Engaging Engineering Students through Environmental Data Science

Dr. Mary Kay Camarillo, University of the Pacific

Dr. Mary Kay Camarillo is an Associate Professor of Civil Engineering at the University of the Pacific in Stockton, CA. She has a PhD in Civil & Environmental Engineering from the University of California, Davis and is a licensed Professional Engineer in California (Civil). Prior to working in academia, Dr. Camarillo worked in the consulting industry, designing and overseeing construction of water and wastewater infrastructure. Her research interests include environmental impacts of energy production, water reclamation and reuse, biomass energy, and urban adaption to climate change. In engineering education she conducts studies on how to best integration technology and data analysis into engineering courses.

Dr. Elizabeth Basha, University of the Pacific

Elizabeth A. Basha is a Professor of Electrical and Computer Engineering at the University of the Pacific. She received a S.M. and Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology. Her research interests are in multi-agent robotics, environmental monitoring, and engineering education.

Engaging Engineering Students Through Environmental Data Science

Abstract

In a world that is increasingly monitored, data management and analysis skills are valued and necessary in engineering. Despite the apparent advantages of a data-savvy workforce, engineering students often have negative attitudes and experiences with programming and data analysis. To improve these skill sets in engineering students, a course on environmental data analysis was developed and taught to a group of engineering graduate students (mostly civil engineering majors). The course relied on R and Excel and used R packages such as those in the tidyverse as well as modeling packages (e.g., fitdistrplus) and discipline-specific packages for accessing environmental data (e.g., DataRetrieval and cder). Real-world data sets were used in the examples and assignments: students analyzed data related to air pollution, climate, reservoir storage, water quality, and river flow. Students worked on importing data sets, data cleaning and wrangling, visualization, geospatial analyses, and modelling. Best practices integrated into the course included good and bad examples of data management, pair programming, live coding, worked examples with labeled subtasks, use of templates for assignments, and project-based learning. Student attitudes and experiences were monitored using surveys at the beginning and end of the term. Polls were conducted to assess specific teaching and learning strategies. The course structure provided a good opportunity for student collaboration and engagement. Environmental data analysis using data mining approaches and real-world data sets was a good way to engage students in programming and analysis, and to prepare them to work in an increasingly data-rich engineering workplace.

Introduction

Engineering students can have negative attitudes about programming, statistics, and data analyses [1,2]. While students may take statistics courses, they may not gain experience in using that knowledge to address real problems and work with large real-world data sets. Such data experiences are lacking from many engineering curricula [3]. Yet, knowledge of data science can be empowering—knowing how to use these skills can help students build confidence in their ability to become practicing engineers and scientists. Additionally, programming and other aspects of data science are beneficial for students because these activities align with modern tools, as well as reinforce critical thinking and analytical skills (e.g., understanding if statements and decision trees). Data science tools are increasingly used in engineering disciplines [e.g., 4] and the skills to use them are sought by employers [5].

This paper examines techniques and best practices to improve students' skills, beliefs, and experiences with programming and data analysis in the context of an environmental data science course. While learning data science skills can occur in many subjects, environmental issues present good applications [6]. All aspects of engineering have environmental issues (e.g., solid waste issues in electronics) and all students have some connection to environmental issues through their experience with air pollution, drought, and other environmental concerns. Because environmental data are often "messy," there are opportunities for students to practice the data cleaning and wrangling techniques of data science. Environmental data also provides opportunities for students to apply visualization techniques and models.

The arguments for teaching a class that incorporates data science using environmental applications are compelling, but the question remains: How do you best teach a class that incorporates programming and data analysis skills that students often lack? To address this issue, we looked to the pedagogy literature for guidance.

Students' struggles with programming and data analysis are not new. In the 1990s active learning in computer science consisted of mini-lectures, handouts containing work-out examples, and class time where students worked independently on projects [7]. This popular method of teaching programming evolved over time with new strategies being suggested and tested [8,9]. One such method is pair programming where students work in pairs at a single computer and periodically switch seats and roles [8]. Another method is live coding where the professor writes code in front of the class while interacting with students [10]. In addition to challenges in teaching programming, teaching statistics has its own challenges and incorporating computational tasks into statistics education is one of them [11,12]. The mini-lecture and active learning model was used by [11] in a data science course taught by faculty in statistics, while [12] recommended an emphasis on applications in a data analytics course. The use of real-world applications was also recommended by [13] in a physics programming course. In an inter-disciplinary course that included students from "business, liberal arts, and engineering and computer science," [14, p.1] reliance on cross-disciplinary collaboration and business applications was used to increase student interest. In their work to incorporate data science modules into multiple STEM courses, [15] encouraged data collection activities as well as visualization and analysis to provide students with stronger connections to the data.

The work by [8] to provide tips for teaching programming is especially compelling. Their tips are intended for any audience and any level of prior preparation, making these tips well-suited for instructors outside of computer science who are incorporating programming and data analysis into their courses (Table 1). The tips include recommendations that address student attitudes and experiences as well as provide specific recommendations regarding course content, assignments, and classroom activities (pair programming and live coding). Given the simple and actionable nature of these recommendations, we used [8] as a guide to teaching an environmental data science course. In reviewing the literature, it appears that evidence-based investigation of data science pedagogy at the course level is needed and this paper addresses that need.

Table 1. Tips recommended by Brown and Wilson [8] for teaching programming.

Number	Tip
1	Remember there is no geek gene
2	Use peer instruction
3	Use live coding
4	Have students make predictions
5	Use pair programming
6	Use worked examples with labelled subgoals
7	Stick to one language
8	Use authentic tasks
9	Remember that novices are not experts
10	Don't just code

This paper first outlines the study objectives and research questions before examining the course and course structure. It then discusses the teaching techniques, outlines the assessment tools and results used, and summarizes what did and did not work to inform efforts in future courses.

Study Objectives and Research Questions

The objective of this study was to increase student engagement and improve data analysis and programming skills in engineering by transforming an existing environmental data analysis course into an environmental data science course. This transformation was accomplished by reducing the math and statistics content and increasing the time that students spent analyzing real-world data sets. Concepts from data science were used to achieve this transition (e.g., data cleaning and wrangling). The research questions explored in this study were:

- Can authentic tasks improve students' attitude and confidence about data analysis and programming? Pre- and post-course surveys were used to address this question.
- Can the following intentional teaching techniques better engage students in learning: worked examples with sub-goals, live coding, pair programming, and presentation of good and bad examples? Polls were used to address this question.

Course and Course Structure

The course is called Environmental Data Analysis. It is a graduate course, open to all engineering and computer science majors. Although an introductory statistics course is not required, it is recommended. The course has been taught four times since 2016, evolving from an applied statistics course into a data science course. Previously, much of the course content was centered on statistics and practice of statistical concepts using textbook problems with a final project applying these concepts to a real-world data set. The last time that the course was taught, in Fall 2022, the statistics content was reduced, a textbook was not used, and the course almost exclusively relied on real-world data sets for lecture examples and homework assignments. Table 2 outlines the lecture topics covered. In Fall 2022, there were 15 students with the following majors: 11 civil engineering, one mechanical engineering, two engineering management, and one computer science. Four of the students were traditional master's students and the remaining 11 were blended students who were simultaneously pursuing undergraduate and graduate degrees. All students had prior programming experience that was typically a course using Matlab.

Table 2. Lecture Topics Covered in Fall 2022 Course.

Initial Topics	Probability and Statistics Testing	Regression
Data collection	Random variables and distributions	Linear regression
Data analysis tools	Fitting distributions	Correlation
Finding and importing data	Using random numbers in eng. calcs.	Logistic regression
Data cleaning and wrangling	Confidence intervals	
Exploratory data analysis	One sample hypothesis testing	
Experimental design	Two sample hypothesis testing	
Spatial data and mapping	One-factor ANOVA	
	Two-factor ANOVA	
	Nonparametric tests	

The software used in the class is R with the graphical user interface of RStudio. Excel was also used in one assignment that relied on pivot tables to organize and summarize data sets. In R, functions of the tidyverse and the "piping" structure were used extensively. Statistical testing and model fitting packages were used (e.g., fitdistrplus) as were geospatial packages (e.g., sf). Data importing was done using various approaches, including the use of the dataRetrieval package to import data from the U.S. Geological Survey's National Water Information System database and the cder package to import data from the California Data Exchange Center. One student used the RCloud online interface, while the remaining students used the desktop version of RStudio. Weekly assignment submissions consisted of the script files with a document containing a summary of answers, tables, figures, and written explanations of the work completed. Mid-semester there was an article review assignment and at the end of the term there was a comprehensive project.

Teaching Techniques

Based on a review of the course content and the research questions posed herein, we ultimately choose to implement eight of the ten tips described by [8] as well as the use of good and bad examples (see Table 1 for the list of tips recommends by [8]). Additionally, the mini-lecture followed by work time model described by [7] was used for most lectures.

Examining application of the tips by [8] in detail, Tips #1 (Remember there is no geek gene) and #9 (Remember that novices are not experts) were adhered to throughout the course. Framing was provided in the first class by showing a video by one of the developers of the tidyverse package, Hadley Wickham, who explained that people shouldn't feel bad if they struggle with tasks in areas where they lack experience. To promote transparency, the paper by [8] was shared with the students and explained. Additionally, the recommendation to stick to one language (Tip #7) was adhered to although there was some use of Excel to discuss pivot tables.

Use of worked examples with labelled subgoals (Tip #6) was implemented extensively throughout the class. All lectures contained at least one worked example that was organized by subtask. Templates were used for all programming assignments. The lectures focused on building skills, step-by-step and the course content led students through that process. After students were proficient at obtaining, cleaning, and visualizing the data, later assignments eliminated these tasks so that students could focus on analysis and modeling. Scaffolding was used to provide a lot of guidance initially and then less guidance in later assignments. While we were concerned that this might represent too much "hand holding", the students "proved" their skills in the final project where no programming guidance was provided.

For class purposes, the use of authentic tasks (Tip #8) most aligned with the research questions and goals of the course. Lecture examples, assignments, and projects relied on real-world data sets. The data sets used connected students with current issues such as air pollution and drought. Many of the sites were familiar (e.g., Shasta Dam and the Colorado River). We also used data from CalEnviroScreen, a tool used to investigate the interactions between environmental and socioeconomic factors and to identify disadvantaged communities. Other data sources included the U.S. Geological Survey, U.S. Environmental Protection Agency, U.S. Bureau of Reclamation, and California Department of Water Resources. Some data were accessed using the California Data Exchange Center. For the final project, students selected their own data sets.

Many of the other tips were tested through lectures or assignments. Live coding (Tip #3), where the instructor writes code in class using input from students, was attempted during one lecture. Pair programming (Tip #5), where students work in pairs at a single workstation and periodically switch roles where one person is writing the code and the other is providing instruction, was also attempted. The students had the option to work on their projects in pairs, so pair programming occurred naturally but less intentionally on the project. Also, most of the class time was spent with students working on assignments and this work time was very collaborative.

The tip to do more than code (Tip #10) was followed by talking about the data sets and sites that were used and discussing current issues (information about droughts and floods featured heavily in the news in Fall 2022). To improve the focus on the underlying data and environmental science questions, an entire lecture was devoted to developing and testing research questions. In one lecture a guest speaker delivered a presentation on environmental monitoring.

The two tips that were not used were peer instruction (Tip #2) and having students make predictions (Tip #4). The decision not to include these tips was based on time limitations.

In addition to the ten tips recommended by [8], the course also used good and bad examples. These examples were intended to show students the benefits of incorporating programming in environmental data analysis workflows and use of a data science approach. As a good example the instructor shared an experience that involved needing to download and compile over 500 hazardous waste shipping manifests (each in a separate csv file). This was a good example of how programming can automate repetitive tasks. As a bad example, the students were given a data set obtained from a government agency and then asked to identify how the data set violated the tenets of "clean data" that were being taught in the class.

Using the tips (other than Tips #2 and #4) plus the use of good and bad examples, the course was restructured to better assist students in learning data science skills. With the new structure came the assessment of the proposed research questions to see if the changes were effective.

Assessment Tools and Results

Surveys and polls were used as assessment instruments. Institutional IRB approval was obtained prior to administering the surveys and polls. All input from students was anonymous. As a result, it was not possible to link pre- and post- responses for individual students. The surveys and polls were distributed using paper copies and the students completed them during class. The instructor left the classroom while the students completed the surveys and polls.

To assess the first research question about students' attitude and confidence regarding their programming and data analysis skills, students took pre- and post-course surveys. The second research question about the effectiveness of specific teaching tools was assessed using polls.

Student Attitude and Confidence

Figure 1 shows the pre- and post- survey questions that were designed to better understand the students' attitude and confidence, and the potential impact that the course had on these attitudes

and their confidence. The pre- and post-course survey questions were identical. Lickert scales of 1-5 were used for all questions except the last question that allowed for open-ended responses.

1. Describe your comfort level in writing **basic** computer code to analyze data.

1	2	3	4	5
not comfortable		neutral		very comfortable

2. Which of the following best describes your attitude about writing computer code?

1	2	3	4	5
don't like it at all		neutral		love it

3. Describe your experience level in working with real-world data sets.

1	2	3	4	5
not experienced		neutral		very experienced

4. How important are data management and analysis skills for engineers?

1	2	3	4	5
not important		neutral		very important

5. Describe your comfort level in using R.

1	2	3	4	5
not comfortable		neutral		very comfortable

6. Describe your comfort level in using Excel.

1	2	3	4	5
not comfortable		neutral		very comfortable

7. Which of the following best describes your attitude about data analysis?

1	2	3	4	5
don't like it at all		neutral		love it

8. Any comments or thoughts regarding the role of data analysis in engineering education? Have you had any positive or negative experiences with coding, data analysis, statistic, etc.?

Figure 1. Pre- and post-course survey questions.

The survey results are first examined visually through the histograms, shown in Figure 2. As the figure shows, there is a clear trend of improvement across all questions. Especially relevant and showing strong trends are the strong improvements in writing basic code (Question 1), working with data (Question 3), comfort with R (Question 5), and attitude towards data analysis (Question 7). Attitude towards coding overall (Question 4) improved although the results look less defined and comfort with Excel (Question 6) also improved with less striking gains. The Excel question helps confirm the survey validity since only one assignment used Excel compared to the full semester of assignments using R.

Statistically examining the results validates the qualitative analysis with statistically significant results for questions 1-3, 5, and 7 (see Table 3). To statistically examine the results, the Wilcoxon Rank Sum test was used as the data for each question do not follow a normal distribution, which is required for a t-test. Note that caution is warranted in interpreting the results of the Wilcoxon Rank Sum test because of the small sample size (n = 14).

Teaching Technique Engagement

Polls were used to examine students' experiences with different teaching techniques, consisting of worked examples with sub-goals, live coding, pair programming, and presentation of good and bad examples. Figure 3 shows the questions asked in the polls. Similar to the surveys, a Lickert scale of 1-5 was used for each poll question.

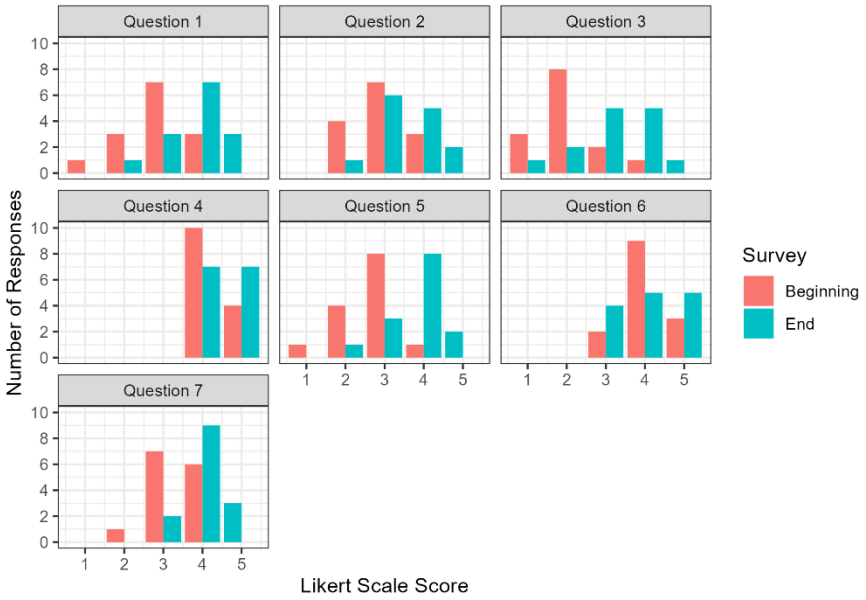

Figure 2. Histograms for student responses to the pre- and post-course survey questions.

Table 3. Wilcoxon rank sum results used to compare pre- and post- survey question responses.

Question	Wilcoxon test statistic	p-value
1	41.5	0.003431
2	58.5	0.02747
3	39	0.002624
4	77	0.1329
5	30	0.0005434
6	97	0.49
7	46	0.004336

Seeing good examples of data analysis is useful.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Seeing bad examples of data analysis is useful.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Pair programming is a good way to learn to write code.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Live coding is a good way to learn to write code.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Using worked examples with labelled subgoals is a good way to learn to write code.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Use of authentic tasks is a good way to learn to write code.

1 Strongly disagree	2	3 neutral	4	5 Strongly agree

Figure 3. Poll questions.

Figure 4 shows the graphical results of each poll using histograms. In trying to understand the student experience through this data, we can see clear preferences for good examples (question 1) and worked examples with labeled subgoals (question 5). Students appreciated using live coding (question 4) and authentic tasks (question 6) although felt less strongly about these approaches. Bad examples (question 2) and pair programming (question 3) are distributed, indicating a less positive student experience with these techniques.

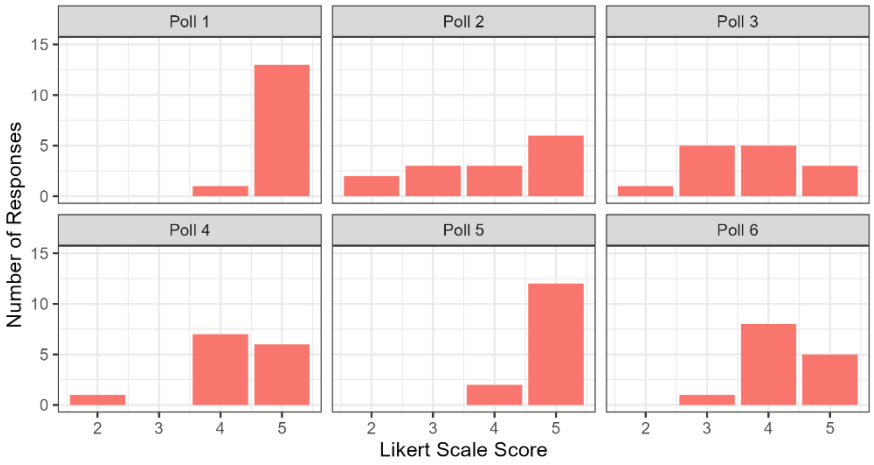

Figure 4. Histograms for student responses to the poll questions.

Discussion

Examining the assessment results provides insights into how an environmental data science course can be taught and lessons learned from the experiment process.

Analyzing Assessment Results

This process examined how authentic tasks improve students' confidence, and if intentional teaching techniques better engage students.

The results show a clear utility to focusing on tasks with real world examples and data in improving student confidence with programming and data analysis. The results are inconclusive regarding student attitude towards these skills. Attitude improved when questioned directly although seeing the direct relevance of the work was not statistically significant.

In examining the specific teaching techniques used (worked examples with sub-goals, live coding, pair programming, and presentation of good and bad examples), students were most engaged with the worked examples and good examples. Live coding also seemed to help although this had mixed experiences from the instructor standpoint as it appeared less effective for the students with slower coding skills. Other instructors report mixed results with live coding [13] although specific advice is available that could improve the experience [10]. Authentic tasks were also not as strongly rated; however, given the focus of the class was on authentic tasks, students may not have seen sufficient "inauthentic" tasks to reasonably assess the difference. Bad examples did not seem to help. This is an area for potential future exploration as perhaps the examples were not authentic enough or the students did not have sufficient understanding to see why the examples were bad.

Finally, pair programming was less well rated by the students. This result is interesting and may suggest some additional research as, observationally, pair programming seemed to work quite well, and students seemed to enjoy it. There are some questions on whether informal pair creation or formal pair creation changes students' experience as well as whether students who did not have a student to informally pair with rated the technique lower. More intentional and consistent use of pair programming could be done to better evaluate this method.

Lessons Learned

Overall, several techniques and decisions worked well and are suggested for others. The structure of the class to use a data science framework with real world data and detailed lecture examples with no textbook helped the course provide an authentic experience for students that improved their confidence. Repeatedly using the techniques of worked examples, scaffolded instruction, and good examples helped students build skills also. Constantly accessing and evaluating real data sets added authenticity to the experience. Ending the course with a final project that used all the techniques and skills learned to explore a topic of interest also helped with repetition, learning, and confidence.

Several areas need further exploration to see if they should be included in future courses. As already mentioned, live coding had mixed experiences, bad examples were not appreciated, and pair programming had some mixed results as well. While class time was dedicated to helping the students efficiently set up R and RStudio at the beginning of the semester, additional efforts could be made create subfolders, install packages, and download initial data sets and script files for easy access and initiation into the course. This assistance would help students better structure their work and reduce the barriers for starting assignments. Additionally, ending the course with a last lecture on communicating and reporting the results of data analyses would better frame the end result of the work for the students and set them up for success in the final report.

A key space for further improvement would be building a narrative around the locations where the data comes from to provide more context for the students. Much of the data are collected from key regions with interesting environmental and historical stories that could help students better connect to the data. One thought is to have short presentations on the sites that contain photos, statistics, historical facts, and summaries of current events.

Conclusion

This paper examined course techniques and structure to improve student confidence, skills, and engagement with programming and data science. This was guided by the work of [8] and others. Our efforts were assessed through student surveys and student polls. Based on this work, it appears that students can learn programming and develop self-efficacy given the right guidance and framework. Carefully curated course content is important, including the use of worked examples with labelled subgoals and templates for assignments. Authentic tasks are effective for engaging students. Interactive and collaborative teaching techniques such as pair programming and live coding can be helpful for engaging students although instructors should be mindful of how these are implemented. Future efforts will include building narratives around monitored sites. Future efforts will also include more detailed assessment and structuring of pair programming and the use of bad examples.

References

[1] C. Snyder et al., “Understanding data science instruction in multiple STEM disciplines,” presented at the ASEE Annual Conference, Virtual Conference, Jul 26-9, 2021. Available: https://peer.asee.org/37955.

[2] C. Torres-Machi, A. Bielefeldt, and Q. Lv, “Work in progress: The strategic importance of data science in civil engineering: Encouraging interest in the next generation,” presented at the ASEE Annual Conference, Minneapolis, MN, Jun 26-9, 2022. Available: https://peer.asee.org/40713.

[3] S. Grajdura and D. Niemeier, "State of programming and data science preparation in civil engineering undergraduate curricula," Journal of Civil Engineering Education, vol. 149, no. 2, p. 04022010, 2023, doi: doi:10.1061/(ASCE)EI.2643-9115.0000076.

[4] J. G. Hering, "From slide rule to big data: How data science is changing water science and engineering," Journal of Environmental Engineering, vol. 145, no. 8, 2019, doi: 10.1061/(ASCE)EE.1943-7870.0001578.

[5] Organisation for Economic Co-operation and Development (OECD), Skills for the Digital Transition: Assessing Recent Trends Using Big Data, Paris, France: OECD Publishing, 2022. [Online]. Available: https://www.oecd-ilibrary.org/content/publication/38c36777-en.

[6] K. Gibert, J. S. Horsburgh, I. N. Athanasiadis, and G. Holmes, "Environmental data science," Environmental Modelling and Software, Article vol. 106, pp. 4-12, 2018, doi: 10.1016/j.envsoft.2018.04.005.

[7] B. S. Gottfried, “Teaching computer programming effectively using active learning,” presented at the ASEE Annual Conference, Milwaukee, WI, June 15-8, 1997. Available: https://peer.asee.org/6813.

[8] N. C. C. Brown and G. Wilson, “Ten quick tips for teaching programming,” PLOS Computational Biology, vol. 14, no. 4, p. e1006023, 2018, doi: 10.1371/journal.pcbi.1006023.

[9] G. Wilson, “Ten quick tips for delivering programming lessons,” PLOS Computational Biology, vol. 15, no. 10, p. e1007433, 2019, doi: 10.1371/journal.pcbi.1007433.

[10] A. Nederbragt, R. M. Harris, A. P. Hill, and G. Wilson, “Ten quick tips for teaching with participatory live coding,” PLOS Computational Biology, vol. 16, no. 9, p. e1008090, 2020, doi: 10.1371/journal.pcbi.1008090.

[11] S. C. Hicks and R. A. Irizarry, “A guide to teaching data science,” Am Stat, vol. 72, no. 4, pp. 382-391, 2018, doi: 10.1080/00031305.2017.1356747.

[12] T. Donoghue, B. Voytek, and S. E. Ellis, “Teaching creative and practical data science at scale,” Journal of Statistics and Data Science Education, vol. 29, no. sup1, pp. S27-S39, 2021, doi: 10.1080/10691898.2020.1860725.

[13] L. Cawthorne, “Invited viewpoint: teaching programming to students in physical sciences and engineering,” Journal of Materials Science, vol. 56, no. 29, pp. 16183-16194, 2021, doi: 10.1007/s10853-021-06368-1.

[14] D. A. Asamoah, D. Doran, and S. Z. Schiller, “Teaching the foundations of data science: An interdisciplinary approach,” ArXiv, vol. abs/1512.04456, 2015.

[15] M. Y. Naseri et al., “A modular approach for integrating data science concepts into multiple undergraduate STEM+C courses,” presented at the ASEE Annual Conference, Minneapolis, MN, June 26-9, 2022. Available: https://peer.asee.org/42010.