Increasing Exposure to Programming: A Comparison of Demographic Characteristics of Students Enrolled in Introductory Computer Science Programming Courses vs. a Multidisciplinary Data Analysis Course

Jennifer Cooper[1] & Lisa Dierker[1]

[1] Psychology Department, Wesleyan Unviversity, Middletown, CT, USA

Correspondence: Lisa Dierker, Psychology Department, Wesleyan University, 207 High Street, Middletown, CT, 06459, USA. Tel: 860-685-2137.

Received: February 6, 2017 Accepted: February 28, 2017 Online Published: March 10, 2017

doi:10.5430/irhe.v2n1p92 URL: https://doi.org/10.5430/irhe.v2n1p92

Abstract

Upward trends in programming careers and increases in programming in less traditionally computer-oriented occupations threaten to further increase the current underrepresentation of both females and individuals from racial minority groups in these disciplines. Using administrative data (2009 – 2014), the present study compares demographic characteristics of students enrolled in a course that introduced programming: a multidisciplinary data analysis course, an introductory programming course, or an introductory computer science (CS) gateway course. The multidisciplinary data analysis course enrolled significantly more female students and students with lower Math SAT scores. Females were overrepresented in the data analysis course and underrepresented in the introductory programming and CS gateway courses relative to the larger campus community, with similar findings for underrepresented minority students. Less emphasis on traditional approaches to introductory programming and increased interdisciplinary opportunities to tackle real world questions may be one way to improve access to programming experiences for students from a wider range of educational, social and economic backgrounds.

Keywords: programming, diversity, access, STEM, interdisciplinary, statistics education

1. Introduction

Currently, over half of the top 25 highest paying in-demand jobs require specialized or general programming skills; for example software engineer, data base administrator, computer hardware engineer, security engineer, data scientist, analytics manager, solutions architect, software architect, etc. (Glassdoor, 2015). In addition to the predicted increase of 17.7% in these and other technical occupations between 2012 and 2022 (Bureau of Labor Statistics, 2013), programming is becoming increasingly relevant in many other growing careers, such as research in the sciences and other domains (Hambrusch et al., 2009; Merali, 2010), which have previously imposed less computational and technological demands (Barnes, 2010; Glassdoor, 2015; Wing, 2006).

Importantly, without considerable changes to the system, the underrepresentation of females and individuals from racial minority groups in these careers is likely to continue. Recent reports from large technology companies (e.g., Apple, Facebook, Google, Twitter, Microsoft, Cisco, Intel, etc.) have again focused attention on this lack of diversity. Employee data from 2014 indicates that most large technology companies employ more than 70% males and with few exceptions, well less than 10% Blacks or Hispanics (Mangalindan, 2014). These disparities are even more pronounced when considering diversity in both leadership and technical roles where, at Google for example, females hold as little as 17% of the technology positions, while Blacks and Hispanics occupy only 3% of positions in corporate leadership (Google, 2015). In academia, females earned less than 20% of the computer science [CS] bachelor degrees in 2012 while Blacks and Hispanics each earned approximately 10% (National Science Foundation [NSF]). As traditionally non-computational careers are increasingly depending on technological skills such as programming, these disparities will potentially expand.

These disparities at all stages are well known, as is the leaky pipeline metaphor (Blickenstaff, 2005), with considerable attention directed towards potential causes (e.g., Varma, 2006; 2010) and potential solutions (e.g., Whittaker and Montgomery, 2012). Though increased funding from both the private (e.g., Apple Inc. and the Gates Foundation) and public sectors (e.g., NSF), along with the launch of university programs (e.g., Alvarado and Dodds, 2010; Eney et al., 2013) and national initiatives specifically targeted at diversifying the pool of students exposed to programming curriculum prior to higher education (e.g., Girls who Code, ARTSI Alliance, and All Star Code) has accelerated in recent years, there is still considerable room for improvement. Overall progress has been slow and limited in scope (for reviews of challenges, see Frieze and Quesenberry, 2013; Peckham et al., 2007; Whittaker and Montgomery, 2012). This is in addition to ongoing efforts to improve computer science education for all students (e.g., Porter et al., 2013).

Though programming curricula are traditionally provided in courses offered through computer science departments, the proliferation of programming across many disciplines (ACM, 2013; Hambrusch et al., 2009) provides an as yet, largely untapped opportunity for evaluating how newer multidisciplinary initiatives might expand exposure to programming (e.g., data science – Hardin et al., 2014, Heinrichs et al., 2011, Squire, 2012; bioinformatics – Tartaro and Chosed, 2015; and geography – Muller and Kidd, 2014). Programming involves learning to plan, write and execute the formal code of diverse computer languages that each rely on such skills as defining variables, creating algorithms, flow control, functions, setting parameters, understanding data structures, requesting input and output, setting up arrays, error handling, and debugging (ACM, 2013; Downey et al., 2012; Nolan and Temple Lange, 2010; Robins et al., 2010). Programming also enhances problem solving skills as students are required to decompose problems and develop appropriate algorithms (Burton and Bruhn, 2003; Robins et al., 2010). These are skills and computational ways of thinking (Wing, 2006) that are becomingly increasingly required in both traditional and non-traditional computational or technological careers.

The hypothesis addressed in the present paper is that that reframing introductory programming experiences from the discipline-specific to an approach that tackles real world questions across multiple disciplines will contribute to improving access to programming skills for students from a wider range of educational, social and economic backgrounds. Previous publications have described the development of a multidisciplinary data analysis course (Dierker, et al, 2012) aimed at engaging students in data-oriented projects that rely heavily on programming. Utilizing a flipped classroom approach (Bishop and Verleger, 2013; Winquist and Carlson, 2014; Yarbo et al., 2014), the course is designed around student research projects of their own choosing and offers individualized hands-on experience in one of four code-based statistical software platforms (SAS, R, Stata, and SPSS). The course introduces students to several basic programming concepts and skills in the pursuit of managing and analyzing real world data. The skills taught are those that would be relevant across disciplines. From reading in their data (file input and output), to managing variables (variable types, creation, and modification), to selecting subsets of data (indexing and control structures), performing descriptive and inferential analyses (using functions with named arguments), and generating graphs (graphics output), students actively use formal syntax to write and execute programs aimed at managing and analyzing data. The course combines an emphasis on individualized research questions and hands-on problem solving during class time with acquiring programming skills in service of answering the students' research question.

Introduced into the curriculum in the fall of 2009 at a selective liberal arts college, one of the goals of the multidisciplinary data analysis course was to increase access to programming opportunities both generally and for students traditionally marginalized within many STEM fields (e.g., women, underrepresented students, educationally disadvantaged students, students pursuing majors that do not require programming courses). To evaluate the potential impact of this multidisciplinary data analysis course on access to programming opportunities, the present paper compares demographic characteristics of students enrolling in the multidisciplinary data analysis course to those enrolling in a traditional introductory programming or the CS gateway course.

2. Method

2.1 Participants

Administrative data were examined for students enrolled between fall semester 2009 and spring semester 2014 in a) a multidisciplinary data analysis course (N = 463), b) an introductory programming course (N = 454), and c) the introductory gateway course to the CS major (N = 132). For students taking more than one of these courses during this 5 year period, only their first course was considered. Students were excluded from the present analyses if they took any of these courses prior to Fall 2009 (n = 11) or enrolled simultaneously in two of the courses (n = 12).

The average enrollment per class section in the multidisciplinary data analysis course was 15.2 (SD = 5.9, range: 6 – 22.4) compared to an average of 16.1 students per section in the introductory programming course (SD = 7.1, range: 4 – 28.0) and 22.6 students (SD = 5.6, range = 16 – 31.0) in the CS gateway course.

The multidisciplinary data analysis course was offered through the Quantitative Analysis Center, a collaborative effort of academic and administrative departments that supports quantitative analysis and programming across the curriculum and provides an institutional framework for collaboration across departments and disciplines in the area of data driven research. Titled "Applied Data Analysis," the course is described in the university's online catalog as a "project-based course, [in which] you will have the opportunity to answer questions that you feel passionately about through independent research based on existing data. Students will have the opportunity to develop skills in generating testable hypotheses, conducting a literature review, preparing data for analysis, conducting descriptive and inferential statistical analyses, and presenting research findings. The course offers unlimited one-on-one support, ample opportunities to work with other students, and training in the skills required to complete a project of your own design. These skills will prepare you to work in many different research labs across the University that collect empirical data." There are no prerequisites.

Offered through the CS department, the introductory programming course is titled "Introduction to Programming" and is described in the on-line catalog as providing "an introduction to a modern high-level programming language including a discussion of input/output, basic control structures, types, functions, and classes. The lectures will also discuss a variety of algorithms as well as program design issues." In courses prior to Fall 2012, the course description also indicated that "while there are no formal prerequisites, familiarity with using computers is assumed."

The CS gateway course is titled "Computer Science I" and is described in the on-line catalog as "the first course in a two-course sequence … that is the gateway to the computer science major." The majority of sections noted that "the course will focus on one particular programming language as well as associated computing and mathematical concepts and formalisms" and that the "course is intended for CS majors and others who want an in-depth understanding of programming and expect to continue programming after this course." Earlier sections noted that while there were no prerequisites, the course was significantly more challenging than the introductory programming course. This was revised starting in spring 2013, to state that some previous acquaintance with computer programming is highly recommended and that students typically are concurrently enrolled in Calculus. No prerequisite courses were listed.

All three courses were open to all students. The data analysis course could be used as one option to fulfill a major requirement for psychology, earth and environmental science, government, neuroscience and behavior, and sociology. Only the CS gateway course counted towards the CS major. All three courses could be counted toward the major requirements for biology or could be applied to the natural sciences and mathematics general education recommendations.

All three courses introduced students to programming concepts such as basic file input and output, variable manipulation and types, indexing, the use of functions from existing libraries, and basic control structures such as if-then. Additionally, all three courses emphasized the need to develop algorithms in order to solve a problem and the need to test the accuracy of that algorithm. Within the multidisciplinary data analysis course, these skills were learned in the context of data and an empirical research question. Within both of the CS courses, these topics, along with more advanced programming concepts, were learned largely through a series of basic and applied exercises. The programming and CS courses typically used either Python or Java, while the data analysis course used syntax-based SAS, R, Stata, or SPSS.

2.2 Measures

Administrative data were supplied by the institutional research office. This data included gender and the following variables:

Race/Ethnicity. Self-reported race/ethnicity included endorsement of one or more of the following categories: Black, White, Hispanic, Asian or other. Those not endorsing any of those categories were considered unknown. Black and/or Hispanic students were categorized as underrepresented minority students (URM).

Financial aid. Students with demonstrated need receiving grants and/or self-help financial aid were compared to those enrolled in the university without financial assistance.

High school type. Students' high school backgrounds were collapsed based on whether they attended a public vs. non-public high school. The non-public category included private schools, religious schools, and homeschools.

Standardized test scores. The majority of the students (84%) provided SAT (Math, Critical Reading, and Writing) scores.

Class year. Class year (freshman, sophomore, junior, senior) was dichotomized into lower vs upper classmen, with students having 4 or fewer semesters until graduation considered upperclassmen.

2.3 Analyses

Binomial tests were conducted to determine how representative each course's enrollment was of the larger university community. Across courses, bivariate analyses were conducted on the relationships between each student characteristic and course enrollment. Chi-square Tests of Independence and ANOVA were used for categorically and continuously measured characteristics respectively. For significant bivariate associations, logistic regression analyses were conducted to evaluate the presence of possible confounding effects of other student characteristics on the association of interest. For significant associations, their interactions were also tested.

3. Results

3.1 Students' Representativeness of the Wider Campus Community

Females were found to be overrepresented in the data analysis course (61%, CI: 56 – 65%; university: 52%, $z = 3.76$, $p < .001$) and underrepresented in the introductory programming (30%, CI: 25 – 34%; $z = 9.58$, $p < .001$) and CS gateway (23%, CI: 16 – 31%; $z = 6.72$, $p < .001$) courses. URM students were also overrepresented in the data analysis course (24%, CI: 20 - 28%; university: 19.4%, $z = 4.24$, $p < .001$) and underrepresented in the introductory programming course (12%, CI: 9 - 16%; $z = 2.46$, $p = .013$). The CS gateway course enrolled statistically similar rates of URM students (14%, CI: 9 – 22%; university: 19.4%) compared to the larger university community and higher rates of students from public high schools (67%, CI: 58 – 75%; university rate: 53%, $z = 3.06$, $p = .002$). Enrollment of students from public high schools was similar to the university rate for both the data analysis course (54%, CI: 49 – 58%) and the introductory programming course (56%, CI: 51 – 60%). Each course enrolled similar proportions of students receiving financial aid in comparison to the larger university community.

3.2 Student Characteristics by Course

Comparisons of student characteristics in the data analysis course versus both the introductory programming course and the CS gateway course are presented in Table 1. Each course enrolled similar proportions of White students, Asian students, and students receiving financial aid. Average SAT scores in critical reading and writing were also statistically similar across the three courses.

The data analysis course however, enrolled significantly more females (60.7%) compared to both the introductory programming course (29.5%, $\chi^2(1) = 88.7$, $p < .001$) and the CS gateway course (22.7%, $\chi^2(1) = 58.8$, $p < .001$). Further, the data analysis course enrolled students with significantly lower average math SAT scores ($M = 685$, $SD = 69$) compared to both the introductory programming course ($M = 722$, $SD = 58$) and the CS gateway course ($M = 739$, $SD = 55$), $F(2, 878) = 50.7$, $p < .001$, Tukey's HSD. While each of these differences persisted when controlling individually for all other student characteristics, a significant interaction between Math SAT and financial aid status, when comparing the data analysis course to the introductory programming course, showed that among students with the highest Math SAT scores (730-800), those receiving financial aid were more likely to enroll in the data analysis course than students not receiving financial aid (See Figure 1).

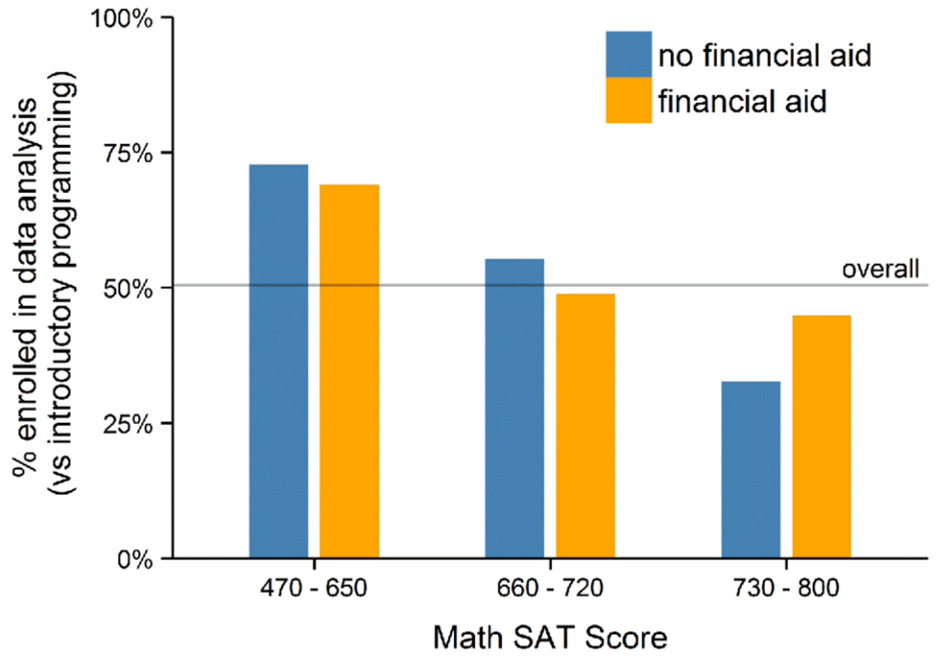

Figure 1. The association between GPA and enrollment by financial aid status

Higher rates of both Black and Hispanic students were enrolled in the data analysis course compared only to the CS gateway course. When analyses were conducted based on under-represented (URM) student status (i.e., Black and/or Hispanic), the data analysis course was found to enroll significantly higher rates of URM students (24%) than both the introductory programming course (12%, $\chi^2(1) = 20.1$, $p < .001$) and the CS gateway course (14%, $\chi^2(1) = 4.9$, $p = .03$). However, when controlling for Math SAT scores, these effects were no longer significant.

Notably, the CS gateway course (67%) was found to enroll a significantly higher proportion of students from public high schools than the data analysis course (54%, $\chi^2(1) = 6.65$, $p < .01$). A significant interaction between public school status and gender ($p = .02$) revealed that this was true only for females.

The data analysis course was found to enroll significantly fewer students in their freshman and sophomore year (37%) than both the introductory programming course (57%, $\chi^2(1) = 38.1$, $p < .001$) and the CS gateway course (85%, $\chi^2(1) = 96.5$, $p < .001$). Overall, effect sizes were found to be small to moderate.

Table 1. Student characteristics by course

	Multidisciplinary Data Analysis (n = 463)	Introductory Programming (n = 454)	CS Gateway (n = 132)	Statistics
First and Second Years	170 (36.7%)[a]	260 (57.3%)[b]	113 (85.6%)[c]	$\chi^2(2) = 108.0$, $p < .001$, V = 0.32
Female	281 (60.7%)[a]	134 (29.5%)[b]	30 (22.7%)[b]	$\chi^2(2) = 115.2$, $p < .001$, V = 0.33
Public High School	248 (53.6%)[a]	252 (55.5%)[a]	88 (66.7%)[b]	$\chi^2(2) = 7.2$, $p = .027$, V = 0.08
Financial Aid	226 (48.8%)	200 (44.1%)	69 (52.3%)	n.s.
Racial Categories[1]				
White	256 (55.3%)	272 (59.9%)	79 (59.8%)	n.s.
Black	59 (12.7%)[a]	28 (6.2%)[b]	9 (6.8%)[a,b]	$\chi^2(2) = 12.9$, $p = .002$, V = 0.11
Hispanic	56 (12.1%)[a]	30 (6.6%)[b]	11 (8.3%)[a,b]	$\chi^2(2) = 8.4$, $p = .015$, V = 0.09
Asian	89 (19.2%)	116 (25.6%)	30 (22.7%)	n.s.
Other	43 (9.3%)	44 (9.7%)	17 (12.9%)	n.s.
Underrepresented(URM)[2]	111 (24.0%)[a]	56 (12.3%)[b]	19 (14.4%)[b]	$\chi^2(2) = 22.4$, $p < .001$, V = .0.15
SAT M (SD)	n = 384	n = 381	n = 116	
Math	684.7 (69.1)[a]	722.3 (57.5)[b]	738.6 (55.0)[b]	$F(2, 878) = 50.7$, $p < .001$, $\eta^2 = .10$
Critical Reading	692.9 (72.7)	688.9 (75.0)	705.3 (69.7)	n.s.
Writing	680.7 (73.3)	680.5 (76.7)	697.6 (74.0)	n.s.

[1] Racial categories are not mutually exclusive.

[2] Underrepresented minority students (URM) are those indicating Black and/or Hispanic ethnicity

4. Discussion

This research considered the potential impact of offering a multidisciplinary data analysis course on access to programming opportunities for students underrepresented in traditional programming courses. Supporting our hypothesis, the data analysis course enrolled significantly more females and students with significantly lower math SAT scores than either of the programming courses offered through the CS department. These findings persisted even when controlling for race, financial aid status, high school type, and class year. Further, the multidisciplinary data analysis course was also found to enroll higher rates of females and URM students compared to their representation in the wider campus community. While each of the three courses enrolled similar rates of students with or without financial aid, among those students with the highest math SAT scores, those receiving financial aid were more likely to enroll in the data analysis course. Taken together, these findings confirm that less emphasis on a traditional approach to introductory programming and more opportunities to tackle real world questions in the context of multiple disciplines may contribute to improving access to programming experiences for students from a wider range of educational, social and economic backgrounds (see Stephenson et al., 2007).

Importantly, however, our findings that the data analysis course enrolled significantly fewer students in their freshman and sophomore year than both the introductory programming course and the CS gateway course suggests that the programming experience students received in the data analysis course may have been less likely to impact their academic trajectory. In particular, earlier involvement in research and longer time periods of research involvement have been shown to promote continued coursework and majoring in CS (Russell et al., 2007; Tashakkori et al., 2011). Overall, quality research participation has positive effects on students' skills and retention in STEM (Kilgo et al., 2015; see Barker, 2009), with some evidence suggesting stronger benefits for underrepresented students (Lopatto, 2007). Programming opportunities that engage students in research within and across disciplines can be used to increase high quality experiences that engage students in both programming and STEM.

Preliminary analyses based on exit survey results from students who completed the data analysis course indicated that more than a third of students were interested in taking a future programming course. This suggests that attracting more freshman and sophomores to the data analysis course would likely provide greater opportunities for students to build on basic programming skills in future coursework. More research is needed to consider the future coursework and career choices of these students and how their experiences in the multidisciplinary data analysis course might offer another pathway into the increasing number of occupations in which programming skills are beneficial.

Unexpectedly, the CS gateway course was found to enroll higher rates of students from public high schools compared to the larger campus community. This held true for female, but not male public high school students in the CS gateway course compared to the data analysis course. In fact, 26 of the 30 females who were first enrolled in the CS gateway course came from public high schools. This may reflect the relatively recent efforts in the public K-12 system to engage female students in high quality computing and STEM experiences (e.g., McGee et al., 2013; Stephenson and Wilson, 2012). Thus, despite few females entering the gateway course as their first college level programming experience, those who did may have been the product of accelerating institutional K-12 efforts to attract women to programming (Armoni and Gal-Ezer, 2014).

Through the use of formal syntax to write complex programs aimed at managing and analyzing data, the data analysis course introduces students to several of the same concepts and skills covered in traditional programming courses (Nolan and Temple Lang, 2010; ACM, 2013). Notably, however, the data analysis course is not described as a programming course in the on-line course catalog. Instead, emphasis is placed on student projects, independent research, and applied skills. Despite this limited reference to programming, students enrolling in the data analysis course following fall semester 2009, were aware through word of mouth that they would be writing programs in the service of their projects. Anecdotally, while a minority of students in the data analysis course did have previous research experience and clearly enrolled for the exposure to a new code based platform, the vast majority did not and were instead motivated by the opportunity to answer their own applied research questions and/or to fulfill a requirement for their major. Importantly, our approach of using contextualized projects and research to motivate students is one that has also been undertaken in CS courses (Jenkins, 2001; Goldweber et al., 2012) and has been shown to be particularly motivating to female students (Rader et al., 2011).

The content and context of the courses is important to consider because programming, well-known to be a difficult skill to acquire, is learned by doing, and requires a considerable investment of thought, practice, and motivation to master (Carter et al., 2011; Jenkins, 2001; Settle et al., 2014). Jenkins (2001) found that a considerable portion of the students in an introductory CS course were motivated to learn programming only because it was required. This presents a problem for both attracting and engaging students. Without an interdisciplinary, applied context (e.g., Guzdial, 2010), it is unlikely that programming will become a common requirement outside of the CS major, although work continues on introducing computational thinking into courses across the curriculum (e.g., Curzon et al., 2009).

A limitation of the present study is a quasi-experimental design that would benefit from replication within educational contexts that might allow for random assignment to diverse introductory programming experiences. Further, the courses compared in the present study are not required of students at the university, though in some cases they count toward specific majors. Thus future research should evaluate introductory programming experiences in terms of their success of retaining students in a variety of STEM majors.

In summary, the present study presents a multidisciplinary applied data analysis approach to introducing programming as one potential way to increase the access of underrepresented students to programming concepts and skills. This research joins a literature that has many examples of such innovations, but this course stands out for the combination of its design (flipped classroom, project-based, etc.) with its emphasis on programming as a problem-solving tool applicable across many disciplines. Importantly, the kind of programming experience that is needed has been changing fairly rapidly with the introduction of new computer languages and application program interfaces (e.g., ACM, 2013). In a recent paper, Gasson and Haden (2014) introduce four non-exclusive types of computer users: the theoretician (focusing on the mathematical formalisms), the practitioner (developing software and hardware), the power user (see Dorn and Guzdial, 2006), and the end user (using non-specialized software). The power user, who will use programming concepts in the service of a problem or task in their own field of expertise, represents the type of programmer whose role is most clearly expanding across diverse occupations. Multidisciplinary project-based courses such as ours may be an important step in attracting larger numbers of students from diverse backgrounds into introductory courses that will begin to provide the education and experience needed to play this kind of role.

Acknowledgements

This research was conducted in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This research was supported by grant 0942246 and 1323084 from the National Science Foundation, Transforming Undergraduate Education in Science, Technology, Engineering and Mathematics (TUES).

References

ACM/IEEE-CS Joint Task Force on Computing Curricula. (2013). *Computer science curricula 2013*. ACM Press and IEEE Computer Society Press. http://dx.doi.org/10.1145/2534860

Alvarado, C., & Dodds, Z. (2010). Women in CS: An evaluation of three promising practices. In *Proceedings of the 41st ACM technical symposium on Computer Science Education* (pp. 57-61). ACM. http://dx.doi.org/10.1145/1734263.1734281

Armoni, M., & Gal-Ezer, J. (2014). High school computer science education paves the way for higher education: the Israeli case. *Computer Science Education*, *24*, 101-122. http://dx.doi.org/10.1080/08993408.2014.936655

Barker, L. (2009). Student and faculty perceptions of undergraduate research experiences in computing. *ACM Transactions on Computing Education (TOCE)*, *9*(1), 5. http://doi.acm.org/10.1145/1513593.1513598

Barnes, N. (2010). Publish your computer code: it is good enough. *Nature*, *467*, 753-753. http://dx.doi.org/10.1038/467753a

Bishop, J. L., & Verleger, M. A. (2013). The flipped classroom: A survey of the research. In *American Society for Engineering Education (ASEE) National Conference Proceedings,* Article 6219, Atlanta, GA. Retrieved from https://www.asee.org/public/conferences/20/papers/6219/view

Bureau of Labor Statistics, U.S. Department of Labor. (2013, December 19). Employment projections: Employment by detailed occupation. Retrieved from Bureau of Labor Statistics: http://www.bls.gov/emp/ep_table_102.htm

Burton, P. J., & Bruhn, R. E. (2003). Teaching programming in the OOP era. *ACM SIGCSE Bulletin*, *35*(2), 111-114. http://dx.doi.org/10.1145/782941.782993

Carter, J., Bouvier, D., Cardell-Oliver, R., Hamilton, M., Kurkovsky, S., Markham, S. *et al*.. (2011). ITiCSE Working Group 2011: Motivating All Our Students?. In, *ITiCSE 2011, 16th Annual Conference on Innovation and Technology in Computer Science Education,* Darmstadt, Germany. Retrieved from http://eprints.soton.ac.uk/272530/

Curzon, P., Peckham, J., Taylor, H., Settle, A., & Roberts, E. (2009). Computational thinking (CT): On weaving it in. In *ACM SIGCSE Bulletin, 41*(3), 201-202. Retrieved from https://www.dl.acm.org/ft_gateway.cfm?id=1562941

Dierker, L., Kaparakis, E., Rose, J., Selya, A., & Beveridge, D. (2012). Strength in Numbers: A Multidisciplinary, Project-Based Approach to Introductory Statistics Education. *Journal of Effective Teaching*, *12*, 4-14. Retrieved from https://eric.ed.gov/?id=EJ1092198

Dorn, B., & Guzdial, M. (2006, September). Graphic designers who program as informal computer science learners. In *Proceedings of the second international workshop on computing education research,* 127-134. 10.1109/VLHCC.2007.35

Downey, A., Elkner, J., & Meyers, C. (2002). *How to think like a computer scientist: learning with python*. Wellesley, Massachusetts: Green Tea Press.

Eney, C., Lazowska, E., Martin, H., & Reges, S. (2013, March). Broadening participation: The why and the how. *Computer*, (3), 48-51. http://dx.doi.org/10.1109/MC.2013.83

Frieze, C., & Quesenberry, J. L. (2013). From difference to diversity: Including women in the changing face of computing. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 445-450). Retrieved from https://www.dl.acm.org/ft_gateway.cfm?id=2445327

Gasson, J., & Haden, P. (2014). Modern digital literacy: The computer science curriculum goes mainstream. In B. Hegarty, J. McDonald, & S. K. Loke (Eds.), *Rhetoric and Reality: Critical Perspectives on Educational Technology. Proceedings ascilite Dunedin* (pp. 49 – 57).

Glassdoor. (2015, February 17). 25 highest paying jobs in demand. Retrieved March 20, 2015 from Glassdoor blog: http://www.glassdoor.com/blog/highest-paying-jobs-demand/

Goldweber, M., Barr, J., Clear, T., Davoli, R., Mann, S., Patitsas, E., & Portnoff, S. (2013). A framework for enhancing the social good in computing education: a values approach. *ACM Inroads*, *4*(1), 58-79.

Google. (2015). Diversity. January 2015 data. Retrieved March 20, 2015 from http://www.google.com/diversity/index.html

Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads*, *1*(4), 4-6. Retrieved from http://www.dl.acm.org/citation.cfm?id=1869747

Heinrichs, L. R., Hutchings, D., Kleckner, M., & Squire, M. (2011). Charting a new curriculum for a data-driven world. *Issues in Information Systems*, *12,* 256-263. Retrieved from http://www.academia.edu/9240902/CHARTING_A_NEW_CURRICULUM_FOR_A_DATA-DRIVEN_WORLD

Jenkins, T. (2001). The motivation of students of programming. *ACM SIGCSE Bulletin, 33*(3), 53-56. http://dx.doi.org/10.1145/507758.377472

Kilgo, C. A., Ezell Sheets, J. K., & Pascarella, E. T. (2015). The link between high-impact practices and student learning: some longitudinal evidence. *Higher Education*, *69*, 509-525. http://dx.doi.org/10.1007/s10734-014-9788-z

Lopatto, D. (2007). Undergraduate research experiences support science career decisions and active learning. *CBE-Life Sciences Education*, *6*, 297-306. http://dx.doi.org/10.1187/cbe.07

Mangalindan, J.P. (2014, August 29). How tech companies compare in employee diversity. Retrieved March 20, 2015 from http://fortune.com/2014/08/29/how-tech-companies-compare-in-employee-diversity/

Mcgee, S., Greenberg, R. I., Reed, D. R., & Duck, J. (2013). Evaluation of the IMPACTS Computer Science Presentations. (2013). *ISTE: SIGCT.* Retrieved from https://www.iste.org/resources/product?id=2853

Merali, Z. (2010). Computational science: Error, why scientific programming does not compute. *Nature*, *467*(7317), 775-777. Retrieved from http://www.nature.com/news/2010/101013/full/467775a.html

Muller, C. L., & Kidd, C. (2014). Debugging geographers: teaching programming to non-computer scientists. *Journal of Geography in Higher Education*, *38*(2), 175-192. http://dx.doi.org/10.1080/03098265.2014.908275

National Science Foundation [NSF], National Center for Science and Engineering Statistics, special tabulations of U.S. Department of Education, National Center for Education Statistics, Integrated Postsecondary Education Data System, Completions Survey, 2002–12. Tables 5-1 and 5-6. Retrieved March 20, 2015, from http://www.nsf.gov/statistics/2015/nsf15311/tables.cfm

Nolan, D., & Temple Lang, D. (2010). Computing in the Statistics Curricula. *The American Statistician*, *64*, 97-107. http://dx.doi.org/10.1198/tast.2010.09132

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., ... & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*, 204-223. http://dx.doi.org/10.1080/08993400500150747

Peckham, J., Harlow, L. L., Stuart, D. A., Silver, B., Mederer, H., & Stephenson, P. D. (2007). Broadening participation in computing: issues and challenges. *ACM SIGCSE Bulletin, 39*(3), 9-13. http://dx.doi.org/10.1145/1269900.1268790

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: What works? *Communications of the ACM*, *56*(8), 34-36. Retrieved from http://www.dl.acm.org/citation.cfm?id=2492007.2492020

Rader, C., Hakkarinen, D., Moskal, B. M., & Hellman, K. (2011). Exploring the appeal of socially relevant computing: are students interested in socially relevant problems? In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 423-428). Retrieved from http://www.dl.acm.org/ft_gateway.cfm?id=1953288

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137-172. http://dx.doi.org/10.1076/csed.13.2.137.14200

Russell, S. H., Hancock, M. P., & McCullough, J. (2007). Benefits of undergraduate research experiences. *Science*, *316*, 548-549. http://dx.doi.org/10.1126/science.1140384

Settle, A., Vihavainen, A., & Sorva, J. (2014). Three views on motivation and programming. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science education* (pp. 321-322). http://dx.doi.org/10.1145/2591708.2591709

Squire, M. (2012). Outline and exercises for a novel introductory course in data science and visualization. *Issues in Information Systems, 13,* 382–390. Retrieved from http://iacis.org/iis/2012/76_iis_2012_382--390.pdf

Stephenson, C., & Wilson, C. (2012). Reforming K-12 computer science education… What will your story be? *ACM Inroads*, *3*(2), 43-46. http://dx.doi.org/10.1145/2189835.2189850

Stephenson, P., Miguel, L., Peckham, J., Hervé, J. Y., & Hutt, R. (2007). Using undergraduate interdisciplinary research to promote computer science. *Journal of Computing Sciences in Colleges*, *22*(3), 98-104. Retrieved from http://www.dl.acm.org/citation.cfm?id=1181849.1181874

Tashakkori, R., Kurtz, B. L., Parks, D. A., Fenwick Jr, J. B., & McRae, A. A. (2011, March). Early participation of CS students in research. *SIGCSE* (pp. 63-68). http://dx.doi.org/10.1145/1953163.1953185

Taulbee Survey, Computing Research Association. (2013). Taulbee Survey. Retrieved March 20, 2015, from http://cra.org/uploads/documents/resources/crndocs/2013-Taulbee-Survey.pdf

Varma, R. (2006). Making computer science minority-friendly. *Communications of the ACM*, *49*(2), 129-134. Retrieved from http://www.dl.acm.org/citation.cfm?id=1113041

Varma, R. (2010). Why so few women enroll in computing? Gender and ethnic differences in students' perception. *Computer Science Education*, *20*(4), 301-316. Retrieved from https://www.unm.edu/~varma/print/CSE_Few%20Women.pdf

Whittaker, J. A., & Montgomery, B. L. (2012). Cultivating diversity and competency in STEM: Challenges and remedies for removing virtual barriers to constructing diverse higher education communities of success. *Journal of Undergraduate Neuroscience Education*, *11*, A44–A51. http://dx.doi.org/101080/07377363.2012.690623

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. Retrieved from https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Winquist, J. R., & Carlson, K. A. (2014). Flipped statistics class results: Better performance than lecture over one year later. *Journal of Statistics Education*, *22*(3), 1–10. Retrieved from ww2.amstat.org/publications/jse/v22n3/winquist.pdf

Yarbo, J., Arfstrom, K. M., McKnight, K., & McKnight, P. (2014). The 2014 extension of the 2013 review of flipped learning. Flipped Learning Network. Retrieved March 20, 2015, from http://flippedlearning.org/domain/41