Harvard Data Science Review • Issue 3.2, Spring 2021

Variations in Data Science Curriculum: A View From Computing Education

Alan David Fekete¹

¹School of Computer Science, University of Sydney, Sydney, New South Wales, Australia

Published on: Jun 07, 2021

DOI: <https://doi.org/10.1162/99608f92.8d7bdf3a>

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)

Contributions of the Berkeley Data Science Curriculum

The wonderful article “[Interleaving Computational and Inferential Thinking in an Undergraduate Data Science Curriculum](#)” by Adhikari et al. (this issue) offers a wealth of insights into a seminal university curriculum for teaching data science. The existence of this curriculum at University of California Berkeley, as well as its details, have been inspiring to educators in many institutions as they grapple with the challenges of designing an undergraduate education for the emerging field of data science. In this discussion I aim to show some aspects of this challenge, as it is seen by computer science educators. In particular, I will convey a variety of different curriculum approaches that have value in different settings.

To start, though, we need to pay attention to many outstanding aspects of the Berkeley curriculum. Among the most influential features of this innovation is the idea of having a campus-wide focus on data science for undergraduates from diverse intellectual backgrounds. Many of the earliest proposals to teach data science were in the context of master’s degrees, which have a deep vocational focus and often are aimed at students who already know statistics or computing. The Berkeley experience shows that data science can be an attractive and valuable study for undergraduates whose main focus is in social science, life science, humanities, and so on; this lesson has been taken on by many institutions.

Just as radical, but less widely adopted, was the way Berkeley used the connector courses to bring leading researchers from diverse disciplines as part of the educational design and delivery team. Instead, many data science curricula were developed by a mix of statistics and computer science faculty, with students exposed to only a few applications beyond those that the core disciplines chose and presented.

The connections between computer science and statistics techniques are of course central in any data science curriculum; however, as I discuss here, few institutions have gone so far in creating new subjects around these connections, compared to the five subjects (Data 8, 100, 140, 102, and 104) described by Adhikari et al. Nonetheless, the example of the Berkeley curriculum (and the wonderful material they produced and made available) has been an inspiration even for those who had to cut back for pragmatic reasons.

The Computing Education Community

Since many readers of *HDSR* seem to come from a statistics background, I want to explain the intellectual community of computing education, whose members provide the computer science contribution in many data science curricula. To begin, a vital feature of this community is that, like most computer science subfields, the dominant way for ideas to spread is through papers published in conferences. Unlike in many traditions, computer science conferences choose the works to appear through a competitive selection, where submissions are refereed by experts. For most computer science disciplines, the most prestigious and visible forum for work is in one of the premier conferences. For computing education, there is the annual ACM SIGCSE (the Association for Computing Machinery’s Special Interest Group on Computer–Human Interaction) conference.¹

This conference is dominated by papers that describe and reflect on innovative approaches to teaching topics in computing; the readers form a community of practice among educators. A similar conference, with more European and less U.S. presence, is the ACM ITiCSE (Innovation and Technology in Computer Science Education) conference series.² A more recent and specialized conference is specifically targeted for papers that use techniques from educational research,³ though papers like this are also a minority in SIGCSE and ITiCSE.

We mention here a few of the key topics that attract attention in computing education: the development of innovative tools to help in teaching and bring industry practice into the classroom, the mental models (and especially misconceptions; that is, incorrect models) formed by students, what leads students to engage or feel confident in computing classes, equity of access to computing education for marginalized populations, computing education for non-majors or for students in K-12, and issues of scaling classes for large cohorts (like the 800 or more mentioned taking Data 8 at Berkeley). The community has been considering how to use data science in teaching computing, or how to teach computing for students whose focus is data science, for several years (see Sullivan, 2013, and Ramamurthy, 2016). In the recent conferences, data science education has been a very active topic (see Adams, 2020; Allen, 2021; Blair et al., 2021; Bressoud & Thomas, 2019; Deb et al., 2019; Fekete et al., 2021; Hassan et al., 2021; Karbasian & Johri, 2020; Rampure et al., 2021; Rosenthal & Chung, 2020; Salloum et al., 2021). Another outcome of this activity has been [a curriculum document](#) setting out appropriate computing competencies from the ACM Data Science Task Force, co-chaired by Andrea Danyluk and Paul Leidig.

Structural Choices and Constraints for a Curriculum

The Berkeley curriculum described by Adhikari et al. is a remarkable educational effort. It is clearly the product of a leading research university with outstanding faculty that includes global leaders in research and teaching. The description shows clearly that a lot of time and effort went into both design of the curriculum and its ongoing delivery. Many institutions that are grappling with the issues in teaching data science lack this level of resourcing. For example, there are many colleges where the total faculty strength in computer science is three or four people who need to cover all the teaching of a full suite of computer science classes, and so don't have much capacity to create new subjects for a data science curriculum. There are many institutions (even in a comparatively wealthy place like California, not to speak of those in less fortunate communities or even the developing world) whose per-student funding is only a tiny fraction of that at Berkeley. There are also often issues of campus politics: The existing computer science or statistics departments, or perhaps central leadership, may fear that a new discipline has lower standards than those established and protected by accreditation processes. Others may worry about the arrival of a popular competitor that takes students away from traditional areas. For all these reasons, many data science curricula are produced under severe institutional constraints. A common consequence is for the curriculum to reuse many subjects that are already offered, in computer science and statistics, and perhaps in application disciplines. In a constrained curriculum of this sort, one usually tries to produce at least one new distinctive subject, to allow students in data science to

form a cohort, to demonstrate their specific skills, and so on. But there are many ways to set this up in a curriculum.

In some institutions, it is institutionally difficult to have a subject that is shared across departments—in such systems, each subject needs to be controlled in one place (and perhaps funding flows to one place only). This may mean that even a subject intended to bring the cohort together feels less like the new field of data science than like the discipline from which it is taught. A data science curriculum set up this way, with each subject clearly owned by one established discipline, may well feel divided: There are subjects that take a computer science view and are taught by computer science faculty, while others feel just like statistics classes. One might end with nothing that shows how data science is different from these older fields. So, many curricula take special care to have one or more subjects that really integrate the approaches. There are three especially notable ways to do this: a distinctive introduction to data science as a discipline that follows after the traditional introductory computer science and statistics subjects; a distinctive capstone project at the end of the bachelor's; or the Berkeley approach of a distinctive introduction that does *not* assume previous study of computer science and statistics.

In many ways, the easiest way for a campus to build a data science curriculum is one in which students study the relevant subjects from computer science and statistics, and then, building on expertise in both styles of thinking, they finally see the whole lifecycle of data science work, and carry out a project that integrates the skills they have learned by attacking a task for some application domain. This takes the least faculty effort, as all the existing content classes don't need to change in any way. The key challenges here are those of any capstone project: finding good project topics and clients for students to work with and managing expectations on both sides (a data science capstone of this style is described by Allen, 2021). However, if one hopes to build a distinctive identity for data science, and to provide support to other disciplines by upskilling their students, it seems less than ideal to leave it until students are almost graduated before they see their first exposure to the key ideas of the data science lifecycle and the connections between computer science and statistics.

The Berkeley approach of Data 8, offering an introduction to data science without prerequisite experience of either computing or statistics, has many pedagogical advantages, and as described by Adhikari et al., it provides an excellent way to teach students both programming ideas and the basics of statistics, with examples that are wonderfully motivating. However, in many settings, there are real concerns with this approach. Especially in times when computer science departments are struggling with huge cohorts, and often have to introduce quotas to limit enrollment, there may not be the available teachers or lab space for more classes that require a lot of support for building skills in programming. There are also many challenges to resolve around cross-crediting: for example, if a student has learned Python programming in the introduction to data science, can they move directly into later computer science classes like databases, data structures or object-oriented programming? If they can, then there are hard decisions about fairness in access to the scarce seats of the higher computer science subjects, and also the teachers of these higher subjects may need to adjust their

lessons to a more diverse set of student backgrounds. If instead the students must take the first computer science class before they proceed in computing, then the teachers in this subject may have difficult situations if they teach things differently, ask for different coding conventions, and so forth, than what a large subset of students are used to.

If students are first expected to complete the usual introductory subjects from computer science and statistics, and then there is an integrative subject that showcases how data science works, building on existing knowledge such as programming skill, statistical tests, and so on, the difficulties above are avoided. However, one instead faces difficulties in clarifying the identity of data science as separate from the prerequisite disciplines, and even more so, this can be very demotivating. The classes for both introductory statistics and introductory computer science have on many campuses reputations as boring, unmotivating, and harshly competitive. Numerous international studies report failure rates in introductory computer science subjects of well over 30%. When students are not allowed to see the attractive aspects of data science until they have surmounted both standard statistics and standard programming, the capacity of data science to build an attractive identity can be compromised.

Another challenge facing the designers of a data science curriculum is which programming language (and which libraries) to use for the class examples and assessments. Data science professionals make use of R or Python as well as other tools such as spreadsheets, MATLAB, statistics packages, and so on. This decision can interact with issues of staffing: Faculty from the statistics department are likely to be more familiar with R, while those from computer science often work with Python (and likely have lots of teaching material already for that language). At my own institution, we decided to insist that each student learn both R and Python; classes taught by statistics faculty use R, those taught by computer science use Python.

Even if one settles on a language, there are many different libraries available (for producing summaries, calculating standard statistics, plotting, machine learning, etc.), and so there are many ways a task can be solved. Some tasks could require pages of complicated algorithmic code if done with the bare language, but with a suitable library the task needs just a few lines. A central issue for the curriculum is how much students should get into the internal algorithms, or just use provided library functions like black boxes. For example, should students learn to code how one adjusts coefficients of a regression, to find the best fit, or just call a library function that produces those coefficients? This choice reflects a tension between getting quickly to solve problems like a data scientist, versus understanding how the computation works (and one hopes, therefore, getting a sense of strengths and limitations of each method).

Berkeley takes an interesting route, teaching with their own special-for-novices Python library in Data 8, and moving to the common-in-industry library Pandas in Data 100. One serious disadvantage of this is that those students who only take the first subject, without proceeding to the rest of the major, don't have knowledge of any of the tools used in practice; they may not be able to apply their knowledge to real projects in their own discipline, if their colleagues are not willing to rely on a peculiar library not used anywhere else. My

experience shows that it can work quite well to start students doing the simplest processing in Python without special libraries, but then we introduce Pandas after about six weeks, and use it thereafter.

The ACM Computing Competencies for Data Science [curriculum document](#) lays out quite a lot of knowledge and skill that they feel every data science graduate should have; it also discusses a variety of different ways to cover those within the scope of a typical major (as this concept is defined in colleges and universities in the United States; other countries have very diverse degree structures, and disciplines may cover much more of the content than is common in the United States). Note that the Berkeley curriculum described by Adhikari et al. predates that ACM document.

Computational Thinking: Some Debates

One final issue seems important for readers trying to connect the Berkeley ideas to the wider conversation in the Computing Education community: while the article by Adhikari et al. uses the phrase ‘computational thinking’ extensively, this term is quite contested among educators. The most influential source for this term was an article by Wing (2006), which does not offer a crisp definition but rather gives many examples; Wing does include the description:

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.

And Wing mentions abstraction, modularity, and efficiency, which are targeted in Adhikari et al. Wing's focus is on ‘thinking like a computer scientist’ as an important skill for all people in modern society. Later work has revealed a tension in the ways educators deal with the term.

Shuchi Grover (2018) wrote [an influential blog post for ACM](#) that is focused on K-12 teaching, but is also relevant for undergraduates. Grover states that there are two views of computational thinking (abbreviated here as CT):

One is a view of CT as a thinking skill for CS classrooms, that includes programming and other CS practices with the goal of highlighting authentic disciplinary practices and higher-order thinking skills used in computer science. The other is CT as a thinking skill/problem-solving approach in non-CS settings—this is often about using programming to automate abstractions of phenomena in other domains or work with data with the goal of better understanding phenomena (including making predictions and understanding potential consequences of actions), innovating with computational representations, designing solutions that leverage computational power/tools, and engaging in sense making around data.

One particular aspect of a wide view of thinking like a computer scientist is a focus on data and its structure and uses, not only on the algorithms. The ACM data science curriculum identifies a number of crucial topics

around data storage and management. For example, students should know how required constraints on the content of the data can be expressed and enforced, how access to data can be controlled, how different structures of the data can be equivalent in the information content, and so on. In the account by Adhikari et al., I felt that this topic was somewhat subordinated to algorithmic thinking. The full Berkeley major in data science includes a traditional computer science database subject, where these topics are taught in the context of database management platforms that have these capabilities built-in; however, many data science projects keep data sets in files, and depend on users to set up mechanisms to manage the data directly. My colleagues and I have designed a curriculum (Fekete et al., 2021) that puts a lot of emphasis on data thinking as part of the computational thinking that data scientists need to learn, discussed in the context of data science projects that store their data sets in various ways.

Conclusion

Anyone planning to introduce a data science education at the undergraduate level should be paying close attention to the innovations at Berkeley, so well described in the article by Adhikari et al. It is clear that the Berkeley students are getting a wonderful opportunity, and those of us elsewhere can take many valuable lessons from what Berkeley has done. As we each work within a distinct educational context, we will not necessarily do things the same way Berkeley has chosen, but we all should seek to bring the insights from computer science and from statistics together for our students, as the Berkeley team has done.

Acknowledgments

I am grateful for discussions about data science education with Mike Franklin (Chicago) and Cathryn Carson (Berkeley), and I have learned much from the detailed work of others involved in the University of Sydney Data Science undergraduate curriculum: Jean Yang, Di Warren, Garth Tarr, Uwe Röhm, and Judy Kay.

Disclosure Statement

Alan David Fekete has no financial or non-financial disclosures to share for this article.

References

- Adams, J. C. (2020, February). Creating a balanced data science program. In *SIGCSE '20: The 51st ACM Technical Symposium on Computer Science Education* (pp. 185–191). <https://doi.org/10.1145/3328778.3366800>
- Allen, G. I. (2021, March). Experiential learning in data science: Developing an interdisciplinary, client-sponsored capstone program. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 516–522). <https://doi.org/10.1145/3408877.3432536>

Blair, J. R. S., Jones, L., Leidig, P., Murray, S., Raj, R. K., & Romanowski, C. J. (2021, March). Establishing ABET accreditation criteria for data science. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 535–540). <https://doi.org/10.1145/3408877.3432445>

Bressoud, T. C., & Thomas, G. (2019, February). A novel course in data systems with minimal prerequisites. In *SIGCSE '19: The 50th ACM Technical Symposium on Computer Science Education* (pp. 15–21). <https://doi.org/10.1145/3287324.3287425>

Deb, D., Fuad, M. M., & Irwin, K. (2019, February). A module-based approach to teaching big data and cloud computing topics at CS undergraduate level. In *SIGCSE '19: The 50th ACM Technical Symposium on Computer Science Education* (pp. 2–8). <https://doi.org/10.1145/3287324.3287494>

Fekete, A. D., Kay, J., & Röhm, U. (2021, March). A data-centric computing curriculum for a data science major. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 865–871). <https://doi.org/10.1145/3408877.3432457>

Grover, S. (2018, November). *A tale of two CTs (and a revised timeline for computational thinking)*. Communications of the ACM [blog]. <https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext>

Hassan, I. B., Ghanem, T., Jacobson, D., Jin, S., Johnson, K., Sulieman, D., & Wei, W. (2021, March). Data science curriculum design: A case study. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 529–534). <https://doi.org/10.1145/3408877.3432443>

Karbasian, H., & Johri, A. (2020, February). Insights for curriculum development: Identifying emerging data science topics through analysis of Q&A communities. In *SIGCSE '20: The 51st ACM Technical Symposium on Computer Science Education* (pp. 192–198). <https://doi.org/10.1145/3328778.3366817>

Ramamurthy, B. (2016, February). A practical and sustainable model for learning and teaching data science. In *SIGCSE '16: The 47th ACM Technical Symposium on Computing Science Education* (pp. 169–174). <https://doi.org/10.1145/2839509.2844603>

Rampure, S., Shen, A., & Hug, J. (2021, March). Experiences teaching a large upper-division data science course remotely. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 523–528). <https://doi.org/10.1145/3408877.3432561>

Rosenthal, S., & Chung, T. (2020, February). A data science major: Building skills and confidence. In *SIGCSE '20: The 51st ACM Technical Symposium on Computer Science Education* (pp. 178–184). <https://doi.org/10.1145/3328778.3366791>

Salloum, M., Jeske, D., Ma, W., Papalexakis, V., Shelton, C., Tsotras, V. J., & Zhou, S. (2021, March). Developing an interdisciplinary data science program. In *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education* (pp. 509–515). <https://doi.org/10.1145/3408877.3432454>

Sullivan, D. G. (2013, March). A data-centric introduction to computer science for non-majors. In *SIGCSE '13: The 44th ACM Technical Symposium on Computer Science Education* (pp. 71–76). <https://doi.org/10.1145/2445196.2445222>

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35. <https://doi.org/10.1145/1118178.1118215>

©2021 Alan David Fekete. This article is licensed under a Creative Commons Attribution (CC BY 4.0) [International license](#), except where otherwise indicated with respect to particular material included in the article.

Footnotes

1. See <https://dblp.org/db/conf/sigcse/index.html> for links to bibliographic entries, and from there to the papers themselves in the ACM digital library. ↵
2. See <https://dblp.org/db/conf/iticse/index.html> ↵
3. See <https://dblp.org/db/conf/icer/index.html> ↵