Neuron

CellPress

NeuroView

The next generation of neuroscientists needs to learn how to code, and we need new ways to teach them

Ashley L. Juavinett[1,*]
[1]Division of Biological Sciences, Neurobiology Section, 9500 Gilman Drive, La Jolla, CA 92093, USA
*Correspondence: ajuavine@ucsd.edu
https://doi.org/10.1016/j.neuron.2021.12.001

Neuroscience education is at an impasse—we need to teach students coding, but many institutions do not have the resources to do so. Here, I outline three major barriers, as well as solutions, to bringing programming education into our undergraduate and graduate programs.

As many researchers are keenly aware, neuroscience data are growing in both size and complexity. Each year we record from more and more neurons simultaneously and build machine-learning models with larger training sets. Further, many labs build bespoke rigs driven by custom code, and trainees are often implicitly expected to know how to run and edit it. As a result, there have been numerous calls over the years to increase the amount of coding exposure in neuroscience education, echoing the calls for bioinformatics training in response to the "-omics" era of biology (Goldman and Fee, 2017; Grisham et al., 2016; Pevzner and Shamir, 2009).

Beyond neuroscience research, coding is increasingly expected in many different career paths. More and more job sectors require programming experience, and the jobs that do pay more. Teaching coding to a diverse generation of students has therefore become not only a matter of practical skill building but also a matter of equity.

Despite the undisputable importance of programming knowledge for future neuroscientists, courses that teach coding are rarely offered in neuroscience degree programs. In a recent study of undergraduate neuroscience majors, only six of 118 institutions required computer science classes, and an additional three schools offered computer science as an elective (Figure 1C; Pinard-Welyczko et al., 2017). At the graduate level, only 15% of neuroscience PhD programs require programming, and 55% include programming as an elective (Figure 1D; Society for Neuroscience, 2017).

Here, I identify three possible reasons for the mismatch between how much coding is needed and how much is taught. With recognition of informal coding education programs and an increase in institutional resources to both hire interdisciplinary instructors and train current instructors, there is hope for quickly closing this gap.

Instructor knowledge of coding

Before offering more coding coursework, institutions need the requisite expertise and teaching capacity. In 2019, I asked practicing neuroscientists in many different career paths whether they had taken coding classes in high school, college, or graduate school. Many individuals who are now data scientists or industry researchers report taking a coding class in college, but these are typically not the people teaching coding or computer science in a university setting. In contrast, less than half of current faculty surveyed had ever taken a coding class or felt comfortable working with code (Figures 1A and 1B). Still, faculty expertise runs the whole gamut, with about a third reporting that they currently feel "a little" or "moderately" comfortable with coding.

These numbers are likely to shift in coming years, as there is clearly a trend of more graduate students and postdoctoral fellows taking coding classes (Figure 1A). About 68% of current neuroscience graduate students that were surveyed are taking coding classes at about the same rate as they are offered in PhD programs (Figure 1D). Even still, this shift will take time and will not address the present need for more instructors who can code. Further, an uptick in individuals who have taken a coding class does not ensure that these individuals have the pedagogical training to convey these skills to students.

Meeting the need for more coding instructors may require training current instructors who have interest in such professional development, an approach that has been successful at K–12 levels (Yadav et al., 2017). Although most instructors who currently teach biology or neuroscience classes have never taken any formal coding courses, many are self-taught coders who can teach entry-level classes or integrate coding lessons into neuroscience courses when given the time and resources to expand their coding knowledge. There is also a wealth of pedagogical knowledge around best practices in teaching coding, such as pair programming, but instructors need time and training themselves to grow their own confidence implementing such techniques (Porter et al., 2013).

To help instructors reach a level of coding at which they feel comfortable teaching others, institutions can offer incentives for instructors to participate in bootcamp style or online coding courses. There are many informal, online degree programs in which students and educators alike can gain coding experience, possibly circumventing the need for all biology departments to offer such courses. Beginning coders can learn via a number of online courses such as the Carpentries (https://software-carpentry.org/), which have a track record of embedding in

Please cite this article in press as: Juavinett, The next generation of neuroscientists needs to learn how to code, and we need new ways to teach them, Neuron (2021), https://doi.org/10.1016/j.neuron.2021.12.001

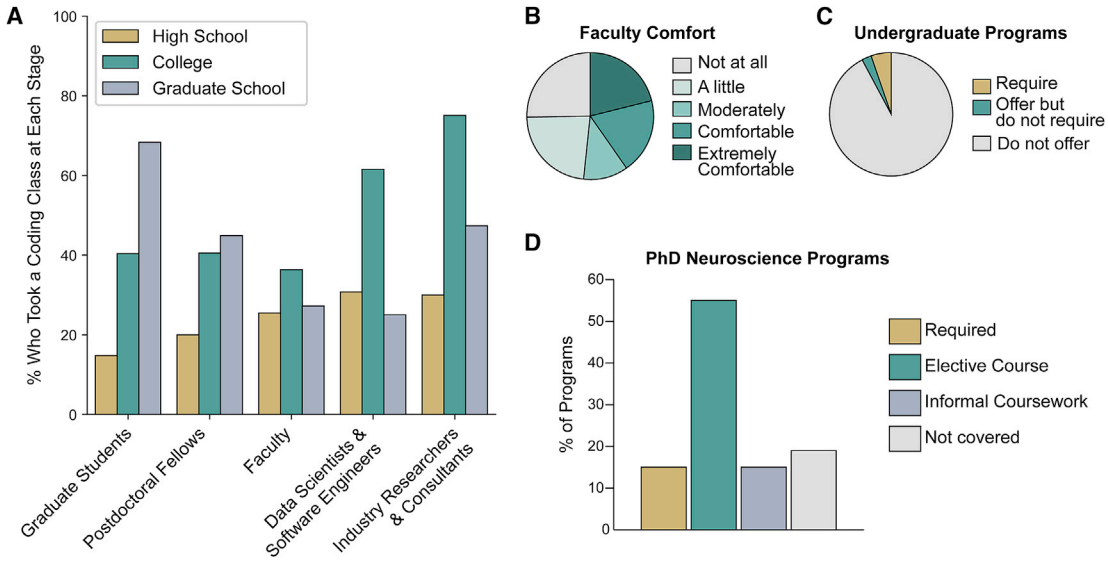

Figure 1. Neuroscience coding experience and course offerings across degree programs

(A) Responses to the question "Did you take any coding, programming, or computer science classes in {high school, college, graduate school}?" for current neuroscience graduate students (n = 116), postdoctoral fellows (n = 80), faculty (n = 55), data scientists and software engineers (n = 13), and industry researchers and consultants (n = 20).

(B) Faculty responses to the question "In general, how comfortable do you feel working with code at this point in your career?" Possible responses were "Not at all (I did not write or work with any code)," "A little (I used and occasionally edited other people's code)," "Moderately (I could easily edit other people's code and occasionally wrote my own code)," "Comfortable (I consistently wrote my own code)," and "Extremely comfortable (I consistently wrote my own production-level code)." Data for (A) and (B) were collected in 2019 by surveying individuals across more than 200 institutions via a Qualtrics survey distributed via email and social media under IRB #191035.

(C) Computer science course offerings in undergraduate neuroscience degree programs. Data are from Pinard-Welyczko et al. (2017).

(D) Computer programming course offerings in PhD neuroscience programs. Data are in response to the question "In addition to your technical neuroscience content, how does your program provide training in the following topics? Select all that apply." where respondents could choose "Required Coursework," "Elective Courses," "Informal Coursework," or "Not covered." Data are from Society for Neuroscience (2017). These data do not capture the number of neuroscience courses that may minimally expose students to coding via activities woven into the biology content.

universities with researchers. For more experienced coders, Neuromatch Academy (NMA) offers a complete curriculum of computational neuroscience coursework for self-guided learners with a basic knowledge of Python (van Viegen et al., 2021; https://academy.neuromatch.io/).

Neuroscience departments should also increase their efforts to hire individuals with an interest in integrating coding into their classes as well as offer interdisciplinary coursework in collaboration with other campus departments. While introductory computer science classes are often impacted by high student demand, instructors teaching those classes can collaborate with those in the life sciences to develop discipline-specific coursework.

Ease of access to coding tools

Furthermore, coding languages and environments vary tremendously in their ease of access and understandability. While high-level languages such as MATLAB are accessible to native English speakers, MATLAB is very expensive without a departmental license. Furthermore, installing necessary software and packages for some coding environments can be a daunting, memory-consuming haul for students and educators alike.

Thankfully, languages such as Python and R are free to use, and Jupyter Notebooks are a user-friendly way to run teaching-oriented code (https://jupyter4edu.github.io/jupyter-edu-book/). Python in particular is extensively used in neuroscience (Muller et al., 2015). Institutions with computing resources can fairly easily set up a JupyterHub, where students can easily launch a coding environment simply from an internet browser (e.g., UC San Diego's DataHub, https://datahub.ucsd.edu/). Usefully, these environments (called Docker containers) can be configured with packages already installed and with data accessible from a global directory, removing the common concern about the multitude of necessary computing packages in Python.

Even without an institutional JupyterHub, Python notebooks can be run for free on Google Colab (https://research.google.com/colaboratory/), as NMA has extensively demonstrated, or Binder (https://mybinder.org). R Studio is also free and provides a comprehensive coding environment for new coders, and R is especially useful in statistics-heavy and bioinformatics research.

In short, recent developments in programming software have made it possible to integrate high-quality coding tools into low-cost classrooms. With minimal time and effort, instructors can use tools such as Jupyter Notebooks to introduce coding with few logistical barriers. The only true barrier is spreading the good news that such free tools are available.

Unknown needs of "applied" versus "basic" programming courses

While introductory coding courses have long been offered in computer science and engineering departments, they are often new arrivals in neuroscience and biology curricula. Although these classes may appear similar in a course catalog

("Introduction to Python"), the needs of end-user researchers are different from those who will be developing analysis pipelines or software from scratch. Therefore, it is important to identify discipline-specific needs to develop applied programming lesson plans and courses that teach relevant skills and feature datasets that will be useful and interesting to neuroscience students.

One consideration is the technical skills that end-user researchers need. In neuroscience, for example, researchers are confronted with a variety of different datasets, from time series to images. Certain bioinformatics programs such as BLAST are best run from the command line, and those working with "-omics" datasets should know how to interact with various databases. And so, students may need exposure to the programming basics that will allow them to analyze and visualize different forms of data and execute programs both from the command line and within a programming environment.

However, there are other skills that are less important for neuroscience researchers. For example, although many introductory programming courses may spend a week or more discussing the development of complex functions or the intricacies of object-oriented programming, these are not necessary skills for the typical neuroscience researcher. Similarly, the quirks of a particular programming language or building interactive software programs may be less relevant to neuroscience students. Ultimately, the underlying concepts—the ways in which researchers think about using programming tools to answer questions about their data—are more important to convey than the syntax of the language.

A more important consideration, however, is that coding content is made relatable to neuroscience and the student experience. Many neuroscience and biology students do not consider themselves "computational" and are often reluctant to take traditional coding courses. A discipline-specific introduction may motivate these students to conquer their fears and dive in. Further, there are well-documented differences in the perception of computational science depending on students' gender and racial identities. Still, even small, carefully constructed coding exercises can improve students' attitudes toward coding. Discipline-specific curricula that integrate coding in a relatable way can also invite educators and students alike into the fold (e.g., Juavinett, 2020), but much work needs to be done to understand how we can best teach coding to a diverse population of biology students.

What this means for the future of teaching and learning neuroscience

It will likely be difficult to find instructors with the perfect trifecta of deep biology content knowledge, strong pedagogical practices, and the technical skills to teach discipline-specific computer science or programming courses. The paramount need to recruit faculty from intersectional and underrepresented populations in STEM is an additional layer to this challenge given the long-standing exclusion of women and minorities in both biology and computer science. We should strive to hire diverse interdisciplinary instructors who can teach innovative, discipline-specific coding classes that practice evidence-based teaching strategies, but we may need stopgap solutions.

While individual institutions address the gaps between the available resources and the need for computing in neuroscience, self-motivated students and instructors can readily learn how to code off campus. Students who do not have access to programming classes within their degree programs should consider pursuing informal coursework, and search committees should recognize this coursework when judging candidates for graduate school or new instructors for hire. In the meantime, instructors who feel just "a little bit" or "moderately" comfortable with coding may need to push their own comfort zones. The next generation of neuroscientists is depending on it.

DECLARATION OF INTERESTS

The author declares no competing interests.

REFERENCES

Goldman, M.S., and Fee, M.S. (2017). Computational training for the next generation of neuroscientists. Curr. Opin. Neurobiol. *46*, 25–30.

Grisham, W., Lom, B., Lanyon, L., and Ramos, R.L. (2016). Proposed Training to Meet Challenges of Large-Scale Data in Neuroscience. Front. Neuroinform. *10*, 28.

Juavinett, A. (2020). Learning How to Code While Analyzing an Open Access Electrophysiology Dataset. J. Undergrad. Neurosci. Educ. *19*, A94–A104.

Muller, E., Bednar, J.A., Diesmann, M., Gewaltig, M.-O., Hines, M., and Davison, A.P. (2015). Python in neuroscience. Front. Neuroinform. *9*, 11.

Pevzner, P., and Shamir, R. (2009). Computing has changed biology–biology education must catch up. Science *325*, 541–542.

Pinard-Welyczko, K.M., Garrison, A.C.S., Ramos, R.L., and Carter, B.S. (2017). Characterizing the Undergraduate Neuroscience Major in the U.S.: An Examination of Course Requirements and Institution-Program Associations. J. Undergrad. Neurosci. Educ. *16*, A60–A67.

Porter, L., Guzdial, M., McDowell, C., and Simon, B. (2013). Success in introductory programming: What works?: How pair programming, peer instruction, and media computation have improved computer science education. Commun. ACM *56*, 34–36.

Society for Neuroscience (2017). Executive Summary of Report of Neuroscience Departments & Programs Survey: Academic Year 2016–2017. https://www.sfn.org/-/media/SfN/Documents/Survey-Reports/Final-SfN_2016-NDP-Survey-Executive-Summary.pdf.

van Viegen, T., Akrami, A., Bonnen, K., DeWitt, E., Hyafil, A., Ledmyr, H., Lindsay, G.W., Mineault, P., Murray, J.D., Pitkow, X., et al.; Neuromatch Academy (NMA) (2021). Neuromatch Academy: Teaching Computational Neuroscience with Global Accessibility. Trends Cogn. Sci. *25*, 535–538.

Yadav, A., Gretter, S., Hambrusch, S., and Sands, P. (2017). Expanding computer science education in schools: understanding teacher experiences and challenges. Comput. Sci. Educ. *26*, 235–254.