that, compared to high school science and mathematics teachers, high school computer science teachers feel less well prepared to teach their respective content.

Roughly half of computer science teachers feel very well prepared to encourage students' interest in computer science, develop students' ability to do computer science, and encourage participation of all students in computer science. However, fewer than a quarter feel very well prepared to differentiate computer science instruction to meet the needs of diverse learners or to incorporate students' cultural backgrounds into computer science instruction. In addition, data on computer science teachers' beliefs about effective teaching show a dichotomy. On the one hand, a large majority hold a number of beliefs about teaching and learning that are in alignment with what is known about effective instruction (e.g., students should learn computer science by doing computer science, teachers should ask students to justify their solutions). On the other hand, a substantial proportion holds views inconsistent with this research. For example, 3 out of 4 computer science teachers believe that students should be provided with definitions for new vocabulary at the beginning of instruction on an idea. Interestingly, classes in schools with the

highest proportions of students eligible for free/reduced-price lunch are somewhat more likely to be taught by teachers with stronger reform-oriented beliefs than those in low-poverty schools.

A large majority of high school computer science teachers have participated in computer science-focused professional development in the last three years, and over half have had more than 35 hours of professional development in that time period. The professional development attended by the majority of computer science teachers has had a heavy focus on deepening their understanding of computer science content and how computer science is done. Professional development is much less likely to have had a heavy emphasis on differentiating instruction to meet the needs of diverse learners or incorporating students' cultural backgrounds into instruction. Interestingly, high school computer science classes with the largest proportion of students from race/ethnicity groups historically underrepresented in STEM are more likely than classes with the smallest proportion of students from these groups to be taught by teachers who have experienced aspects of effective professional development.

Data on computer science courses indicate that 70 percent of high school students in the nation have access to one or more computer science courses at their schools. However, fewer than half of high school students have access to an AP computer science course. Equity analyses revealed statistically significant differences in the percentages of schools that offer computer science by region, school size, and community type, with larger schools more likely to offer computer science than smaller schools and rural schools less likely to offer it than suburban or urban schools. Unsurprisingly, equity analyses of AP computer science offerings revealed similar patterns to computer science courses more generally. However, for AP course offerings, the difference between high-poverty schools and low-poverty schools was significant, with high-poverty schools being less likely to offer AP courses. Although about half of the high school student population is female and a similar proportion of students are from race/ethnicity groups historically underrepresented in STEM, fewer than one-third of students in high school computer science courses are female or from these race/ethnicity groups.

The most common modes of instruction in high school computer science courses are having students work on programming activities and lecture. In terms of engagement with the practices of computer science, students are engaged in activities related to developing, testing, and refining computational artifacts at least once a week in most computer science classes. However, they are engaged in practices related to end users much less often. There were no statistically significant differences by equity factors in the frequency with which students are engaged in practices of computer science, although classes with a higher proportion of students from race/ethnicity groups historically underrepresented in STEM fields are more likely than classes with fewer students from these groups to emphasize reform-oriented objectives. Similarly, classes in schools with a higher proportion of students eligible for free/reduced-price lunch are more likely than classes in schools with fewer eligible students to emphasize these types of objectives.

Only about a quarter of high school computer science classes have instructional materials designated for them, and instruction is most commonly based on teacher-developed lessons, with lessons from free websites a distant second. When teachers do use commercially published materials, they often modify them, supplementing and skipping elements for a variety of reasons. In terms of other resources for instruction, most, but not all, schools provide required computing and data storage technology for students.