Data Science as a Route to AI for Middle- and High-School Students

Shriram Krishnamurthi[1], **Emmanuel Schanzer**[1], **Joe Gibbs Politz**[2], **Benjamin S. Lerner**[3], **Kathi Fisler**[1], **Sam Dooman**[4]

[1] Brown University and Bootstrap
[2] UCSD and Bootstrap
[3] Northeastern University and Bootstrap
[4] Work done at Brown University; current affiliation: Down Dog Yoga
Contact email: schanzer@bootstrapworld.org

Abstract

The Bootstrap Project's Data Science curriculum has trained about 100 teachers who are using it around the country. It is specifically designed to aid adoption at a wide range of institutions. It emphasizes valuable curricular goals by drawing on both the education literature and on prior experience with other computing outreach projects. It embraces "three P's" of data-oriented thinking: the promise, pitfalls, and perils. This paper briefly describes the curriculum's design, content, and outcomes, and explains its value on the road to AI curricula.

Modern AI is heavily data-centric. Becoming a successful student of AI requires a reasonable facility with data in several dimensions: writing code to process it; employing statistics to summarize and understand it; and engaging with society to understand both its power and its potential for harm.

In this paper we outline the Bootstrap:Data Science (BS:DS) curriculum. This curriculum is part of the Bootstrap project (https://www.bootstrapworld.org/), which is over a dozen years old and is one of the leading curriculum providers for middle- and high-school computing in the USA.[1] Bootstrap has been particularly successful in its oldest curriculum, which teaches key algebra concepts (Schanzer, Fisler, and Krishnamurthi 2018), through the strategy of *integration*: rather than teaching computing directly, it piggybacks atop existing courses and expertise in schools. BS:DS leverages this visibility in school systems and awareness amongst both computing and math teachers.

AI and Data Science

BS:DS is primarily a *data science* curriculum, not an AI curriculum. Nevertheless, we believe it is deeply connected to the subject of this workshop.

First, many of the central concepts—writing programs over data, using these programs to answer questions, and using statistical techniques for this task—are common to both. Second, while BS:DS does not currently handle all the forms of data found in AI—such as plain text or streaming data—it provides a strong foundation atop which these materials can be added (possibly with collaborators who wish to work with us). Finally, because a facility with data is a prerequisite for many AI systems, BS:DS can serve as a prerequisite for AI courses that educators wish to build.

Furthermore, bringing AI to schools requires far more than just "teaching AI". There are numerous logistical, pedagogic, developmental, and cognitive issues that need to be addressed. For the foreseeable future, most US schools do not have room to add whole AI classes. Students need to feel confident tackling this material, rather than opting out of it as being not for "people like" them (as reinforced by both leaders and media portrayals). They need to gain comfort with the techniques, which can be demanding, and depend on a wide range of STEM skills (which are already often in short supply). Finally, many of these issues also apply at the level of teachers, for their preparation and support.

Because BS:DS already addresses many of these issues, it enables educators to focus more on their core ideas and less on these logistics, which would otherwise also be their bailiwick. Indeed, imagine if an AI course could just *assume* every incoming student already knew basic programming, statistics, and data analysis—think how far it could go! We therefore hope this paper will spur interesting conversations and potentially also collaborations with AI researchers.

Curricular Goals and Constraints

As outlined in a recent CACM paper (Schanzer, Krishnamurthi, and Fisler 2019), the Bootstrap project has three goals across all curricula. These bear repeating here, because they significantly impact the design and content of BS:DS:

Equity Equity is the goal that the curriculum be accessible across various student communities. There are numerous aspects to equity that BS:DS pays attention to; here are a few of them:

- Specialized offerings like opt-in courses, after-school courses, etc. have self-selecting student groups: either those advantaged to take it (e.g., those who don't need to hold down an after-school job) or those who identify with the course (e.g., the gender and ethnicities already

[1] The term "Bootstrap" in this paper will always refer to this *project*, and never refers to the statistical or computing *concepts* called bootstrapping.

present in such courses, which are highly skewed). Thus, it is best to try to put material into courses that all students in a school already take; required math courses have been a particularly productive avenue for Bootstrap. These have the same gender and underrepresentation ratios as the school itself.

- Putting material in required courses means addressing the learning needs of *all* students in that course. As one illustrative example: though a notable percentage of students are visually impaired (https://nfb.org/resources/blindness-statistics), many computing courses are largely or entirely inaccessible to them. For this reason, Bootstrap has spent significant effort making both the IDEs and the programming languages accessible to blind and other visually impaired students (Schanzer, Bahram, and Krishnamurthi 2019).
- Just because material is introduced into a course does not mean it is *appealing* to all the students in the course. Giving students the ability to customize their learning, without disrupting the material's learning goals, is an important way to make it appeal to broad populations. Our prior study in Bootstrap:Algebra (Schanzer, Krishnamurthi, and Fisler 2018) shows that even a *small* amount of carefully targeted customization is enough to make students feel a great deal of ownership and pride in their work. Though we have not yet formally studied the same issue in BS:DS, we use the same "formula", as described later in this paper.

Rigor Rigor should be fairly self-evident, but is often missing in many curricula that aim for equity. There is an undertone in some middle- and high-school computing curricula that students reject rigorous material, and the frustrations of programming (not the same); that including these will therefore make students less interested in the subject; and that these should therefore be minimized. In contrast, BS:DS embraces the concept of productive struggle (Hiebert and Grouws 2007), and uses program design methods (Felleisen et al. 2001) that employ Bruner's notion of scaffolding (Wood, Bruner, and Ross 1976) and Vygotsky's concept of zones of proximal development (Vygotsky 1978) to help a student make steady and measurable progress without compromising the content.

Scale Finally, the Bootstrap project cares deeply about scale. It is relatively easy to create highly rigorous curricula in very specialized settings: the experience of several of this paper's authors, who teach introductory computing at their universities, is that a small number of schools have (almost surprisingly) deep computing content covering several topics, especially AI.

However, these approaches do not currently scale: they depend on hardware (that may be expensive but even more importantly needs to be *maintained*), trained teachers (when there is already a significant national shortage of computing teachers), time in the curriculum and space in the school, etc. In fact, at many public schools in the US (the primary audience of Bootstrap), even more basic assumptions cannot be met: many schools have locked-down computers (to prevent viruses), and hence cannot install new software; and at many schools, students cannot be assured daily computer access. While some situations may be nearly impossible to address, we feel that scaling (by definition) means tackling the needs of roughly the 20th percentile, not the 90th.

Success also requires attention to the factors that influence teacher uptake. These include teacher preparation, alignment with standards, making the material engaging to different groups of students, respect for the amount of time teachers can make in a class, and so on.

Curriculum Description

Major Components

At a high level, the BS:DS curriculum has three main aspects:

Introduction to Programming A central part of our curriculum is to program over data. For now, BS:DS focuses on *structured* data.[2] We focus on tabular data, which are a useful abstraction over many data sources, from streams of sensor readings to census bureau reports and much more. We primarily use Pyret (https://www.pyret.org/), which can be thought of as a simplified, more student-friendly version of Python. It has a sophisticated (Baxter et al. 2018) Web implementation that hides several complexities of the Web platform from students.

Students entering BS:DS increasingly have *some* programming familiarity, such as using Scratch (Resnick et al. 2009) in primary school. Some students, of course, have not had this prior exposure. Even for those who have, however, in our experience, these exposures tend to be quite superficial, do not transfer particularly well to textual programming (see (Grover, Pea, and Cooper 2015)), and are highly data-impoverished, leaving students with no real facility for thinking about processing data. Thus, we effectively have to start programming all over again, which we do not find problematic since this is easier than trying to cater to a variety of backgrounds.

Introduction to Statistics BS:DS is used in several subjects: general math, statistics, business, and social studies. In many of these, it is necessary to also include introductory material on statistical reasoning. Over time, we have found this material is needed not only for the students but also to train some of the teachers who use the curriculum. While we can assume a basic knowledge of the mean, median, and mode, even regression is not a topic we can assume. In fact, the more statistics material we include rather than assume, the more demand we find for it from teachers, and hence have to keep expanding it.

The Three P's As noted earlier, our curriculum revolves around "three P's": promise, pitfalls, and perils. We discuss each below:

Promise The promise of data-oriented methods are not at all universally familiar to teachers and students in

[2] We very much wish to incorporate various forms of unstructured data (such as raw text) in the future, but we believe this requires more attention paid to the statistics components.

schools. Thus, a major portion of the curriculum is having students wrestle with real-world data sets and learn things from them. Both teachers and students find it revelatory that they have the power to formulate hypotheses and test them against data.

Pitfalls Programming with large quantities of data also introduces pitfalls. There is a well-known tendency to assume an output is correct because it came from a computer. Furthermore, so long as an output does not wildly deviate from the programmer's expectations (which may themselves be highly, and even implicitly, biased), there is not much reason to question it. Instead, BS:DS asks teachers and students to create representative data samples and formulate tests on these, ensuring that their program is producing the desired output on *known* data sets before trusting their output on unknown ones.

Perils Over the past several years, journalists of various stripes have done a great deal to expose some of the potential perils of data-oriented thinking (e.g., the seminal ProPublica study on parole (Angwin et al. May 23 2016)). As data-driven AI systems are embedded even deeper into society (e.g., China's social credit system), it is vital for students to understand the role of these systems in their lives and the problems they can cause. For many BS:DS students, these issues are deeply personal, as they may even affect immediate family members. *We believe every data-oriented curriculum has a moral obligation to teach not only the methods but also the responsibilities that come with data-centricity.*

In addition, the curriculum has two important facets that have proven invaluable:

Notice and Wonder Real data demand reflection: to understand its structure, meaning, value, and problems. However, simply telling students to "think" about data is not likely to be effective. We instead structure this thinking through the technique of "notice and wonder" (Ray 2013), which carries over superbly to reasoning about data. This requires asking students to study the data to first see what they notice; and then using their observations to ask what they wonder. Having a class collectively share their observations and questions is a revealing experience, and gives students different ways to shine: a student may not be very facile at programming, but may be particularly good at observing weaknesses in a data set.

Authenticity Finally, BS:DS is attractive to teachers because we offer *authentic* embeddings into different contexts. For instance, as noted above, some of our teachers teach social studies. Social studies is a superb area for introducing data-oriented thinking. However, most of these teachers do not much care about *programs*; their unit of currency is the *research report*. We present programs (and statistics) as *yet another way of answering questions*, a concept both familiar and welcome to these teachers. Programs then become just one more tool in their arsenal, which already includes concepts like literature reviews and surveys. The curriculum thus leads to the creation of a report: starting with a question, then finding relevant data, then determining the statistics and programming them... but then also closing the circle, returning to answer the question and justifying the answer based on the computed statistics. This has the authenticity that is necessary for teachers to comfortably incorporate seemingly alien material into their classes, thus greatly increasing the reach of BS:DS.

Unit	Data	Programming
1	Introduction to data: tables, categorical, and quantitative data.	Introduction to data types in programming.
2	Learning to ask questions about data.	Defining functions, filtering.
3	Preparing logical subsets of data. Focus on categorical data.	Defining and using filter functions. Displaying data.
4	Histograms and barcharts (and their difference); manual construction; interpretation.	Doing these programmatically.
5	Central tendencies and spread.	Doing these programmatically.
6	Table manipulation. Trusting data.	Methods for building and transforming tables.
7	Scatter plots. Correlations.	Doing these programmatically.
8	Linear regression. More on correlations.	Doing these programmatically.
9	Threats to validity.	Wrapping up projects and reports.

Figure 1: BS:DS Modules

Module Structure

The curriculum currently has 9 units (fig. 1). Each unit is divided into data-oriented and programming-oriented learning, which proceed roughly in lock-step.

A key idea is personalization: early in the curriculum, students choose a data set of interest to them. *We believe that all students are inherently data scientists*: even a student who "hates math" will passionately scour for and deploy statistics to score a point about, say, their favorite sports player. The key is to find the data sets that students are passionate about. Thus, a great deal of effort in BS:DS goes towards identifying and curating suitable data sets, from income and schools to cancer rates to movies to sodas and cereals to Pokémon characters to sports.

One topic that is *not* currently a part of BS:DS is data cleansing. We introduce the idea and have students understand the messy nature of real-world data, but the set of both programming and statistical techniques needed to effectively cleanse data are well beyond middle- and most early high-school courses. Instead, we provide a set of curated data sets that have already been cleansed by us, and work with teachers to add new data sets of interest to them.

Deployment

BS:DS has trained about 100 teachers so far, several of whom are in various stages of deployment. The curriculum is currently in use in six different US states and has some international interest as well. The entire material is available for free from https://www.bootstrapworld.org/materials/data-science/.

The Role of Standards

Many teachers need to align their classroom content with standards. Currently, these standards say little to nothing about AI. However, the CSTA standards (https://www.csteachers.org/page/standards) and K-12 Framework (https://k12cs.org) highlight data and analysis as a major content theme. Required elements within this theme include presenting and making claims from data, cleaning data, and creating and refining models based on data. Some of this content is highly relevant to learning about AI. However, the *practices* side of the standards—the part that deals with the habits of thinking and behavior that underlie effective work in computing—fall short in ways that a good data science curriculum should mitigate. For instance, designing (and using!) test data, and making observations about data prior to drawing conclusions from it, are key skills that underlie responsible AI. The BS:DS curriculum fills these gaps, and other curricula should endeavor to as well.

A Caution

Collegiate educators are used to AI being in the middle of a large DAG of courses. However, this is unlikely to happen at most schools. Their AI course could become the *only* "computer science" course in many schools. As the primary or even sole reference point, the habits it encourages are the ones that students will carry forward.

This means an ends-at-all-costs approach will have bad side-effects. For instance, the rough-and-ready software style espoused in some projects—and not prohibited by standards practices—will then become students' default mode of operation. BS:DS recognizes this danger and therefore builds topics like systematic software construction, paying attention to validation, etc., into the materials. We exhort other curriculum designers to attend to these same issues.

Ongoing and Future Work

The current module structure of BS:DS is largely an artifact of our pathway to adoption. Presenting whole-semester, all-or-nothing options to teachers is usually a non-starter. Instead, the Bootstrap curricula begin as a collection of modules that teachers can intersperse through their existing courses while meeting state and national standards.

As the curricula grow in adoption, however, we have the freedom to take up more time. For instance, some regions and states are now looking at using Bootstrap curricula over a year or even two years across math classes. In these settings, our hope is to expand BS:DS on three fronts: statistics, threats to validity, and improved testing and validation. Of course, some teachers prefer to go into more advanced programming as well: for instance, methods for tabular operations are effectively higher-order functions, a topic that a few teachers want to explore in its own right.

A separate direction, which is unlikely to directly impact AI but will likely leave students much better prepared for it, is new material we are piloting for middle-school history. Students who have performed programming-driven data analysis several times over the course of their history curriculum will likely come into a future AI-centric class far better prepared to start at a more advanced stage and to take on its challenges.

Acknowledgments

We thank Leigh Ann DeLyser for her support for BS:DS over several years. We are grateful for support from the US National Science Foundation and from Bloomberg, the Robin Hood Foundation, and the Seigel Family Foundation.

References

Angwin, J.; Larson, J.; Mattu, S.; and Kirchner, L. May 23, 2016. Machine bias. ProPublica. https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, last accessed 2017-03-25.

Baxter, S.; Nigam, R.; Politz, J. G.; Krishnamurthi, S.; and Guha, A. 2018. Putting in all the stops: Execution control for JavaScript. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*.

Felleisen, M.; Findler, R. B.; Flatt, M.; and Krishnamurthi, S. 2001. *How to Design Programs*. MIT Press.

Grover, S.; Pea, R.; and Cooper, S. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education* 25(2):199–237.

Hiebert, J., and Grouws, D. A. 2007. The effects of classroom mathematics teaching on students' learning. In Lester Jr., F. K., ed., *Second Handbook of Research on Mathematics Teaching and Learning*. National Council of Teachers of Mathematics. 371–404.

Ray, M. 2013. *Powerful Problem Solving*. Portsmouth, NH: Heinmann.

Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B.; and Kafai, Y. 2009. Scratch: Programming for all. *Commun. ACM* 52(11):60–67.

Schanzer, E.; Bahram, S.; and Krishnamurthi, S. 2019. Accessible AST-based programming for visually-impaired programmers. In *ACM Technical Symposium on Computer Science Education*.

Schanzer, E.; Fisler, K.; and Krishnamurthi, S. 2018. Assessing Bootstrap:Algebra students on scaffolded and unscaffolded word problems. In *ACM Technical Symposium on Computer Science Education*.

Schanzer, E.; Krishnamurthi, S.; and Fisler, K. 2018. Creativity, customization, and ownership: Game design in Bootstrap:Algebra. In *ACM Technical Symposium on Computer Science Education*.

Schanzer, E.; Krishnamurthi, S.; and Fisler, K. 2019. What does it mean for a curriculum to succeed? In *Communications of the ACM*.

Vygotsky, L. S. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.

Wood, D.; Bruner, J. S.; and Ross, G. 1976. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry* 17(2):89–100.