Teaching Programming Skills in Methods Courses is an Opportunity, not a Burden*

Rob Williams†

July 09, 2021

Word count: 3,068

As political scientists have become more skilled in the tools of computational social science, we have begun to pass these skills on to our undergraduate students. Students in undergraduate quantitative methods courses today often learn to program in statistical software alongside the more traditional topics of mathematics and research design. Many undergraduates are eager to learn these skills due to their desirability in increasingly common data science jobs. An equally large number may harbor anxieties about having to 'learn to code' for the first time, especially in departments where completing a quantitative methods course is a required component of the degree. These worries are especially pronounced for students who belong to groups that have been historically excluded from science, technology, engineering, and math (STEM) fields.[1]

Some faculty may feel that teaching programming skills detracts from the social scientific goals of a course, and prefer to use point and click software such as Stata or SPSS to avoid this instructional burden. However, teaching programming with tools such as the R statistical computing environment represents an opportunity for instructors to cultivate a more engaging classroom experience for students. Understanding statistical concepts can be a time-consuming process with a steep learning curve, but computing skills offer students a succession of smaller, more easily-attainable victories that can build confidence and maintain

*I thank Rachel Porter, Santiago Olivella, Ted Enamorado, Leah Christiani, Christopher Lucas, and two anonymous reviewers for helpful comments.

†Postdoctoral Research Fellow, Washington University in St. Louis, rob.williams@wustl.edu, jayrobwilliams.com

engagement with conceptual material. This approach can help students from historically excluded groups gain the confidence to pursue subjects they may not have otherwise, and give more advanced students the chance to learn by teaching their peers.

1 Coding as motivation

Learning statistics can be a difficult process for undergraduate political science students. Many students choose to major in political science, at least in part, due to the perceived absence of quantitative requirements for the discipline. Concepts such as null-hypothesis hypothesis testing are far from straightforward and require previous knowledge of probability distributions, which in turn requires knowledge of calculus. Even learning how to read statistical material can be challenging if students are not familiar with mathematical notation. Taken together, this material adds up to a semester-long journey where the importance of each individual component may not be apparent for many weeks.

In contrast, programming is characterized by immediate feedback and frequent discrete confirmations that students are doing things correctly. Each time a student runs a line of code without encountering an error is a small victory. When students learn to create a scatterplot, they can immediately and intuitively assess the strength of a bivariate linear relationship, whereas computing the correlation coefficient requires first understanding what a z-score is and how to compute it.

While coding in R can often be frustrating, students' successes during in-class coding sessions can serve as the highlight of instructional time. The tangible nature of the code that students produce in-class can also encourage some to apply their new programming skills to projects outside the course. One student told me that they planned to continue learning R so they could use sports analytics techniques for their fantasy sports team, while another said that they hoped to use R to conduct the analyses for a psychology research project they were planning. The fact that R is open source software and freely available, unlike less programming-intensive options such as Stata or SPSS, means that students can easily apply the skills they learn in the course to their own projects. This sense of transferable skills helps students connect to what can easily be abstract and disconnected material. Job listings for data science positions mention R twice as frequently as any other statistical software typically taught by political scientists (Li 2019), and many students are willing to learn it because of this connection to data science.

2 The lure of data science

The emergent field of data science represents the intersection of statistics, computer science, and substantive knowledge (Davenport and Patil 2012). Beyond the basic quantitative skills that employers desire in applicants (National Association of Colleges and Employers 2020), data science skills are in particular demand (Davenport and Patil 2012). A higher proportion of data scientists have non-computer science backgrounds than many other tech industry jobs (Lindner 2018), making it possible for students without a degree in computer science to pursue these opportunities

Multiple students in my undergraduate methods course noted on their pre-course surveys that they chose my course to fulfill their methods major requirements instead of less quantitatively intense alternatives due to an interest in data science. Many nonmajors also expressed that the possibility of learning data science skills contributed to their decision to take the course. Emphasizing and advertising the programming content of undergraduate quantitative methods courses can be beneficial to departments by boosting enrollment in political science courses.

Students can be discouraged from studying data science because they believe that they must be "very smart" and "good at science and math" to learn computer science (Google 2015, 15). In contrast, political science attracts many students who feel that they are not skilled in math or science. I pursued political science as an undergraduate largely to avoid taking additional math courses, and did not discover an appreciation for math until my graduate coursework paired it with the motivational benefits of coding and substantive material that interested me. Teaching programming in quantitative methods courses thus represents an opportunity to impart skills, and potentially help students discover a passion for data science or adjacent disciplines, that they might otherwise not take the opportunity to acquire. This last benefit is especially important when we consider *which* students are least likely to follow a more traditional path to data science.

There is a widespread perception of "coding as an impossibly hard technical skill [that] has been used to push people out of the field and diminish the contributions of whole groups of people" (Shugars 2021). Despite the prevalence of women in the early days of computing, computer science has come to be a largely male-dominated field (Abbate 2012; Beyer 2014); the early contributions of Black engineers and computer scientists to the development of computers and the internet have been similarly marginalized in the popular imagination (McIlwain 2020). In a study of 7th-12th grade students in the United States, both girls and boys rated boys as considerably more interested in computer science (Google 2015). Male students cite more early exposure to computers as reason to major in computer science (Funke, Berges, and Hubwieser 2016), and are more likely to have early exposure to computers (Papastergiou 2008). Female students are also less likely to pursue computer science as biased teachers early in their education may steer them towards less technical fields and weaken their self-confidence (Carlana 2019). In addition, parents who lack experience in computer-related fields may rely on and reinforce societal gender stereotypes around them (Cheng and Huang 2016, 281). Black and Hispanic students are more likely to fall off the advanced math track when transitioning from middle school to high school even after controlling for academic performance (Irizarry 2021), and once in high school are less likely to be exposed to computer science and express lower levels of confidence in their ability to learn computer science (Google 2015, 21). Teaching programming in quantitative political methodology courses represents one way to counteract these forces and introduce nonmale and nonwhite students to data science who may be deterred from computer science departments.

Due to the surging demand for computer science education in the United States, many departments have responded by enacting competitive enrollment policies (Singer 2019). Students in introductory level computer science courses in departments with competitive admission cite such competition as one reason for not continuing in the major (Lewis, Yasuhara, and Anderson 2011), and competitive admissions policies lead to lower levels of self-perceived belonging in the department and self-efficacy for nonwhite and nonmale students (Nguyen and Lewis 2020). As political science departments rarely limit enrollment in quantitative courses, they are not subject to the same competitive forces. Quantitative methods courses in political science thus represent an alternative pathway for students from historically excluded groups to encounter coding as a practice and data science as a discipline.

3 Instructional setup

Realizing these benefits in a quantitative methods course requires careful attention to course design to integrate programming skills with substantive material. Simply adding a programming component may lead to worse course outcomes for students who have less exposure to computing due to their gender, race, and socioeconomic status backgrounds. I attempt to accomplish these goals by structuring the course to spend a significant amount of instructional time engaged in collaborative coding activities. Spending more time on programming necessarily entails spending less elsewhere, so care should be taken that the material omitted is not essential to a basic understanding of research design and statistical inference.[2] This course structure requires at least partially adopting a flipped classroom model that exposes students to R programming outside of class. Websites such as Dataquest provide a ready-made solution that combines in-depth instruction and the ability to write and test R code in a browser. Another option is the `learnr` R package, which offers a similar interactive learning format, but requires students to have a working RStudio installation. Effectively using class time to allow students to explore the more complex aspects of coding works best with either a small enrollment or a group of teaching assistants for larger courses, so that a member of the instructional team is always available to assist with issues as they arise.

Having students work in groups, whether their neighbors for in-class activities or assigned groups for semester long scaffolded final projects, has many benefits. This type of collaborative programming exercise can be especially helpful for female students, making them feel more engaged and more confident (Ying et al. 2019). Working in groups also allows students who grasp a given concept quickly to assist their groupmates, both reinforcing their own learning and freeing the instructor to assist with more complex issues that other students encounter.

Equally important to assisting students with resolving those issues is demonstrating how I overcame them. When a student has a computing error that I was not able to quickly diagnose and correct, I get the class's attention and work through finding and implementing the answer. This process typically involves using the classroom projector to show the specific Google search term I use, highlighting the relevant lines in the resulting web page, and then successfully correcting the error in RStudio.

How you as an instructor go about helping students in class is more important than the specific content of any assistance. In pre-course surveys, many students are concerned about the prospect of learning to program. Multiple students noted in course evaluations that a 'judgment free atmosphere' was important to their success in the course. Students often lack fundamental computer literacy skills such as an understanding of folder structures,[3] so an instructor must truly convince them that there are no dumb questions or dumb errors.

One tactic that goes a long way towards accomplishing this latter goal is to convey the frequency and simplicity of your own coding errors to students. I often recounted ways in which I had failed to correctly use a specific function as students learned it for the first time. Live-coding your way through a task on the classroom project presents ample opportunities for errors to organically arise and allows you as the instructor to model how to respond to them. Many students seem paralyzed when encountering an error message, so demonstrating how to respond and resolve them is an invaluable skill that they use when completing problem sets or working on final projects outside of class time. The goal is to use class time as effectively as possible to give students the tools they need to complete work independently.

Providing this level of detailed attention to students requires time and substantial effort, so departments should provide instructional support for those teaching them. When departments are unable to offer small class sizes or teaching assistants to instructors, they should provide course releases in recognition of the additional prep work required to successfully teach programming. They should decrease the weight given to student evaluations for instructors of these courses. Student evaluations are systematically lower for instructors of quantitative courses (Uttl and Smibert 2017), so instructors should be not penalized for choosing to teach more demanding courses. Racial and gender bias are endemic to student evaluations (Mitchell and Martin 2018; Chávez and Mitchell 2020), with one study finding that women experienced the largest difference in evaluations from men when teaching quantitative courses (Mengel, Sauermann, and Zölitz 2018). Departments seeking to support faculty from historically-excluded groups should be particularly aware of these dynamics, as these faculty face significantly higher demands on their time than faculty from historically-included groups (Gewin 2020).

4 Conclusion

It is tempting to view programming skills as extraneous to the core competencies students must develop in an undergraduate quantitative methods course in political science. This is a mistake given the increasing importance of data science skills both in the labor market and in academic research. When instructors integrate programming throughout the course, they can increase student engagement by providing skills that students can use in projects outside the course.

While political scientists may not have the same level of expertise in statistics and programming as statisticians and computer scientists, political science can provide a lower stakes introduction to these concepts to a wide array of students. Although some students may eventually turn to these departments for additional training, political scientists are better-equipped to teach students how to pair subject matter expertise and basic research design with these more technical skills. This blend of research and technical skills is what data science jobs require, so political science is well-positioned to impart data science skills to students who computer science departments have failed to welcome, and train them to bring the tools of data science to bear on questions of real substantive importance.

Notes

1. While students from many backgrounds may lack a strong foundation in these skills at the start of postsecondary education, Section 2 discusses the ways that stereotypes and societal barriers specifically discourage women and students of color from pursuing these fields.

2. At UNC Chapel Hill, where I received my Ph.D., undergraduate quantitative methods is now a two course sequence to allow for sufficient instructional time to be devoted to programming.

3. This issue arises each time students must load a dataset from a file, and is particularly problematic because the devices that students use most (smartphones and tablets) have no visible folder structure.

References

Abbate, Janet. 2012. *Recoding Gender: Women's Changing Participation in Computing*. History of Computing. Cambridge, Mass.: MIT Press.

Beyer, Sylvia. 2014. "Why Are Women Underrepresented in Computer Science? Gender Differences in Stereotypes, Self-Efficacy, Values, and Interests and Predictors of Future CS Course-Taking and Grades." *Computer Science Education* 24 (2-3): 153–92. https://doi.org/10.1080/08993408.2014.963363.

Carlana, Michela. 2019. "Implicit Stereotypes: Evidence from Teachers' Gender Bias." *The Quarterly Journal of Economics* 134 (3): 1163–1224. https://doi.org/10.1093/qje/qjz008.

Chávez, Kerry, and Kristina M. W. Mitchell. 2020. "Exploring Bias in Student Evaluations: Gender, Race, and Ethnicity." *PS: Political Science & Politics* 53 (2): 270–74. https://doi.org/10.1017/S1049096519001744.

Cheng, Ching-Ching, and Kuo-Hung Huang. 2016. "Stereotypes And Technology Education: Different Perceptions Of Computer Career Among Elementary School Students." *Journal of Baltic Science Education* 15 (3): Continuous.

Davenport, Thomas H., and D. J. Patil. 2012. "Data Scientist: The Sexiest Job of the 21st Century." *Harvard Business Review*, October.

Funke, A., M. Berges, and P. Hubwieser. 2016. "Different Perceptions of Computer Science." In *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, 14–18. https://doi.org/10.1109/LaTiCE.2016.1.

Gewin, Virginia. 2020. "The Time Tax Put on Scientists of Colour." *Nature* 583 (7816): 479–81. https://doi.org/10.1038/d41586-020-01920-6.

Google. 2015. "Images of Computer Science: Perceptions Among Students, Parents and Educators in the U.S."

Irizarry, Yasmiyn. 2021. "On Track or Derailed? Race, Advanced Math, and the Transition to High School." *Socius* 7 (January): 2378023120980293. https://doi.org/10.1177/2378023120980293.

Lewis, Colleen M., Ken Yasuhara, and Ruth E. Anderson. 2011. "Deciding to Major in Computer Science: A Grounded Theory of Students' Self-Assessment of Ability." In *Proceedings of the Seventh International Workshop on Computing Education Research*, 3–10. ICER '11. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2016911.2016915.

Li, Ruoxi. 2019. "Teaching Undergraduates R in an Introductory Research Methods Course: A Step-by-Step Approach." *Journal of Political Science Education*, September, 1–19. https://doi.org/10.1080/15512169.2019.1667

Lindner, Chris. 2018. “Where Do Data Scientists Come From?” *Indeed Engineering Blog*. https://engineering.indeedblog.com/blog/2018/12/where-do-data-scientists-come-from/.

McIlwain, Charlton D. 2020. *Black Software: The Internet and Racial Justice, from the AfroNet to Black Lives Matter*. New York, NY: Oxford University Press.

Mengel, Friederike, Jan Sauermann, and Ulf Zölitz. 2018. “Gender Bias in Teaching Evaluations.” *Journal of the European Economic Association*. https://doi.org/10.1093/jeea/jvx057.

Mitchell, Kristina M. W., and Jonathan Martin. 2018. “Gender Bias in Student Evaluations.” *PS: Political Science & Politics* 51 (3): 1–5. https://doi.org/10.1017/S104909651800001X.

National Association of Colleges and Employers. 2020. “Job Outlook 2020.”

Nguyen, An, and Colleen M. Lewis. 2020. “Competitive Enrollment Policies in Computing Departments Negatively Predict First-Year Students’ Sense of Belonging, Self-Efficacy, and Perception of Department.” In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 685–91. SIGCSE ’20. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3328778.3366805.

Papastergiou, Marina. 2008. “Are Computer Science and Information Technology Still Masculine Fields? High School Students’ Perceptions and Career Choices.” *Computers & Education* 51 (2): 594–608. https://doi.org/10.1016/j.compedu.2007.06.009.

Shugars, Sarah. 2021. “Dr. Sarah Shugars: Data Science Is for Everyone.” *NYU Center for Data Science*. https://nyudatascience.medium.com/dr-sarah-shugars-data-science-is-for-everyone-54f7a3bc4229.

Singer, Natasha. 2019. “The Hard Part of Computer Science? Getting Into Class (Published 2019).” *The New York Times*, January.

Uttl, Bob, and Dylan Smibert. 2017. “Student Evaluations of Teaching: Teaching Quantitative Courses Can Be Hazardous to One’s Career.” *PeerJ* 5 (May): e3299. https://doi.org/10.7717/peerj.3299.

Ying, Kimberly Michelle, Lydia G. Pezzullo, Mohona Ahmed, Kassandra Crompton, Jeremiah Blanchard, and Kristy Elizabeth Boyer. 2019. “In Their Own Words: Gender Differences in Student Perceptions of Pair Programming.” In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 1053–59. Minneapolis MN USA: ACM. https://doi.org/10.1145/3287324.3287380.