Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges

Sean Kross
UC San Diego
La Jolla, California, USA
seankross@ucsd.edu

Philip J. Guo
UC San Diego
La Jolla, California, USA
pg@ucsd.edu

ABSTRACT

Data science has been growing in prominence across both academia and industry, but there is still little formal consensus about how to teach it. Many people who currently teach data science are practitioners such as computational researchers in academia or data scientists in industry. To understand how these practitioner-instructors pass their knowledge onto novices and how that contrasts with teaching more traditional forms of programming, we interviewed 20 data scientists who teach in settings ranging from small-group workshops to large online courses. We found that: 1) they must empathize with a diverse array of student backgrounds and expectations, 2) they teach technical workflows that integrate authentic practices surrounding code, data, and communication, 3) they face challenges involving authenticity versus abstraction in software setup, finding and curating pedagogically-relevant datasets, and acclimating students to live with uncertainty in data analysis. These findings can point the way toward better tools for data science education and help bring data literacy to more people around the world.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

data science education; teaching programming

ACM Reference Format:
Sean Kross and Philip J. Guo. 2019. Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, UK*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3290605.3300493

CHI 2019, May 4–9, 2019, Glasgow, UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300493

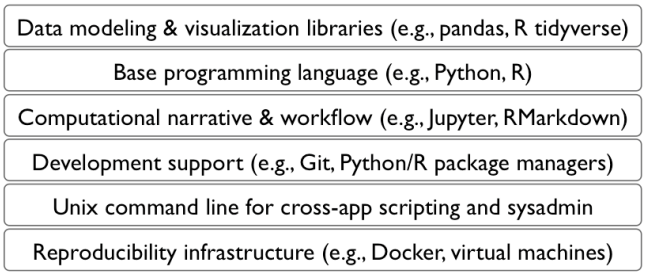

Figure 1: An example technology stack that modern data scientists must learn to do their job of writing code to obtain insights from data in a robust and reproducible manner. They often learn these skills from their fellow data scientists, not from formal computing instructors.

1 INTRODUCTION

People across a wide range of professions now write code as part of their jobs, but the purpose of their code is often to obtain insights from data rather than to build software artifacts such as web or mobile apps. Although programmers have been analyzing data for decades, in recent years the popular term *data science* has emerged to encapsulate this kind of activity. Data scientists are now pervasive throughout both industry and academia: In industry, it is a fast-growing job title across many sectors ranging from technology to healthcare to public policy [49]. In academia, data scientists are often STEM graduate students, postdocs, and technical staff who write code to make research discoveries [32].

Despite its blossoming across many fields of practice, data science has only recently begun to formalize as an academic discipline, so there is still little consensus on what should go into a data science curriculum [16, 21, 35, 37]. Many novice data scientists are currently learning their craft and associated technology stacks (e.g., Figure 1) on the job from expert practitioners rather than from full-time teachers. To understand how these *practitioner-instructors* pass their knowledge onto novices and what challenges they face, we conducted interviews with 20 data scientists (five men and fifteen women)

who teach in both industry and academic settings ranging from small-group workshops to large online courses. Our participants come from backgrounds ranging from the life sciences to the behavioral sciences to the humanities; none have formal degrees in computer science.

We chose to study practitioner-instructors because they are the ones defining both the technical and cultural norms of this emerging professional community. Their insights can inform the design of new programming tools and curricula to train this growing population of diverse professionals who are responsible for making advances across science, technology, commerce, healthcare, journalism, and policy.

While prior work has studied what data science practitioners do on the job [32, 41, 42, 44, 64, 65], to our knowledge, we are the first to systematically investigate how they teach their craft to junior colleagues and students.

Our study extends the rich lineage of HCI research on how people learn programming to pursue different career goals. On one end, there is a long history of studies on teaching computer science and engineering skills to those who aspire to become professional software engineers [33, 60, 68]; on the other end, there is a parallel literature on the learning needs of end-user programmers [28, 47, 73]. Data scientists are a distinct and so-far understudied population in between those two extremes: They share similarities with both software engineers (they aspire to write reusable analysis code to share with their colleagues) and end-user programmers (they view coding as a means to an end to gain insights from data).

We found that data science instructors must empathize with a diverse array of student backgrounds and expectations. Also, despite many of their students viewing coding as merely a means to an end, they still strive to teach disciplined workflows that integrate authentic practices surrounding code, data, and communication. Finally, they face challenges involving authenticity versus abstraction in software setup, finding and curating pedagogically-relevant datasets, and acclimating students to cope with uncertainty in data analysis.

These findings can point the way toward the design of specialized tools for data science education, such as block-based programming environments, better ways to find and synthesize datasets that are suitable for teaching, and fostering discussions around data ethics and bias.

In sum, this paper's contributions to HCI are:

- A synthesis of the technical workflows that data science practitioners teach to novices, along with challenges they face in teaching. These findings advance our understanding of a growing yet understudied population in between end-user programmers and professional software engineers.
- Design implications for specialized tools to facilitate data science education.

2 RELATED WORK

Our study was inspired by prior work in end-user programming, teaching data science, practitioners as instructors, and broadening computing education to learners who do not self-identify as programmers.

Data Science and End-user Programming

Data science is a broad term that encompasses a wide variety of activities related to acquiring, cleaning, processing, modeling, visualizing, and presenting data [35, 41]. Although data visualization is a highly active area of HCI research, what is more relevant to our study is prior HCI research on programming as performed by non-professional programmers.

Kandel et al. found great variation in levels of programming ability amongst data scientists [41]. Many of them write code in languages such as Python and R [21, 25, 35, 37], but they are not professional software engineers; moreover, many do not even have formal training in computer science. Much of data scientists' coding activities can be considered end-user programming [46] since they often write code for themselves as a means to gain insights from data rather than intending to produce reusable software artifacts. Related terms for this type of insight-driven coding activity include exploratory programming (from Kery et al. [42, 43]) and research programming (from Guo's dissertation [32]).

However, as we discovered in our study, modern data scientists are not merely writing ad-hoc prototype code. They are now developing increasingly mature technology stacks for writing modular and reusable software (e.g., Figure 1). In the terminology of Ko et al., they are now engaging in *end-user software engineering* [46] with more of an emphasis on code quality and reuse; in Segal's related terminology, data scientists are now becoming *professional end-user developers* [65]. Along these lines, software engineering researchers such as Kim et al. have studied the role of data scientists within industry engineering teams [44].

In contrast to prior HCI work that focuses on what data science practitioners do on the job, our study instead focuses on how they pass on those skills to novices via teaching.

Teaching Data Science

Data science is now a highly in-demand subject within both academia and industry: Many universities are launching new data science majors [69, 70], research labs are organizing hands-on workshops [75], and MOOCs and coding bootcamps focused on data science are some of the most popular offerings [1, 2]. But despite this growing interest over the past few years, there is still little agreement on what a data science curriculum should contain [16, 21, 35, 37].

To our knowledge, there does not yet exist a systematic research study on how data science is currently being taught.

The only publications on this topic are course design guides and experience reports of how instructors have taught *specific* courses within their own fields. These papers fall into two categories: descriptions of courses taught by computer science (CS) faculty, and those taught by faculty in other disciplines. CS faculty have written about their experiences teaching data science both to enrich introductory computing courses with data-oriented applications [16, 17, 25, 34] and in courses intended to serve non-CS-majors [14, 20, 61]. And faculty in fields ranging from bioinformatics [71], business [21], and statistics [35, 37] have written field guides on teaching data science in their respective majors. In particular, data science within statistics curricula places more of an emphasis on computational workflows and tools rather than on theoretical aspects of the underlying mathematics [35, 37].

Outside the classroom, instructors have also documented their experiences teaching in informal settings. For instance, the Software Carpentry [75] and Data Carpentry [4] organizations hold workshops to teach computing and data analysis to academic researchers; they also publish course design guides. Related groups have organized data-oriented hackathons [15], hack weeks [39], and apprenticeships [67] to train academic researchers in data science best practices.

In contrast to the aforementioned experience reports, to our knowledge, ours is the first academic research study that attempts to provide a broad overview of how modern data science is taught by practitioner-instructors across both industry and academia—synthesizing findings in a way that transcends anecdotal experiences within individual courses.

Practitioner-instructors

Most of our participants were practitioner-instructors: data science practitioners who also teach students. Practitioner-instructors are often found in settings such as medical schools (clinical faculty) [50], art schools, business schools, and law schools, where they are sometimes known as professors of the practice [59]. Two noted benefits of learning from practitioners are that they are likely up-to-date on the latest tools in their field [53] and that they are more direct members of the *community of practice* [48] that their students aspire to join. However, they often lack formal pedagogical training: Wilson refers to them as *end-user teachers* [76] (as an analogue to end-user programmers) since they teach but are not formally trained as professional teachers. To our knowledge, researchers have not yet studied practitioner-instructors in computing-related settings such as data science.

Computing Education for Broader Populations

Our study contributes to the growing body of HCI and computing education work on teaching programming to broader learner populations. Specifically, it extends prior work that target people who do not self-identify as programmers.

Although much of computing education research targets learners who aspire to become computer science majors or professional programmers [60], there is a growing body of studies on learners with other professional identities. For instance, Ni et al. studied the challenges faced by high school teachers who are learning programming in order to become CS teachers [54, 55]. Dorn et al. studied graphics and web designers who identify more as artists [27, 29]. Chilana et al. studied industry professionals in non-programming roles (e.g., sales, marketing, product management) who try to learn programming to communicate better with their engineering colleagues [23, 73]. Dasgupta and Mako Hill extended the Scratch blocks-based programming environment to enable K-12 children to perform analysis on data generated by members of the Scratch online community [26]; although children do not yet have professional identities, they are able to use Scratch programming as a conduit to develop computational and data-oriented thinking skills. What all of this work has in common is that it focuses on teaching programming to learners who do not self-identify as programmers.

Along similar lines, the instructors we interviewed self-identified as data scientists, data analysts, researchers, or more generally, the umbrella term "scientist"; since their students are junior members of their peer groups, they would also likely identify as such. To our knowledge, we are the first to characterize the challenges involved in teaching the topic of data science in diverse professional settings. Some of our findings corroborate those of prior work on how programming is perceived as a means to an end rather than as something to be intrinsically enjoyed for its own sake [23, 29].

3 METHODS

For this study we interviewed 20 data scientists who teach in a diverse variety of settings across industry and academia. We recruited participants in-person at both corporate and academic conferences, online through social media posts and emails, and via snowball sampling.

Each interview lasted 45 to 60 minutes and was conducted either in-person or via video conferencing software. Participants were not paid. Interviews were semi-structured and focused on what material is being taught in their courses, how they perceived student experiences, and what challenges they faced. We encouraged, but did not require, each participant to bring sample teaching materials to walk through together at our interviews. Guiding questions included:

- Describe the overall setting(s) in which you teach.
- What are the core concepts you teach in your courses?
- Which programming languages and tools do you use to teach? What technological challenges have you faced?
- Can you walk through the structure of a typical meeting of your course? [with optional course materials]

ID	Gender	Age	Degree	Field	Sector	Workplace	Teaching setting(s)	Students
P1	F	25–34	PhD	Biostatistics	Academia	R1 university	workshops, online	1000+
P2	M	25–34	PhD	Biostatistics	Academia	R1 university	workshops, online	1000+
P3	F	25–34	MS	Genomics	Industry	R&D nonprofit	workshops, online	1000+
P4	F	25–34	PhD†	Education	Industry	Startup company	online	350
P5	F	25–34	PhD	Genetics	Academia	R1 university	ugrad/grad courses	20
P6	F	25–34	MPH	Medical stats	Academia	Medical school	workshops	20
P7	F	35–44	PhD	Marine biology	Academia	Research institute	workshops	15
P8	M	25–34	PhD	Statistics	Academia	R1 university	grad course, workshops	20
P9	F	35–44	PhD	Neuro/genomics	Academia	R1 university	grad course, workshops	20
P10	M	25–34	PhD†	Biostatistics	Academia	R1 university	grad course	20
P11	F	35–44	PhD	Psychology	Academia	Medical school	grad course, online	1000+
P12	F	45–54	MS	Psychology	Industry	Coding bootcamp	bootcamp	30
P13	F	35–44	BS	Sci/tech studies	Industry	Mid-sized company	workshops	20
P14	F	25–34	PhD	Statistics	Academia	Liberal arts college	ugrad course, workshops	30
P15	F	35–44	PhD	Statistics	Academia	R1 university	ugrad course, workshops	30
P16	M	25–34	PhD	Neuroscience	Industry	Pro sports franchise	online video livestreams	20
P17	M	25–34	BS	Math/business	Industry	Startup company	online	1000+
P18	F	25–34	MS	Library sci.	Academia	R1 university	ugrad/grad courses	15
P19	F	25–34	BS	English/stats	Industry	Mid-sized company	workshops	20
P20	F	45–54	MS	Management	Industry	Open-source nonprofit	workshops	25

Table 1: The 20 data science practitioner-instructors we interviewed: F=female, M=male. For PhD†: P4 left a PhD program, and P10 is currently a PhD student. R1 means major research university. 'Students' is approximate number of students per class.

- What, if anything, is especially challenging about teaching this material? Where do you see students struggling most?

The lead author recorded notes and quotations during each interview. After all interviews were completed, the research team (two members) iteratively categorized them together into major themes using an inductive analysis approach [24].

Interview Participant Backgrounds

Table 1 summarizes our 20 participants' demographic and professional backgrounds. We strove for diversity across multiple dimensions, such as gender, age, field, and occupation. Participants' academic degrees include bachelors, masters, and PhDs in fields ranging from the life sciences to the behavioral sciences to the humanities. Most notably, none of the participants had formal degrees in computer science (CS) or related fields. This sample is representative of our anecdotal experiences that most working data scientists today do not come from formal CS backgrounds.

Participants work at a wide range of institutions: 8 in industry, 12 in academia. Workplaces included startups, mid-sized companies, nonprofit organizations, and universities. Most notably, almost none of our participants (*even those in academia*) work as full-time data science instructors: Instead, they are scientific researchers, business analysts, or data scientists who teach part-time for supplemental income or as volunteer outreach for their professional community. P14 is the only exception; she was a visiting assistant professor of data science. This distribution of occupations reflects our anecdotal observations that, at the moment, there are relatively few people who teach data science as their primary job. (Faculty who teach in data science interdisciplinary programs also usually teach and do research in their home departments.)

Related to this diversity of occupations is the diverse variety of settings where these practitioner-instructors teach. These include standard university courses of varying sizes (ugrad=undergrad, grad=graduate course in Table 1), day- or week-long workshops such as Software/Data Carpentry [4, 75], months-long bootcamps, online courses with thousands of students, and even livestreams (see P16 below).

To showcase examples of the range of instructor backgrounds, we highlight P7 and P16: P7 is a marine data scientist at an oceanography research institute who travels around the world both to perform research fieldwork and to teach data science workshops to researchers. P16 is a neuroscience PhD currently working as a sports data analyst for a U.S. professional sports team; as a hobby, he offers free data science lessons via video livestreams on Twitch.tv [38].

Study Design Limitations

Although we strove to include instructors from a diverse array of demographic and professional backgrounds, our personal participant recruitment and snowball sampling led to some limitations: We found only two participants in the 45–54 age range, and nobody who was 55 or older. All of our participants identified as cisgender. None were underrepresented minorities in STEM fields. Everyone except for P8 (Australia) and P9 (Canada) was based in the United States. Follow-up studies that recruit from broader demographics would improve the external validity of our findings.

In terms of technology stacks, all participants taught using open-source languages and tools (e.g., Python, R), so we were not able to study data scientists who work in closed-source proprietary ecosystems such as Matlab, Mathematica, or Stata. We could not reach data scientists within corporate or government settings that were restricted by nondisclosure agreements or security clearances. This sampling bias means that our findings likely apply more to open-source and open-science cultures rather than to closed-source settings.

We studied only instructors who taught formal courses (albeit in a wide variety of settings), so that means we did not cover informal learning via on-the-job apprenticeships or getting on-demand help from colleagues.

We interviewed only data science instructors but did not directly study their students. We chose to focus on instructors because they are the ones defining both the technical and cultural norms of this emerging professional community, and they must also try to understand and address the challenges faced by a wide range of students. However, without directly studying students, our insights about student struggles will necessarily be limited to their instructors' interpretations.

Finally, we did not interview computer science (CS) professors who teach programming to non-CS-majors, even though some of these courses would likely be useful for training aspiring data scientists. Instead we focused exclusively on data science practitioners who teach their junior colleagues since this population has not yet been studied in prior work.

4 DIVERSE STUDENT BACKGROUNDS AND EXPECTATIONS

We present three main sets of findings from our interviews: student backgrounds, technical workflows, and teaching challenges. First, many participants brought up the importance of empathizing with diverse student backgrounds and varying expectations for what a data science course ought to offer.

Varying Backgrounds and Prior Coding Experience

Students who enroll in data science courses tend to be of varying ages, at very different stages of their education, and from a variety of academic backgrounds. For instance, a STEM postdoc may take a Software Carpentry workshop (e.g., taught by P3, P9), or a mid-career business analyst may take an online course on the DataCamp [5] platform. 6 of our 20 participants [P2, P8, P9, P10, P16, P19] mentioned that they regularly teach students who have never done any sort of programming before, and that these students are sometimes intermixed with others who have significant programming experience.

Due to such widely varying backgrounds, it is difficult to establish a common ground from which instruction can begin. P9, a neuroscientist who teaches graduate courses and workshops, mentioned that: *"Student heterogeneity is higher than any of us could have anticipated."* P3, an instructor from industry, faces a similar issue: *"It is always a challenge to not make assumptions about what people know or don't know. There is a huge diversity of learner backgrounds."* This issue persists even for P6, who teaches at a medical school, where a seemingly more narrow set of student demographics still manifests a wide array of backgrounds: *"The people in my workshop are all professionals: mostly professors, statisticians, and clinical data coordinators. And there's still a big variety of programming and math backgrounds."* Thus, instructors faced the challenge of creating courses that could incorporate engaging problems for students with different kinds of backgrounds and prior knowledge.

Despite these instructional design challenges, since none of our participants had formal computer science training, they sometimes felt *better* equipped to empathize with their students, who also do not come from computer science backgrounds. For instance, P17, a data scientist at a startup who teaches online courses, mentioned that: *"From a teaching perspective, I feel blessed that I didn't study computer science. I'm self-taught, and I feel that makes it easier for me to empathize with my students and anticipate their problems."*

Student Expectations and Motivations for Coding

> *"Students are grudgingly learning to program; they're really interested in analyzing their data."* -P3

> *"Most people I see have to learn to code in an absolute panic for their thesis."* -P7

Unlike many students in introductory computer science (i.e., CS1) courses [60], data science students are not enrolling because they want to learn about programming or to become full-time programmers. Rather, they are motivated to learn to solve concrete problems in their own work via data analysis. The above quote from P3 comes from a part of our conversation where she explained how grad students seek out her workshops at the moment when the amount of data or the sophistication of required analyses outgrows the capabilities of spreadsheet software (e.g., Excel) they have been using.

P7's quote recalled similar experiences, where the scientists she teaches already formulated hypotheses and collected data before they realize they need to learn programming to look for appropriate insights in their data: *"Their entry point to coding is when they're already deep into a scientific question."* We also heard from others in academia that their students do not seem to seek out programming out of a general desire to "learn to code," but rather see it as a means to render their data into meaningful scientific results. Therefore the challenge for instructors is providing just enough of an understanding of computing environments and programming to demonstrate relevant data analysis methods.

Instructors in industry face similar challenges. P12 teaches at a coding bootcamp for rural U.S. residents where the data that students learn to analyze is provided by potential employers. Student motivations for joining the nine-month bootcamp tended to be more directly career-oriented, and they also have widely varying levels of prior experience: "*We have helped many folks retrain for new jobs. Some students have never written code while others are experienced developers.*"

Eight instructors reported some students being motivated to join their course because they were excited to learn about *one specific tool* for data visualization, manipulation, or modeling [P1, P2, P4, P5, P6, P7, P13, P17]. For instance, P6 teaches medical researchers, who in this case had heard about the utility of R's ggplot2 [7] data visualization library: *"They're very excited about one specific thing: plotting, making dashboards, basically any immediately useful data product."* Instructors found these concrete expectations as both a valuable avenue for motivating students and as a source of frustration. They were challenged by students who were not motivated to understand how a tool fits into the overall technology stack such as Figure 1. P6 continued: *"I have heard, 'I don't care how it thinks, I just want to make a cool graph.'"* This frustration is exacerbated by hype around certain tools, which sets high expectations from their students' managers or supervisors for what that tool can do for their team. For instance, P1 teaches both graduate students and professionals who want to demonstrate immediate value with a tool soon after taking a seminar: *"Students are under lots of pressure to take away particular skills [back to their team]."*

5 TEACHING DATA-ANALYTIC WORKFLOWS

Although many students viewed coding as a means to an end (see prior section), nonetheless instructors emphasized teaching a more disciplined data-analytic workflow using a modern stack of open-source tools (e.g., Figure 1). In other words, they did not simply want students to create one-off scripts but rather wanted to provide them with the skills to write more robust and reproducible scientific code.

As instructors walked through the technical contents of what they taught, we noted the most salient points they raised that differed from what is typically taught in CS-oriented programming courses [22, 30, 68]. Most notably, these instructors emphasized workflows that centered on the integration of code, data, and communication rather than on the more algorithmic foundations of computing.

Teaching Data-Analytic Programming

Although data can certainly be analyzed and visualized using spreadsheets or other specialized GUI tools (e.g., Tableau), our participants all opted to teach programming languages such as Python (N=6) and R (N=14) so that: a) students could construct more reproducible scripts to automate their workflows, and b) students could learn to access the vast ecosystems of statistical and data analysis libraries in those languages, which is likely what they will be doing on the job.

Five instructors mentioned teaching how to create programmatic workflows for shaping data and moving it through an analytic process [P2, P3, P7, P11, P15]. For example, P11 demonstrates a workflow where she uses several R libraries to read data into her computing environment from sources on the web and from local files; she then combines these into one dataset using several other libraries, until she has one canonicalized *tidy* [74] data table (i.e., "data frame") where each row is an observation from an experiment and each column represents a variable that was measured. She then computes different statistics about groups in this table, creates figures and statistical models using other libraries, and finally writes a resulting narrative using R Markdown [9], a computational notebook for R. P14 echoes P11's strategy of synthesizing multiple data inputs into one rectangular data frame: *"It's rectangle-based teaching or pipeline-based programming. Everything is based on modifying one data frame."*

This sort of *data-analytic programming* [14, 32] differs from the style of programming that is typically taught in introductory CS courses. Drawing from all of our interviews, the main data structure taught by these instructors is a tabular ("rectangular") data frame; traditional CS1/CS2 data structures such as linked lists, binary trees, stacks, queues, and hash tables were rarely mentioned. Canonical operations on these tabular data frames include filtering and rearranging rows/columns, combining groups of rows and columns to create derived datasets, and creating new columns based on combining or splitting other columns. These operations are performed with calls to special vectorized functions that operate across an entire data frame at once; thus, instructors do not need to teach students how to iterate through data with explicit control flow such as for-loops, while-loops, or recursive function calls.

Similarly, teaching students to create abstractions such as functions, classes, and modules is common in CS-focused

programming courses [22, 68], but data science instructors often do not emphasize their importance, since many data science tasks can be done without these abstractions. For instance, P14 mentioned that *"maybe ten percent of the people I teach are going to need to write their own R function."* Instead, instructors emphasized that programming for data science involves *connecting existing APIs together* in order to shape them for the analytic tasks at hand. For example, a data scientist may need a software library to import geospatial data, another library to shape that data, another library to calculate statistics or build models from that data, and then yet another library to visualize the data or resulting models.

In addition to programming, instructors also emphasized data management skills. P7 mentioned: *"A ton of time is spent just showing people how to manage folder architecture and organization."*. Data science projects often involve gathering a collection of raw data files, metadata for each data file (i.e., codebooks with column descriptions), several stages of processed data files, and other data products that are created during the analysis such as rendered figures and reports. These files must all be carefully organized in a directory structure so that analysts can track provenance and so that the correct versions of files can be programmatically accessed.

Teaching Data-Oriented Communication

In addition to teaching programming, statistics, and analytical thinking, the instructors we interviewed also placed a heavy emphasis on the importance of writing, public communication, and framing analysis results in a broader societal context. P15 teaches both industry workshops and undergraduate courses, and mentioned that communication is the centerpiece in both settings: *"Communication about ideas is much more important [than code] and arguably the goal of data science. I think this is not as much the case in computer science."* She is familiar with the computer science undergraduate curriculum at her university, and by comparison says that her students do significantly more writing and public speaking. P12 is an instructor at a coding bootcamp who pushes her students to write detailed prose for their analyses and to present their findings in class: *"Students are constantly presenting and articulating their insights."* Similarly, P18 teaches social science graduate students who often want to incorporate more quantitative methods into their qualitative research:

> *"In my programming class I make them write essays. It's important that I have them talking about their project every single week. I take the communication component of the class as my primary focus."* -P18

Tools for communicating data science outputs are just as emphasized as tools for programming or statistical modeling. The majority of participants (14 out of 20) tightly integrated computational notebooks, including Jupyter [3] and R Markdown [9], into their curriculum to enable students to interleave runnable code and explanatory text. Both of these tools allow students to easily write in a literate programming style [45], where code and prose coexist in the same document. These instructors also used notebook technologies for delivering their course materials. For example, P5 illustrates statistical concepts such as the law of large numbers in a notebook, which enables students to adapt her code in order to play with this law's statistical properties.

Instructors also mentioned that they felt an important difference between data science courses and more CS-based programming courses is that students are able to create polished data artifacts that they can communicate to others even with relatively little training time. For instance, students can learn to visualize their own research data in just a one-day workshop by using the proper API calls. In contrast, it can take much longer in a CS course to go from "Hello World" examples to building compelling and useful real-world apps. P15 uses example datasets to motivate students during the first class meeting: *"By minute 10 of class they need to be able to have made a data visualization."* She often starts lessons by showing a data product that students will learn to produce that day: *"When making a cake you look at pictures of the end result cake. You don't look at pictures of eggs and milk!"*

Teaching Authentic Practices

Since our instructors were data science practitioners, they emphasized teaching students authentic work practices with tools that they actually used on the job. They taught exclusively open-source technologies for data analysis and communication, made materials that they built for their courses publicly available, and, most notably, distributed those materials *using the same tools that they teach their students to use* for sharing code, data, and analyses. P7, a marine science researcher who teaches small-group workshops to her peers, mentioned: *"I made all of my materials available on GitHub beforehand for reference."* This way, her students can follow along with her during class, and they can refer back to her materials after the course has finished. She also believes it is important to guide students through the emotional tribulations of understanding these tools, so she writes in a personal style unlike that of traditional reference guides: *"I wrote my own materials to share how I was feeling when I was learning."*

Instructors' uses of the GitHub platform are not limited to just distributing their own course materials [77]. Although GitHub is not thought of as a data science tool, our instructors showed students how to use it to connect to the broader data science community. P9 brought up the importance of

structuring the data science masters degree program she is helping launch so that students can build a public-facing portfolio of data science projects: *"All of the courses are project based and all projects are done on GitHub. It helps them build a portfolio."* Instructors see having an online portfolio as an authentic work practice in several ways: Practitioners often share their analyses as notebooks on GitHub so that others can expand upon and comment on them [64]. They also share code on GitHub to get feedback and contributions from the community. Finally, employers often ask for the GitHub profiles of data science job applicants, so instructors are motivated to help students create shareable projects.

By showing students the *same* tools they use in their daily practice, these data scientists are not merely serving as instructors, but rather as exemplars of authentic expert behavior that sets an example for junior members of their community of practice [48]. This contrasts with, say computer science university professors or K-12 teachers, who are *not* in the community of practice of most of their students (i.e., the majority of CS students do not aspire to become CS teachers).

6 CHALLENGES IN TEACHING DATA SCIENCE

The instructors we interviewed faced three widely-mentioned sets of challenges in their teaching: authenticity versus abstraction, finding and curating data sets, and acclimating students to living with uncertainty in data analysis.

Authenticity versus Abstraction in Software Setup

In addition to decisions surrounding instructional content (i.e., what to cover in their course), instructors must also decide the extent to which they are going to teach students about managing the details underlying their computing environments. Although maintaining these environments requires significant technical knowledge, these system configuration and administration logistics are usually unrelated to the data analytic skills taught in the rest of the course. While lamenting already-limited class time, P17 rhetorically asked: *"How much do we really want to teach about system administration and .bashrc?"* We identified three approaches that instructors took: 1) *Desktop*: Ten instructors taught students to configure their own personal computers at the start of class. 2) *Server*: Five instructors set up a pre-configured server computing environment that can be accessed through a web interface. 3) *Web application*: Five used a specialized web app (e.g., DataCamp [5]) that emulates a scientific computing environment and guides students through lessons with videos and coding exercises. Each approach has tradeoffs:

1) Desktop setup: Most instructors taught students to set up an authentic computing environment and toolchain on their own computers. P10 reported that companies who hire their students often do not have standardized analysis tools in place, so *"employers expect students to bring their own tools."* P11 felt that her sense of self-sufficiency in setting up her own environment informed her decision to teach about tooling: *"It's important to teach students how to work on their setup. I want them to be able to work the same way that I work."* Teaching these system administration skills, though unrelated to data analysis or to any scientific domain, provides a more authentic experience so that students can understand how the pieces of the stack (e.g., Figure 1) fit together.

However, this desktop approach is challenging for instructors because of the wide array of versions of operating systems, programming languages, and libraries each student may require to be configured on their machine. P2 dealt with a bevy of issues in the workshops that he teaches, including insufficient user permissions on work computers, outdated operating systems, and hieroglyphic configuration errors: *"These students bring in a wide variety of computers each with their own installation, permissions, and dependency issues."* He spends the start of many workshop sessions battling these complications on student computers. This process is daunting for students as well; P4 mentioned that *"data science requires a level of intimacy with your computer that my students are not used to."* For many students, it is their first time installing and using software through a command-line interface.

2) Server setup: Instructors can avoid this perennial start-of-class setup struggle by setting up a computing environment on a web server. Server-based environments such as JupyterHub [8] and RStudio Server [10] provide access to a virtual file system, command-line interface, Python and R interpreters (respectively), and hosting for computational notebooks. These systems have the potential to enable a more equitable computing experience to all students since they can be accessed from web browsers on low-cost or communal machines. For instance, P20 often gives seminars about how to use Jupyter. She emphasizes how these server-based systems put her students on the same playing field, instead of certain students being disadvantaged because they cannot afford to buy the latest hardware: *"Using shared resources like JupyterHub provides more equitable access to a computing environment. I don't need to be the wealthy kid with a new computer, in fact all I need is a [low-cost] Chromebook."*

A server-based configuration shifts much of the setup burden onto instructors, thus letting students worry less about their environment and focus more on learning data science. However, students miss out on the authenticity of learning how to configure their own machines. Also, instructors (most of whom are not full-time teachers) need to work with their local institutions to maintain a cloud-based computing environment for as long as they continue to teach, and must also figure out how to procure sustainable funding to pay for it.

3) Web application setup: The third and most abstracted strategy for setting up a computing environment involves creating courses on a fully-hosted web application such as DataCamp [5] or Dataquest [6]. Both are web apps that pair a Python/R console with guided tutorials that walk students through programming exercises and videos. Each exercise evaluates the correctness of commands that are entered into an emulated console, or they evaluate the correctness of scripts that are written by students in a simple text editor included in the web app. Instructors write lessons in a domain-specific markup language, and then upload lesson files to the web application. Students must pay to access most lessons on both DataCamp and Dataquest.

The advantage of these web applications is that they require no configuration or maintenance by *either* instructors or students: Instructors only need to write a lesson and then students can access it as long as they have an internet connection. However, participants [P2, P3, P11] mentioned several limitations: a) The environment provided by these services was not always congruent with the behavior of the real desktop computing environments they are attempting to emulate. Therefore, correct coding answers are sometimes flagged as incorrect, while incorrect answers sometimes passed automated test cases. b) In addition, these web applications are not able to integrate with existing command-line tools, external libraries, and other desktop applications in a data scientist's real-world workflow. c) Lastly, these web applications are used only for learning on the given examples and cannot be used for working on arbitrary data science tasks, so students may have trouble transferring what they learn here to their jobs. In sum, this setup makes it the easiest for instructors to focus on teaching the contents of data analysis but greatly sacrifices the authenticity of actual work practices.

Finding and Curating Datasets

Data science often takes place within the context of another discipline. P4 mentioned: *"We shouldn't teach data science alone outside of any domain."* Providing the right context for learning a new analysis concept requires first finding data that illustrates that concept well without being overly complicated. For example, illustrating a statistical concept like Simpson's Paradox [72] requires a realistic-looking dataset where the overall data has a positive correlation but the correlation within groups is negative. Illustrating domain-specific concepts within specific scientific or business fields can require datasets with features that are even more subtle.

Instructors struggle to find datasets in their domain that both feel authentic and are useful for teaching specific concepts, but without overwhelming students with their size or complexity. P1 confronted many of these problems teaching both online and in workshops: *"It's hard to find a dataset that exactly fits your problem. I've spent weeks looking for a dataset to teach with."* She eventually found a solution by asking her students to use data collected from their own personal computing devices: *"One solution is to get students to use their own data! It's interesting and personal to them."* But even if students have access to these devices, that kind of personal informatics data will only be relevant in certain disciplines.

Student-provided data is not always ideal, though. For example, P7 is a marine scientist who works with researchers that each have very different types of field data. In order to teach certain core tools of data science, and to make sure that her course is relevant for all students, she mindfully abstracts away the specifics of working with a particular kind of data. She tells her students: *"We are deliberately not using your data in order for you to learn about how to think about data itself."* Instead of having each student use data that they collected as part of their research, she curates simpler datasets that she finds online, which allow her to illustrate shaping and cleaning tasks that all of her students will need to know.

Data repositories—websites where researchers make their data publicly accessible—are another source that instructors commonly explore for teachable datasets. Unfortunately some come with licensing limitations. For example, P1 was frustrated by the stipulations of the Pew Research Center's data repository [18]: *"There are great data repositories like Pew, but they won't let you modify or distribute their data."* Pew distributes data that is relevant to P1's area of teaching, but she is prohibited by their data sharing agreement to embed the data within her course materials.

Dataset search within repositories is also challenging. For instance, P11 teaches biomedical data analysis, so the datasets that are relevant to her teaching are very specific to that field. She has to constantly monitor data repositories in hopes of finding better datasets: *"I periodically comb through PLoS open data, Data Dryad, and Harvard Dataverse looking for data to teach with."* Even after many searches, she believes it is still hard to know what data in these repositories will be useful for her course: *"People who share data tend to do complex analysis which doesn't make it great for teaching."* Even if an instructor can find data that looks relevant to their course, they then need to invest a considerable amount of time exploring the dataset to ensure that it illustrates the analytic concepts that they want to teach. P5 added: *"There is a major upfront cost to familiarizing your self with a dataset."*

Finally, there remains a gap where data that is used to teach students does not resemble the data that those students will later see on the job. P11 was concerned about how this impacts her ability to prepare students to work effectively after they leave the classroom: *"It's hard to find data that looks like the data my students will get in their jobs."*

Coping with Uncertainty

> *"Everything is always on fire! How do we teach people to live with this reality?"* -P4

P4's sentiment reflects the reality that data science practitioners often sit at the intersection of multiple disciplines and must adapt to rapidly-changing needs from stakeholders such as academic research colleagues, corporate managers, and software engineers. Thus, they wanted to teach not only the technical skills involved in data science, but also the meta-skills for coping with uncertainty on the job.

To this end, participants highlighted the importance of understanding their students' emotions while programming, particularly their frustrations when debugging high-level API calls (such as data visualization libraries) that hide many details behind each line of API code. They show students that it is normal not to know everything about the libraries that they are working with, especially by taking frustrating moments and using them as a teaching opportunities. For instance, they make sure not only to show students how to search the web efficiently, but to also normalize this practice for them. Whenever P7 is unsure about how to answer a student's question, she walks them through how to find the answer online: *"I say 'I don't know' all the time. If I can't find the answer in the documentation then we Google it together right then."* This sort of highly-personal classroom interaction is challenging for instructors to maintain, especially as class sizes grow, since it requires both personalized one-on-one attention and also an emotional investment in individual student needs.

Another important meta-skill that our participants teach is how to keep one's technical skills up-to-date. Maintaining relevance in the face of fast-changing tool ecosystems requires being able to work effectively with tools despite not achieving much mastery over them. However, there is still no consensus on what specific technologies data scientists ought to know, so this reality results in *uncertainty for instructors* about what should be included in their courses. P9 has already revised the graduate curriculum for her department's new data science masters program: *"It's hard to figure out what's essential considering that the field is changing so quickly."* P5 had a similar experience with frequently updating the contents of her course to keep up-to-date: *"Every year the technology could be different in a data science class."*

7 DISCUSSION AND DESIGN IMPLICATIONS

Our findings reveal a contrast in expectations between the novice data scientists who are taking these courses and the expert practitioners who are teaching them. Novices come into courses as end-user programmers [46] who want to learn just enough coding to be able to solve their own personal data analysis problems (e.g., *"in an absolute panic for their thesis [work]"* -P7). Instructors must empathize with that desire, but at the same time they also strive to teach a more disciplined technical workflow that integrates professional tools for code (e.g., Python or R libraries), data (e.g., manipulating tabular data frames), and communication (e.g., computational notebooks). In essence, they would like for students to not merely create one-off ad-hoc personal scripts, but rather to be able to eventually acquire the skills necessary to join the community of practice [48] of professional data scientists—for them to transform from being simply end-user programmers to being end-user software engineers [46] or professional end-user developers [65].

To facilitate this transition, instructors teach using authentic tools (e.g., Figure 1) that they use on the job. However, tools for professional use may not necessarily make the best tools for teaching. *How can we design better tools for teaching data science?* We explore some ideas in the rest of this paper.

Designing New Data Science Learning Environments

Our participants' approach to teaching was to try to scale up an apprenticeship model [11, 33] by bringing production-grade tools to their students, but those tools were not originally designed with teaching in mind. This approach does not provide a gentle point of entry for newcomers to data science who may not have prior experience in programming, performing statistical analyses, or even thinking critically about data.

Years before data science became a popular term, the statistics community had been reflecting on this rift between tools optimized for doing statistics and those for learning it. Biehler outlined a vision for the components that an integrated tool for both learning and doing statistics would require [19], and McNamara built on those ideas to advocate for tools inspired by developments in the computing education community [52]. In sum, she envisioned a *"blocks-programming environment along the lines of Scratch [62]."*

Transferring this vision into data science, there have been recent efforts to extend Scratch with data access blocks [26], to add functional programming constructs into blocks languages such as GP [12] that could be adapted for vectorized data manipulation, and to extend other pedagogical environments such as Racket with data science APIs (e.g., Bootstrap Data Science [20]). However, those languages were not originally designed with teaching data science in mind.

Whereas block-based languages like Scratch tend to be object-oriented (i.e., on-screen sprites interacting with one another), we envision a block-based language for data science being *data-oriented*. By data-oriented we mean that such a language should focus on entities representing datasets and output data products such as statistical models and visualizations. In the way that a "block" in Scratch represents a block of code in a language like Java, the representations of

data-analytic programming blocks should reflect the workflows that data scientists use and ideally help prevent novice misconceptions about those workflows. One of our participants (P12) mentioned: *"Different programming languages give different mental models of data manipulations."* Thus we believe that a block-based language for data science should be designed afresh from the ground up, not by putting layers atop existing imperative or object-oriented blocks languages.

Obtaining High-Quality Datasets for Teaching

A central challenge for many of our instructors was the difficulty of finding datasets that were relevant for a particular data-analytic concept that they wanted to teach. Instructors were highly motivated to find interesting datasets, both for illustrating general statistical phenomena and for attributes that are specific to their domain. We envision two future research directions for improving access to high-quality datasets for teaching: better tools for finding data, and new tools for synthesizing data.

Tools for Finding Data: Several instructors we interviewed vigilantly monitor online data repositories for new datasets they can use for teaching. Instead of having to monitor data repository sites, ideally they should be able to use a dataset search tool to look for data that might interest them.

There already exist a number of prominent dataset search tools, some of which focus on specific domains including data.gov [13], ICPSR [58], NCBI [66], and Google Dataset Search [57]. These tools allow search queries on some combination of the data itself and metadata about them. This approach is useful for finding datasets that include a specific kind of variable or pertain to a particular topic. However, it *fails to capture the notable features of a dataset*: e.g., the various correlations, associations, relationships, and quirks within the data that are often the essence of what an instructor would like to illustrate in class. Thus, one could design an improved search system where those notable features could be included in queries. For example: searching for a dataset that shows increasing periodicity in electromagnetic intensity from a star, or finding a factor that confounds the relationship between two other factors in gene expression data. An alternate query-by-example mechanism is to specify a model or parts of a model, and the system will use that specification to perform searches. For example: searching for population growth rate and another variable that grows logarithmically.

New Tools for Synthesizing Data: Even if more advanced dataset search tools did exist, there is still no guarantee that the data an instructor is looking for is actually available in the wild. Ideally instructors should be able to synthesize artificial datasets that would both appear realistic and exhibit the desired features for their teaching.

One approach for building such a dataset synthesis tool would be a constraint-based system (inspired by program synthesis techniques [36, 63]) where an instructor would iteratively build relationships between variables. One could, for example, specify the range of variables X and Y, and then generate their correlation. Each additional variable and additional relationship introduced to the generated dataset would need to not interfere (or interfere only within a set level of tolerance) with previously specified relationships. Data would not necessarily have to be generated from scratch; users could start with a real dataset and then append new variables and relationships onto it. One could imagine the user interaction with such a system would be similar to the work on *Same Stats, Different Graphs* [51]. Such a system would enable instructors (and students) to creatively explore the design space of example datasets that meet the given constraints.

An even more speculative approach for synthesizing data could be inspired by *image style transfer* [31, 40, 78], a deep learning technique where the style of one image, such as van Gogh's painting of *The Starry Night*, is "transferred" onto a target image, like a portrait—resulting in a wispy swirling impression of a person. By analogy, one could create a *dataset style transfer* system to transfer the "style" of one dataset (e.g., its salient properties such as periodicity, multifactorial associations, skewness/kurtosis [56]) to another dataset. Such a system would allow for more creative control for instructors to borrow subtle patterns in data from other domains that they could apply in their own desired domain. It would also allow students to each bring their own personal datasets to class but let the instructor transfer the style of a canonical dataset that exhibits the properties they want to teach onto each student's dataset; this empowers each student to work with their own individual data but learn the same lessons.

8 CONCLUSION

We have presented an interview study of 20 data scientists who teach in diverse settings across industry and academia. Despite the fact that none of them come from formal computer science backgrounds, they teach a set of sophisticated technical skills that form a coherent stack of technologies to enable open and reproducible science. They also emphasize teaching students to communicate and contextualize the outputs of their analysis work. These instructors work to integrate their students into their own communities of practice by using real-world tools with authentic datasets.

Data science is a technical specialty that continues to grow in prominence across many disciplines. In the coming years, we should work toward providing its practitioners and learners with the same levels of support in terms of both tools and community that have so far been developed for more traditional programming fields. In addition to new technical

systems for teaching data science, it is also critical to design ways to help novices understand the social systems that underlie such tools. For instance, discussions about equity, ethics, and algorithmic bias are critical for how data science is taught and who ends up even receiving such an education.

In sum, data science education is now a quickly growing form of computing and end-user programming education that is distinct from other related genres commonly studied in HCI (e.g., end-user programming, conversational programming [23], interaction designers learning programming), with its own unique challenges that require researchers to design new kinds of tools and workflows to support. We view this paper as an invitation to the HCI community—which has already produced myriad research insights in computing education and end-user programming—to increasingly study the emerging frontier of data science learning environments.

ACKNOWLEDGMENTS

Thanks to the UC San Diego Design Lab, rOpenSci, Elissa Redmiles, and Os Keyes for their feedback, and NSF award #1735234 for funding.

REFERENCES

[1] 2017. The 50 Most Popular MOOCs of All Time. http://www.onlinecoursereport.com/the-50-most-popular-moocs-of-all-time/.

[2] 2017. The Complete List of Data Science Bootcamps & Fellowships. http://www.skilledup.com/articles/list-data-science-bootcamps.

[3] 2017. Project Jupyter. http://jupyter.org/.

[4] 2018. Data Carpentry: Building communities teaching universal data literacy. https://datacarpentry.org/. Accessed: 2018-09-20.

[5] 2018. DataCamp: Learn R, Python & Data Science Online. https://www.datacamp.com/. Accessed: 2018-09-20.

[6] 2018. Dataquest: Learn Data Science With Python And R Projects. https://www.dataquest.io/. Accessed: 2018-09-20.

[7] 2018. ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. https://ggplot2.tidyverse.org/. Accessed: 2018-09-20.

[8] 2018. JupyterHub: A multi-user version of the notebook designed for companies, classrooms and research labs. http://jupyter.org/hub. Accessed: 2018-09-20.

[9] 2018. R Markdown: Analyze. Share. Reproduce. https://rmarkdown.rstudio.com/. Accessed: 2018-09-20.

[10] 2018. RStudio for the Enterprise. https://www.rstudio.com/products/rstudio-server-pro/. Accessed: 2018-09-20.

[11] 2018. TPI: Teaching Perspectives Inventory. http://www.teachingperspectives.com/tpi/. Accessed: 2018-09-20.

[12] 2018. Welcome to GP! GP is a free, general-purpose blocks programming language. https://gpblocks.org/. Accessed: 2018-09-20.

[13] U.S. General Services Administration. 2018. The home of the U.S. Government's open data. https://www.data.gov/. Accessed: 2018-09-20.

[14] Ruth E. Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. 2015. A Data Programming CS1 Course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 150–155. https://doi.org/10.1145/2676723.2677309

[15] Craig Anslow, John Brosz, Frank Maurer, and Mike Boyes. 2016. Datathons: An Experience Report of Data Hackathons for Data Science Education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 615–620. https://doi.org/10.1145/2839509.2844568

[16] Austin Cory Bart, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 2018. Reconciling the Promise and Pragmatics of Enhancing Computing Pedagogy with Data Science. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 1029–1034. https://doi.org/10.1145/3159450.3159465

[17] Austin Cory Bart, Ryan Whitcomb, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 2017. Computing with CORGIS: Diverse, Real-world Datasets for Introductory Computing. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 57–62. https://doi.org/10.1145/3017680.3017708

[18] Nick Bertoni and Scott Keeter. 2018. How to access Pew Research Center survey data. http://www.pewresearch.org/fact-tank/2018/03/09/how-to-access-pew-research-center-survey-data/. Accessed: 2018-09-20.

[19] Rolf Biehler. 1997. Software for learning and for doing statistics. *International Statistical Review* 65, 2 (1997), 167–189.

[20] Bootstrap. 2018. Data Science Curriculum (Spring 2018 edition). http://www.bootstrapworld.org/materials/spring2018/courses/data-science/english/. Accessed: 2018-09-01.

[21] Robert J. Brunner and Edward J. Kim. 2016. Teaching Data Science. *Procedia Comput. Sci.* 80, C (June 2016), 1947–1956. https://doi.org/10.1016/j.procs.2016.05.513

[22] Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. 2016. Developing a Computer Science Concept Inventory for Introductory Programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 364–369. https://doi.org/10.1145/2839509.2844559

[23] Parmit K. Chilana, Rishabh Singh, and Philip J. Guo. 2016. Understanding Conversational Programmers: A Perspective from the Software Industry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1462–1472. https://doi.org/10.1145/2858036.2858323

[24] Juliet M. Corbin and Anselm L. Strauss. 2008. *Basics of qualitative research: techniques and procedures for developing grounded theory*. SAGE Publications, Inc.

[25] Sarah Dahlby Albright, Titus H. Klinge, and Samuel A. Rebelsky. 2018. A Functional Approach to Data Science in CS1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education (SIGCSE '18)*. ACM, New York, NY, USA, 1035–1040. https://doi.org/10.1145/3159450.3159550

[26] Sayamindu Dasgupta and Benjamin Mako Hill. 2017. Scratch Community Blocks: Supporting Children As Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3620–3631. https://doi.org/10.1145/3025453.3025847

[27] Brian Dorn and Mark Guzdial. 2006. Graphic Designers Who Program As Informal Computer Science Learners. In *Proceedings of the Second International Workshop on Computing Education Research (ICER '06)*. ACM, New York, NY, USA, 127–134. https://doi.org/10.1145/1151588.1151608

[28] Brian Dorn and Mark Guzdial. 2010. Discovering Computing: Perspectives of Web Designers. In *Proceedings of the Sixth International Workshop on Computing Education Research (ICER '10)*. ACM, New York, NY, USA, 23–30. https://doi.org/10.1145/1839594.1839600

[29] Brian Dorn and Mark Guzdial. 2010. Learning on the Job: Characterizing the Programming Knowledge and Learning Strategies of Web Designers. In *Proceedings of the SIGCHI Conference on Human Factors*

in Computing Systems (CHI '10). ACM, New York, NY, USA, 703–712. https://doi.org/10.1145/1753326.1753430

[30] Mohammed F. Farghally, Kyu Han Koh, Jeremy V. Ernst, and Clifford A. Shaffer. 2017. Towards a Concept Inventory for Algorithm Analysis Topics. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 207–212. https://doi.org/10.1145/3017680.3017756

[31] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2414–2423.

[32] Philip J. Guo. 2012. *Software Tools to Facilitate Research Programming*. Ph.D. Dissertation. Stanford University.

[33] Mark Guzdial. 2015. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics* 8, 6 (2015), 1–165.

[34] Olaf A. Hall-Holt and Kevin R. Sanft. 2015. Statistics-infused Introduction to Computer Science. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. ACM, New York, NY, USA, 138–143. https://doi.org/10.1145/2676723.2677218

[35] J. Hardin, R. Hoerl, Nicholas J. Horton, D. Nolan, B. Baumer, O. Hall-Holt, P. Murrell, R. Peng, P. Roback, D. Temple Lang, and M. D. Ward. 2015. Data Science in Statistics Curricula: Preparing Students to "Think with Data". *The American Statistician* 69, 4 (2015), 343–353. https://doi.org/10.1080/00031305.2015.1077729

[36] Andrew Head, Elena Glassman, Gustavo Soares, Ryo Suzuki, Lucas Figueredo, Loris D'Antoni, and Björn Hartmann. 2017. Writing Reusable Code Feedback at Scale with Mixed-Initiative Program Synthesis. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale (L@S '17)*. ACM, New York, NY, USA, 89–98. https://doi.org/10.1145/3051457.3051467

[37] Stephanie C. Hicks and Rafael A. Irizarry. 2017. A Guide to Teaching Data Science. *The American Statistician* 0, ja (2017), 00–00. https://doi.org/10.1080/00031305.2017.1356747

[38] Suz Hinton. 2017. Lessons from my first year of live coding on Twitch. https://medium.freecodecamp.org/lessons-from-my-first-year-of-live-coding- on-twitch-41a32e2f41c1.

[39] Daniela Huppenkothen, Anthony Arendt, David W. Hogg, Karthik Ram, Jacob T. VanderPlas, and Ariel Rokem. 2018. Hack weeks as a model for data science education and collaboration. *Proceedings of the National Academy of Sciences* (2018). https://doi.org/10.1073/pnas.1717196115 arXiv:http://www.pnas.org/content/early/2018/08/17/1717196115.full.pdf

[40] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, 694–711.

[41] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise Data Analysis and Visualization: An Interview Study. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (Dec. 2012), 2917–2926. https://doi.org/10.1109/TVCG.2012.219

[42] Mary Beth Kery, Amber Horvath, and Brad Myers. 2017. Variolite: Supporting Exploratory Programming by Data Scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1265–1276. https://doi.org/10.1145/3025453.3025626

[43] Mary Beth Kery and Brad A. Myers. 2017. Exploring exploratory programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 25–29. https://doi.org/10.1109/VLHCC.2017.8103446

[44] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2016. The Emerging Role of Data Scientists on Software Development Teams. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 96–107. https://doi.org/10.1145/2884781.2884783

[45] Donald E. Knuth. 1984. Literate Programming. *Comput. J.* 27, 2 (May 1984), 97–111. https://doi.org/10.1093/comjnl/27.2.97

[46] Andrew J. Ko, Robin Abraham, Laura Beckwith, Alan Blackwell, Margaret Burnett, Martin Erwig, Chris Scaffidi, Joseph Lawrance, Henry Lieberman, Brad Myers, Mary Beth Rosson, Gregg Rothermel, Mary Shaw, and Susan Wiedenbeck. 2011. The State of the Art in End-user Software Engineering. *ACM Comput. Surv.* 43, 3, Article 21 (April 2011), 44 pages. https://doi.org/10.1145/1922649.1922658

[47] Andrew J. Ko, Brad A. Myers, and Htet Htet Aung. 2004. Six Learning Barriers in End-User Programming Systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing (VLHCC '04)*. IEEE Computer Society, Washington, DC, USA, 199–206. https://doi.org/10.1109/VLHCC.2004.47

[48] J. Lave and E. Wenger. 1991. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press.

[49] Steve Lohr. 2017. Where the STEM Jobs Are (and Where They Aren't). New York Times.

[50] Geraldine Mason and Annette Jinks. 1994. Examining the role of the practitioner-teacher in nursing. *British Journal of Nursing* 3, 20 (1994), 1063–1072. https://doi.org/10.12968/bjon.1994.3.20.1063 arXiv:https://doi.org/10.12968/bjon.1994.3.20.1063 PMID: 7827455.

[51] Justin Matejka and George Fitzmaurice. 2017. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics Through Simulated Annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1290–1294. https://doi.org/10.1145/3025453.3025912

[52] Amelia Ahlers McNamara. 2015. *Bridging the gap between tools for learning and for doing statistics*. Ph.D. Dissertation. UCLA.

[53] Kevin Mickey. 2013. The best teacher is a practitioner. http://polis.iupui.edu/index.php/the-best-teacher-is-a-practitioner/.

[54] Lijun Ni. 2011. *Building Professional Identity As Computer Science Teachers: Supporting High School Computer Science Teachers Through Reflection and Community Building*. Ph.D. Dissertation. Atlanta, GA, USA. Advisor(s) Guzdial, Mark. AAI3500584.

[55] Lijun Ni and Mark Guzdial. 2012. Who AM I?: Understanding High School Computer Science Teachers' Professional Identity. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, New York, NY, USA, 499–504. https://doi.org/10.1145/2157136.2157283

[56] NIST.gov. 2018. Engineering statistics handbook: Measures of Skewness and Kurtosis. https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm. Accessed: 2018-09-20.

[57] Natasha Noy. 2018. Making it easier to discover datasets. https://www.blog.google/products/search/making-it-easier-discover-datasets/. Accessed: 2018-09-20.

[58] The University of Michigan. 2018. ICPSR Timeline. https://www.icpsr.umich.edu/icpsrweb/content/about/history/timeline.html. Accessed: 2018-09-20.

[59] American Association of University Professors. 2018. Professors of Practice. https://www.aaup.org/report/professors-practice. Accessed: 2018-09-20.

[60] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. 2013. Success in Introductory Programming: What Works? *Commun. ACM* 56, 8 (Aug. 2013), 34–36. https://doi.org/10.1145/2492007.2492020

[61] Bina Ramamurthy. 2016. A Practical and Sustainable Model for Learning and Teaching Data Science. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16)*. ACM, New York, NY, USA, 169–174. https://doi.org/10.1145/2839509.2844603

[62] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for All. *Commun. ACM* 52, 11 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779

[63] Reudismam Rolim, Gustavo Soares, Loris D'Antoni, Oleksandr Polozov, Sumit Gulwani, Rohit Gheyi, Ryo Suzuki, and Björn Hartmann. 2017. Learning Syntactic Program Transformations from Examples. In *Proceedings of the 39th International Conference on Software Engineering (ICSE '17)*. IEEE Press, Piscataway, NJ, USA, 404–415. https://doi.org/10.1109/ICSE.2017.44

[64] Adam Rule, Aurélien Tabard, and James D. Hollan. 2018. Exploration and Explanation in Computational Notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 32, 12 pages. https://doi.org/10.1145/3173574.3173606

[65] Judith Segal. 2007. Some Problems of Professional End User Developers. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC '07)*. IEEE Computer Society, Washington, DC, USA, 111–118. https://doi.org/10.1109/VLHCC.2007.50

[66] Kent Smith. 2013. A Brief History of NCBI's Formation and Growth. https://www.ncbi.nlm.nih.gov/books/NBK148949/. Accessed: 2018-09-20.

[67] Sarah L.R. Stevens, Mateusz Kuzak, Carlos Martinez, Aurelia Moser, Petra M. Bleeker, and Marc Galland. 2018. Building a local community of practice in scientific programming for Life Scientists. *bioRxiv* (2018). https://doi.org/10.1101/265421

[68] Allison Elliott Tew and Mark Guzdial. 2011. The FCS1: A Language Independent Assessment of CS1 Knowledge. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*. ACM, New York, NY, USA, 111–116. https://doi.org/10.1145/1953163.1953200

[69] Rachel Treisman. 2017. Yale to offer new major in data science. http://yaledailynews.com/blog/2017/03/08/yale-to-offer-new-major-in-data-science/.

[70] Alexa Vanhooser. 2018. UC Berkeley announces data science pipeline program for students. The Daily Californian.

[71] Allegra Via, Thomas Blicher, Erik Bongcam-Rudloff, Michelle D. Brazas, Cath Brooksbank, Aidan Budd, Javier De Las Rivas, Jacqueline Dreyer, Pedro L. Fernandes, Celia van Gelder, Joachim Jacob, Rafael C. Jimenez, Jane Loveland, Federico Moran, Nicola Mulder, Tommi Nyronen, Kristian Rother, Maria Victoria Schneider, and Teresa K. Attwood. 2013. Best practices in bioinformatics training for life scientists. *Briefings in Bioinformatics* 14, 5 (2013), 528–537.

[72] Clifford H Wagner. 1982. Simpson's paradox in real life. *The American Statistician* 36, 1 (1982), 46–48.

[73] April Y. Wang, Ryan Mitts, Philip J. Guo, and Parmit K. Chilana. 2018. Mismatch of Expectations: How Modern Learning Resources Fail Conversational Programmers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 511, 13 pages. https://doi.org/10.1145/3173574.3174085

[74] Hadley Wickham. 2014. Tidy Data. *Journal of Statistical Software* 59, 1 (2014), 1–23. https://doi.org/10.18637/jss.v059.i10

[75] G. Wilson. 2006. Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive. *Computing in Science Engineering* 8, 6 (Nov 2006), 66–69. https://doi.org/10.1109/MCSE.2006.122

[76] Greg Wilson. 2018. End-User Teachers. http://third-bit.com/2018/06/20/end-user-teachers.html. Accessed: 2018-09-01.

[77] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. 2015. The Emergence of GitHub As a Collaborative Platform for Education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 1906–1917. https://doi.org/10.1145/2675133.2675284

[78] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *IEEE International Conference on Computer Vision*.