UC San Diego

UC San Diego Previously Published Works

Title

When Coding Meets Biology: The Tension Between Access and Authenticity in a Contextualized Coding Class

Permalink

https://escholarship.org/uc/item/7h27k9kr

Authors

Zuckerman, Austin L

Juavinett, Ashley L

Publication Date

2024-03-07

DOI

10.1145/3626252.3630966

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-ShareAlike License, available at https://creativecommons.org/licenses/by-sa/4.0/

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

When Coding Meets Biology

The Tension Between Access and Authenticity in a Contextualized Coding Class

Austin L. Zuckerman
Mathematics and Science Education, UC San Diego
La Jolla, CA, USA
San Diego State University
San Diego, CA, USA
alzucker@ucsd.edu

Ashley L. Juavinett
Neurobiology, UC San Diego
La Jolla, CA, USA
ajuavine@ucsd.edu

ABSTRACT

As programming skills become more demanded in fields outside of computer science, we need to consider *how* we should be teaching these skills to our students. One option is to encourage students to pursue introductory computer science courses; however, these courses are often geared towards computer science (CS) majors and without important discipline-specific context. Other avenues include short coding modules within disciplinary courses or full courses that blend CS with another discipline. Guided by insights from an introductory CS course in the context of biology, we describe a key tension when coding meets biology: while contextualized programming classes are often perceived as more accessible, students may also view them as less authentic. Taken together, these observations point to specific recommendations for educators who choose to integrate coding and biology in this way. Ultimately, we conclude that discipline-specific programming education is essential to improve equity in computing education.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; **CS1**; *Computing literacy*; **Computing education**; **CS1**; • **Applied computing** → **Life and medical sciences**; **Life and medical sciences**.

KEYWORDS

Contextualized coding, CS1, Computing in Biology, Equity & Inclusion

ACM Reference Format:
Austin L. Zuckerman and Ashley L. Juavinett. 2024. When Coding Meets Biology: The Tension Between Access and Authenticity in a Contextualized Coding Class. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20–23, 2024, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3626252.3630966

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike International 4.0 License.

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630966

1 THE NEED FOR COMPUTING EDUCATION IN BIOLOGY

In the 21st century, programming is a skill required not only for computer scientists, but for a wide range of professionals spanning many different disciplines [28, 30, 38]. In biology in particular, many have pointed to the increasing need for computational approaches to analyze large datasets and construct complex models of biological systems [2, 15, 27]. However, undergraduate curricula that include adequate computational training for life sciences majors remain elusive, especially for students who pursue such majors without a computational specialization.

Proposed reasons for the mismatch between the demand and inclusion of computational skills in biology curricula include instructors' lack of experience, barriers to programming tools, and challenges in identifying discipline-specific programming needs [19, 37]. Further, there is limited research about the experiences, dispositions, and outcomes of students who journey into opportunities to bridge coding and biology. And so, it is unclear *how* we should be teaching non-computer science (non-CS) majors how to code: should we invite them into computer science (CS) courses, or should we integrate CS education into other curricula? In this experience report, we draw from our experience teaching a biology CS1 to identify specific considerations when integrating programming and biology, which can offer transferable implications for other non-CS fields.

1.1 Considerations when integrating programming & biology

There are several considerations when inviting non-CS majors into computer science spaces. Decades of research have shown that even students *choosing* to major in CS have preconceived notions about what it means to learn computer science and who can be a programmer [7, 14]. For example, one of the most pervasive stereotypes of programmers – even now – depicts males with singular focus coupled with an asocial, competitive personality [6, 24]. These stereotypes are especially held by women and students from minoritized backgrounds because of their continued exclusion in these fields [8, 14]. Field-specific stereotypes interact with self-identity effects such as stereotype threat, which can hinder students' self-efficacy, sense of belonging, persistence in a field, and ultimately, their academic performance [17, 21].

In addition to these stereotypes about CS, biology students may conflate mathematics and computing, further hindering the participation of students who are math-averse [1, 5, 34]. Women and first

generation students especially have negative task values (which summarize interests and perceptions) for mathematics in a biology context [1]. Our biology students are therefore especially likely to hold such notions, given that many of them chose their majors instead of computational or mathematical majors. When we invite students into CS spaces, we therefore need to acknowledge and directly address the mindsets students bring with them.

A second barrier in encouraging non-CS majors to develop CS skill sets is access to relevant coursework. Depending on the university, non-CS majors may not be able to enroll in introductory computer science (CS1) courses: CS1 may be restricted to computer science or engineering majors or have limited spots for non-majors. Further, in some cases these courses have prerequisites that non-majors do not have or may not be counted as courses towards a student's non-CS major. Non-CS majors can pursue informal coding opportunities (such as workshops or bootcamps), but these may not afford the same professional development or spring boarding into future computational learning or career paths. Students, colleagues, and future employers also may not recognize opportunities outside of the university setting as legitimate [33]. The design of our curricula therefore communicates a specific message about who can and should access these skills.

1.1.1 Course vs. Modular Approaches. In response to these challenges with bringing non-CS students into computing education, some educators have developed discipline-specific coding opportunities. One notable example is a biology-based CS1 (CS1-B) at Harvey Mudd College, which has been taught for over a decade and has demonstrated that contextualized coding equally prepares students for future computing coursework [11]. There is also an important story about gender equity: CS1-B students are more likely to be female than those in non-biology CS1, demonstrating that contextualized coding courses recruit student populations not served by traditional courses [12]. Dodds et al. (2021) argue that computing is a literacy and means of inquiry that should be shared by all, and that disciplines such as biology can preserve their identity while integrating computing. Yet despite the strong example set by CS1-B, such interdisciplinary courses are rare.

Other educators have pioneered more modular inclusion of computing in biology curricula, predominantly through hands-on laboratory courses that involve computing or bioinformatics modules, which naturally lend themselves to introductions to computing [9, 18, 25, 31]. For example, Custer et al. (2021) implemented an introductory tutorial to statistical programming using R across a variety of courses, including for students majoring in natural sciences and found that students' perceptions of R became more favorable after completing the tutorial [9]. Similarly, Madlung (2018) found that biology students valued a module with applied bioinformatics and were interested in receiving such instruction earlier in their undergraduate biology education [25]. However, students desiring a deeper, more thorough introduction to computing will still need access to full courses that cover the primary content of CS1.

2 A CS1 IN THE CONTEXT OF BIOLOGY

Inspired by CS1-B and other contextualized computing courses, we developed an introductory Python course in biology that was accessible to non-CS majors. In this section, we provide details about the course to contextualize our insights into the following research objectives:

- Q1: Is course enrollment more diverse than would be expected based on enrollment in CS majors at our university?
- Q2: Do students perceive an CS1 course in biology as more accessible than a non-biology CS1?
- Q3: How do students perceive computing skills in a biology context?

2.1 Course Context

"Introduction to Python for Biologists" is taught in a biology department at a large research-intensive (R1) public institution where CS1 course enrollments are typically in the hundreds. The course is open to non-CS majors with no prerequisites and has approval as an elective course for a variety of non-CS majors.

2.1.1 Participants. There have been three offerings of this course in the past two years (2022-2023), with 42, 80, and 90 students in each offering. This course is currently taught by faculty in the biology department, one with degrees in neuroscience and the other with degrees in physics. While this course focused on teaching applications of Python in biological sciences, students were not required to be enrolled as biology majors (about half were, and many other students were chemistry/biochemistry, environmental science, or cognitive science majors). Certain colleges at our university require computing courses, but not all. Student demographics reported below were collected in a pre-course survey that was not linked to student grades. Approximately 95% of students filled out this survey. Institutional demographic data reported below was obtained from http://ir.ucsd.edu and represents the 2021-22 school year.

2.1.2 Learning Objectives & Course Content. The learning objectives for the course were as follows:

- Read, write, edit, and run basic Python programs in both Jupyter Notebooks and the command line, recognizing the structures used (i.e. variables, conditionals, loops, functions) and explaining how they work
- Manipulate and create objects in Python, including data structures and classes
- Visualize and analyze simple datasets in Python
- Implement common algorithms for analyzing biological data (e.g., time series, images)

The 10-week course covers similar content to a typical CS1, but with several notable exceptions (Table 1). First, wherever possible, code examples are related to biology. For example, string slicing is done with DNA or RNA sequences – for example, finding three-letter strings in a longer string of nucleotides ('GCTGTCAGTC') – to illustrate the relevance of this skill to bioinformatics datasets. Second, there is a strong emphasis on data wrangling, analysis, and visualization in the second half of the course, as these are key skills for biology researchers. Lastly, the final weeks of the course cover applications that are specific to biology, such as relevant Python packages for scientific analysis and topics such as image processing and time series analysis. Although the second half of the course is biology application-based, it also serves to reiterate concepts from the first half – for example, in the neuroinformatics

Table 1: Course Overview

Week	Topic
1	Course introduction, tools, Python syntax
2	Data structures (lists, tuples, and dictionaries), conditionals, and functions
3	Loops
4	Object-oriented programming
5	Scientific computing (NumPy & Data visualization)
6	Data analysis in Pandas
7	Review and preparation for final projects
8	Signal and image processing in Python
9	Neuroinformatics
10	Documentation, version control, code collaboration

module, students need to navigate nested data structures and write functions. This course was built with biology majors in mind and was not intended to serve the needs of students who are continuing into CS education. It is, however, approved as a substitute course for other introductory Python courses in non-CS majors.

Most of the course content is delivered via partially-completed Jupyter Notebooks which students complete during interactive live-coding class sessions. All of the course content can be found on GitHub (www.github.com/BILD62). Educators who wish to know more about course specifics are encouraged to reach out to the corresponding author with questions or suggestions.

2.1.3 Assessment. Throughout the quarter, students complete weekly problem sets which constitute about half of their grade. Students are tested on their fundamental programming knowledge with a multiple choice open-notes exam featuring Parson's problems and questions requiring students to describe several lines of code in their own words. At the end of the course, students complete a final group project: either a small program (e.g., a biology-inspired Wordle game) or an analysis workflow (e.g., analyzing data collected in their research lab or a publicly accessible dataset).

2.2 Focus Groups

To address our research questions, we conducted focus group interviews with 22 students (3-5 per group) in three different quarters of instruction. These interviews were conducted towards the end of the course, engaging students in both summative and retrospective reflection. Focus groups were conducted by the first author and within a larger mixed-methods study of student attitudes, which included pre- and post-surveys (not discussed here). The focus group protocol expanded upon the survey's open-response questions and broadly explored students' attitudes towards programming. Transcripts were first iteratively read and then analyzed using an inductive coding scheme. Quotes from these focus groups are included here to provide student voice to our observations.

2.3 Positionality Statement

A. Zuckerman is a PhD student who operated independently from the course and was primarily responsible for overseeing the collection and analysis of the focus groups. He does not teach this course in any capacity. A. Juavinett was the instructor for two iterations of the course. Her experiences teaching the course and interacting with students provided additional anecdotal evidence that complemented the insights from the focus groups.

3 DISCIPLINE-BASED PROGRAMMING IMPROVES ACCESS

One of the primary motivations for developing an introductory computing course in biology was to provide access to these skills for non-CS majors. After multiple course iterations, we have noted that our course enrollment is more diverse than the enrollment in typical engineering and data science courses at our university. These numerical observations are supported by student comments.

3.1 More diverse course enrollment and higher retention compared to CS fields

As observed in other discipline-based computing courses [12], we noted a high enrollment 60%) of female students in our course. As seen in Figure 1, these numbers are particularly striking when compared to the number of female students in the computer science (24%) or data science (29%) degree recipients at our university, and are on par with how many biology degree recipients identify as female (65%) and the number of female students in life sciences more broadly (60%; Figure 1). For broader context, in the United States and Canada, about 22% of doctoral degree recipients in CS are female [39], and at our same institution, CS1 enrollment is typically 19-31% female, depending on whether students are in a "prior experience" track or not [32]. It is clear from this data that discipline-based coding, perhaps particularly in biology (which has many female students), can improve participation for women despite gendered expectations of coding classes [24].

We also observed higher rates (24%) of students from underrepresented racial groups (URM; includes any student who did *not* identify as either White or Asian American or did not include a response) than those who received degrees in either Computer Science & Engineering (11%; see also [32] for CS1 data) or Data Science (14%; Figure 1). Further, our course enrollment is more aligned with Life Sciences major enrollment than Computer Science (Figure 1, bottom). It should be noted that participation rates for URM students in CS doctoral programs are even lower than the undergraduate statistics shown here (3.9% of CS doctoral recipients in 2021-22;[39]).

Finally, course retention can speak directly to the accessibility of a course. Indeed, this course also had very high retention across all three quarters (92-98%), unlike what is typical for CS1 courses, which are often in the ballpark of 60-80% [32]).

3.2 Perceived accessibility

Such demographic data raises an important question: *why* might a discipline-based coding class be more accessible? Student feedback, obtained via focus groups, provides several key insights here.

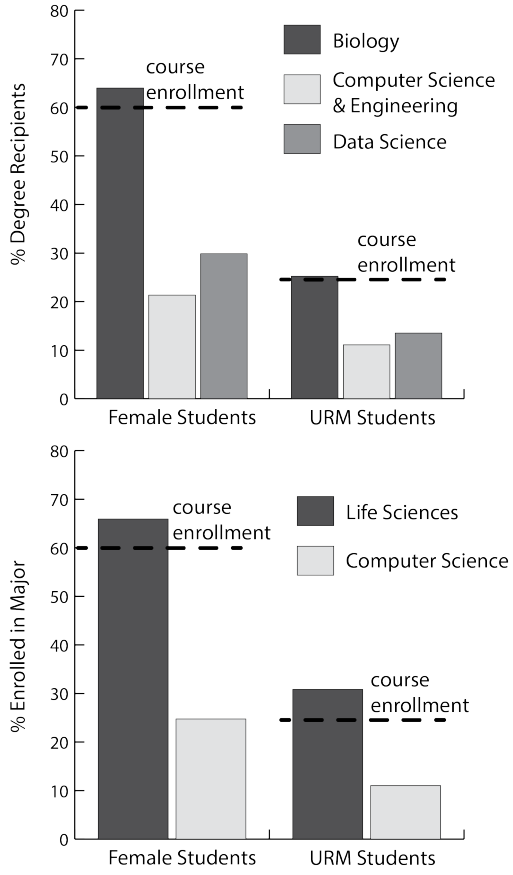

Figure 1: Degree recipients (top) and major enrollments (bottom) by gender and race/ethnicity at our institution. Black dotted line indicates % of female and URM students enrolled in our course, for comparison. Majors chosen represent the closest to those of interest that were available in institutional data; "Computer Science & Engineering" also includes all fields of engineering. Underrepresented minorities (URM) include African American/Black, American Indian/Alaska Native, Chicanx/Latinx, Native Hawaiian/Pacific Islander.

First, participants identified this course as an opportunity to gain exposure to fundamental programming skills, yet there were variations in their descriptions regarding the perceived difficulty or accessibility of the course. Many participants expressed uncertainty about the course content, but attended to certain aspects of the course title, "Introduction to Python for Biologists," when formulating their impressions about the course's difficulty and accessibility. The course's focus on teaching both fundamental programming skills and how these skills could be directly applied in biological applications was particularly inviting to some students:

> I felt like it was a little too late to start, since every coding class I thought about joining had a bunch of other people who have started since like high school or middle school. But this one was called, like, Intro to code, but also for specifically, like bio. And I thought that was just very welcoming.

This class was designed to specifically recruit biology students, and many students were biology majors. We were therefore not surprised that many students highlighted the fact that elements of the class directly linked to biology: "[The nicest thing] is when we would learn like the basics in [Jupyter] notebooks and then later on, we'd see them in the context of a biology problem and like something you might see in a lab." In another student's words, "Because I'm a bio major, and it's bio-centered, I thought it seems a little bit more accessible than some of the other coding classes."

Some participants were expecting the course to be manageable because this course specifically focused on learning Python, which they felt was a more accessible and transferable programming language. One participant stated that they "understood Python [as] the easiest coding language to learn, so [they were] a little less intimidated by this class." In contrast, other participants stated that they expected learning Python to be difficult based on a previous programming experience, stating that "I had taken math, and part of the class was to learn Matlab, which is another language, and that language was really infuriating because I felt like the syntax was very picky…I thought that Python would be the same."

In addition, we learned that many students were pleased that aspects of the class were immediately rewarding and relevant to other coursework, particularly data analysis and visualization:

> I didn't think I'd be able to utilize Python packages quite the way that we have, like, with Matplotlib I can generate a plot. And that's something that I can actually already go and take into like my recombinant DNA techniques class. I wasn't expecting to already have things that I could visualize and use that quickly.

Another student linked the content to their identity as a learner: "I'm a visual learner, so I think, like you, says I'm able to create something that can visualize and like actually understand."

In sum, the foregrounding of biology applications seems to have recruited many students into the class and helped retain them by demonstrating the immediate value of computing skills.

3.3 Perceptions of a normalized error climate in the classroom

In addition to the perceived accessibility of the course, high retention may also be attributed to student perceptions of a learning culture that explicitly normalized errors as a productive part of the coding process. For example, one participant recalled the course instructor using the internet in real time to troubleshoot coding errors. This observation allowed them to see that a universal understanding of programming concepts was not required to engage in the coding process.

> I think like watching the even the Professor like mess up in class and being like, oh, it's okay, I still understand what I'm doing, even though I messed up my one line of code and like watching them look up things online and being like, oh, you know it's not that I know everything. [I now] feel confident and understand how to troubleshoot.

As another participant described, the course experience gradually improved their confidence with the coding process, especially as seeking assistance became a normalized part of this process.

> At the beginning, I was very scared. I limited myself in a lot of the coding process, because I was scared of the error messages. So as the weeks went on, and then I started doing more assignments, I kind of became okay with that error message and I started to like to see my coding, my code in different ways, in order to like know why each line was doing and then if there was an error in that line, how can I fix it by myself and if I couldn't fix it by myself, then you know, ask for help.

These comments indicate that the error climate of this course was often perceived to be positive and welcoming, especially due to the collaborative nature of the course. The perceived accessibility that inspired students to enroll in the course was overall sustained as students engaged with the course material and learned to persist through challenges in the coding process.

4 IS CODING IN BIOLOGY CONTEXTS "REAL" CODING?

One unexpected observation from our focus groups is that students were grappling with whether the flavor of coding learned in our course was "real" coding, akin to the kind of programming learned in a typical CS1 course. In some instances, students viewed coding in biology as being a different, more applied form of coding from other types of coding. Part of this skepticism came from the course materials. Several students cited the fact that the course was taught in Jupyter Notebooks, which they did not see as "real" coding:

> In the sense of like just using Jupyter notebooks which are almost like a little bit, I think they feel almost like separate from computer coding because they are like a separate form or place to code.

Another point of skepticism came from the focus on data analysis, which some students did not see as synonymous with coding, even though it involved writing code:

> I mean the coding is still there it's just I thought it'd be more emphasis on the coding but this has been more emphasis on what these things do in relation to data or something like that.

While students acknowledged that basic familiarity with coding is becoming increasingly essential as a highly demanded skill across many disciplines, they also perceived that coding skill sets are narrower or more specialized in biology. As one participant noted, the scope of coding techniques used by professionals in the biotechnology industry are narrower and more specialized than in other programming disciplines:

> I think in biotech stuff it's usually going to be, I think a lot simpler because there's usually just small amounts of code needed to do certain tasks... [C]ompared to like what they do in Google, like fixing bugs and stuff, you don't really have to deal with that as much because I feel like you're just writing code to do tasks for you, you don't really have to worry about having like something that other people could then work with and all that stuff.

Another participant described that "with biology, it's mostly working with stuff that's already premade or at least learning how to use functions and packages, rather than entirely making something new," indicating a perception that coding is more scaffolded when used in biological applications. These student comments suggest that we should think critically about how we portray authenticity in applied CS contexts.

5 LESSONS LEARNED FOR FUTURE BIOLOGY & CODING COURSES

As the above data and observations suggest, teaching coding in a biology context may lead to two slightly opposed outcomes. On one hand, contextualized coding may serve a broader population of students, but on the other hand, students may also perceive that their learning is less authentic.

5.1 On the question of authenticity

In contemplating the authenticity of computing in the course, students made several accurate observations: most biologists who use code are indeed using packages written by others, often running scripts with small modifications. In the course, we used partially-completed Jupyter Notebooks, which may also simulate a biology researcher receiving a data analysis notebook from a colleague. However, the observation that students perceived data analysis with code as qualitatively different than "coding" is alarming, though in line with other observations about concerns about the legitimacy of other modes of computational education, such as more visual, block-based coding [35]. In a recent study on the use of Code.org block-based coding, one student response particularly resonated with our observations: "It was fun, yet it didn't actually teach me code" [13]. Such beliefs may undermine the stated goals of the course to improve student attitudes towards coding and ultimately accessibility; although we did not assess this directly, we speculate based on other research that the perception of an illegitimate programming education could negatively impact their self-identity as a coder and role as a legitimate peripheral participant in a computational community [16], the opposite effect of incorporating authentic learning into STEM contexts [4].

While further research is undoubtedly warranted to track how well our course (compared to traditional CS1) prepares students for future computational work as well as how persistent this skepticism about the legitimacy of coding in this context is, we feel it was a clear enough signal in our observations to permit a recommendation to other educators: namely, this skepticism needs to be addressed, in one of several ways. First, instructors could highlight the actual work of professional data scientists or biology researchers and ensure course materials are aligned with the community of practice that we are preparing students to join [16, 23]. Instructors could, for example, demonstrate the typical code that a biology researcher may encounter, illustrating that indeed much of their time is spent working with pre-written packages or wrangling figures.

Further, our insights here dovetail with work in biology education which underscores the importance of authenticity in coursework, particularly lab courses and those that strive to improve data

literacy [4, 18, 20, 26]. Likewise, in contextualized coding, students should see and work with real data sets and packages, perhaps even replicating analyses and figures in published scientific work. To this end, the final project in this class was designed so that students could, if they chose, work with real-world data. Indeed, several students chose to work with data from their own research labs or publicly-available datasets, such as ones related to COVID-19.

5.2 Additional takeaways

In addition to the specific insights about perceived accessibility and authenticity of coding in the context of biology, we have gained several broadly applicable insights into developing these courses.

First, course title matters. In student comments, several highlighted the fact that our course title included the word “introduction.” Students are keen on these details – “introduction” means a course for beginners, and likely without any prerequisites. Similarly, setting a welcome, inclusive tone for all learners on the first day of class will reiterate the introductory nature of the course [22].

A second consideration is institutional context. Our course is taught at a large university to a student body that contains many students on medical tracks. Students are accustomed to lecture-based classes in which they may never interact with another student and may indeed be in direct competition with the students around them for spaces in desirable medical programs. These cultural factors directly influence the classroom environment. In this class, we implemented live coding and normalized making mistakes, even in front of other students, a considerable shift in culture from other classes but a necessary one for overcoming preconceived notions about the isolating nature of coding. Institutions with more intimate classroom cultures may find it much easier to encourage collaboration in introductory coding courses.

Further, instructors should carefully consider the inclusion of math (or math prerequisites) in introductory CS courses for non-CS majors. Although mathematics is often thought of as a prerequisite for skill building in programming, on the contrary programming may actually improve students’ self-efficacy and performance in mathematics [29]. Although mathematics and computer science share a history and *should* have considerable cross talk, many life sciences students are deeply math-averse and wrongly believe that math and programming are identical skill sets [34]. One strong signal from the literature is that if math is introduced in computer science, it should be clearly contextualized [3].

In designing our course, we grappled with the question of how much mathematics should be taught, especially if we are introducing algorithms that are commonly used with biological data. For example, signal filtering is handy for the processing of time series, and to teach filtering by convolution, instructors can choose graphical or mathematical means. In most cases, we feel that a graphical intuition is sufficient for an introductory CS course, and that this will leave students encouraged to pursue additional, more mathematically-grounded coursework (a necessary component of a thorough biology education), rather than discouraged by the inclusion of topics they were not expecting.

5.2.1 When biology meets generative AI. As we write this, computing education is rapidly changing in the wake of chatGPT and other large language models. The wide availability of these tools will (and should) change computational education, as many have noted [10]. These tools also have specific ramifications for contextualized coding. Students in all disciplines struggle with syntax and are often frustrated with the detail-oriented nature of writing code. Generative AI tools can write code with clear instructions and explain small errors to students, potentially making writing code *de novo* obsolete in coding education. While this means that many of us will need to re-write our assignments and exams such that they are less likely to be solved by generative AI, it also opens a welcome opportunity to teach more than just syntax.

Computational thinking, which involves "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science," is more than just replicating syntax [28, 38]. Courses like ours can now spend less class time on colons and more time on code design and critical thinking. When we’re not confused as to why our sequence slice is ['ATG'] instead of 'ATG', we can ask higher-level questions like "how do we design robust code for sequences of different lengths?" In other words, we can move away from programming and towards computational thinking.

As noted above, many students – especially those who have been excluded in computational fields – approach CS topics with apprehension and preconceived notions. It is possible that generative AI may serve as a "mediating tool" that can help students engage in authentic coding problems, and therefore *should* be incorporated into introductory contextualized coding education [36]. However, considering the concerns about perceived authenticity, we also need to be sure to tell students that relying on generative AI for certain tasks is appropriate and that such tools are indeed becoming commonplace in the professional workforce as well.

5.3 Conclusion

Contextualized coding courses such as ours serve a great proportion of students that are typically underrepresented in computational fields. Therefore, such classes can serve as important curricular structures for increasing access and equity in computing education. With these insights, we can adapt discipline-based coding courses to empower students to see their learning experiences as legitimate. We hold that it is entirely feasible to teach a contextualized coding class that is perceived as wholly legitimate and will thoroughly prepare students for an evolving economy and scientific workforce. Doing so can help ensure that we are not leaving students behind.

ACKNOWLEDGMENTS

We are thankful to Philip Guo and Catherine Hicks for their feedback on this manuscript. We are also grateful to Shannon Ellis and Tom Donoghue for sharing materials which were adapted for this course.

REFERENCES

[1] Sarah E. Andrews and Melissa L. Aikens. 2018. Life Science Majors’ Math-Biology Task Values Relate to Student Characteristics and Predict the Likelihood of Taking Quantitative Biology Courses. *Journal of Microbiology & Biology Education* 19, 2 (Jan. 2018), 19.2.60. https://doi.org/10.1128/jmbe.v19i2.1589

[2] Teresa K Attwood, Sarah Blackford, Michelle D Brazas, Angela Davies, and Maria Victoria Schneider. 2019. A global perspective on evolving bioinformatics and data science training needs. *Briefings in Bioinformatics* 20, 2 (March 2019), 398–404. https://doi.org/10.1093/bib/bbx100

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

[4] Margaret E. Beier, Michelle H. Kim, Ann Saterbak, Veronica Leautaud, Sandra Bishnoi, and Jaqueline M. Gilberto. 2019. The effect of authentic project-based learning on attitudes and career aspirations in STEM. *Journal of Research in Science Teaching* 56, 1 (2019), 3–23. https://doi.org/10.1002/tea.21465

[5] Alicia M. Caughman and Emily G. Weigel. 2022. Biology Students' Math and Computer Science Task Values Are Closely Linked. *CBE—Life Sciences Education* 21, 3 (Sept. 2022), ar43. https://doi.org/10.1187/cbe.21-07-0180

[6] Sapna Cheryan, Allison Master, and Andrew N. Meltzoff. 2015. Cultural stereotypes as gatekeepers: increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology* 6 (2015). https://www.frontiersin.org/articles/10.3389/fpsyg.2015.00049

[7] Sapna Cheryan, Victoria C Plaut, Paul G Davies, and Claude M Steele. 2009. Ambient Belonging: How Stereotypical Cues Impact Gender Participation in Computer Science. *Journal of Personality and Social Psychology* 97, 6 (2009), 1045–1060. https://doi.org/10.1037/a0016239

[8] Sapna Cheryan, Victoria C. Plaut, Paul G. Davies, and Claude M. Steele. 2009. Ambient Belonging: How Stereotypical Cues Impact Gender Participation in Computer Science. *Journal of Personality and Social Psychology* 97, 6 (2009), 1045–1060. https://doi.org/10.1037/a0016239

[9] Gordon F. Custer, Linda T. A. van Diepen, and Janel Seeley. 2021. Student perceptions towards introductory lessons in R. *Natural Sciences Education* 50, 2 (2021), e20073. https://doi.org/10.1002/nse2.20073

[10] Marian Daun and Jennifer Brings. 2023. How ChatGPT Will Change Software Engineering Education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ACM, Turku Finland, 110–116. https://doi.org/10.1145/3587102.3588815

[11] Zachary Dodds, Ran Libeskind-Hadas, and Eliot Bush. 2012. Bio1 as CS1: evaluating a crossdisciplinary CS context. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. ACM, Haifa Israel, 268–272. https://doi.org/10.1145/2325296.2325360

[12] Zachary Dodds, Malia Morgan, Lindsay Popowski, Henry Coxe, Caroline Coxe, Kewei Zhou, Eliot Bush, and Ran Libeskind-Hadas. 2021. A Biology-based CS1: Results and Reflections, Ten Years in. In *SIGCSE 2021 - Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, Vol. 21. Virtual Event, 796–801. https://doi.org/10.1145/3408877.3432469

[13] J Du, H Wimmer, and R Rada. 2016. "Hour of Code": Can it change students' attitudes toward programming? *Journal of Information Technology Education: Innovations in Practice* 15 (2016), 52–73.

[14] Allan Fisher and Jane Margolis. 2003. Unlocking the clubhouse: women in computing. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03)*. .23. https://doi.org/10.1145/611892.611896

[15] William Grisham, Barbara Lom, Linda Lanyon, and Raddy L. Ramos. 2016. Proposed Training to Meet Challenges of Large-Scale Data in Neuroscience. *Frontiers in Neuroinformatics* 10 (July 2016), 1–6. https://doi.org/10.3389/fninf.2016.00028

[16] Mark Guzdial and Allison Elliott Tew. 2006. Imagineering inauthentic legitimate peripheral participation: an instructional design approach for motivating computing education. In *Proceedings of the second international workshop on Computing education research*. ACM, Canterbury United Kingdom, 51–58. https://doi.org/10.1145/1151588.1151597

[17] Aniko Hannak, Kenneth Joseph, Daniel B. Larremore, and Andrei Cimpian. 2023. Field-specific ability beliefs as an explanation for gender differences in academics' career trajectories: Evidence from public profiles on ORCID.Org. *Journal of Personality and Social Psychology* (June 2023). https://doi.org/10.1037/pspa0000348

[18] Ashley Juavinett. 2020. Learning How to Code While Analyzing an Open Access Electrophysiology Dataset. *The Journal of Undergraduate Neuroscience Education* 19, 1 (2020), 94–104.

[19] Ashley L. Juavinett. 2022. The next generation of neuroscientists needs to learn how to code, and we need new ways to teach them. *Neuron* 110, 4 (Feb. 2022), 576–578. https://doi.org/10.1016/j.neuron.2021.12.001

[20] Melissa K. Kjelvik and Elizabeth H. Schultheis. 2019. Getting Messy with Authentic Data: Exploring the Potential of Using Data from Scientific Research to Support Student Data Literacy. *CBE—Life Sciences Education* 18, 2 (June 2019), es2. https://doi.org/10.1187/cbe.18-02-0023

[21] Sophia Krause-Levy, William G. Griswold, Leo Porter, and Christine Alvarado. 2021. The Relationship Between Sense of Belonging and Student Outcomes in CS1 and Beyond. In *Proceedings of the 17th ACM Conference on International Computing Education Research (ICER 2021)*. Association for Computing Machinery, New York, NY, USA, 29–41. https://doi.org/10.1145/3446871.3469748

[22] A. Kelly Lane, Clara L. Meaders, J. Kenny Shuman, MacKenzie R. Stetzer, Erin L. Vinson, Brian A. Couch, Michelle K. Smith, and Marilyne Stains. 2021. Making a First Impression: Exploring What Instructors Do and Say on the First Day of Introductory STEM Courses. *CBE—Life Sciences Education* 20, 1 (March 2021), ar7. https://doi.org/10.1187/cbe.20-05-0098

[23] Jean Lave and Etienne Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, Cambridge, MA. https://doi.org/10.1017/CBO9780511815355

[24] Colleen M. Lewis, Ruth E. Anderson, and Ken Yasuhara. 2016. "I Don't Code All Day": Fitting in computer science when the stereotypes don't fit. In *ICER 2016 - Proceedings of the 2016 ACM Conference on International Computing Education Research*. Association for Computing Machinery, Inc, New York, NY, USA, 23–32. https://doi.org/10.1145/2960310.2960332

[25] Andreas Madlung. 2018. Assessing an effective undergraduate module teaching applied bioinformatics to biology students. *PLOS Computational Biology* 14, 1 (Jan. 2018), e1005872. https://doi.org/10.1371/journal.pcbi.1005872

[26] Irina Makarevitch, Cameo Frechette, and Natalia Wiatros. 2015. Authentic research experience and "big data" analysis in the classroom: Maize response to abiotic stress. *CBE Life Sciences Education* 14, 3 (2015), 1–12. https://doi.org/10.1187/cbe.15-04-0081

[27] Florian Markowetz. 2017. All biology is computational biology. *PLOS Biology* 15, 3 (March 2017), e2002050. https://doi.org/10.1371/journal.pbio.2002050

[28] Dave Mason, Irfan Khan, and Vadim Farafontov. 2016. Computational Thinking as a Liberal Study. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, Memphis Tennessee USA, 24–29. https://doi.org/10.1145/2839509.2844655

[29] Sarantos Psycharis and Maria Kallia. 2017. The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science* 45, 5 (Oct. 2017), 583–602. https://doi.org/10.1007/s11251-017-9421-5

[30] Sarah Monisha Pulimood, Kim Pearson, and Diane C. Bates. 2016. A Study on the Impact of Multidisciplinary Collaboration on Computational Thinking. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, Memphis Tennessee USA, 30–35. https://doi.org/10.1145/2839509.2844636

[31] Hong Qin. 2009. Teaching computational thinking through bioinformatics to biology students. In *SIGCSE'09 - Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. ACM Press, New York, New York, USA, 188–191. https://doi.org/10.1145/1508865.1508932

[32] Adrian Salguero, Julian McAuley, Beth Simon, and Leo Porter. 2020. A Longitudinal Evaluation of a Best Practices CS1. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. ACM, Virtual Event New Zealand, 182–193. https://doi.org/10.1145/3372782.3406274

[33] Kyle Thayer and Amy J. Ko. 2017. Barriers Faced by Coding Bootcamp Students. In *Proceedings of the 2017 ACM Conference on International Computing Education Research*. ACM, 245–253. https://doi.org/10.1145/3105726.3106176

[34] Lucas P. Wachsmuth, Christopher R. Runyon, John M. Drake, and Erin L. Dolan. 2017. Do Biology Students Really Hate Math? Empirical Insights into Undergraduate Life Science Majors' Emotions about Mathematics. *CBE—Life Sciences Education* 16, 3 (Sept. 2017), ar49. https://doi.org/10.1187/cbe.16-08-0248

[35] David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, Boston Massachusetts, 199–208. https://doi.org/10.1145/2771839.2771860

[36] James V. Wertsch. 2017. Mediation. In *Introduction to Vygotsky* (3 ed.). Routledge.

[37] Jason J. Williams, Jennifer C. Drew, Sebastian Galindo-Gonzalez, Srebrenka Robic, Elizabeth Dinsdale, William R. Morgan, Eric W. Triplett, James M. Burnette, Samuel S. Donovan, Edison R. Fowlks, Anya L. Goodman, Nealy F. Grandgenett, Carlos C. Goller, Charles Hauser, John R. Jungck, Jeffrey D. Newman, William R. Pearson, Elizabeth F. Ryder, Michael El Sierk, Todd M. Smith, Rafael Tosado-Acevedo, William Tapprich, Tammy C. Tobin, Arlin Toro-Martinez, Lonnie R. Welch, Melissa A. Wilson, David Ebenbach, Mindy McWilliams, Anne G. Rosenwald, and Mark A. Pauley. 2019. Barriers to integration of bioinformatics into undergraduate life sciences education: A national study of US life sciences faculty uncover significant barriers to integrating bioinformatics into undergraduate instruction. *PLOS ONE* 14, 11 (Nov. 2019), e0224288. https://doi.org/10.1371/JOURNAL.PONE.0224288

[38] Jeannette M. Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. https://doi.org/10.1145/1118178.1118215

[39] Stuart Zweben and Betsy Bizot. 2023. 2022 Taulbee Survey Record Doctoral Degree Production; More Increases in Undergrad Enrollment Despite Increased Degree Production. (2023).