UNIVERSITY OF CALIFORNIA SAN DIEGO

Prototyping the Developer Experience for Data Science Practitioners and Instructors

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Cognitive Science

by

Sean Michael Kross

Committee in charge:

Professor Philip J. Guo, Chair
Professor Shannon E. Ellis
Professor James D. Hollan
Professor Julian McAuley
Professor Bradley Voytek

2022

The Dissertation of Sean Michael Kross is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2022

DEDICATION

For Toby Kross, Fred and Renee Kramer, and Jeffrey Elman

EPIGRAPH

The main business of humanity is to do a good job of being human beings,
not to serve as appendages to machines, institutions, and systems.

—From *Player Piano* by Kurt Vonnegut

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I have been supremely blessed with the support of many collaborators, mentors, family, and friends. I would not be here today without the daily encouragement of my parents Marc and Donna, and my brother Max. Philip, I feel like I owe you the kind of debt that a child feels they owe to a parent. I moved to California just to work with you, and our relationship and the work we have been able to accomplish together has exceeded my wildest dreams. Jim, Brad, Shannon, and Julian: I am so lucky to have each of you on my committee, I could not ask for a better assembly of role models in science. I have benefitted from the mentorship of so many incredible researchers who I would like to thank in chronological order: Ashley Bate, Patrick Eichenberger, Joseph Paulson, Mihai Pop, Elissa Redmiles, Michelle Mazurek, Jeffery Leek, Roger Peng, Brian Caffo, Jake Hofman, Daniel Goldstein, Eszter Hargittai, and Thomas Ball. I would also like to thank several other scholars who have significantly influenced my thinking including Lilly Irani, Federico Rossano, Julia Lowndes, Ian Drosos, Sam Lau, Dorothy Howard, Amy Fox, Stephan Kaufhold, Catherine Hallsten, Emily Winokur, Victor Gandarillas, Mateus Guzzo, and Alexandra Neuman. Finally I would like to thank the National Science Foundation and the Alfred P. Sloan Foundation for their generous financial support.

Chapter 3, in full, is a reprint of the material as it appears in the proceedings of the 2019 CHI Conference on Human Factors in Computing Systems as Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. Sean Kross and Philip J. Guo. 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in the proceedings of the 2019 IEEE Symposium on Visual Languages and Human-Centric Computing as End-User Programmers Repurposing End-User Programming Tools to Foster Diversity in Adult End-User Programming Education. Sean Kross and Philip J. Guo. 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, is a reprint of the material as it appears in the proceedings of the 2021

ACM Conference on Computer Supported Cooperative Work as Orienting, Framing, Bridging, Magic, and Counseling: How Data Scientists Navigate the Outer Loop of Client Collaborations in Industry and Academia. Sean Kross and Philip J. Guo. 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in part, is a reprint of the material as it appears in the proceedings of the 2021 CHI Conference on Human Factors in Computing Systems as as Datamations: Animated Explanations of Data Analysis Pipelines. Xiaoying Pu, Sean Kross, Jake Hofman, Daniel Goldstein. 2021. The dissertation author was one of the primary investigators and author of this paper.

VITA

2012	Bachelor of Arts in Biology, New York University
2015	Bachelor of Science in Computer Science, University of Maryland College Park
2020	Master of Science in Cognitive Science, University of California San Diego
2022	Doctor of Philosophy in Cognitive Science, University of California San Diego

PUBLICATIONS

Sean Kross, Philip J. Guo. 2021. Orienting, Framing, Bridging, Magic, and Counseling: How Data Scientists Navigate the Outer Loop of Client Collaborations in Industry and Academia. In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 2021).

Xiaoying Pu, Sean Kross, Jake M. Hofman, Daniel G. Goldstein. 2021. Datamations: Animated Explanations of Data Analysis Pipelines. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2021).

Sean Kross, Philip J. Guo. 2019. End-User Programmers Repurposing End-User Programming Tools to Foster Diversity in Adult End-User Programming Education. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2019).

Sean Kross, Philip J. Guo. 2019. Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2019).

ABSTRACT OF THE DISSERTATION

Prototyping the Developer Experience for Data Science Practitioners and Instructors

by

Sean Michael Kross

Doctor of Philosophy in Cognitive Science

University of California San Diego, 2022

Professor Philip J. Guo, Chair

Data science encompasses the most prominent collection of methods for creating scientific knowledge in the 21st century. Currently, data scientists must navigate a wide-ranging and often incoherent ecosystem of tools, in addition to organizing sociotechnical interactions with colleagues across many fields of expertise.

This predicament motivates my thesis: The elements of data science work that are based in human expertise and social relationships must be integrated into existing programming workflows to create the developer experience that data scientists require to be successful.

This dissertation supports my thesis by presenting three empirical studies and two tools. First, I investigated how professional data scientists teach novices about data science focused

programming workflows, including how to adapt software development tools to their work, how to navigate the full depth of the stack of technologies that data science relies on, and how to use their tools to help communicate their findings. Then I explored how a team of academic data scientists repurposed the tools from their everyday data science work to create a data science course designed to reach traditionally underrepresented groups in computing. Finally, I examined how consulting data scientists interact with their clients, how their working relationships take them beyond well-characterized programming-oriented cycles, and how they achieve success by integrating designerly work into their data analysis process.

These studies inspired me to develop two tools: 1. Datamations animates each step in a data analysis pipeline via transitions that show how rows, columns, and cells move within a data frame. 2. Tidy Data Tutor creates step-by-step interactive illustrations for a data analysis pipeline, so that every individual cell can be tracked.

The main research findings of this dissertation are that data scientists adapt software engineering tools to fit into their own workflows, and that data scientists must communicate the uncertainty that they face in their work to novices. Additionally, this dissertation found that several nested cycles are required for data scientists to achieve success in collaboration with their colleagues. Finally, my prototype tools showed that animations and illustrations derived from data wrangling code can help convey a clearer understanding of data analysis pipelines.

Chapter 1

Introduction

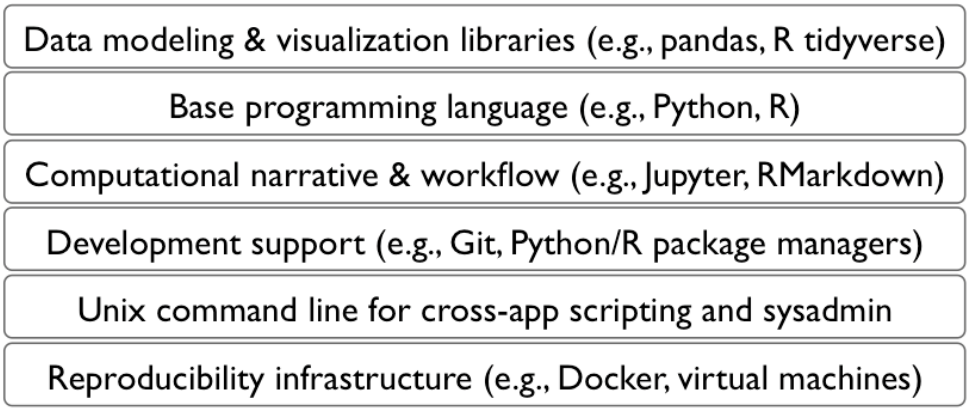


Figure 1.1. The full technology stack that data scientists must navigate in their day-to-day work, as characterized in my CHI 2019 study [138]. In Chapter 3 this dissertation discusses how data scientists teach novices to manage the interactions between these different layers.

Data analysis is the engine of scientific progress in the 21st century, driven by the availability of existing data, the relative ease with which new data can be collected, increasingly inexpensive computing infrastructure, and the growing number of people with the skills required to competently perform data analysis. A prominent mode of doing data analysis is via *data science*, a process where data are explored, transformed, and analyzed by writing computer code [65, 226]. The requirements for doing data science work are comparable to software engineering: as Figure 1.1 shows, practitioners need a sophisticated understanding of their computing environment, the code that they are writing, preexisting code they are using, and the data objects that they are manipulating [91]. However, unlike software engineers, data

scientists are most often not building production-grade computer systems. Instead they are building *data products* — the artifacts that are produced during a data analysis that are often for the consumption of others, like data analyses, data graphics, and new datasets [196].

Serious attention has been paid to building tools for software engineers to help them with their jobs [146, 166], and in the past decade the study of user experience design has been applied more prominently to the experience of using software development tools. The study of *developer experience* therefore focuses on the interactions between software developers and their tools [78]. This includes evaluating the extent to which different tools increase programmer productivity [160], imagining new representations to improve programmers' understanding of complex software systems [92], and better characterizing programmer's emotional reactions to features of different tools [83]. The rapidly evolving developer experience landscape over the past decade includes coding environments like the Microsoft Visual Studio Code Extension platform, artificial-intelligence-based tools like GitHub Copilot for context-aware code completion, and developer-focused integrations into productivity applications like Slack, Jira, and Confluence. Each of these systems required significant corporate investment in either open source community building (in the case of VS Code), large language models built with deep learning (in the case of GitHub Copilot), or bringing an awareness of developer needs into traditional business applications (in the case of Slack, etc).

In contrast, even though the number of data scientists has grown significantly in the last decade, **there has been no proportional proliferation of tools that cater specifically to the developer experience of data scientists.** There are IDEs and notebooks like RStudio and Jupyter, however the open source ecosystem of Jupyter extensions is largely developed by individual hobbyists and academics, with relatively little corporate or institutional investment [141].

In this way, the experience of a modern data scientist in 2022 is similar to the experience of being a software engineer ten or twenty years ago. Some of the most valuable resources are in blog posts, technical documentation, and question and answer forums [138] but little

Chapter	Contributions	Findings
Chapter 3: Practitioners Teaching Data Science in Industry and Academia	• An analysis of 20 interviews with data science practitioner-instructors. • A synthesis of technical workflows that data science practitioners teach to novices.	• Data scientists' workflows are adapted to confront the significant amount of uncertainty in their work. • Novice data scientists must learn to navigate the full breadth of the technology stack.
Chapter 4: Repurposing End-User Programming Tools to Foster Diversity in Adult Data Science Education	• Results from 9 interviews with a team of academic data scientists and how they repurposed their tools to create a data science course targeting traditionally underrepresented groups in computing.	• Data scientists feel empowered to adapt tools that are a part of the their regular research workflow to new tasks. • This comes as second-nature since the tools available to them don't fully meet their needs.
Chapter 5: How Data Scientists Navigate the Outer Loop of Client Collaborations in Industry and Academia	• A study of 10 consulting data scientists and their interactions with clients. • A perspective on data science workflows that goes beyond programming-oriented cycles.	• There are several nested cycles of work that data scientists must regularly engage with for the sum of their work to be successful. • These cycles largely depend on designerly interactions with colleagues.
Chapter 6: Tools for Visualizing Data Analysis Pipelines (Datamations and Tidy Data Tutor)	Two interactive systems deployed online: • Datamations creates animations from data analysis pipeline code. • Tidy Data Tutor interactively visualizes each step in a data analysis pipeline.	• Animations of data analysis pipelines may help people recognize paradoxical relationships within a dataset. • Visualizing transformations of a data frame can be useful for debugging and teaching.

Figure 1.2. An overview of the structure of this dissertation.

of this knowledge or awareness of important and common challenges has been integrated into production-grade data science developer tools in a way that is interactive or intelligent. Unlike the software developer experience, there is not enough awareness of what issues are important to data scientists, and therefore there is currently little guidance about where significant investments should be made in the developer experience of data scientists.

Reflecting on the state of the art as someone who has worked as a data scientist, an educator who has co-developed two Coursera massively open online course specializations [139], and a researcher of data scientists and their practices throughout my PhD, I have arrived at the following point of view, my thesis statement:

> *The elements of data science work that are based in human expertise and social relationships must be integrated into existing programming workflows to create the developer experience that data scientists require to be successful.*

My dissertation supports this thesis statement by laying the foundation for understanding what data scientists need in their own developer experiences. This is accomplished with a series of interview studies with data science practitioners about their day-to-day work, analyses of the requirements that data science instructors have for their classrooms, and the development

of interactive systems focused on helping data scientists understand data-science-centric code. Taken together, this dissertation makes the following contributions:

- Chapter 3 explores the central concerns of twenty working professional data scientists while they are teaching data science in different modalities. This includes an analysis of how they teach students to adapt software development tools like Git to help them in their data science careers, how they make decisions about much computer system administration knowledge they think their students should have, and how to cope with the uncertainty of ever-changing software library APIs and code that is often not straightforward to understand. This work shows how data scientists must adapt existing software engineering focused tools to fit their own workflows, and it details how professional data scientists communicate the incongruity between their needs and the available tools to newcomers to data science.

- Chapter 4 is a deeper examination of some of the personas discussed in Chapter 3: it describes how a team of academic data scientists repurposed the data science tools they use in their day-to-day work and built new tools because of their dissatisfaction with the available tools for creating data science focused learning materials. They claim that this transformation to their workflow was the only way they could build and maintain a sequence of data science courses designed specifically for adults from populations that are not traditionally reached by computing education efforts. This chapter shows how academic data scientists had to draw on their previous experience doing reproducible research to string together a solution for doing good in their community and satisfying their own engineering needs.

- Chapter 5 shifts focus from practitioner-instructors to prioritize the experience of practitioners with an investigation of how data scientists collaborate with their clients via a series of interviews with ten data scientists who specialize in consulting work. The cyclical nature of data science work has previously been explored in detail as a process that revolves

around programming tasks like importing, cleaning, modeling, and visualizing data. I describe this process as an "inner loop" where most modern programming innovations are currently focused, in contrast to an "outer loop" of interpersonal and technical interactions that I characterize in this chapter. This outer loop includes the stages of groundwork, orienting, problem framing, bridging the gap, magic, and counseling. Each of these stages adds context to decisions that a data scientist must make while doing their programming work. This study found that current developer experience tools do little to incorporate these nuances into a data scientist's final work product.

- Chapter 6 draws considerable inspiration from opportunities identified in the preceding studies and proposes two prototype systems for improving the developer experience of data scientists, specifically for visualizing data science code: Datamations and Tidy Data Tutor. Both systems use data from the user's computing environment and the user's own data pipeline code to build a visualization. Datamations animates each step of every transformation being applied to a data frame in a data analysis pipeline. The animation is meant help a data scientist understand what is happening inside of a data analysis pipeline, especially in the case where they might want to modify an existing pipeline that they are not familiar with. Tidy Data Tutor illustrates how each individual cell in a data frame is rearranged or transformed according to each step in a data analysis pipeline. Tidy Data Tutor achieves this by displaying a "before" and "after" state of the data frame for each pipeline step. It then annotates these two data frames by drawing arrows, coloring cells, or applying other kinds of markup.

- Chapter 7 concludes with a discussion of the implications that this dissertation has for new representations, educational opportunities, and novel programing paradigms in data science.

Chapter 2

Related Work

This dissertation was inspired by prior research into understanding how data scientists work (Section 2.1), new methods for data and code visualization (Section 2.2), and explorations of data science and computing education (Section 2.3). Data science is still an emerging field and data scientists have a broad scope of professional responsibilities, therefore a portion of this prior work that I review in this chapter attempts to define the boundaries of what kinds of work should be considered *data science*. My dissertation work has also been influenced by research into data science education, since skills that are taught as data science should be fundamental to the kinds of work data scientists are expected to perform. Developing impactful interventions, mental models, and computational representations to improve the developer experience of data scientists requires an understanding of the boundaries of data science work. By focusing on the common experiences of data scientists, this related work provides insight into opportunities for a data scientist's expertise and relationships to be integrated into their developer experience.

2.1 Data Science as a Practice

This line of research addresses who data scientists are, patterns in their professional and educational backgrounds, and what data scientists do in their day-to-day work. This includes understanding what technologies data scientists commonly use, how their work is organized, and how they interact with their colleagues. This work especially informs Chapter 3 and Chapter 5.

2.1.1 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 [101, 125]. Although data visualization is a highly active area of HCI research, what is more relevant to this work 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 [125]. Many of them write code in languages such as Python and R [44, 58, 101, 110], 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 [136] 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. [128, 129]) and research programming (from Guo's dissertation [91]).

However, 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 3.1). In the terminology of Ko et al., they are now engaging in *end-user software engineering* [136] with more of an emphasis on code quality and reuse; in Segal's related terminology, data scientists are now becoming *professional end-user developers* [202]. Along these lines, software engineering researchers such as Kim et al. have studied the role of data scientists within industry engineering teams [131].

2.1.2 Types of Data Scientists

We first provide context for our study in Chapter 5 by specifying the types of data scientists that we focus on. We focus specifically on data scientists who work with clients to provide analytical services for them to make data-informed decisions. In industry these clients may come from other parts of the company (e.g., a data scientist may work with the

marketing growth team for a new product line) or from outside (e.g., external business partners). In academia, data scientists are usually in 'soft money' positions (e.g., postdoc or research staff) where their salaries are funded by collaborative grants with various PIs (usually faculty); these academic data scientists may need to juggle multiple grant-funded projects to pay their salaries and thus concurrently work with several clients in different subfields. Client project engagements usually last from a few months to a few years.

This type of client-oriented role is common throughout both industry and academia. For instance, Zhang et al. [235] found five distinct roles on data science teams in industry, including communicator, manager/executive, domain expert, researcher/scientist, and engineer/analyst/programmer. Under this taxonomy, the data scientists we study are 'analysts' whose clients are managers/executives and domain experts. Within academia, Mao et al. [154] studied the collaboration between data scientists and their clients who were biomedical researchers. Kim et al. [131] further honed in on five types of working styles of industry data scientists: insight providers, modeling specialists, platform builders, polymaths, and team leaders. The data scientists we study are mostly insight providers who *"with a strong background in statistics [their] main task is to generate insights and to support and guide their managers in decision making."* [131] Similarly, using a broader survey of 19,000 data scientists on Kaggle, Hayes found that the most common role was to *"analyze and understand data to influence product or business decisions"* (63% of respondents) [103]. Some industry practitioners also use the term 'Type-A' [187, 60] data scientist to denote this type of *Analyst* role.

In contrast, other types of data scientists (whom we did not study) function more like software engineers who build data-driven platforms and apply machine learning to create predictive models. Some still work with clients, but others have more fixed long-term roles to build specific software infrastructure, such as Netflix's movie recommendation engine. In Kim et al.'s taxonomy [131], they are platform builders and modeling specialists. In Zhang et al.'s taxonomy [235], these are researchers/scientists (usually with a machine learning background). Hayes found that 35% of Kaggle survey respondents built machine learning infrastructure [103].

And some practitioners use the term 'Type-B' [187, 60] data scientist to denote this type of *Builder* role.

2.1.3 Inner-Loop Collaborative Data Science Workflows

The bulk of research around data science centers on the technical workflow shown in Figure 5.1. Much of this research involves tools to facilitate all of these workflow stages, such as data wrangling systems [124, 94], computational notebooks [141], coding assistance tools [72], and data visualization interfaces [233, 197]. This dissertation does not explicitly focus on tool use, although we discuss implications for collaborative tool design in Section 5.9.5. Rather, we studied how data scientists and their clients collaborate, so the closest related work to ours are those that study data science collaborations.

In industry, two notable studies of data science collaboration were both done at large multinational technology companies: Zhang et al. conducted a survey of 183 IBM employees who worked in data science related roles (whom they call *data science workers* [165] to emphasize that people with varying job titles work closely with data) to investigate collaboration practices [235]. Kim et al. interviewed 16 data scientists at Microsoft about how they collaborate with software engineering teams in particular [131]. In academia, Mao et al. interviewed 22 collaboration participants containing a mix of data scientists and biomedical domain scientists working together on research projects.

These studies all took a holistic approach of characterizing many facets of collaboration, including who usually collaborated with whom, what roles each collaborator played, how multi-domain teams were structured, and what tools they used to work together. They provide important context for the types of data scientists that we study; as we discussed in Section 2.1.2, we focus on the 'analyst' persona from Zhang et al. [235] and the 'insight provider' persona from Kim et al. [131].

Most notably, these studies focused on how collaboration took place within the inner-loop technical workflow stages of Figure 5.1: Kim et al.'s study listed stages such as collecting data,

cleaning data, building predictive models, hypothesis testing, and operationalizing models [131]. Zhang et al.'s study referenced the workflow in Figure 5.1b and emphasized collaboration within this workflow in its paper abstract: *"We found that data science teams are extremely collaborative and work with a variety of stakeholders and tools during the six common steps of a data science workflow (e.g., clean data and train model)."* [235] Mao et al. [154] presented a similar inner-loop technical workflow diagram for academic collaborations between biomedical scientists and data scientists, shown in this chapter as Figure 5.1c.

This dissertation complements and extends this lineage of work by zooming out of the usual technical workflow stages to consider how data scientists interface with their clients at a higher level both before and during an analysis. While prior studies characterized the day-to-day technical aspects of collaborative work (e.g., who takes on what roles, what tools they use), this dissertation focuses on the *strategic and emotional considerations* of how data scientists collaborate with clients throughout the lifetime of a project. ***One of our contributions is to synthesize a more general workflow showing an outer loop of analyst-client interactions*** (Figure 5.2) that progresses from laying the groundwork before an analysis even begins, to orienting, to problem framing, to doing "magic" (which is what many of our participants' clients referred to their technical work as), to helping clients cope with sub-optimal analysis results. The inner-loop technical workflows that prior studies were all situated within appear inside the "magic" node of our more general workflow diagram.

2.1.4 How Data Scientists Work with Clients

Other projects related to work presented in this dissertation are those that not only study general data science collaborations (see prior section) but that also zoom in on how data scientists work with clients.

Mao et al. interviewed 16 data scientists and 6 of their clients who were biomedical scientists in academia [154]. They analyzed client interactions through the theoretical lens of the Olsons' framework from *Distance Matters* [175] and found a variety of challenges in building

common ground in these collaborations. This dissertation corroborates some of their findings, especially as expressed in the Problem Framing and Bridging the Gap stages of our outer-loop workflow (Section 5.5). Specifically, data scientists and their clients face a tension when defining the actual problem to solve and need to balance both knowledge from general data-analytic methods and from the target domain. Mao et al. elegantly expressed this tension as one between 'find the right answer' versus 'ask the right question.' We add to their contributions here by making a connection from data science to what professional designers do in the design thinking process [55, 56] when iteratively framing the problem. In Section 5.9.1, we further contextualize our findings in relation to this work.

This dissertation extends the scope of Mao et al.'s work in several main ways. First, we characterize the legwork that a data scientist must do *before a collaboration even begins* (the Groundwork stage in Section 5.3) as well as the five main ways in which they enter into collaborations (the Orienting stage in Section 5.4). In contrast, Mao et al.'s study focuses mostly on collaborations that are already underway. That said, they do discuss aspects of collaboration readiness such as motivations for collaborating on data-centric open science projects in academia. We extend those findings by detailing how data scientists build reputation and trust in themselves and then use five distinct entry points (Section 5.4) to enter into collaborations throughout large companies, startups, and academic institutions. Next, we surfaced the variety of emotional labor [90] involved in data scientists doing problem framing (Section 5.5.1) and counseling clients (Section 5.7) when results look unfavorable, especially when there are power imbalances. Furthermore, this dissertation covers different but complementary topics as Mao et al. in terms of the 'inner loop' when a collaboration is well underway: Mao et al. use the Olsons' framework to characterize technology readiness of how specific software tools can mediate analyst-client interactions. This dissertation covers new topics such as how data scientists handle the emotional reactions of clients to their technical 'magic' work in the Magic and Counseling stages (Sections 5.6 and 5.7, respectively). In sum, this dissertation augments Mao et al.'s findings by synthesizing an end-to-end workflow of how data scientists lay groundwork, enter

into collaborations, and emotionally interface with clients.

Another related study was done by Hou and Wang, who performed observations and interviews at two civic data hackathons that connected data scientists with clients from nonprofit organizations (NPOs) [115]. They found that having a group of student volunteers serve as "brokers" (i.e., intermediaries) between data scientists and NPO representatives helped the two groups work better together, especially during the short timespans of hackathons that last for one or two days. The settings we studied differ in that data scientists collaborate with clients over longer time frames (a few months or years) in a variety of industry and academic settings, rather than in a short hackathon with NPOs. Also, it is up to the data scientist themselves to "broker" those client interactions without third-party involvement; we depict this brokering activity as the Bridging the Gap stage in Figure 5.2 (Section 5.5.2). Lastly, we cover how data scientists enter into such collaborations in the groundwork and orienting stages, whereas Hou and Wang focus on what occurs when stakeholders (e.g., NPOs, client teams, and data scientists) have already been established for the hackathon event.

Related to above, more distantly-related CSCW studies of data scientists working with open government data for social good [51] and crisis-response scenarios [109] also reveal the importance of broker roles in establishing common ground between collaborators. However, those study settings are more decentralized, often rely on part-time and volunteer workers, and involve a vast array of stakeholders, in contrast to the tightly-focused relationship between a data scientist and their client that we study in this chapter. Similarly, CSCW studies of large-scale, multi-disciplinary, geographically-distributed scientific collaborations (sometimes called e-Science, e-Research, or collaboratories) reveal the socioemotional challenges of remotely coordinating domain scientists and software developers in high-performance computing [120, 143, 176]. The work in this dissertation differs in that it focuses on finer-grained direct interactions between a data scientist and their client in a colocated setting. In Section 5.9.1 we further discuss our findings in relation to these lines of work.

2.1.5 Other Types of Consultative Work with Clients

Zooming out beyond data science, this dissertation continues the lineage of CSCW research on consultative work involving clients in other fields. The closest analogue is design consulting, where UX or product designers are brought in to help ideate and prototype with various business stakeholders [55, 56, 89]. Similar to our data scientists, design consultants must lay groundwork to find clients (Section 5.3) and engage in extensive problem framing (Section 5.5.1) before even starting to prototype. However, data science work has a larger variety of entry points (Section 5.4) and involves techniques that are more opaque (Section 5.6) than design work, whose processes and outputs are usually visible to clients. Our setting also shares some similarities with I.T. (information technology) consulting, such as Stager's study at a university computing center [207] and Lending and Dillon's work in industry [145]; in particular, I.T. consulting also involves groundwork and problem framing, as *"users often do not know what type of information or services a consultant is able to provide"* [207]. More distantly-related are business and management consulting, which often involve training consultants to empathize with clients [212] and using workflow tools to coordinate business metrics and deliverables [158]; data science work shares some similarities but also involves more perceived technical 'magic' (Section 5.6) than business consulting, which is more easily understandable by clients who are business stakeholders. Finally, since some branches of data science come from academic statistics departments [65], statistics consultants within universities are similar to modern-day data scientists in some ways since they help researchers to design experiments, apply the appropriate statistical tests, and interpret results [43, 81, 150]; however, statisticians usually work with more fixed data sets and engage in less data wrangling and programming work than data scientists do. Overall this dissertation differentiates itself from prior work in other fields by revealing domain-specific insights about how data science consultants work with clients and synthesizing them into an end-to-end workflow.

2.2 Representations of Data Science Artifacts

Embedded within data science work practices are the artifacts that data scientists produce and use for themselves to understand their own work and to communicate the results of their work to others. The outputs of data science work are often complex and are historically aided by data visualizations and data tables, however these outputs often fail to illustrate the analysis process and the decisions that were made that lead to their creation. As sharing data, code, and specifications for computing environments has grown easier, prior work has tried to integrate those variables into visual data representations. The prior work presented in this section is most relevant to Chapter 6.

2.2.1 Understanding Data and Analysis Behind Visualizations

When presented with a plot or table it can often be difficult to understand what led to the results encoded in that figure. Information about the raw data and the analysis pipeline are often placed elsewhere, in the form of code and written paragraphs. The disconnect between data representations and analysis is evident in Rule *et al.* [195], where users (data analysts) reportedly share their data analysis results as emails and slides, excluding the computational notebooks that generated the findings [195]. When users lack easy access to the context of data, they can misunderstand the data patterns they see. A concrete data scenario is Simpson's paradox, where data trends in one grouping contradicts trends in another grouping [177]. Without explicit efforts such as a detection algorithm [95], users may miss out on such aspects of the data or struggle to understand these apparent contradictions [178]. In Chapter 6 we introduce Datamations and Tidy Data Tutor to enhance data frames with details from the data analyses that generated them, and test their effectiveness using an instance of Simpsons' paradox.

2.2.2 Using Animations to Communicate Data

By using animations to explain plots and tables we hope to take advantage of their visual appeal and potential explanatory benefits. As one example, the GapMinder[1] animation communicates how life expectancy changed throughout the world in an engaging and informative manner[82]. Fisher suggests that such animations can be broadly useful when they follow the appropriate design principles [82]. By adding animated transitions between statistical graphics, Heer & Robertson find improvements in graphical perception in two ways—tracking objects and estimating changes [105]. Tracking objects—in our case data points in a table—can be essential for understanding data analyses, as the user may want to know which data points are related to which results. In addition, users have been found to prefer animations over their static counterparts [133, 27]. Hypothetical outcome plots (or HOPs) are one recent and promising example of using animations to add to the information shown in static plots and boost reader comprehension [123, 117]. Specifically, HOPs augment static visualizations such as error bars with animated frames of random draws from the underlying sampling distribution. With HOPs, animation conveys randomness and uncertainty in final results, but unlike Datamations, the goal of HOPs is not to communicate the underlying data analysis pipeline that led to these results.

With Datamations and Tidy Data Tutor we aim to surface visually the *entire* analysis pipeline behind how data frames are created. The high-level idea behind both systems is similar to that of programs that teach complex systems and algorithms, as reviewed in Tversky *et al.* and Hundhausen *et al.* [118, 216], but with a more specific scope and different goal. As we discuss in Section 6.2, Datamations and Tidy Data Tutor communicate a particular data analysis process with concrete steps and visual analogies. Instead of helping readers learn abstract algorithms in an educational setting, both systems are meant to help people understand specific analysis results.

Since creating animated transitions can be low-level and time consuming, previous work

[1] gapminder.org

has developed ways to automate the creation of these transitions. Kim & Heer [132] propose a visualization grammar and a recommender system on the visual mark level (*cf.* the Grammar of Graphics [227]) to augment specification. In addition, Drucker & Fernandez design a framework for transitioning between unit visualizations [73], and Canis is a declarative language for creating chart animations in SVG [87]. These approaches for automating animated transitions tend to work with a single data state and do not explicitly consider the semantics of animation such as aggregation or grouping. Our efforts on Datamations and Tidy Data Tutor take inspiration from and extend this work: We outline a general purpose framework for communicating entire analysis pipelines, such that there is a direct mapping from code that executes an analysis to an animation that helps to explain it.

2.2.3 Probing the Data Analysis Pipeline

Datamations animate sequential, separable data operations, each of which can be thought of as a step in a "pipeline", where verbs operate on the data at each step and pipes chain together the results of each operation. Here we present an implementation of Datamations centered around the programming language R, which has recently seen broad adoption of the pipeline paradigm through packages such as `magrittr` (which creates an explicit pipe operator within R, denoted by `%>%`) [28] and verbs to operate on data within a pipeline through the `tidyverse/dplyr` package [225]. That said, the pipeline approach is a general purpose one inspired by relational algebra [224] and has been adopted in systems ranging from Unix to Javascript [106]. As a result our work on Datamations is applicable to this entire range of data analysis frameworks.

Ordinarily, data analysis pipelines do not directly expose the intermediate data states involved in an analysis. To this end, there are existing solutions for probing pipelines, such as text-based tools [183, 71, 193] and static illustrative comics [222] to surface intermediate results. Furthermore, a grammar for "data tweening" has been proposed to generate easy-to-follow, static data table visualizations to step through and explain database queries [130]. Our work on Datamations builds on these efforts by presenting a framework to automatically generate visually

compelling animations that achieve similar goals.

2.2.4 Provenance

Our efforts on visualizing data analysis pipelines are also related to work on visualizing *data provenance* [188]. Provenance is a broad research topic on the history of changes in the process of analysis and the creation of visualizations [188], and Chevalier *et al.* identify the use of animation as "the most under-explored" for replaying and summarizing history (provenance) [49].

The research community has designed many systems that capture and communicate data provenance during visual exploratory analysis and present it to users [204, 75, 203, 122, 39, 57]. These provenance-capable systems focus on preserving and presenting all the *alternatives histories* of user action. As a result, such systems often visualize provenance as abstract node-link diagrams [108]: the nodes represent data states, and forking links represent alternative actions or transitions between states. Partly due to the spatial layout of node-link diagrams, the exact data actions are often presented abstractly, in text or as glyphs [108]. In comparison, Datamations and Tidy Data Tutor only communicate *one version* of an analysis pipeline, but emphasize more of the semantics of analysis operations ("verbs") in the corresponding animations. In addition, most visual provenance tools operate within visual analytics interfaces and lack the flexibility of Datamations and Tidy Data Tutor, which translate potentially more complex code to animations.

2.3 Data Science and Computing Education

The importance of useful representations of data science artifacts is especially important in data science education, where visualizations and diagrams are key tools for shedding light on challenging concepts. Prior work about what topics have been emphasized in data science education can illuminate what skills and technologies data scientists believe are important for students to know. Examining where earlier work describes the focus of data science programs and understanding how data scientists are adapting existing patterns in computing education to

data science classrooms can provide insight for where meaningful interventions can be made for improving the experience of doing data science work. This prior work is particularly relevant to the next two chapters, Chapter 3 and Chapter 4.

2.3.1 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 [217, 215], research labs are organizing hands-on workshops [229], and MOOCs and coding bootcamps focused on data science are some of the most popular offerings [9, 10]. But despite this growing interest over the past few years, there is still little agreement on what a data science curriculum should contain [31, 44, 101, 110].

Research about how data science is currently being taught is sparse: the majority of 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 [31, 32, 58, 99] and in courses intended to serve non-CS-majors [25, 38, 189]. And faculty in fields ranging from bioinformatics [218], business [44], and statistics [101, 110] 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 [101, 110].

Outside the classroom, instructors have also documented their experiences teaching in informal settings. For instance, the Software Carpentry [229] and Data Carpentry [13] 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 [26], hack weeks [119], and apprenticeships [208] to train academic researchers in data science best practices.

In contrast to the aforementioned experience reports, to our knowledge, this dissertation contains 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.

2.3.2 Practitioner-Instructors

Most of the participants in the instruction-focused studies were practitioner-instructors: data science practitioners who also teach students. Practitioner-instructors are often found in settings such as medical schools (clinical faculty) [156], art schools, business schools, and law schools, where they are sometimes known as professors of the practice [173]. Two noted benefits of learning from practitioners are that they are likely up-to-date on the latest tools in their field [162] and that they are more direct members of the *community of practice* [142] that their students aspire to join. However, they often lack formal pedagogical training: Wilson refers to them as *end-user teachers* [230] (as an analogue to end-user programmers) since they teach but are not formally trained as professional teachers. To our knowledge, there is limited research about practitioner-instructors in computing-related settings such as data science.

2.3.3 Computing Education for Broader Populations

This dissertation 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 [185], 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 [167, 168]. Dorn et al. studied graphics and web designers who identify more as artists [66, 68]. 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 [50, 220]. 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 [59]; 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, the work in Chapter 3 and Chapter 4is 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 [50, 68].

2.3.4 Diversity in Computing

Chapter 4 specifically addresses diversity in computing, and to our knowledge, the majority of efforts around this topic have been for students in K-12 and university settings. In contrast, Chapter 4 is a study of a program to teach computing to adults who are not in school settings.

At the K-12 level (elementary, middle, and high schools), researchers have developed domain-specific programming environments to broaden interest in computing amongst traditionally underrepresented groups. For instance, Storytelling Alice [127] focused on engaging female middle school students, and Scratch [192] was deployed to after-school programs to foster computing interest amongst low-income African American and Latinx youths from 8 to 18 years old [152]. Beyond programming, the Glitch Game Tester project [63, 64, 62] hired African American high school students as game testers, which sparked their interest in computing careers.

Project Rise Up 4 CS [76] used in-person mentorship and financial incentives to encourage African American high school students to succeed on the annual AP Computer Science exam.

At the university level, research-based diversity initiatives have focused on two fronts: curriculum design and activities inside the classroom. On the curricular front, alternative pathways into computer science [214], more flexible threads of courses for different interests [84], and service learning opportunities [42] have improved diversity in computing majors. Within the classroom, pair programming, peer instruction [185, 186], and media computation activities [96, 191] have improved retention for students from underrepresented groups.

In contrast to K-12 and university initiatives, in Chapter 4 we study the development of a free computing education program targeted at adults who do not have access to formal schooling. Prior research on adult learning of computing has studied how working adults take online courses [33], how older adults over 60 years old [93] learn to code on their own, and how end-user programmers learn to code on the job [66, 68, 50]. However, these adults often already have higher education and plentiful access to technology. Despite these differences in learner demographics, the educational program that we study addresses some of the same challenges of adult education that prior work found, most notably lack of time given other life responsibilities and feelings of isolation due to lack of in-person support. Finally, the work presented in Chapter 4 is unique in showing how a team attempted to address diversity in computing by *using the tools of end-user programming at their disposal* to develop an adult data science education program.

Chapter 3

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

3.1 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 [147]. In academia, data scientists are often STEM graduate students, postdocs, and technical staff who write code to make research discoveries [91].

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 [31, 44, 101, 110]. Many novice data scientists are currently learning their craft and associated technology stacks (e.g., Figure 3.1) 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

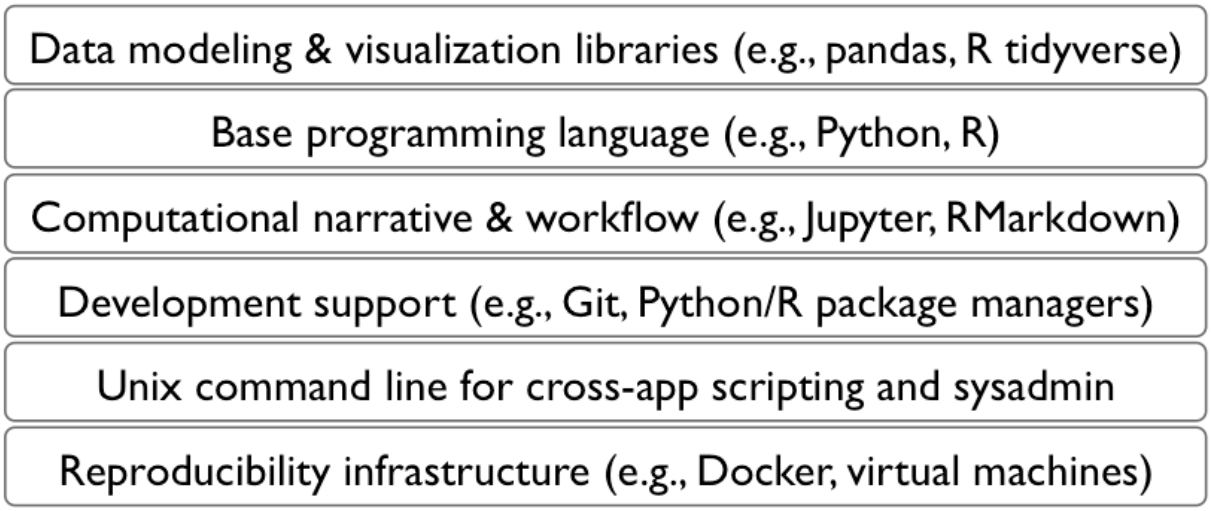


Figure 3.1. An example technology stack that modern data scientists must learn to do their job of writing code to obtain insights from data in a robust and reproducible manner. They often learn these skills from their fellow data scientists, not from formal computing instructors.

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 [91, 125, 128, 195, 202, 131], 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 [98, 185, 213]; on the other end, there is a parallel literature on the learning needs of end-user

programmers [67, 137, 220]. 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 chapter'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.

3.2 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

Table 3.1. 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.

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

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]
- 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 [54].

3.2.1 Interview Participant Backgrounds

Table 5.1 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 5.1), day- or week-long workshops such as Software/Data Carpentry [13, 229], 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 [113].

3.2.2 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.

3.3 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.

3.3.1 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 [14] 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."*

3.3.2 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 [185], 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 [16] 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 3.1. 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]."*

3.4 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 3.1). 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 [46, 79, 213]. 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.

3.4.1 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* [224] 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 [18], 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* [25, 91] 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 [46, 213], 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.

3.4.2 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 [12] and R Markdown [18], 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 [135], 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!"*

3.4.3 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 [234]. 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 [195]. 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 [142]. 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).

3.5 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.

3.5.1 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 [14]) 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 3.1) 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 [17] and RStudio Server [19] 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 [14] or Dataquest [15]. 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.

3.5.2 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 [219] 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 [35]: *"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."*

3.5.3 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."*

3.6 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 [136] 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 [142] of professional data scientists—for them to transform from being simply end-user programmers to being end-user software engineers [136] or professional end-user developers [202].

To facilitate this transition, instructors teach using authentic tools (e.g., Figure 3.1) 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 chapter.

3.6.1 Designing New Data Science Learning Environments

Our participants' approach to teaching was to try to scale up an apprenticeship model [20, 98] 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 [36], and McNamara built on those ideas to advocate for tools inspired by developments in the computing education community [159]. In sum, she envisioned a *"blocks-programming environment along the lines of Scratch [192]."*

Transferring this vision into data science, there have been recent efforts to extend Scratch with data access blocks [59], to add functional programming constructs into blocks languages such as GP [21] 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 [38]). 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.

3.6.2 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 [22], ICPSR [172], NCBI [206], and Google Dataset Search [171]. 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 [104, 194]) 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* [157]. 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* [86, 121, 236], 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 [169]) 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.

3.7 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 [50], 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 chapter 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.

3.8 Acknowledgements

Chapter 3, in full, is a reprint of the material as it appears in the proceedings of the 2019 CHI Conference on Human Factors in Computing Systems as Practitioners Teaching Data Science in Industry and Academia: Expectations, Workflows, and Challenges. Sean Kross and Philip J. Guo. 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Repurposing End-User Programming Tools to Foster Diversity in Adult Data Science Education

4.1 Introduction

A widely-acknowledged deficit in computing fields is the lack of historically under-represented groups on teams that build software and make engineering decisions [155, 170]. This deficit of perspectives is especially impactful considering how algorithmic-driven decision-making has become a fundamental part of modern life. Algorithms determine who is approved for bank loans [40], which job applications are considered for job openings [153], and what plan of care certain patients end up receiving [144]. Data-driven systems have also re-enforced racial biases [170] and denied essential social services to members of historically marginalized groups [77]. The reasons for these failings are multifaceted, but they are compounded when contributions from the people who would be most affected are excluded from the system design process [170, 77].

In recent years both researchers and community organizations have addressed these issues by diversifying the population of students who choose to study computing. For instance, academic projects such as Storytelling Alice [127], Scratch [192], and App Inventor [231] have fostered more inclusive programming communities at the K-12 level, especially in middle and

high schools. Nonprofits such as Code2040 [3], Girls Who Code [4], and Black Girls Code [2] strive to improve both gender and racial diversity amongst K-12 students interested in computing. At the university level, research-backed curricula such as media computation [96] and diverse paths into computing [214, 84] have made advances in the proportion of computing majors from underrepresented groups. In addition, college scholarships and mentoring programs have helped retain such students as they advance through school. However, the majority of such efforts target K-12 and university students, so there is a lack of knowledge about how to provide these benefits to adults who are *not in school*.

To address this gap, this chapter presents a case study of a team of academic research scientists who partnered with a local community organization to teach data science to adults living in a high-poverty area of a large U.S. city. The typical student in this program is an adult member of an underrepresented and marginalized group who did not complete high school; they may have grown up in foster care or may have experienced extended periods of unemployment or homelessness. The program's goal is to equip these adults with basic data science skills required to get entry-level jobs doing tasks such as spreadsheet data entry, data cleaning, wrangling, and validation. These types of data-oriented jobs offer an on-ramp into computing careers while being more within their reach than full-time software developer positions, which require much more extensive training.

To implement this grassroots initiative on a short time frame with a small budget, the team had to perform end-user programming [136] to repurpose existing tools from their research workflows and to create new ad-hoc tools to support course development. Specifically, they developed text-based programmatic workflows based on R Markdown computational notebooks [7] that they use in their research lab.

Our study is the first, to our knowledge, to analyze how a team of end-user programmers (i.e., academic research scientists) applied the philosophy of end-user programming (i.e., repurposing/building software tools for personal use) to diversify end-user programming (i.e., data science) education to reach traditionally underrepresented groups. While this chapter reports a

single case study, we believe its findings have generalizable research value to the field of end-user programming. Specifically, it advances the idea that *end-user programming can be a vehicle for positive social change by enabling a small team of non-specialists to repurpose software tools to serve their user population quickly and at low cost*.

This chapter makes the following contributions:

- Findings from a case study of a nine-member research lab on how they performed end-user programming to repurpose tools from their research to foster education.
- Implications for end-user programming research and practice, especially related to social good, diversity in computing, and broadening educational opportunities.

4.2 Methods

We performed a case study of the development process of *Cloud Based Data Science* (CBDS), a free online course described in Table 4.1. The goal of CBDS is to teach basic data science skills using spreadsheets and the R language in order to prepare students to obtain jobs as entry-level data scientists. In essence, it is training students to become end-user programmers who write code as a means to an end to clean data and produce analysis outputs.

For this case study we interviewed everyone involved in creating CBDS: eight research scientists at a large U.S. university and the project's program administrator, who was a research administrator in their lab. None of the nine team members' full-time jobs were to create educational programs or to write software; CBDS was a voluntary effort. The first author conducted all interviews (each lasting 45 to 60 minutes) using video conferencing software.

The interviews were semi-structured with questions focusing on the motivations each team member had for working on this project, their use and development of software tools during the project, and how these tools affected their interactions with other team members. Interview questions included:

- How did you first get involved in CBDS?

- What was your role in developing CBDS?
- What existing tools have you used for educational content development?
- What was your level of expertise with these particular tools before CBDS? (if they mentioned specific tools)
- Did you have to build any of your own software tools to develop CBDS? If so, which ones?
- How were development tasks distributed throughout the team?
- How did you coordinate work between team members?

The first author took notes and recorded verbatim quotations during every interview. After all interviews completed, the research team (two members) iteratively categorized them into major themes using an inductive analysis approach [54].

4.2.1 Study Design Limitations

This project was a case study of a specific team of academics at a U.S. university who attempted to develop a nontraditional educational program. Thus, we do not have large-scale replicable data and cannot claim that the CBDS team's experiences generalize to other related efforts. Also, we are relying solely on interviews and did not perform an ethnography to observe the team when CBDS was first being developed. Note that CBDS is still under active development, so many of the details are fresh on participants' minds.

Since CBDS is still in its early stages, having enrolled only around a dozen students so far, it is too early to tell the long-term outcomes of this program in terms of sustainability and impacts on its alumni. We also did not have direct access to the students and thus cannot report on their experiences. This study focuses solely on the CBDS development team.

Table 4.1. Curriculum of the CBDS (Cloud Based Data Science) course.

Module	Subject
1	Introduction to the CBDS Program
2	How to use Your Chromebook Laptop
3	How to use Web Applications
4	Organizing a Data Science Project
5	The Command-Line and Version Control
6	R Programming
7	Data Wrangling
8	Data Visualization
9	Connecting to Data Sources
10	Data Analysis
11	Communicating Analysis Results
12	Getting a Data Science Job

Table 4.2. Backgrounds of the nine members of the CBDS team that we interviewed, along with their prior end-user programming experience and whether they created course content or served as in-person tutors.

ID	Gender	Field	Job Title	End-User Programming Experience	Created Course Content?	In-Person Tutor?
P1	M	Biostatistics	Research Lab P.I.	> 5 years	Yes	No
P2	F	Genetics	Postdoc	$1-5$ years	Yes	Yes
P3	M	Biostatistics	Research Scientist	> 5 years	No	No
P4	M	Biostatistics	Research Scientist	> 5 years	Yes	No
P5	F	Biostatistics	Research Scientist	> 5 years	Yes	No
P6	F	Biostatistics	Ph.D. Student	$1-5$ years	Yes	No
P7	F	Liberal Arts	Administrative Staff	none	No	Yes
P8	M	Economics	Postdoc	< 1 year	Yes	Yes
P9	F	Genetics	Ph.D. Student	< 1 year	Yes	No

4.3 CBDS Goals: Diversify End-User Programming

We report our case study's findings by first detailing the goals of CBDS and the motivations of its volunteer development team. Then we describe their end-user programming activities throughout project development.

Table 4.1 shows the curriculum of CBDS, a self-paced online course that anyone can take for free. However, being free and online is nowhere near sufficient for ensuring that it is accessible to many members of underrepresented groups. Over the past decade of research into MOOCs (Massive Open Online Courses), a widely-acknowledged finding is the notable lack of diversity in who takes and benefits from them: MOOC students are mostly white or Asian males with at least college- or graduate-level degrees [41, 100, 97].

Interview participants P1 and P3 saw this lack of diversity firsthand since they had prior experience creating data science MOOCs. P1 explained his motivation for starting CBDS: *"Why aren't certain groups of people using our existing MOOCs? Maybe they didn't have access to hardware, they lacked prerequisite knowledge, or they were just unaware that data science was a thing."* P4 mentioned that existing courses assume prior educational experiences that exclude people without access to such opportunities: *"The problem with data science programs is that the material is pretty advanced. They're geared towards master's students."* The CBDS team believed that with a more accessible curriculum and personalized teaching approach, they could bring data science to a group that has not traditionally been reached by MOOCs. Specifically, P1's goal was to target students with a 10th-grade level of math literacy. The team also augmented CBDS with in-person support to help members of underrepresented groups enroll, remain in, and successfully complete the course.

First they worked with a local community organization to recruit potential students. To reach its target audience, the CBDS team partnered with the Historic East Baltimore Community Action Coalition (HEBCAC) [5], a nonprofit that specifically serves the historically disenfranchised low-income neighborhoods surrounding the university where the team works. HEBCAC

serves a community where many residents did not complete high school, grew up in foster care, or experienced extended periods of joblessness or homelessness. HEBCAC steps in to help them complete their GED diploma (the equivalent of a U.S. high school degree), place them in jobs, or help them arrange further educational opportunities such as community college. The majority of people served by HEBCAC are African American, Hispanic, and Latinx adults. HEBCAC was a critical bridge between potential students and the CBDS team. Otherwise these students would not likely know about the existence of data science as a career path that could be within their reach.

Once students enroll, they are given a free Chromebook laptop and the opportunity to meet in-person with volunteer tutors twice per week during 90-minute office hours; P2, P7, and P8 served as tutors. Students can also ask online questions to course staff in a private Slack chat channel. Finally, to encourage retention in the program, students are paid a modest stipend for successfully completing each module in Table 4.1; this stipend is designed to be comparable to the wage they would earn from working in the kinds of jobs that HEBCAC normally helps them obtain.

Once students finish the course, the CBDS team and HEBCAC work with them to do resume and interview preparation and to refer them to entry-level data science jobs in the area.

4.4 Motivations of CBDS Development Team

Not only was CBDS's goal to train new end-user programmers (i.e., data scientists), its course development team also consisted of end-user programmers. Table 5.1 shows team members' backgrounds. Everyone on the team works in the same life science research group at a large U.S. research university. P1–P6 all had several years of end-user programming experience, in the form of using bioinformatics pipelines and programming as part of statistical data analysis for their research. P8 and P9 had limited programming experience before working on CBDS, while P7 had no programming experience before joining. None of the team members have a

degree in computer science or experience doing professional software development. All are cisgendered (5 female, 4 male).

Why was this team motivated to create CBDS when their primary job was as research scientists? Their workplace is located in the same neighborhood served by HEBCAC, an area that has been historically disenfranchised. Decades of societal inequity has led to increased rates of poverty, which everyone on the team sees around them. Thus, all team members reported their primary motivation as wanting to create opportunities for adults in the surrounding neighborhood who could not normally afford to pay the tuition for a traditional education like that offered at their university.

P7 was closest to the target student community. She does most of the administrative work for CBDS and serves as a volunteer tutor for it. She grew up near the area served by HEBCAC, so she was very motivated to see people from her community succeed in this program: *"My personal excitement about joining in the first place was to help my people."* Besides growing up in the area, P7 felt that she was also able to relate to the students because she had only recently started learning how to code: *"I really appreciate my position in the program because I believe I am the least experienced staff member in terms of programming. So I experience the same frustrations and joys when a program crashes or when my graph turns out how I thought it would."* Also, as the only member of the team who was not on a Ph.D.-oriented research career path, she felt that students could be more honest and open with her: *"I definitely think it was a good idea to have somebody on the team who they weren't intellectually intimidated by."*

4.5 End-User Repurposing of Existing Tools

Because CBDS was developed by a team of volunteer non-specialists, they needed to engage in a variety of end-user programming activities to make this program work with relatively little time and money. The first set of activities we describe here, while not "programming" per se, invoked the spirit of end-user programming by repurposing existing hardware and software to

develop a data science curriculum.

4.5.1 Repurposing Low-Cost Chromebook Hardware

Keeping costs as low as possible was a major concern for the team. P1 described how prohibitively expensive it would be to build CBDS as an official university course or MOOC: *"In a traditional college setting if you had assembled several faculty to build this program it would have cost millions of dollars. We did not have that!"* Cost minimization was not just a concern in terms of development, but it was critical to the team's mission to make data science education available to underserved members of their local community. One initial obstacle they faced was simply making the technology required for doing data analytic work available to students. The students that they wanted to reach typically did not even own personal computers, or their computers were too old to install modern data science tools on: *"We wanted to reduce the cost of the hardware you need to get started. For low income folks these small costs are insurmountable."* (P3)

The team's solution was to provide a Chromebook laptop for free to every student in the program. Many Chromebooks now cost less than $300, which is affordable compared to the typical hardware that data scientists use and well within the constraints of the seed funding provided to launch the program with the first dozen students. In addition, Chromebooks are sometimes available to check out for free from public libraries.

Each Chromebook runs Chrome OS, an operating system geared for web applications. Instead of relying on a computer with powerful hardware, students used RStudio Cloud [8], a free data science run-time environment for the R language that is hosted on a web server and accessed via a browser-based IDE. This web-centric setup also helped students start coding in their browser without the frustrations of software installation, which prior work found to be a major barrier to getting started [138]: *"The goal was to minimize tool setup for students. Everything had to be done in the browser."* (P2)

In short, the CBDS team repurposed Chromebooks, which have traditionally been used

for casual web browsing, as end-user programming tools for aspiring data scientists.

4.5.2 Repurposing Tools for Open and Reproducible Science

As academic researchers who practice open and reproducible science [148, 182], the CBDS team were adept users of computational notebooks – especially R Markdown [7] – to do end-user programming for their research. R Markdown allows users to write prose in the lightweight Markdown format, while interleaving graphs, diagrams, and runnable code in several programming languages such as R and Python. Users can write R Markdown in a text editor or in RStudio Cloud, which renders it as a notebook-like interface similar to Jupyter [12]. Since these are text documents, changes can easily be tracked in version control systems like Git. To create lessons for CBDS, the team repurposed this computational notebook – one of the central tools in their daily scientific workflow – to become the substrate for the educational content they built.

A related example of repurposing was the fact that years of the team's data analysis code and documentation were already in R Markdown, so they could be curated, simplified, and re-used for teaching. For instance, P9 took software documentation she had already written for internal lab use and adapted it into course content: *"The fact that all of the content is plain text makes making those changes super easy."*

Another example of repurposing was led by P1 and P3, who had both developed MOOCs before. MOOC providers like Coursera offer an in-browser rich-text editor where instructors are supposed to write lessons and assignments for their course. Compared to the team's usual open science workflows, which take advantage of the command line, Git, and other programmatic tools, the team felt hindered by the compulsory use of manually-driven web-based GUIs. P1 explained: *"The way the Coursera platform is set up, which isn't as simple as 'push to GitHub,' it makes [content updates] difficult."* Thus, to better integrate the end-user programming workflows they already used for scientific research into their desired teaching workflow, the team partnered with online publisher Leanpub [6] to develop a new course platform. Leanpub is currently a

platform for taking Markdown-formatted documents and compiling them into eBooks. P1 and P3 already had experience publishing eBooks there, and they shared Leanpub's ethos of working with Markdown files. The team also valued Leanpub's pricing philosophy and applied it to CBDS: content on Leanpub follows a pay-what-you-want pricing model, which enables people to get it for free if desired.

The team worked directly with Leanpub, which built them a custom web application where they can upload R Markdown files and have them render as course webpages and assessments. This web app recognized custom Markdown syntax for elements such as multiple-choice questions and programming assignments with test cases. P3 appreciated how it was compatible with their existing text-based workflow: *"Leanpub catered very much to the idea of 'text to course.' Assessments as plain text was a very important feature."*

Using this custom platform, seven of nine team members (P1, P2, P4, P5, P6, P8, P9) developed technical course content solely in R Markdown files, tracked changes using Git, and collaborated on developing course modules (Table 4.1) using GitHub across 25 different repositories. Course material development began in February 2018, and the first in-person cohort for CBDS started at the end of May 2018. The team credited the ability to repurpose their research workflows as critical for launching this initial version in just three months. Significant updating of materials continued as the first cohort made their way through the program, as changes were made based on student feedback.

Lastly, the CBDS team also had to grapple with teaching modern data science software libraries that were continuously updating and changing their APIs. When they previously used MOOC platforms like Coursera, whenever a library or API changed, they would need to spend hours navigating menus and GUIs to modify relevant course content that mentioned that library. This manual workflow was antithetical to the team's practice of reproducible research [182, 209], where all of the figures, tables, and reported statistics in a data analysis can be automatically re-compiled with one command whenever the underlying dataset is updated. Working with Markdown course materials allowed the team to use command-line tools to find and appropriately

replace outdated content, similar to how they would update an outdated statistic or graph in light of updated input data while they were doing research.

Plain-text data formats, coupled with command-line tools and Leanpub's platform, enabled the CBDS team to *take an end-user programming approach to course development* instead of relying on manually-driven GUI-based content management portals typically used for online courses.

4.6 End-User Programming to Build New Tools

Seven team members (P1–P6, P8) had previously developed bespoke R-language packages for performing domain-specific analysis tasks or for sharing algorithms and data from their published research studies. Besides repurposing the tools of computational science to programmatically generate online course materials, these team members also engaged in end-user programming to build custom tools for themselves. Here we detail two such tools: Didactr for validating course content and Ari for expediting video production.

4.6.1 Didactr: Custom Software to Validate Course Content

The team developed various R packages to help them create, check, and deploy the data science lessons that went into CBDS. One of those packages, called Didactr, allowed team members to automatically validate lessons to make sure their Markdown was structured correctly before being uploaded to Leanpub. Lessons comprised two types of files: lecture videos that explained course concepts (see next section on Ari), and R Markdown files containing lesson readings, example code, and assessments to practice after each lesson.

Didactr parses these files using heuristics to make sure they are formatted to display properly on Leanpub's online course platform. Compiling the lessons and checking for errors locally with Didactr was faster and provided more useful error messages compared to uploading an error-laden lesson to Leanpub and manually checking on the web: *"Compiling courses on Leanpub takes time. Didactr allowed me to preempt errors that I would get on Leanpub so I*

could fix them locally and shorten the correctness-evaluation loop." (P2) Ultimately Didactr served as a "command center" package which allowed the CBDS team to test and track the dozens of rapidly-evolving content files that constituted the course.

In the spirit of end-user programming for one's own needs, Didactr's features were built piecemeal in response to recurring bottlenecks the team faced while making course content. For instance, P3 worked closely with several other team members to better understand tasks they were initially doing manually: *"I asked the content creators, 'Show me what you do' and tried to then find APIs that would allow us to automate as much as possible."* And P2 recalled how closely she worked with teammates to extend Didactr on-the-fly: *"When I would tell [P3] there was a feature I wanted, he would sit with me and build it right in front of me."*

4.6.2 Ari: Custom Software for Text-Based Video Production

Other than writing lessons and assignments, the most time-consuming part for the CBDS team was recording and editing lecture videos. Videos in CBDS often feature an instructor giving a real-time demo of writing and running code or showing how to think through a data analysis task. If the API for a function in the featured analysis changes, then significant portions of the video must be re-recorded and re-edited. This problem is particularly pronounced in fields like data science, where industry-standard libraries are rapidly evolving. P1 and P3 remembered how costly it was to re-record videos for their past MOOCs whenever the code they were teaching had their APIs updated: *"With content that changes so often it's not feasible to reshoot videos, re-edit, et cetera, every time an API changes"* (P3).

To make it easier to create and update these code- and slide-based lecture videos, the CBDS team developed a custom R package called Ari that allowed them to automatically generate narrated lecture videos from the R Markdown documents they were already writing as part of their course materials. To use Ari, a creator first passes in a set of lecture slides and an accompanying text narration script. Ari uses Amazon's text-to-speech web API [1] to synthesize a machine voice to speak out the script and FFmpeg [11] to stitch together the lecture slides

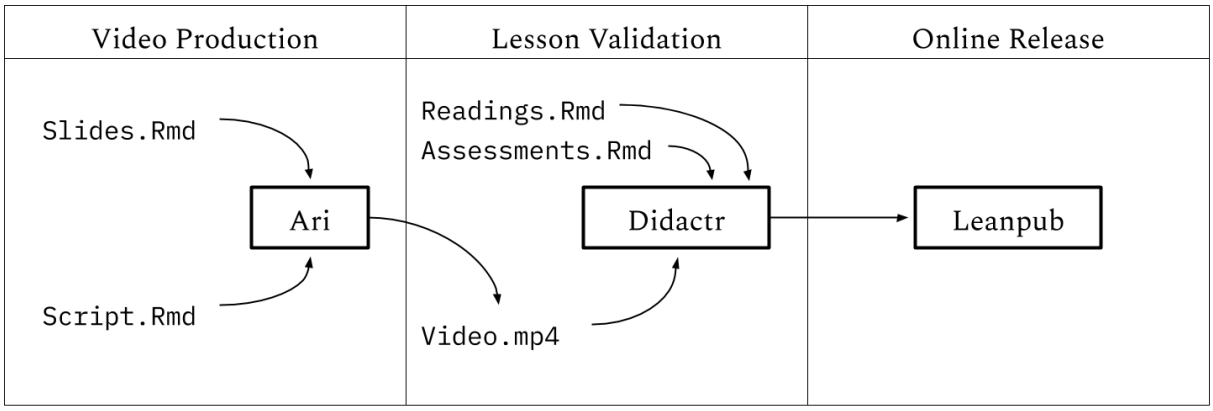


Figure 4.1. The software workflow that the CBDS team developed to produce, validate, and release course content from R Markdown (`*.Rmd`) source files.

and spoken audio into a final compiled video file with the proper timings. Lecture slides can be generated from R Markdown files, but often team members opted to use more traditional GUI-based presentation tools like Google Slides.

Ari helped the CBDS team take an end-user programming approach to video production, turning it into a process of editing text and compiling it into videos with R scripts instead of manually recording and editing using heavyweight video production GUI software. To make bug fixes or updates to their videos, they can simply edit text files and recompile. This format also allows them to easily track video edits in Git version control. P3 discussed how Ari's workflow aligned with the team's research philosophy: *"The videos are fully reproducible [from text-based sources], just like our scientific work."* P3 elaborated that having this more modular format meant they could iterate more quickly: *"This allows us to only modify content without changing presentation, delivery, et cetera. So we can take a much more experimental approach to making lecture videos."*

4.7 Toward End-User Software Engineering

Figure 4.1 shows the team's current course production pipeline where textual R Markdown files (`*.Rmd` extension) get compiled into videos (`*.mp4`), lesson webpages, and assessments, validated with Didactr, and then released online to Leanpub's web platform.

The CBDS team's goal was to create a free data science course for members of their local community, *not* to create a production-scale course development platform. However, to create such a course as a volunteer effort on a short time frame (3 months from inception to launch), they had to repurpose end-user programming skills from their research careers to create tools to make themselves more productive. But now that the course is in progress, the team found themselves transitioning from end-user programming into *end-user software engineering* [136, 45] with issues of tool maintenance, robustness, and updates from new contributors on their minds, especially as some of the original team members depart.

For instance, P1 was concerned about the ease with which future CBDS team members could interact with both the tools and course content: *"I know the original builders of the program will graduate soon, so lots of knowledge about maintaining the program will leave. This fact informed all of the technology decisions."* As our study was being conducted P2 finished her postdoc and moved to a new institution, and P1 explained the extent to which her departure was already testing the robustness of their tools: *"We have already done lots of maintenance and restructuring and our system is working. Team members who have then left are still regularly fixing bugs, which shows how easy the material is to maintain."*

A related concern was the extent to which tools could enable future maintenance and expansion of course content. P1 said the motivation behind building tools in the first place was the question: *"How can we make [course] maintenance costs as asymptotically close to zero as possible?"* But now the tools themselves need to be maintained and updated as well.

4.8 Discussion

We conclude by reflecting on our findings in light of implications for future end-user programming and computing diversity research, end-user programming for education and social good, and the paradox of scale and access to education.

4.8.1 Implications for Future Research

This case study presents only a single snapshot, but we believe it can open the doors to future research on the interplay between end-user programming and diversity in computing. For instance, there is at least an order of magnitude more end-user programmers than professional software developers [200, 199], and they likely come from more diverse demographics than those who specialized in computing fields. Thus, one of the most practical and scalable ways to further broaden diversity in computing is to channel the energy of end-user programmers. How can institutions that employ such programmers foster these kinds of initiatives without making them too bureaucratic and thus undermining their bottom-up grassroots spirit? Can lessons from these volunteer-run efforts inspire new practices for designing collaboratively-constructed educational experiences?

Switching gears, how can systems researchers develop tools to better support the extensibility of end-user programming environments to stretch far beyond their original intended uses? In our case study, the CBDS team repurposed the R language ecosystem, which was originally designed for statistical research, to build an online course development platform. While experts in educational technology could probably come up with a "better" toolchain, the fact is that this is the toolchain these research scientists already know well, so tools should meet them where they are. But must every ecosystem reinvent the same wheels in an ad-hoc non-reusable manner? Or are there more general principles for constructing modern software platforms that we can abstract out into language-agnostic tools that developers can plug into whether they are working in R or Python or JavaScript or even spreadsheet environments?

4.8.2 DevOps Patterns in End-User Programming for Education

Reflecting on our nine interviews in this case study, one recurring theme was how much technical infrastructure was involved behind the scenes to keep CBDS running. It reminded us of how the past decade saw the emergence of DevOps [74, 237, 30], a practice that combines

software *development* with the *operations* required to deliver and maintain that software. In industry, DevOps engineers write custom code to monitor the lifecycle of software products (especially web applications) throughout development, deployment, testing, and release. In a similar vein, the CBDS team are not only producing educational content like faculty normally do, but they are also writing custom software to manage the lifecycle of that content. In essence, they are mirroring the patterns in DevOps while performing end-user programming for education.

Like DevOps engineers, the CBDS team has significant influence on the design of their program (developing content), the programming tasks involved in delivering their "product" (building software to support that content), and monitoring its status (interacting with students to see where they get confused). Every CBDS team member can both make observations about what parts of their system (Figure 4.1) need to be improved and are empowered to make those improvements.

Also like DevOps engineers, the CBDS team repurposed or built custom software tools for each stage of the course lifecycle. They used the same tools that they would normally use to do their research to create course modules, deployed those modules on GitHub so other team members could collaboratively build upon them, developed their own monitoring software to test whether modules were formatted correctly, and released online and iterated based on student feedback. If not for their knowledge of appropriate tools to cobble together, they would likely not have been able to deliver CBDS on top of their normal duties as researchers. That said, some team members like P3 had concerns with the technical challenges of continued maintenance and scaling, given their multifaceted job roles: *"How can we be expected to be scientists, security experts, system administrators, and good instructors?"*

More broadly, we believe that treating educational artifacts like software artifacts by borrowing patterns from fields like DevOps could become a promising strategy as the demand for computing education grows in the coming years.

4.8.3 End-User Programming for Social Good

Another unique aspect of the CBDS project was how end-user programming was applied for broader social good. This project originated from the team's desire to create data science education opportunities for an underserved adult population that would not otherwise encounter an on-ramp into computing careers. Although the CBDS team was composed mostly of academic data scientists, it was not their data- or research-related skills that allowed them to build CBDS; rather it was their ability to design a code-based workflow that enabled rapid collaborative iteration on their course materials via end-user programming, software engineering, and DevOps.

The speed and relatively low cost with which CBDS was launched opens up the question: Who is in the best position to create such opportunities for underrepresented minorities to enter computing fields? Many existing diversity efforts have their origins at the top levels of organizations, whether it is CEOs diversifying hiring practices or nonprofits offering scholarships. This top-down approach, though impactful, is often not closely connected to the communities which these opportunities are designed for. Conversely, there are thousands of bottom-up local organizations working to help historically disenfranchised communities on the ground, but they are often not aware of paths into viable computing careers, especially in newer professions like data science. CBDS took more of a bottom-up approach: The team was not highly-positioned within the organizational structure of their university, and they relied on a partnership with the HEBCAC community organization to recruit interested students from the local area.

This case study points to the compatibility between a grassroots vision for positive social change and the spirit of end-user programming. CBDS was developed such that all team members could contribute to not just course materials but also to the custom software tools they developed for their own workflow. The flat structure and transparency within the team meant that they could address their students' concerns more quickly. Although the success of the program has yet to be determined, it may be good for top-down decision makers to rethink who should be empowered to help foster greater diversity in computing fields. Simply using free software

or putting free course materials online is not enough to create lasting change in terms of who has access to computing education. It appears there was no single tool or innovation by the CBDS team that allowed this program to come to fruition. However, this program could not have been built without end-user programming, which equipped each team member with the level of technical agility required to respond to the needs of their student population.

We believe that the CBDS team's ability to write bespoke software to manage their course production pipeline, combined with their proximity to the target student population in their neighborhood, made them well-positioned to deliver such an educational program. In contrast, a traditional university instructor would likely not be able to produce and support a complex technical course outside their normal teaching duties and would also not have funding to hire professional engineering staff to help them. On the other end, a MOOC provider such as Coursera or Udacity would likely not create such a "small" program due to lack of perceived market size and revenue potential; to our knowledge, no major MOOC provider has yet partnered with local community organizations to produce courses for underserved adult populations.

4.8.4 Rethinking Scale and Access to Educational Opportunities

One critique of CBDS might be, *"How will this ever scale?"* At present, it does not scale, since the CBDS team must staff the online Slack chat channels and in-person office hours themselves on top of their day jobs as researchers. The team has ideas for how to gradually scale, such as using alumni as volunteer tutors for subsequent cohorts and fundraising to buy more Chromebooks. However, we believe the fact that the team did *not* initially think about scale was what led to this program being developed in the first place. If they had thought about scale from the beginning, they would have likely created an ordinary MOOC like what P1 and P3 have done before.

People who take MOOCs tend to already have higher incomes and higher levels of prior education; this is especially the case for courses that teach technical subjects such as computer science or data science [41, 100, 97]. It appears that if a course is designed upfront for reaching

the largest possible audience in terms of enrollment, then it does not usually reach populations that have been historically excluded from educational opportunities. Thus, paradoxically, designing courses for scale might mean *less access* for those who are unlikely to find those resources on their own.

In contrast, CBDS presents an alternative to online courses or university outreach programs. It takes an approach where free course materials are designed to scale online but are also formatted in a way so that course creators can iterate on them quickly. But it was the team's willingness *to do what does not scale* – adapting to their students by partnering with the HEBCAC community organization – that enabled them to tailor the program continuously as new needs arose throughout deployment. Working with HEBCAC and the students face-to-face does not scale, but we believe this approach provides a path for greater access to computing opportunities.

Lastly, CBDS points the way toward future hybrids of online and in-person education. One idea here for scaling is that paid versions of CBDS could partially fund in-person versions that target historically underrepresented groups. With this financial model, those who have had more access to educational opportunities can pay to enroll and thus indirectly fund those who have not had access to the same opportunities. Beyond financial sustainability, online courses can take advantage of their ability to scale by transforming their online communities into in-person communities of support: Online alumni could be recruited to help with in-person tutoring in underserved communities and also provide guidance and networking opportunities related to computing careers.

4.9 Conclusion

We presented a case study of how a team of academic scientists repurposed end-user programming skills and tools from their research to create an adult education program to cultivate diversity in computing. The team provided an easily-accessible learning environment with free

Chromebook laptops, a web-based coding platform, and weekly in-person office hours and online help. They customized R Markdown computational notebooks to develop and publish course content. And they built custom tools such as those to validate lessons and to compile textual scripts into lecture videos. This study shows how the bottom-up grassroots spirit of end-user programming can advance social good. Hopefully both small teams and large organizations can repurpose these lessons for positive ends.

4.10 Acknowledgements

Chapter 4, in full, is a reprint of the material as it appears in the proceedings of the 2019 IEEE Symposium on Visual Languages and Human-Centric Computing as End-User Programmers Repurposing End-User Programming Tools to Foster Diversity in Adult End-User Programming Education. Sean Kross and Philip J. Guo. 2019. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Orienting, Framing, Bridging, Magic, and Counseling: How Data Scientists Navigate the Outer Loop of Client Collaborations in Industry and Academia

5.1 Introduction

Data science has grown so ubiquitous over the past decade that anyone familiar with the field must have seen workflow diagrams like those in Figure 5.1, which often appear throughout the literature [91, 226, 221, 232, 235]. A Google image search for 'data science workflow' shows many similar diagrams from across the web. While details vary, these all depict the *technical* workflow of data science, usually with stages such as acquiring, wrangling, cleaning, exploring, modeling, and visualizing data. Many researchers have studied how data scientists work both individually and collaboratively in each of those stages [23, 125, 232, 154, 235, 131] and highlighted the myriad challenges they face; they have also built numerous tools to support all of these stages [72, 141, 94, 91, 124, 197, 233].

Yet despite all of these advances in our understanding, we argue in this chapter that the ubiquitous data science workflows we are familiar with (e.g., Figure 5.1) *capture only a portion of what data scientists actually do in practice*. Specifically, they represent the tight "inner loop" of iterative day-to-day technical work that is required to turn raw data into insights. But before

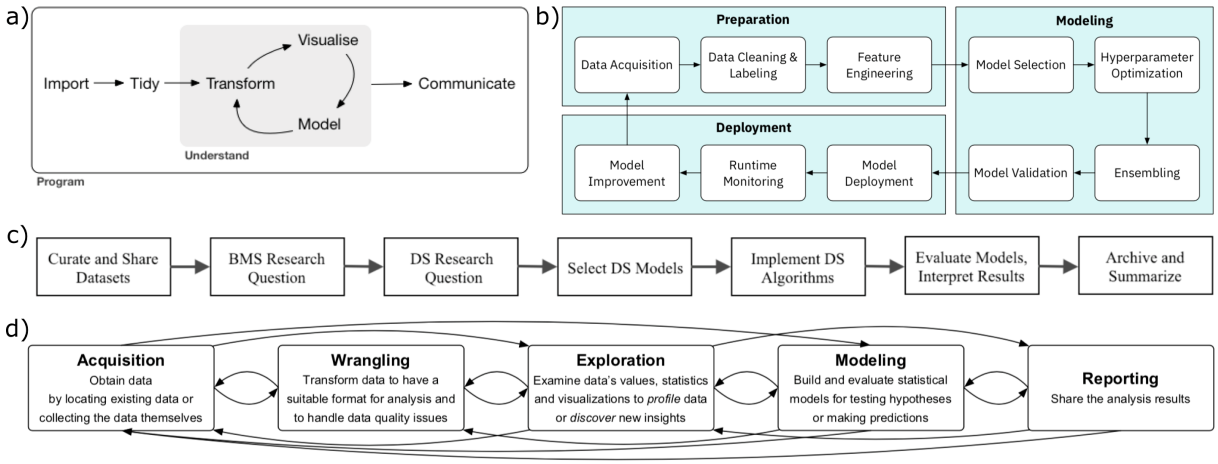


Figure 5.1. Examples of technical data science workflows from a) Wickham and Grolemund [226], b) Wang et al. [221] also used by Zhang et al. [235], c) Mao et al. [154], d) Wongsuphasawat et al. [232]. In this chapter we show how these workflows are actually the inner loop of a more general collaborative data science workflow that we illustrate in Figure 5.2.

any technical work can occur, data scientists must find clients to collaborate with and establish the parameters of those working relationships. For instance, in academia a postdoc in statistics may do data science work for a cancer research lab PI on a grant that partially funds their salary. In industry, a freelance data scientist may run their own independent consultancy and work with a variety of external clients; and within large companies or government agencies, data scientists often act as internal consultants to provide analytical services to different clients within their organization.

Once a working relationship has been established and the project is underway, data scientists periodically interface with their clients, usually in weekly or monthly meetings. The few prior studies of data science collaborations [131, 154, 235] have focused on how teams collaborate within the context of technical workflows like those in Figure 5.1. But since clients are often not involved in the inner loop of day-to-day technical work, their collaborative activities and interpersonal dynamics with data scientists likely take different forms. ***What does the end-to-end workflow of a data scientist's collaboration with clients look like throughout the lifetime of a project?*** Understanding this client-oriented workflow is important for capturing a more complete picture of what data scientists actually do in practice beyond their technical work,

since their end goal is often to analyze data to meet clients' needs [61].

To investigate this question, we performed semi-structured interviews with ten data scientists (5 female, 4 male, 1 non-binary) in diverse roles across industry and academia. We focused our conversations on how they find clients (or how clients find them), how they navigate these client relationships throughout a project, and what challenges they face in communicating with clients.

We distilled our interview findings into an *outer loop workflow* of collaborative data science shown in Figure 5.2, which subsumes the inner-loop diagrams in Figure 5.1. Our workflow has 6 stages:

- **Groundwork**: Data scientists must build trust, reputation, and social capital in order to establish working relationships with clients before a project even begins.
- **Orienting**: Data scientists join projects at different times and must rapidly orient themselves to the constraints of their given environment. We found five ways that they enter into client engagements: 1) at the very start of a project, 2) when the client already has an analysis technique or 3) data set in mind, 4) when the client wants them to compute a specific value, and 5) when the client has already tried and failed to do an analysis themselves.
- **Problem Framing**: Data scientists engage in conversations to probe the client's assumptions and work with them to frame the actual underlying problem they want to solve with data.
- **Bridging the Gap**: Concurrent with problem framing, data scientists must also bridge the gap between their analytical expertise and the client's domain-specific knowledge.
- **Magic**: Data scientists report that clients refer to their day-to-day technical work as "magic" since clients often do not see or understand it. The inner-loop technical workflow (e.g., Figure 5.1) resides within this stage, so that is how our outer-loop workflow subsumes it.

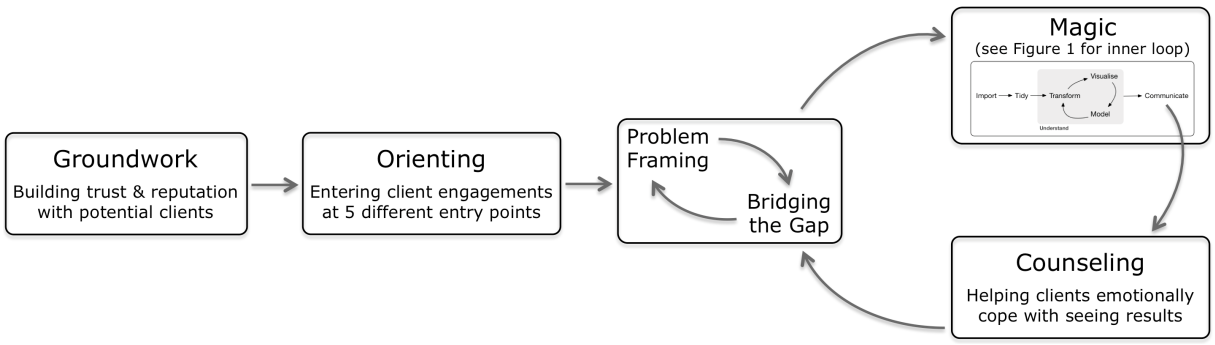


Figure 5.2. The outer loop of data science collaboration with clients, which we discovered via an interview study of ten data scientists. The inner-loop technical workflows in Figure 5.1 all reside within the "magic" stage.

- **Counseling**: When showing results to clients, data scientists must also counsel them to provide emotional reassurance and help them cope with seeing less-than-favorable results.

Taken together, this outer loop workflow that we discovered in our study (Figure 5.2) captures the end-to-end collaborative work that professional data scientists engage in from before a client is found all the way to the end of a client engagement. ***It contributes to CSCW by expanding the notion of what collaboration means in data science beyond the day-to-day technical work*** of Figure 5.1 that has been covered by prior work [131, 154, 235].

More broadly, our study findings extend the scope of findings from prior CSCW studies in data science collaborations [154, 115, 235, 131], consultative work with clients [207, 145, 212, 89], and data management lifecycles [161, 184]. Specifically, we contribute novel insights about the extensive groundwork that occurs before collaborations can even begin, the myriad ways that data scientists must orient themselves at the start of projects, and the emotional labor they engage in to manage client relationships in the face of power imbalances (e.g., during the problem framing, bridging the gap, and counseling stages).

We conclude by discussing the implications of our findings for data science education, parallels to design work, and unmet needs for tool development. First, we discuss how this outer loop workflow leads data scientists to build *socio-emotional infrastructure* to support their clients' needs, which complements the technical infrastructure provided by computer-based tools.

However, socio-emotional work such as building trust, orienting into a project's prior constraints, and helping clients emotionally cope with analysis results are not currently taught in most data science curricula, which instead focus mostly on technical education. Next, we draw parallels between the work of data scientists and those of professional designers, who also must work with clients under various constraints. However, unlike designers who have visually-expressive tools like CAD, Photoshop, and Figma, data scientists often lack expressive representations to make their ongoing work transparent and legible to clients; as a result, clients often perceive what data scientists do day-to-day as opaque "magic," which leads to communication barriers. Thus, we identify an unmet need for future data science tools to construct more expressive representations to foster collaborative work.

In sum, this chapter's contributions to CSCW are:

- A novel characterization of client-oriented collaborative work in data science beyond the usual technical workflow of acquiring, cleaning, analyzing, modeling, and visualizing data.
- Findings from an interview study of ten data scientists who work with a variety of clients across industry and academia, synthesized into a six-stage outer-loop collaborative workflow.
- A call for more expressive representations to foster collaborative work in data science.

5.2 Methods

For this study we interviewed 10 data scientists across industry and academia who regularly collaborate with a variety of clients to provide analytical services for them (see Section 2.1.2 for details). We recruited participants online via social media posts and snowball sampling from our professional networks. We strove for a diverse sample in terms of demographics, work environments, and client profiles.

Each interview was conducted by the lead author remotely on Zoom and lasted from 1 to 1.5 hours. Participants were not paid. Interviews were semi-structured and focused on how participants interacted with their clients throughout the lifetime of data science projects. The interviewer encouraged, but did not require, participants to show us artifacts from past projects (e.g., business presentations, academic publications) to help ground the conversations. Guiding questions included:

- At your workplace, how do new data science projects come to you? Do you proactively look for clients, or do clients find you?
- How do you establish new client relationships?
- Walk me through the course of a typical client engagement.
- What, if any, difficulties in communication have you had with clients in the past?
- Have you ever had to deliver bad news to a client? If so, describe how that went.

Most notably, we kept conversations at the level of their client interactions rather than diving into inner-loop technical mechanics of how they use specific data science tools (e.g., Figure 5.1).

5.2.1 Interview Participant Backgrounds

Table 5.1 shows our 10 participants' diverse demographic and professional backgrounds. 5 are female, 4 male, and 1 non-binary; and they are evenly split between industry and academia. Within industry, workplaces include startup companies, a well-known large tech company, a government agency on environment protection, and an independent consultant (P6). Participants in academia usually work at large research universities or medical research facilities. Our participants had between 5 to 25 years of firsthand experience doing data science work[1].

[1] includes time spent working in various data analysis and statistics roles before 'data scientist' became an official job title

Table 5.1. The 10 data scientists in our study, their demographics, workplace, and who their clients typically are. G=gender (F=female, M=male, NB=non-binary). PI=Principal Investigator on a research grant.

ID	G	Sector	Workplace	Typical Clients
P1	F	Industry	Healthcare startup	business executives, software engineers
P2	F	Academia	Dept. of biostatistics	medical doctors, biomedical research PIs
P3	F	Academia	Dept. of medicine	medical doctors who have research grants
P4	M	Academia	Medical research center	faculty PIs in biology and statistics
P5	F	Industry	Well-known tech company	junior and senior executives, manager
P6	NB	Industry	Freelance data consultant	companies and government agencies
P7	M	Academia	Dept. of biostatistics	PIs in biology, genetics, and statistics
P8	F	Industry	Federal government agency	government representatives, managers
P9	M	Industry	Data-driven health startup	junior and senior executives, manager
P10	M	Academia	Brain research center	faculty PIs in various fields

P1, P5, P8, and P10 work on a team of data scientists, while the other six work individually. We did not investigate teamwork dynamics in this study, though, since we focused on client interactions. The 'Typical Client' column in Table 5.1 shows where clients usually came from. In industry, clients include various managers, business executives at different levels, and sometimes software engineers. In academia, clients are mostly PIs (Principal Investigators) who hire our participants to do data analysis and statistics consulting for a particular grant-funded project.

5.2.2 Data Overview and Analysis

The lead author recorded notes and prominent quotes during each interview. In addition, all interviews were recorded, transcribed, and viewed by the entire research team (two members). As we studied the videos and transcripts, we iteratively categorized our observations together into major themes using an inductive analysis approach to build a grounded theory [54] of how these data scientists collaborated with their clients. Two researchers performed open coding on interview transcript data as it arrived and met regularly to reconcile our codes when there were disagreements. We decided to stop after 10 participants since we felt we had reached a reasonable saturation point [198]: several participants often brought up the same themes without any prompting by the interviewer, and it became more difficult to find distinctly new themes.

Fifteen intermediate themes emerged during this iterative process, which we eventually merged into six stages. Example intermediate themes included participants' discussions about managing client expectations, defining success in a collaboration, and details about how new projects are first presented to them. After all interviews were completed, we collaboratively merged our codes into a hierarchy that resulted in six final high-level themes (Groundwork, Orienting, Problem Framing, Bridging the Gap, Magic, and Counseling), which we used to construct the workflow diagram shown in Figure 5.2. During our meetings, we initially discussed having more stages in our model such as a 'success/failure' node at the end. But we eventually decided that negotiating the meaning of success happens throughout different stages (most notably, Counseling) so we interspersed those findings into our six stages.

5.2.3 Study Design Limitations

Although we strove to include participants from a variety of demographic and professional backgrounds, our personal recruitment and snowball sampling led to some limitations: Everyone was from North America, those in academia usually worked with PIs in the life sciences and medical research, and those in industry may have not been able to disclose all the details of their work due to corporate confidentiality agreements since we did not work at the same companies that they did.

We concluded our study after interviewing 10 participants since we felt we had reached a reasonable saturation point [198] (e.g., several participants independently brought up the same themes, it became harder to find new themes). Since this was a qualitative study, we were not trying to quantify event occurrences or divide our interview data into segments. However, we acknowledge that having more participants in more varied settings may have yielded additional findings. In addition, we believe our study population is representative of the common types of client-facing analysts roles described in Section 2.1.2. However we did not cover machine learning or platform-building oriented roles, so our findings may not generalize to those types of data scientists.

Furthermore, we focused on how data scientists interact with their clients, but we did not account for greater structural issues such as how their work environments, team dynamics, or organizational hierarchies might have affected their workflows. Related work by Zhang et al. [235], Kim et al. [131], and Mao et al. [154] cover these organizational aspects of team-based interactions.

Finally, we did not interview the clients whom our participants worked with, so we can report only from the perspectives of data scientists. Thus, our findings are representative of what data scientists think their clients' needs, problems, and concerns are throughout their engagements.

5.3 Groundwork: Building Trust with Clients in Positions of Power

We report findings in the order we illustrated in Figure 5.2. First, interview participants emphasized the importance of building trust with potential clients before a collaboration can even begin.

5.3.1 Finding clients and building initial trust

Clients seek out data scientists based on trust built from prior working relationships. For example, P1 mentions how clients typically find her:

> *I've noticed the people that I have a closer relationship with, maybe we've worked together on different projects in the past, they feel really comfortable asking me directly for things or they feel like they know the path to me.*

Some need to proactively seek out clients, though. For instance, P8 works on a data science team in a governmental agency on environmental protection and describes the publicity work she must do internally to advertise their services:

> *Now we are sort of, let's say marketing our team to some degree. I think a lot of that comes through things like internal blog posts, community-of-practice websites, obviously presentations, but you only have so much time where you can*

give webinars or go to another ministry or go to another group.

Once a potential client is identified, data scientists must build trust by showing that they care about the client's domain-specific goals. P1 worked for over a decade at a research hospital and recently joined a healthcare startup; she mentioned how it is important in both academia and industry to demonstrate her motivation to understand her client's domain:

> *That helps you show I am interested in this business and I know how to bring value to this business. I'm not just like churning out numbers, whether that's churning out a p-value for your academic publications or churning out a dashboard – yet another dashboard – for your industry stuff.*

Clients also have their own "clients" who come from a broader audience that consumes the final results of an analysis. For instance, in academia the client may be a lab PI, but the audience for analysis results includes readers of the publications produced by the lab; and in industry the client may be a division director, but the audience is fellow business executives who attend presentations that the director gives. The end-user audience who is consuming analysis results will also likely have similar domain knowledge as the direct client. To that end, P2 mentioned empathizing with the audience as well:

> *If you want to be successful, like if you want to be good at your job, essentially you have to understand that there's an audience who's consuming that analysis. And your goal is to get them to believe you, to trust you, to have confidence that what you're producing makes sense.*

5.3.2 Power dynamics in establishing client relationships

Clients are often in positions of power (e.g., a senior PI in academia or a business executive in industry), so data scientists must also take power dynamics into consideration when establishing these relationships. For instance, some clients may implicitly expect a foregone conclusion, such as a senior design manager who believes one variant of a website's homepage will lead to higher sales volume compared to another variant of the same page, and they want to get some data analysis results to back up their hunch. Thus, data scientists must balance the client's expectations with their sense of duty to be faithful to the rigor of the analysis process.

Similarly, P4 mentioned that it is important to be able to give honest feedback to their client to help them avoid unrealistic expectations before entering into a collaboration, but power dynamics can make it hard to do so:

> *I think you should always respect your collaborators, but you need to also understand at times they want you to be their check. They want you to push back, but it's very difficult when there's a huge power difference.*

The way to build this trust is to have a proven track record of doing high-quality work and accruing social capital within the organization or broader data science community. As P5 explains,

> *I think that it was really important to have built up credibility, like an institutional reputation, so people believed there were a lot of things I was right about before, or I had given an answer that people believed and made sense to them. Then to come to this kind of situation and to be able to say, um, "I'm sorry, we can't do this." Then I have to spend the capital. It's like you earn the capital and then you spend it.*

Seven participants (P1, P3, P4, P5, P8, P9, P10) mentioned similar experiences in terms of earning and spending social capital to manage client expectations.

5.4 Orienting: Five Entry Points into Data Science Collaborations

Having established trust (see previous section), a data scientist joins an analysis project at different stages depending on the client's needs. One common sentiment amongst our participants was that they felt they were "parachuting in" to a foreign environment whose constraints had already been established before they joined, so they needed to *orient themselves* before starting the technical analysis work. We found five entry points into data science collaborations, each with different kinds of communication that occur between data scientists and clients during this orienting process.

5.4.1 At the very start of a project

Some data scientists enter an analysis project before anything has been decided or any data has been collected, and they are often relied upon throughout the entire lifetime of the project. This is the ideal case in many of our participants' minds since they get to engage clients in substantive conversations about how to set up the data collection, experiment, or analysis from the ground up. However, the reality is that oftentimes the client has already established some constraints before bringing them in. For instance, P1 has over ten years of experience playing different data science roles in medical research and healthcare settings; here she describes the ideal case of being brought in early, along with a range of other entry points that we will describe later in this section:

> *When [my client] would ask me for things, it would range from "we need you to design this study" to "we need you to help define the research question" to "we need you to make this figure because we want to put this figure in a journal [paper]" or, "Hey, can you do this re-analysis because reviewer two wanted this re-analysis?"*

5.4.2 Client already has an analysis technique in mind

Some clients bring in a data scientist at the start of a project, but they already have a technique in mind that they want to try out. For instance, a biomedical lab PI may have heard about the latest advances in neural network-based deep learning for image recognition in medical applications, and they want to apply deep learning to detecting a certain kind of cancer cell that they study in their lab. Their hope is that these new techniques could reveal hidden insights that could lead to, say, notable publications in academia or product growth in industry. The main upfront conversation that a data scientist must have with clients in this case is to help them manage expectations about their desired technique. For instance, P3 said this about her clients in a medical research context:

> *I feel like people have over-hyped data science, artificial intelligence, these words, I hate even using them now. Because people expect very glamorous sorts of*

> *results. So now since I do a lot of question-asking upfront, I see that people have an expectation from all the hype, and then when they don't get it they feel a little let down, and they're disappointed.*

Similar to the power dynamics mentioned in the prior section, it can be hard for data scientists to push back against a client's expectations when the client is in a position of greater authority and often funding the work. Thus, our interview participants sometimes offer to start with applying simpler methods (e.g., general linear regression models, gradient boosted decision trees) before agreeing to invest the large upfront time in applying more complex methods such as deep learning.

5.4.3 Client already has a data set in mind

Some clients bring in a data scientist after they have already collected data. In academia this data may come from a variety of lab experiments or from data sets obtained from external collaborators. And in industry this data often comes from usage logs of how customers are using their product (e.g., server logs for web applications, telemetry for mobile apps or internet-connected devices).

These clients have already put in lots of upfront effort and cost to collect the data, so they want to somehow "capitalize" on it. P9 recounted such a situation where a client had collected thousands of log entries from wearable fitness devices which their participants had worn for several years:

> *I recall one instance of someone saying like, "I have all this data and I want to show that it's significant." And I had to have moments of being like, well, they're very "significant" because you spent all this time running experiments, collecting that, they're very meaningful, but let's maybe unpack what that means.*

What P9 meant by 'significant' here is that he is reassuring the client that there is probably *something* meaningful that can come out of this data, although he cannot guarantee any specific set of results. There is a delicate balance between giving the client emotional reassurance that their efforts in collecting that data has been worthwhile and also setting realistic expectations that real-world data is messy and incomplete, so they cannot guarantee any desired outcomes.

Again, in the ideal case the data scientist would have been involved in the project from the very beginning to help design the data collection process itself, but they are often parachuted in after data has been collected.

5.4.4 Client wants to compute a specific value

Clients sometimes call on a data scientist when they want to get a specific value from their data, which may be a summary statistic of one variable, or a few extreme values from a range of data points. For instance, P1 mentioned one case where a collaborator was insistent on getting her to compute a readmission rate in order to understand how often patients who were admitted to the hospital for a disease were later readmitted after a medical intervention:

> *And then there's the person who's like, "I don't really know what [data] you need to get to calculate this readmission rate, but I just need a readmission rate. I just know that I need a readmission rate." Right? You have to work with them a bit to be like, "Okay, but what exactly do you mean by readmission?" And you have to ask some follow-up questions.*

One challenge in these situations is that the data scientist needs to establish context as to why the client wants a certain value to be computed, what data is needed for such an analysis, and whether alternative computations would be more appropriate. Again they are dropped into the middle of an analysis project without being able to design the research questions or data collection methods.

5.4.5 After the client already tried to do analysis themselves

Finally, clients may call on a data scientist after they have already tried to do part of the analysis themselves but without success, often due to their lack of data science expertise. For instance, a lab researcher in genomics may have only basic data analysis skills and unsuccessfully tried computing gene expression values for a paper; that failure motivates them to bring in someone with more experience in analyzing larger-scale data. Surprisingly, this can cause a client to under-appreciate the amount of time and effort that data science work takes, as P10

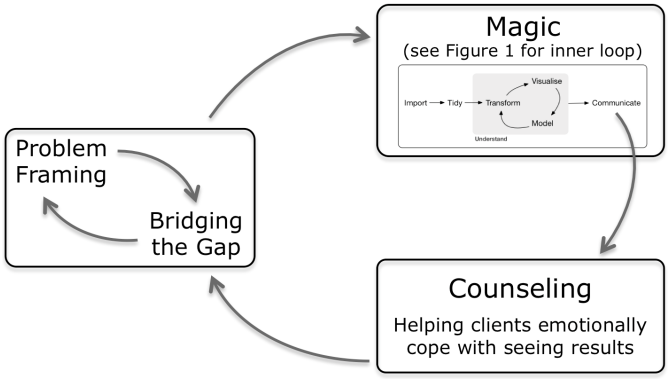


Figure 5.3. The outer loop of collaborative data science (reproduced from Figure 5.2), with all the technical work occurring in the inner loop within the "magic" stage.

describes here:

> *Sometimes people have tried it by themselves. They're frustrated. They're like, "Oh, like this is just getting in my way. Can you just do it? And, we know you're good at this. Can you just do it fast?"*

The main communication challenge here is for the data scientist to provide emotional reassurance to their client, who is likely already frustrated by their failed attempts, while at the same time setting realistic expectations that the desired work may take considerable time and effort to complete. These interactions can be sensitive since, again, clients are often in a position of power over the data scientist (e.g., it may be their research lab's PI who tried to do some analysis but failed). In Section 5.6 we further discuss how the invisibility and opaqueness of data scientists' technical work often makes clients underestimate the amount of effort involved.

5.5 Problem Framing and Bridging the Gap

After orienting themselves at an entry point, the data scientist and client enter into the main loop of their collaboration. As Figure 5.3 shows, this loop involves problem framing and bridging work (this section), magic (where all the technical data analysis work occurs), and counseling the client about analysis results.

When the main loop begins, the data scientist does not start writing analysis code right away. Rather, similar to the practice of professional designers [55, 56], they first engage in

problem framing by having conversations with clients to get at the underlying problem they want to solve, which may be different than what they claim to need. During these conversations they also work together to *bridge the gap* between the data scientist's analytical expertise and the client's domain expertise.

5.5.1 Problem framing: asking questions to get at the underlying problem

All ten data scientists we interviewed engaged in *problem framing* conversations before starting analysis work, which is akin to a practice that professional designers engage in where they question the client's initial assumptions and brainstorm with them to get at their underlying issue [55, 56]. For instance, P1 explains how even a seemingly simple request for one data value can lead to a detailed back-and-forth conversation as she tries to understand what the client really wants:

> *I'll get a random message from someone saying, "Hey, do you know what this number is?" And I'll say, "Maybe, let's say yes, I do know what that number is. But tell me why you're asking? Is there a bigger problem than I can help solve? Are you having conversations that are going to lead to more projects later, or does this reflect the fact that you don't have immediate access to this number?" There's always a question under the question, right?*

Uncovering the 'question under the question' can be a tricky process since a client's ultimate end goal may not be clear at the beginning of an analysis project, and the attributes of what the data analysis should achieve may not be precisely defined. P3, who investigates digital interventions in public health, explains the kinds of questions she asks to get to the bottom of these issues:

> *When someone comes to me and says, "I want to do X," I ask a ton of clarifying questions: What will this data be used for? Is it going to be like a one-and-done publication? Or is it going to be a platform that's rolled out like a website? Or, how far in advance do you want to predict [using some model]? Would it be successful if it was predicting one month out or are you looking for 12 months of prediction?*

One specific problem framing technique that some participants used was to ask their

clients for what kind of output they want. Focusing on asking questions about concrete analysis outputs like tables, graphs, and models can help data scientists productively drive conversations about what questions an analysis should try to answer, as P3 recounts from doing analysis for a public health research project: *"I asked [my client] to lay out what kind of table they would like to see, and then I'm able to move in that direction or give them something very close. And having that conversation where I say, 'Tell me the ideal tables you'd like to see in your paper,' makes it so much easier for me."*

Finally, data scientists need to be careful to respect the domain expertise of clients while communicating that their client's goals for an analysis may not be feasible. P6 outlines the difficulty of trying not to offend a client who they were working with to refine the premise of an analysis of a product A/B test:

> *I think the hard thing is saying [to a client]: "I'm not questioning you or your job, I just need to understand what you're doing this for. So that I can actually do my job properly and get you the right thing rather than just giving you what you want, which probably isn't correct."*

5.5.2 Bridging the gap between data science and domain knowledge

While in the midst of problem framing, data scientists and clients often need to learn more about each others' respective expertise. We refer to this as bridging work, as it involves trying to establish common ground [52] by building bridges across the gap between data science and domain knowledge. That is why problem framing and bridging work are shown together in the same stage in Figure 5.3: It reflects how data scientists and clients move fluidly between these two kinds of complementary activities as they try to understand both each other and the underlying problem better.

Bridging work occurs in two directions: the data scientist teaching the client about technical analysis methods, and the data scientist getting the client to *teach them* about the target domain.

Data scientists teaching their clients about data analysis methods

The first direction is when data scientists teach clients about methodological ideas to help those clients think about how their business or scientific problem can fit into a known data analysis workflow. These conversations occur when a data scientist believes they can help their client frame their project in terms of specific statistical phenomenon or analysis methods. For instance, they may want to explain how power law distributions arise when talking to a client who is an online bookseller trying to understand sales patterns (i.e., the majority of sales will come from only the few most popular books). Data scientists hope that clients can articulate their requests in more realistic terms if they understand the analysis tools and methods that the data scientist can feasibly use. P10 mentioned calibrating expectations as another benefit of this bridging work:

> *I can talk multiple languages when I talk to different people. And I think that in a lot of situations, that's also the scenario where people might not know as much about data science, right? So if they don't know the language, some of the basics, then they might have unrealistic expectations. So that's also part of my vision is setting realistic expectations.*

Data scientists learning domain knowledge from their clients

The opposite bridging direction is when the data scientist wants to learn more about the client's domain to hopefully integrate the client's knowledge into the analysis and to ask more informed questions when they return to doing problem framing. Since data scientists often work in a consulting capacity with different groups of clients, each project they undertake may be in a slightly different domain and thus require specialized knowledge. P1 emphasized the importance of learning about those domains from clients so she can craft more refined analyses:

> *The more I learned about either the clinical domain in my [past] academic life, or now either the clinical or the business domain [in my job], it helped me ask better questions. [...] I can say, "Oh wait, do you think this variable is maybe a confounder for this and this, because this is how they're related?" And we can have more intelligent conversations about that, and that might inform how I build the model or how we report results.*

P2 expanded on this theme by mentioning that a data scientist can unburden the client from thinking about data science issues if they can learn enough about the client's domain. That way, the client can focus their efforts on thinking about their own domain, where they are most comfortable:

> *If you have communicated to them, essentially, what the problems are in an understandable fashion, then in their mind they're thinking about the biology [...] They're thinking about the actual context, that domain context that they know and love very dearly, to solve whatever problem that they're after.*

5.6 Magic: the opaqueness of technical analysis work leads to communication breakdowns

After some initial problem framing and bridging work, the technical work of data analysis begins. Technical work in data science has been widely-studied [23, 125, 232, 91], so we will not repeat those details here. In short, data scientists spend most of this time in an "inner loop" (e.g., Figure 5.1) writing analysis code in environments based on R, Python, or other tool ecosystems. Note that this process is iterative, so they may return to another round of problem framing after getting some analysis results and showing it to the client, as the outer loop in Figure 5.3 shows.

Prior studies of data science collaboration [131, 154, 235] focused on the ways in which data science teams and other stakeholders collaborate on the technical aspects of their work (e.g., what tools they use or how data gets passed between people). Thus in this section we focus on how clients (who may not know all these technical details) perceive this work when they are not involved in the day-to-day implementation.

5.6.1 Clients perceive data science work as "magic"

Most notably, half the participants we interviewed (P1, P3, P4, P5, P9) mentioned the word "magic" when describing how their clients perceived their work, without any specific prompting from the interviewer. We were surprised by the frequency with which participants talked about magic, and the way they all used the word to describe similar phenomena. P4, who

works at a research-driven medical center, took pride in his work with doctors, but also pointed out how little insight those doctors have into the data analysis process:

> *Usually a lot of them are clinicians, like grateful physicians, who essentially think you do magic, right? They don't know how you do it. They are never going to figure out how you do it. And they're just like, "that's amazing" and "do more!"*

Although P4's clients make him feel like a wizard, the opaqueness of technical work from the client's perspective can also make data scientists feel like their work is underappreciated or misunderstood. P1 expressed how she feels her analysis work is trivialized by clients since it is to opaque to them: *"There's a lot of like, you know, 'You go and do your magic and then come back with the numbers.' Which I hate, I really hate that. Can you tell? I just had a meeting like this last week."*

5.6.2 How perceptions of "magic" lead to communication breakdowns

Others mentioned how the 'magic' aspect of data analysis work reflects a breakdown in communication between them and their client. Since most day-to-day technical analysis work is opaque to clients, it is difficult for them to understand which parts are easy and which are hard for the data scientist to complete. P3 explains how this issue manifests in her work:

> *We have a lot of communication issues. Those mostly center around like how requests are made to me. For example, I'll be asked for data, and it's kind of like data is this magic thing that they think I just have on my computer. Like, five minutes later they expect me to say: "Here's your data!"*

P3 continued to explain how since she had access to certain data sets that were relevant for her client, the client assumed she had equally-easy access to related data that they had not discussed before (which was not true).

This opaqueness also makes it hard for clients to set realistic expectations about how long some analysis task should take. P8 works in a large federal environmental agency in North America, and here she explains the challenges of showing the client that she is making consistent progress even though much of her work is invisible to them:

> *When you're solving problems with data, there's this idea that you should be able to hit the ground running. And of course we all know that you could spend weeks and weeks and weeks just trying to actually get that data, and get it into a format and understand it and read metadata. And so I think one challenge of a data science team is to be able to figure ways to keep everyone engaged and show value in those early stages of a data science work or data analysis workflow that are kind of less exciting. Often not much to show.*

The tremendous effort involved in acquiring and cleaning data has been well-studied, and it is widely known that it amounts to a significant portion of a data scientist's working time [124, 125]. However, this fact is still not visible to many clients, as P1 explained when telling the interviewer about a time she spent two weeks cleaning and wrangling human-generated free-text data:

> *If you ask our business stakeholders what I've been doing this whole time, and they would say, "I actually don't know. I think she's working hard. She seems busy, but I don't really know what she's doing." That work is not visible to them, nor should it be. This is not for them to consume. This is to build the foundations of a product that will then be for them to consume. But it's very necessary.*

In sum, since the day-to-day technical work seems "magical" and opaque to many clients, they often have a hard time communicating productively with data scientists until they can see some partial results (see next section), which may not come for a few weeks at a time.

5.7 Counseling: Showing results to clients and helping them cope

After some period of technical analysis work (e.g., a few days or weeks), the data scientist shows some results to their client. This section describes the most salient types of communication that occur when the client sees these partial results.

5.7.1 Helping clients cope with unexpected analysis results

Without prompting, many of our participants mentioned doing some informal counseling during result presentation meetings to help clients "cope" with unfavorable results. Like in many

fields of science, technology, and design, there may be no data or experimental specification that will yield a result that clients are happy to receive. In data science, clients may want to show via a data analysis that their vaccine candidate is effective at preventing disease, that a governmental policy has reduced air pollution, or that they can predict with high accuracy which customers will spend the most money on their e-commerce site. But it is not clear whether these aims have been achieved until a data scientist has completed their analysis after a span of weeks or months. Even though P7 has over 20 years of experience working as a quantitative geneticist, he still struggles when delivering results that clients will not like to hear: *"How to present bad news? Um, I don't really know how to do it. And I would say from my experiences in my career, I still haven't figured it out."*

Data scientists help clients cope with unfortunate circumstances throughout the course of an analysis, not just when presenting results. For instance, P2 was brought in after the data had already been collected, but then she had to tell her client about a serious flaw she eventually discovered:

> *Some respond better than others. I've had one extreme with a collaborator who we discovered had a confounded study design. They had already done the experiment and so we had to go to them and explain to them what happened. They were all excited about the biology, but once we told them, they went through the full seven stages of grief in that one hour.*

Also, data scientists acquire their training from a variety of academic fields [138], which often have different norms for canonical statistical methods, reporting of findings, and thresholds for significance of results. Here P9 recounted his experience doing counseling ('playing therapy') to reassure a client of the soundness of his results when the client had different field-specific norms:

> *There's definitely also like a tension of field-specific expectations or norms and your principles or values as an outside consultant or analyst. And then it gets manifested as like "my analyst did something wrong" or "they didn't do what I told them." And then you just sort of feel like you're playing therapy with them, trying to reason out why what you did is actually okay.*

5.7.2 Power dynamics when presenting results to clients

Power dynamics appear here as well: Some of our participants work in environments (e.g., large companies) where there are many data scientists that clients can consult with. Therefore, they are afraid that they may be 'replaced' if they deliver results that a client does not like. P3 laments that clients do not understand how much they may be giving up by finding a new data scientist (whom they hope will produce more favorable analysis results) rather than sticking with the current one:

> *When someone doesn't get the result they want as fast as they want, they think that it could be your fault. And so they say, "Well let's just bring in another one of you!" But they don't understand that I can't just give you my code. They don't understand that it doesn't just transfer without any effort, like the other person has to understand what's going on.*

Ultimately, helping clients cope with analysis results comes down to the data scientist's ability to empathize. They often have to find a balance where they are not so empathetic as to make undesirable results look better than they actually are, but to still be mindful about how clients are going to be affected by bad news. For instance, P1 explained her approach to empathy for her clients and how she frames analysis results in terms of how it will affect her client's future plans:

> *If something doesn't go the way that somebody likes, how do you handle that? I think I'm a pretty empathetic person. And so being able to put myself in their shoes and think, "Ugh, this might mean they're not going to get that grant." Or, "Oh, this means the hard work that their team put in is not reflected in our data. And therefore we're not going to be able to prove that [a medical intervention] helped."*

In sum, the main loop in Figure 5.3 continues until the end of the client engagement. Ideally there is an outcome that satisfies the client, but sometimes data scientists must help clients cope with a less-than-desirable ending. Here P8 reflects on different ways that engagements can end:

> *Success is not just the thing you build, but was it a successful process? And many*

times things are not successful in that the analysis either didn't get finished or you didn't find anything, right? One of the things about looking for patterns in data is like, well, we look for patterns in data and there were no patterns.

5.8 Participants' thoughts about data science education

For our final set of findings, we report on an unexpected theme that many participants raised. Even though the focus of our study was not on data science education, we found it interesting that the majority of our participants (P1, P2, P3, P4, P6, P8, P10) brought up this topic during their interviews without prompting. Specifically, they recalled that they either spent no time or minimal time learning how to navigate the socio-emotional work of data science (i.e., the 'outer-loop' workflow stages we have described so far) during their formal education. For instance, P1 said:

> *I think a lot of it, you can't really learn in school. It's hard to not learn it on the job. [...] There's so much that I could not have learned in grad school about how to work with stakeholders, the psychology of it all, like when to use what kind of conversational style or rapport or how to report my results.*

P2 mentioned how the teaching of software tools for doing data science is well-understood in academia, but that is not sufficient for helping students to become effective data scientists:

> *We are master tool teachers. We teach tools, we teach software, we teach methods. But by no means does that mean when a student walks out the door, they know what it means to produce a good data analysis. [...] We have to be able to tell students in a 12 to 16 week course what it means to be able to have that initial conversation [with a client] in an effective manner, and then how to get them to build a 'successful' data analysis and an effective, efficient manner.*

In Section 5.9.3 we discuss some implications of these findings for data science education.

5.9 Discussion

We reflect on our study findings in terms of its relationship to prior work in CSCW, the socio-emotional infrastructure that our clients must build, implications for education, the

parallels between data science and design work, and implications for tool design.

5.9.1 Relationship to Prior CSCW Work

Our findings enhance the perspective of data scientists' involvement in computer-supported cooperative work. In the following subsections we discuss how our findings in all stages of our outer-loop workflow from Figure 5.2 both complement and extend relevant prior work in CSCW.

Groundwork

During the Groundwork stage, data scientists try to build relationships and establish trust with clients before a collaboration can even begin. Sometimes these relationships are already arranged beforehand; for example, the participants in the studies of Mao et al. [154] and Hou and Wang [115] already have relationships with clients within the context of an ongoing research consortium or a hackathon, respectively, so those studies did not cover the Groundwork or Orienting stages (next subsection) that we did. Mao et al. contributed novel insights about building common ground during data science collaborations [154], and we extend that notion to laying groundwork before these collaborations even begin.

The importance of establishing trust is also explored in work by Lawrence [143] and (separately) Tang [212], who point out that trust in the beginning of client-consultant relationships is dependent on reports from third-parties and overall reputation. Our work extends their findings by adding firsthand descriptions of how data scientists do their own "marketing" to build their reputations both within their organizations and externally in the data science community. Lawrence showed how large e-Science research projects are facilitated by years-long relationships and foundations of trust that started long ago when leaders of those projects were training together or had advisor-advisee relationships in formal Ph.D. programs [143]. Our work extends those findings by showing what needs to be involved at the very beginning of client-based relationships. As data science work becomes more widespread and democratized, fewer people will be able to rely on traditional relationships fostered during long-term academic training or Ph.D.-level

professional networks; therefore, understanding how these cooperative relationships start in diverse settings is important for improving data science practices.

Orienting

This is the process through which data scientists first enter into collaborations and 'find their bearings.' Since our participants are well-established, with 5 to 25 years of experience (Section 5.2), their collaborations start when a client approaches them through one of the five entry points from Section 5.4. However this is not always the arrangement in all data science collaborations. For example, Choi and Tausczik describe the experiences of citizen scientists, journalists, and volunteers working on open civic data projects that either directly benefit a community or their local government [51]. In their case the people working as data scientists are the ones initiating the collaborations, and often their progress on a project is self-directed rather than client-directed. Mao et al. describe a few large-scale crowdsourcing collaborations where many domain experts are consulting with many data scientists; those data scientists reported that it could be difficult to find entry points for projects because many collaborations are short-lived, which does not give them enough time to develop relationships that would facilitate follow-up contact with previous clients [154]. Overall, our work extends prior CSCW studies by characterizing the orienting activities of data scientists who have worked in years-long collaborations through many project cycles.

Problem Framing

The process of Problem Framing involves asking the right questions: Asking questions allows data scientists to narrow down exactly what data-informed question a client is interested in addressing, which then allows data scientists to propose technical methods that could be used to answer this distilled question. Lawrence's findings show that it can be unclear when to ask questions and whose role it is to ask and answer questions when large groups of people are collaborating in multidisciplinary e-Science projects [143]; in our setting the roles are more

clear-cut since the data scientist takes the lead on problem framing. Mao et al. focus extensively on question-asking: among their most important findings is that asking the right question can lead to new lines of research and scientific discoveries [154]. We extend those findings by showing that what separates Problem Framing from just asking the right questions is that Problem Framing is a methodical process that encompasses more than honing in on questions: it also helps them plan for future products and find agreement with clients about deliverables (e.g., what graphs or tables do they want to see?); we also connected this process to how professional designers work [55, 56]. In addition, our findings in this stage extend prior CSCW study findings by revealing the emotional labor [90] that data scientists have to engage in to make their clients feel at ease when put in the uncomfortable position of frequently questioning clients who are often in positions of power.

Bridging the Gap

To bridge the gap between data and domain expertise, Mao et al. reported that data scientists in their study frequently asked their collaborators (biomedical domain scientists) about what the ambiguous variables in their data represented [154]. Third parties can also serve as bridges, like how Hou and Wang's "client teams" acted as brokers during a data hackathon for nonprofit organizations to inform volunteer data scientists about the origin and domain of their data [115]. Additionally, Choi and Tausczik describe open civic data projects where domain experts explain the features of government data to data scientists [51]. In all of this prior work, though, most of the knowledge transfer *is in one direction*: from domain expert to data scientist. We extend these prior findings by showing how the data scientists in our study revealed that they provide the social and contextual scaffolding necessary for teaching and learning in both directions: Our participants reported that this both helps them understand the client's domain (which has been well-covered in prior work) and also transfers knowledge from data scientist to domain expert. Most notably, we discovered that data scientists can help their clients to work more effectively with them if they proactively educate clients about the capabilities and

limitations of relevant data-analytic methods.

Magic

The problem of opaqueness pervades many kinds of data science collaborations. For instance, Mao et al. [154] and Hill et al. [112] describe the pervasive interpretability problems of "black box" methods, where project collaborators, or in some cases the data scientists themselves, are unable to understand or explain the results produced by data science or machine learning algorithms. While this is an important problem, it is different from what our participants meant when they mentioned 'magic' during interviews. Our participants meant that clients perceived an opaqueness *in the actual process of how data analysis work was done*, even when the outputs were interpretable. Thus, instead of focusing on the opaqueness of algorithms (which is well-covered by prior work), our findings shed new light on the opaqueness that clients perceive in the data analysis process. This sometimes results in their dismissive attitudes about technical work (e.g., quotes like "go do your magic") as clients chose to exclude themselves from the nitty-gritty of technical processes. This attitude frustrated our data scientist participants since it belittled their role in the project, as if they were themselves a "black box" like an algorithm. It also hindered communication, since technical decisions during data analysis project may need finer-grained input from clients. Lawrence touches on some of these issues in her findings about the complexities of communication in a distributed e-Science consortium [143]. In sum, clients' *perceptions about* and *willingness to participate in* technical data analysis work can dramatically affect both the work itself and the emotional state of data scientists who work with them.

Counseling

Prior studies of data science collaborations [154, 115, 235, 131] have not yet investigated the emotional labor [90] required when data scientists present unexpected or sub-optimal results to clients. Thus, our findings add to this literature by highlighting what goes on in the Counseling stage. Our investigation of these issues is influenced by Slovák and Fitzpatrick's call for

more integration between CSCW and lessons from the field of social and emotional skills development [205], and Hochschild's work on the role of emotional labor as a professional skill [114]. Our findings also include participants' reports about their own feelings as the bearers of bad news, and how their awareness of the attention and resources that their clients have put into a project affects how they frame potential disappointments for clients. This contribution is novel within the literature at the intersection of CSCW and data science, but more broadly it follows prior work by Moncur about how HCI researchers protect themselves from emotional harm [163] and work by Raval and Dourish about the emotional labor that rideshare drivers perform for their clients [190].

5.9.2 Socio-Emotional Infrastructure for Data Science

The outer-loop workflow that our findings reveal (Figure 5.2) consists not of technical activities but rather of *socio-emotional* activities such as developing trust, building common ground, and providing emotional support to clients throughout the lifetime of a project. In addition to all of the technical work required to be a professional data scientist, our participants conveyed how they navigate relationships with their clients and the work culture in which they are situated, as well as how they manage emotions and reactions that arise during an analysis. Our investigation shows that relationship-building and maintenance with clients is made up of many systematized interactions that are common and repeated both during an individual data scientist's career and across data scientists working in different domains in industry and academia. We characterize this kind of work as *socio-emotional infrastructure* – interpersonal scaffolding that data scientists are expected to provide for clients. This sort of human-to-human infrastructure complements the technical infrastructure that has been widely-studied in prior work, such as computational notebook platforms [141], cloud computing services, and I.T. infrastructure for data processing [228].

According to our participants, a major purpose for providing this infrastructure is to avoid dysfunction and encourage better practices throughout an analysis. For example, in the orienting

stage a client may have an analysis method they want to use that the data scientist believes is inappropriate given the analysis question they are trying to answer. In this case, the data scientist may need to spend some of the social capital they accrued earlier (during the groundwork stage) to help the client avoid the mistake of using a method that may be too costly or technically burdensome. In this way the data scientist is lending their expertise as infrastructural support for making decisions about an analysis, which can lead to better results and cost savings down the line.

Without this analytical expertise and contextual support, clients would otherwise be left to navigate these decisions themselves. Considering the democratization of data science tools and the proliferation of AutoML [221] (i.e., AI support for data science) and 'push-button' cloud-based data science products, it is easier than ever for an end user to upload their data to one of these services and get results without ever interacting with a human being. Although this method of doing data science may have less friction and lower costs, there is little guidance in these products to indicate whether the end user is using appropriate methods for their circumstances, whether the methods are working as they intended, whether the user has the right data to answer the questions they are interested in, or how their results should or should *not* be interpreted. We believe our findings show how ***the socio-emotional infrastructure that data scientists provide will not be easily automated away***, even as increasingly sophisticated AutoML methods become commoditized.

5.9.3 Implications for Data Science Education

Even though the focus of our study was not on data science education, many participants brought up this topic, which we describe in Section 5.8. Most notably, they mentioned a lack of training in socio-emotional skills that surround the technical work. Considering how these socio-emotional skills are vital to our participants on the job, we believe that data science curricula should place more emphasis on teaching them. This is a known gap in current curricula, since a recent survey of data science course descriptions does not mention any focus on building

these skills [139]. We hypothesize that socio-emotional skills may be rarely taught in data science programs because the concepts required to excel in the outer-loop workflow of Figure 5.2 are more challenging to teach than, say, specific programming languages, technical tools, or statistical methods. But tools-related knowledge is increasingly becoming commoditized via free online tutorials, MOOCs, and data science coding bootcamps. However, what is much harder to commoditize are the subtle interpersonal dynamics that occur between data scientists and their clients, so perhaps where formal education can add the most value is in fostering these skills in a more rigorous way.

5.9.4 Data Scientists as Designers

Our study findings highlight the ways in which data scientists function as designers and co-designers of analyses when working alongside their clients. We noticed that many of the conversations and behaviors they described run parallel to the kinds of design processes that have been studied by ethnographers of design [55, 56]; some data science practitioners have also discussed these parallels in research papers, blog posts, and podcasts [181, 180, 179, 210]. For example, our participants reported needing to reconcile many competing interests – including the client's ambitions for a project, the limitations of available data sources, and constraints about the time and resources available to work on an analysis. Ultimately data scientists need to strike the right balance between all of these factors to create a coherent analysis that meets their own standards for ethics and rigor while also properly serving their client's needs. Designers must similarly manage this socio-emotional process of negotiating with stakeholders and working within prescribed limitations on client projects.

Like designers, data scientists help their clients approach problems via the iterative process of problem framing [55]. Several of our participants mentioned how even when a client would approach them with a problem that the client thought was well-defined, they would still begin with a series of questions to get the client to think more deeply about their prior assumptions. This made clients realize that their initial problem was unclear, and then our

participants would work with them to re-frame their problem and revise their goals. Similar to how designers work, this style of breaking down problems and then re-framing them means that data scientists are often embracing uncertainty and helping clients work through the resulting ambiguity. Coping with uncertainty is such an important skill in data science that it has been incorporated into some data science courses [138]. Also like designers, data scientists need to cyclically return to the problem framing stage if new data or new constraints are introduced as an analysis proceeds (Figure 5.3).

In addition, both data scientists and designers draw upon a repertoire of examples from past projects that they creatively adapt to new challenges. For a data scientist, starting a new analysis project by seeing how it is similar to a past project of theirs can help constrain the design space of techniques to apply. However, a problem faced by data scientists that may not be as common in other design disciples is the lack of publicly-available analysis examples due to sensitive or proprietary data [182, 111]. Thus, there are missed opportunities for data scientists to learn from the experiences of *others*, whereas designers in many fields can draw upon many publicly-visible examples to learn from (e.g., in visual design, user interface design, industrial/product design).

Finally, like designers, data scientists use various concrete representations to aid in problem framing, bridging knowledge gaps, and discussing results with clients. In design studies, visual representations are often used for communication, namely for creating a shared space to work out ideas and make them more concrete [56]. Data scientists frequently create graphs, diagrams, tables, statistical models, and computational notebooks as representations to communicate with clients. In the next subsection we will discuss some ways that these current representations fall short.

5.9.5 Tool Design: Lack of Expressive Representations for Collaborative Data Science

We conclude by discussing some implications of our study findings on tool design for collaborative data science. In the prior section we discussed ways in which data scientists work like designers. One salient attribute of design work is the ability to use representations as an external memory for the working state of a problem and as an artifact to facilitate communication with clients. For instance, in visual design a common representation may be an Adobe Illustrator or Photoshop file that resides in a shared folder. *The communication challenges in our study findings suggest that data scientists may lack representations with the same expressiveness in facilitating client interactions as those used in other design fields.* To show a concrete example of this contrast, we will compare the work of data scientists to that of product designers and web designers.

Product designers use computer-aided design (CAD) software to create 3D models of an object that will eventually exist in the real world. They use these models as prototypes to see if the object fits a client's specifications, and clients can easily see and comment on parts of the object's design that may need to be changed. Product designers can also place their models into simulated environments so that clients can see how the object would look in its intended environment (e.g., on a kitchen counter); they can even 3D-print low-fidelity prototypes to place in real physical environments. Similarly, web designers prototype their work using computer-aided 'CAD-like' tools such as Figma, Sketch, or Adobe XD. Even though these designs exist purely as pixels under glass in a virtual world, there is still a known set of physical analogues such as space layout constraints, UI widgets such as scrollbars and menus, and interaction physics like scroll momentum and touchscreen gesture responsiveness. With tools like Figma, web designers can make prototypes look and feel 'real enough' for clients to give rich feedback on it throughout the design process.

Tools like CAD and Figma both allow designers to work at the level of their expert

understanding, while still allowing clients to intuitively experience the artifact that is being designed (e.g., a 3D object or a responsive mobile website). For example, in Figma a designer can manually specify advanced CSS rules, yet clients can intuitively grasp the effects of CSS changes on-screen. This allows clients to easily give feedback like *"Can you make this menu bar a darker shade of blue and a bit taller?"* without needing to understand the technical details of how CSS layouts work.

In contrast, data scientists still lack tools like CAD, Photoshop, or Figma which would make their ongoing work so transparent that their clients can ask the data science analogues of *"Can you make this menu bar a darker shade of blue and a bit taller?"* Although hundreds of tools, frameworks, libraries, and computational notebook technologies exist for data scientists, these representations are made to facilitate the productivity of those who are analyzing data, *not* those who need to consume the eventual analysis results. For instance, if a data scientist walked into a client meeting, opened up a Jupyter notebook filled with hundreds of lines of specialized Python code coupled with a CSV file with thousands of rows, then that would probably not be a very productive meeting! Instead, data scientists must painstakingly distill these opaque representations into summary charts, graphs, and tables before showing clients, which ends up losing a lot of richness of the original 'uncompressed' representations. In contrast, designers can show the original CAD, Photoshop, or Figma files to clients and engage in meaningful conversations about those *primary* representations.

Data scientists also need to do bridging work with clients (Section 5.5.2) without the benefit of rich representations. This work requires the data scientist to educate the client about how certain methods are used in an analysis, or how statistical phenomena affect the client's problem. Our participants reported significant difficulty in explaining these concepts to clients, often relying on their own experiences teaching and learning statistics to convey a concept. They could not benefit from basic design representations like sketches or drawings, because clients often do not have enough of an understanding of data science methods to understand the statistical concepts that underlie these drawings (e.g., properties of statistical distributions or

how certain factors interact).

In sum, we believe that in order to create collaborative tools to facilitate the outer loop stages discovered by our study, we need to invent more expressive representations. Existing CSCW tools for data science such as real-time collaborative Jupyter notebooks [141], version control, and project management systems help coordinate the inner-loop of technical work, but those representations are still at the code and data level, which is not as useful for interfacing with clients. Unlike in product or web design, in data science there is often no shared point of reference between the data scientist and their client: Although data are collected from observations in the physical and digital world, the ongoing work in analyzing that data often occupies an abstract mathematical world that remains opaque (and thus "magical") to clients. It is an open question how we might design a tool with the transparency of CAD, Photoshop, or Figma for the outer loop of collaborative data science; our hunch is that it will require a representation that is richer than code or data.

Perhaps one step toward these richer representations is to reify workflow diagrams such as ours in Figure 5.2 and finer-grained versions of it into a workflow system [69]. Since the early days of data science, workflow diagrams have been used to track data provenance and manage collaborative work in a data-centric way, such as in graphical workflow programming systems like Kepler [149], Taverna [174], and VisTrails [201]. We could potentially generalize this technology to track progress in the 'outer-loop' stages that we identified in this chapter in addition to the 'inner-loop' mechanics of data flows. Doing so would turn workflow diagrams into boundary objects akin to design sketches or UX flows in user-centered design collaborations [37]. Our envisioned diagrams could be embedded into contemporary workflow systems such as Asana, Notion, or Jira[2] so that data scientists can keep their clients up-to-date with the state of their project; this could also help to reveal some of the opaque 'magic' throughout the analysis process. However, as Dourish, Suchman, and others in the CSCW community have pointed out, workflow technologies risk hindering flexibility in creative work (such as those performed

[2]https://asana.com/, https://www.notion.so/, https://www.atlassian.com/software/jira/

by the data scientists that we studied) because they tend to categorize work into fixed classes of tasks [69, 211]; moreover, usually those with more power are the ones determining the categories [211]. Thus, we must be careful to design a workflow technology that preserves such flexibility while still providing sufficient process transparency for clients.

5.10 Conclusion

This chapter has argued via an interview study that we should view collaborative data science work beyond the usual lens of technical workflows like those in Figure 5.1 that now pervade the field. Rather, we zoom out to characterize the interpersonal dynamics between data scientists and their clients by synthesizing a novel outer-loop workflow to capture this form of collaborative work. This workflow involves laying groundwork by building trust and reputation before a project begins, orienting to five possible entry points of client engagements, problem framing, bridging the gap between data science and domain expertise, technical magic, and counseling to help clients emotionally cope with analysis results. A unique aspect of our study is that interview participants working in both industry and academia provided their insights, thus moving the conversation forward about the professional needs of data scientists beyond the usual suggested refinements for technical tools. We believe that their perspectives illustrate a broad spectrum of experiences that can inform the intersection of data science and design practice. One potential avenue for future work is to dig deeper into possible differences between industry and academic outer-loop workflows. By studying practitioners in both settings, we aim to identify common problems (and notable differences) so that we can propose generally-applicable solutions to challenges in outer-loop workflows. We conclude by showing how this workflow is not well-supported by current data science education, tool development, or theories of data science practice, and we look forward to addressing those challenges in future work.

5.11 Acknowledgements

Chapter 5, in full, is a reprint of the material as it appears in the proceedings of the 2021 ACM Conference on Computer Supported Cooperative Work as Orienting, Framing, Bridging, Magic, and Counseling: How Data Scientists Navigate the Outer Loop of Client Collaborations in Industry and Academia. Sean Kross and Philip J. Guo. 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 6

Tools for Visualizing Data Analysis Pipelines (Datamations and Tidy Data Tutor)

6.1 Introduction

As the world becomes more data-driven, we are increasingly presented with plots and tables that convey the results of complex analyses involving intricate datasets. A great deal of work has been done on how to make these figures easy for readers to comprehend, for instance in helping people decode data values [53, 151, 107], make high-level inferences [223], and perceive related uncertainties [117, 80]. All of this work has led to vast improvements in data visualization and data communication, and yet it has focused on conveying only a small part of what is involved in creating these plots and tables.

By the time a reader is presented with a figure, the underlying data have most likely been extensively processed (*e.g.*, filtered, grouped, aggregated, augmented, and reshaped), but conventional plots and tables show only the end results of the analyses that led to them. As such, while the reader is often able to decode the values in a plot or table, they may be left wondering how those values were arrived at and what they actually mean. In short, it is easy to see a figure and be unsure of exactly what went into creating it and how that affects what one should (and shouldn't) take away from it.

One solution to this problem is giving the reader more context around the steps that led to any given plot or table. Often times this is done through text and captions written to accompany a

figure. Such write-ups can be time-consuming to produce and are often only an approximation to the actual steps that were taken in any data analysis pipeline. As a result, these descriptions can introduce ambiguities, gloss over important steps in an analysis, or fail to convey analysis steps in an easy-to-comprehend manner. Another potential solution is to simply share the code that generated a figure, but this can have the opposite problem: while code is precise and provides exact information on the process that led to a figure, it is often difficult for novices and experts alike to read and understand someone else's code.

This is where we see an opportunity to improve how data analyses are communicated to readers. In this chapter, we introduce two systems: *datamations* and *tidy data tutor*. These systems automatically generate animations or illustrations that explain the data analysis pipeline that produced a given plot or table. The idea behind these systems is relatively simple: each step in a data analysis pipeline can be programmatically mapped onto a visual transformation of the underlying dataset, and these transformations can be chained together to produce an animated explainer for a plot for table. Datamations can, for instance, show data being filtered, split into different groups, and aggregated into summary statistics.

To clarify our contribution, we are aware of existing examples of animations in data journalism [29] and education [140] that have been created to explain data analyses. These serve as inspirations for our work, but many of them are custom animations that require a great deal of time and manual effort to create, and do not readily generalize to other scenarios. Furthermore, custom animations for data analysis pipelines have not to our knowledge been tested in controlled experiments to assess the degree to which they help readers understand plots and tables they are shown. Thus, our contribution is threefold: first, to formalize the idea of datamations and tidy data tutor for explaining entire data analysis pipelines; second, to provide a framework by which such animations can be automatically generated to explain both plots and tables; and third to run large-scale experiments to better understand the potential benefits of (and issues with) these animations.

In the remainder of the chapter, we present a framework for formalizing and automatically

generating datamations, and an explanation of tidy data tutor. Then we apply this framework to generate and test example datamations that explain an instance of Simpson's paradox [177], where a seemingly paradoxical reversal in final results can occur based on a small but important change to the underlying data analysis pipeline. We chose to study Simpson's paradox because it is a case where understanding the data analysis pipeline is critical for understanding the corresponding results. The problem that we presented participants involved a dataset of salaries where, looking across industry and academia, people with master's degrees make more money on average than people with PhD degrees overall, yet within both academia and industry people with master's degrees make less money on average than people with PhD degrees. We created plots and tables describing these results along with accompanying datamations that explain the process leading to these figures. We then used these datamations as stimuli in two pre-registered experiments that involve over 1,200 participants to test whether datamations are able to improve comprehension of Simpson's paradox. We investigated whether seeing datamations helps readers correctly identify that such a reversal is possible and/or whether it helps them choose the correct explanation for the reversal. We also used this experiment as a chance to collect qualitative feedback from participants on the benefits of datamations and ways in which they can be improved. We conclude with a discussion of participants' feedback, and we present thoughts for future research to be done in this area.

6.2 Design of Datamations

Datamations are animations that explain data analysis pipelines. Our design of datamations is informed by an abstraction of states and transitions, summarized in Table 6.1. Given a data analysis pipeline, we define states as all intermediate and final data values, and transitions are operations on these data. From there, we build mappings from data values to plots and tables, as well as mappings from data operations to different types of animations. We provide a prototype implementation of datamations as an R package.[1]

[1] https://github.com/seankross/datamations

Table 6.1. The datamations framework. Data values are mapped to states (shown as plots or tables) and operations on those values are mapped to transitions (shown as animations between plots or tables).

	Code	*map onto*	Visual
State	Data Values	→	Plot or Table
Transition	Data Operation	→	Animation

Table 6.2. Data operations in datamations and their corresponding animations.

Operation		Animation
`group_by`	→	Translate observations to depict splitting into groups
`arrange`	→	Reorder observations to depict sorting by a variable
`mutate`	→	Highlight observations to depict creating or modifying variables
`filter`	→	Observations disappear to depict removal from dataset
`summarize`	→	Observations collapse into summary values for each group

6.2.1 Data value-plot/table mapping

Datamations present intermediate and final data values as plots or tables. For tables, we assume that the data is in a "tidy" format [224], where each row corresponds to an observation and each column a variable, and there is one value per cell. The table visuals for datamations directly reflect the rows and columns in the data, supplemented with labels of variable names and values to help comprehension. Figures 6.7 and 6.8 are examples of table visuals, with the latter being a sample of the frames in the table-based datamations used in our experiments.

We have more flexibility when presenting data values as plots, and a large body of literature compares visualization types [53, 117, 134, 34] and even looks at automating the process of selecting the best type of visualization for a given dataset [164]. To facilitate the design of plot-based datamations, we make two design choices: first, we use natural-frequency encodings as often as possible, and second, we constrain the datamation so that the final plot (keyframe) is identical to the plot it is meant to explain. For frequency encodings, we use icon arrays and jittered scatter plots. The cells in tables and icon arrays, and the points in scatter plots, convey to the reader the sizes of different subgroups in the data, so that readers can intuitively

judge the relative proportions of these groups. We use these frequency encodings because they have been shown to lead to improved comprehension [88, 24] and decision making in various settings [102, 85, 126]. Figures 6.4 and 6.5 are examples of plot-based visuals, with Figure 6.5 showing frames in the plot-based datamations used in our experiments.

6.2.2 Data operation-animation mapping

Datamations support a set of operations in data analysis pipelines as described in Table 6.2. Given the plots and tables we generate from data states, the process of animating often amounts to translating visual markers representing individual data points from one coordinate to the next. As discussed above, here we show operations corresponding to R's `tidyverse/dplyr` verbs [225] informed by data science tutorials[2] and common transition and interaction types in the visualization literature [82, 116], but the framework we outline is general purpose and not language specific. For instance, regardless of the language an analysis is carried out in, the idea that points might be grouped together and summarized by an average remains broadly applicable.

This abstraction also incorporates *modularity* in the composition of datamations, which facilitates their automatic generation. Data analysis pipelines are very flexible and small changes—placing an aggregation operation before or after a group by operation—can have large consequences, some of which introduce logical (but not syntactical) errors. Since datamation transitions are directly determined by data operations (*e.g.*, `summarize()`) and intermediate data states are already computed in the code, datamations automatically reflect any such changes in the underlying pipeline, which can have the effect of alerting users to these important differences.

6.2.3 Following animation design principles

We follow design principles and empirical findings in the visualization literature when creating datamations. Dragicevic *et al.* find that when animating between views, it is best to use "slow-in/slow-out" transitions [70]. Accordingly, we use an exponential function[3] for easing

[2] https://rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf
[3] https://gganimate.com/reference/ease_aes.html

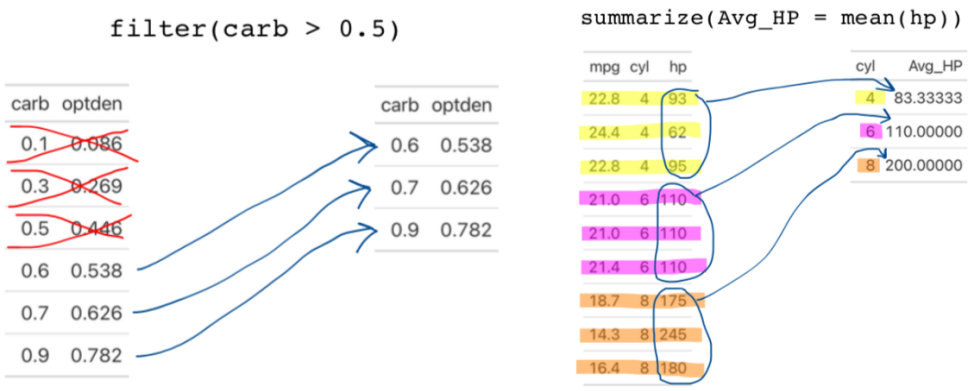


Figure 6.1. Hand drawn annotations showing how each function, with corresponding arguments, transforms a data frame from one state to the next. These kinds of illustrations inspired the illustrations created by Tidy Data Tutor.

the transitions between datamation states. When there is grouping in the data, all groups move at once instead of one group after another. This design choice is based on there being little advantage to "staggering" group animations [48]. In addition, data points within a group move in the same way. This is supported by *Gestalt* psychology: people perceive things similar in motion as belonging to the same group [47].

If we view the analysis pipeline as a whole, datamations are staged in that they animate one data operation at a time. *Staging* makes the animation easier to follow [238]. Datamations also generate only *necessary motion* and *meaningful motion* [82], as each animated transition is directly linked to a step in the analysis.

6.3 Tidy Data Tutor: Interactive Visualizations of Data Analysis Pipelines

The design of datamations significantly influenced the design of Tidy Data Tutor. Tidy Data Tutor was inspired by diagrams that data science instructors commonly draw by hand, like those in Figure 6.1, to explain how different functions transform data frames. Tidy Data Tutor is designed to help instructors create easy to share, self-contained interactive diagrams for teaching how different functions work in data analysis pipelines.

```r
penguins %>%
  select(species, bill_length_mm) %>%
  group_by(species) %>%
  arrange(desc(bill_length_mm), .by_group = TRUE) %>%
  slice(1)
```

Figure 6.2. A block of R code that implements a data analysis pipeline. The first four lines use the pipe operator (%>%) to move data frames from one function to the next. The first line only has the variable name of a data frame (`penguins`) and then every line afterwards uses a function to transform the data frame. Programmers can easily read code that uses the pipe operator since it minimizes the need for variable assignment and organizes each transformation for the data frame as a step in the pipeline.

Tidy Data Tutor is a publicly available web application that presents the user with a simple text editor where they can write R code. Users are advised that the last R expression in their code (which can take up multiple lines) should be a data analysis pipeline that uses the pipe operator (%>%). The user can then click the "Visualize %>% pipeline on last line" button and all of the code is sent to a server, where our backend serializes the R code into JSON. This JSON payload is then sent to the frontend of the web application which visually renders every step in the data analysis pipeline.

6.3.1 Transforming R Data Analysis Pipelines into JSON

The R code that a user types into the Tidy Data Tutor text editor is sent to a backend server that takes one R code file as input and produces a JSON payload that contains a trace of how a data frame is transformed at each step of the data analysis pipeline. The entire R code file is evaluated in a new execution environment, and every line of the file before the last line is evaluated normally. Only the last expression is evaluated in a way that traces every step of the data analysis pipeline. This way, any R code can be run before the data analysis pipeline, allowing users to do potentially complicated tasks to help properly set up the pipeline.

If the last line of the R code file is a data analysis pipeline that uses the pipe operator, then that line is parsed, and then each line of the pipeline is run in succession. This way, the state of the data frame at every step in the pipeline can be recorded. Each step of the pipeline is

analyzed using the state of the data frame before and after an individual pipeline step, and using the function and arguments that correspond to that step in the pipeline. This analysis begins by detecting which function is used at a particular step. Each function performs a different transformation on a data frame, so based on which function is used, certain assumptions can be made about how the data frame has been transformed. Depending on the function, the backend may look for rows and columns in the data frame have been rearranged, deleted, added, or renamed. Each of these transformations is added to a JSON trace containing the full list of transformations, such that there are as many steps in the JSON trace as their are pipe operators in the data analysis pipeline. Each step in the trace contains the following components:

- The function that is being visualized.
- All of the code for the step in the pipeline.
- A JSON converted version of both data frames before and after the transformation.
- A list of individual elements of the diagram that include:
  - The type of illustration for this element of the diagram.
  - Whether this illustration will apply to a row, a column, or an individual cell of the data frame.
  - Which of the two data frames (before and after the transformation) this element will be illustrating.
  - Information about the index, position, or group of this element.

After every step in the pipeline has been analyzed, the backend sends the trace back to the user's browser as a JSON payload. We impose several limitations on the backend computing environment to prevent misuse and abuse. Submitted code files can only be 5 kilobytes in size, which corresponds to about 100 lines of code. There is no way to upload files or data to the backend, so if users want to use their own data we recommend they copy and paste a data frame

into the code editor. There are only 512 megabytes of RAM available, and the backend will only run for 10 seconds to protect against infinite loops that might lock up resources. Since anyone could run arbitrary R code on our backend, every code execution runs in it's own isolated environment to protect our backend infrastructure.

6.3.2 Visualizing Data Analysis Pipelines

Anyone can go to `tidydatatutor.com` to interact with the frontend of our application. The homepage of Tidy Data Tutor features dozens of examples to help newcomers get started, or users can start out with an empty text editor where they can write or copy and paste in R code. Users can click the "Visualize %>% pipeline on last line" button and as long as the last line of R code is a data analysis pipeline, then the pipeline will be visualized below the text box.

An illustration is drawn for each step in the data analysis pipeline. Each illustration has the corresponding line of code at the top, and two data frames on the left and right side, representing the state of the data frame before the function is applied, and the state of the data frame afterwards. If all of the rows and columns cannot be displayed in the viewport, then our frontend collapses the rows or columns that are displayed. This collapsing feature is optimized to show rows and columns that illustrate how the data frame is being transformed. Users can also slide the collapsed rows and columns to reveal the parts of the data frame that have been hidden.

Different manipulations of the data frame use different annotations to highlight the type of transformation. For example, if a function extracts individual columns from a data frame, then an arrow is drawn from the columns that are selected in the data frame on the left, to show where the columns are positioned in the data frame on the right. If groups are assigned to each row based on a column, the an outline is drawn around the grouping column and each row gets the highlight of a different color that corresponds to that row's group assignment. Figure 6.3 shows a detailed walkthrough of the illustrations that would be created for the data analysis pipeline in Figure 6.2.

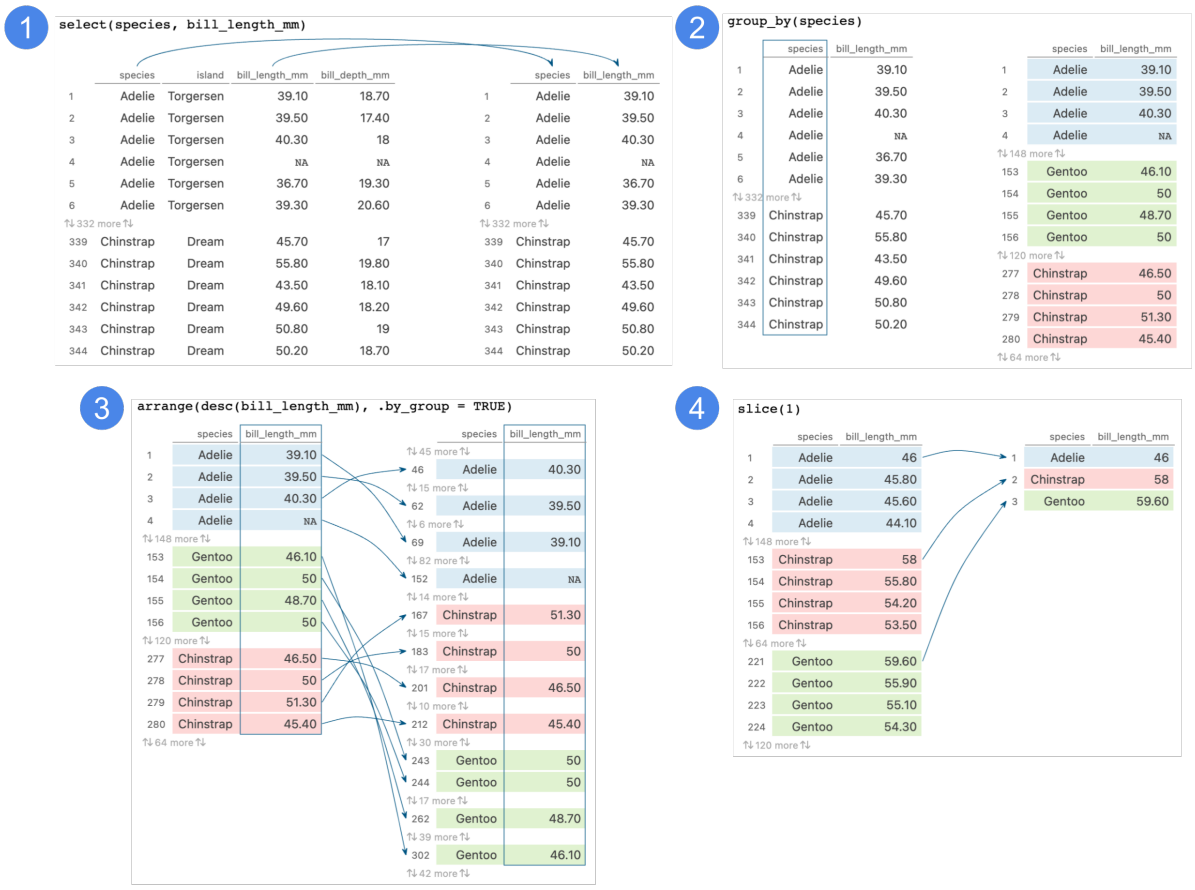


Figure 6.3. Illustrations rendered by Tidy Data Tutor, each corresponding to one pipe in Figure 6.2. 1. Two columns are picked out of the data frame using `select`. Arrows are drawn from the conserved columns in the *before* state of the data frame to the *after* state. 2. Grouping metadata is applied to the rows of the data frame. The column that is specified by `group_by` is outlined in the *before* data frame, and rows are colored based on their grouping in the *after* data frame. 3. The rows of the data frame are rearranged in descending order within their groups. Arrows indicate where each row moves relative to every other row, and the column that the rows are being arranged by is outlined. 4. The first row of each group is kept while all of the other rows are discarded. The arrows show which rows are conserved.

6.4 Overview of Experiments

We present two experiments where we gathered data from over 1,200 participants to gain a deeper understanding of the potential benefits of (and issues with) datamations for explaining data analysis pipelines. Both experiments introduced the same scenario to each participant: they were presented with the results of a hypothetical survey about employment analyzed in two different ways, with a seemingly contradictory reversal of results in average earnings due to an instance of Simpson's paradox. As the main intervention, we showed participants these results as either a set of static images only or with a set of datamations. Participants were then asked whether it was possible that these two sets of results could have come from the same underlying dataset. After answering this question, we revealed to participants that both results were in fact from the same dataset, and asked participants to select an explanation (from a set of eight possible choices) for how this potentially puzzling outcome is possible. We concluded by asking participants for their preferences about datamations compared to static images and for free text feedback on datamations.

For our main analyses we tested two pre-registered[4] hypotheses with this design:

- **H1: Acknowledging that a reversal is possible.** Participants who see datamations of data analysis pipelines will be able to correctly identify that a reversal is possible in the dataset more often than participants who see only static images of final results.
- **H2: Identifying the correct explanation.** Participants who see datamations of data analysis pipelines will choose the correct explanation for the existence of the apparent paradox in the dataset more often than participants who see only static images of final results.

We tested each of these hypotheses for two different settings in order to evaluate the two types of

[4]Pre-registration at: https://aspredicted.org/72qc9.pdf

datamations we have developed: one where people are shown plots and another where they are shown tables. Plot-based datamations resemble animated scatterplots and point-range plots, and in this experiment they were tested against a static image of point-range plots. In Table-based datamations, cells in a data table are animated as part of a grid, and they were tested against a static image of a data table. In total we conducted two between-subjects experiments (one for plot-based figures and one for table-based figures), each with two conditions (comparing static images to datamations).

For both experiments, we report all sample sizes, conditions, data exclusions, and measures for the main analyses mentioned in the pre-registration document, with all data and analyses available as supplementary material.[5] We determined the total sample size of 1,300 participants for our experiments based on estimates from a pilot study to enable detection of a difference of at least 10 percentage points in the probability of correct answers between the static final figures condition and the datamation condition with 80% power in a one-sided[6] comparison of proportions using the `prop.test()` function in R. All plots and datamations described below visualize synthetic data that we created for the purpose of being used in these experiments. Our experiments were approved by the IRB committee at Microsoft Research. A live version of the experiments can be found online[7], and the plots, tables, and animated GIFs containing datamations used as stimuli in the experiments are available in the online supplement. We ran the two experiments using one common task on Amazon's Mechanical Turk for ease of administration, but describe and analyze them separately (as declared in our pre-registration) as they apply to different use cases (plots versus tables). As per our pre-registration, we excluded the 27 participants who took part in previous pilots of the experiment before conducting any analyses.

[5] See https://osf.io/85njc/ for all supplemental material.

[6] We conducted one-sided tests because we are only interested in whether datamations improve upon the status quo of static figures.

[7] https://jhofman.github.io/datamations-chi2021-paper/simpsons_multiple_choice_all/ randomly redirects to one of the experimental conditions.

6.5 Experiment 1: Plot-Based Datamations

The goal of our experiments is to understand whether datamations help people better comprehend the plots and figures they are shown by exposing more information about the underlying data analysis pipeline that led to these figures. To evaluate this we designed experiments to see if datamations could help people understand an occurrence of Simpson's paradox within a dataset compared to a static figures containing the same results. We chose to study an instance of Simpson's paradox because it is a case where awareness of the underlying data analysis pipeline is crucial for resolving seemingly contradictory results.

We created a synthetic dataset which contains the results of a survey about employment, where each respondent provided information about whether they have a master's degree or a PhD, whether they work in industry or in academia, and their annual salary. This dataset was constructed to show that on average respondents with master's degrees made more money than respondents with PhDs, however when the data are grouped according to whether respondents work in academia or industry, then PhDs make more money on average within each group. In this experiment we created static plots that showed this reversal along with datamations that did the same, and randomly exposed participants to one of these two stimuli.

6.5.1 Stimuli

The first stimulus for this experiment (the entirety of Figure 6.4) is a static image of two point range plots that are the result of a modest data analysis pipeline. Both plots show average salaries and standard errors between groups of different degree holders, and the second plot further distinguishes between degree and work setting (academia versus industry).

The second stimulus for this experiment (illustrated in Figure 6.5) is a looping animated GIF file meant to show transitions between different parts of the a data analysis pipeline, which ultimately ends in a plot similar to what is displayed in Figure 6.4. While the static version shows two charts, this version shows two datamations: one that only accounts for degree when

This chart shows all data from the 100 respondents. **People with master's degrees make more money on average than people with PhD degrees.**

This chart shows all data from the 100 respondents. Within both academic and industry work settings, **people with master's degrees make less money on average than people with PhD degrees.**

Figure 6.4. In our first experiment we tested whether plot-based datamations help participants to understand data analysis pipelines. Participants were randomly assigned to see either two static plots (shown in this figure) or a animation (illustrated in Figure 6.5).

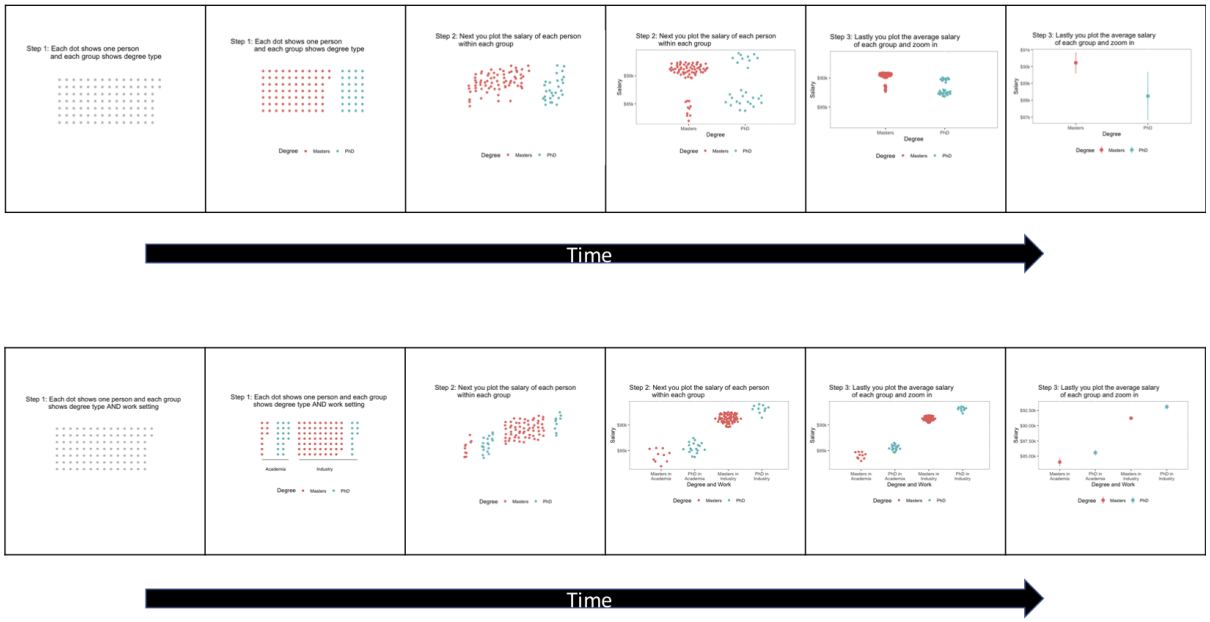


Figure 6.5. The timeline above illustrates two series of animations that were shown to participants in sequence. In the first datamation, individuals in the survey are displayed as grey circles, then they are colored in according to whether they have a master's degree or a PhD. The points then move to show a bee swarm plot according to each individual's salary by their degree. Finally, the points in the bee swarm converge to show line plots featuring means and 95% confidence intervals. The average salary for an individual with a master's degree appears to be higher than the average salary for someone with a PhD. The second datamation is similar to the first, except each point is separated further according to whether that individual works in industry or academia. In the last frame we can see that the trend is reversed: it appears PhD holders make more money on average than people with master's degrees.

calculating the average salary, and a second datamation that shows the contrasts for both degrees in industry and in academia separately. These plot-based datamations start as a grid of points that are then colored and arranged according to what type of degree an individual has (in the first case) or their degree and their work setting combined (in the second). These points then shift into a bee swarm plot that show salary on the Y axis and category (according to degree and work setting) on the X axis. Finally these points contract onto one point representing the group average salary, ending the transformation as a point range plot.

6.5.2 Procedure

First we provided participants with an overview of the study and presented them with a consent form. Specifically we told participants that they would be asked questions about a dataset and that they should not consult external resources to formulate their answers. After this introduction participants saw the following:

> Imagine that you are an analyst working for a think tank. You conducted a salary survey with 100 respondents in June 2018. Each respondent worked in either industry (companies) or academia (colleges and universities) at the time of the survey. Also, each respondent had either a master's or a PhD degree. Each of the 100 respondents reported:
>
> - **Work setting**: whether they worked in academia or industry at the time of the survey.
> - **Degree**: their highest education level obtained (master's or phd degree).
> - Their current annual **salary**.

After clicking a button acknowledging that they had read this information, participants were shown one of two conditions: either a static image, or a series of datamations. The static image that participants saw is in Figure 6.4, and an illustration of the series of datamations that participants saw can be found in Figure 6.5. For participants who saw datamations, they were presented with six datamations in sequence, with each datamation corresponding to a step in the data analysis pipeline. Each of these datamations played on a loop so that the animation restarted after it reached the last frame. After seeing each abbreviated datamation in sequence,

participants in this condition saw the entire datamation pieced together, showing an animation of the entire data analysis pipeline. Finally participants in the datamation condition were shown both the complete datamations and the final static images in Figure 6.4.

After seeing either only a static image in one condition, or a series of datamations and a static image, participants were asked:

> Note that compared to people with PhDs, people with master's degrees make more in the left chart, but make less in the right chart. Does it seem impossible to you that the results would come out this way?

Participants could wither select *Yes, it seems impossible* or *No, it seems possible*, the latter being the correct answer.

After submitting their answer participants were provided with the correct answer (that the results are in fact from the same dataset, without any data exclusions or manipulations) and then asked to select one of eight multiple choice explanations that they thought best accounted for the seemingly paradoxical reversal. The text informing them of the correct answer is below:

> **It turns out that these two charts are made up of the exact same data, just grouped differently.** That is, 100 people's salaries are represented in the left graph and the same 100 people's salaries are represented in the right graph. There is no mistake in the charts, but it may seem like a **paradox** when both are true:
>
> - The left chart shows that people with master's degrees make **more** money than people with PhD degrees on average.
> - The right chart shows that people with master's degrees make **less** money than people with PhD degrees on average, both inside industry and inside academia.
>
> Which of the following could explain how both statements are true? **This is the main point of this experiment. Please take it seriously.**
>
> Participants could then select from the following answer choices:

- Most people with a master's degree work in industry, which pays more and drives up the average master's salary in the left chart.
- People with neither master's nor PhDs are factored into the right chart, which biases the averages in opposing directions.

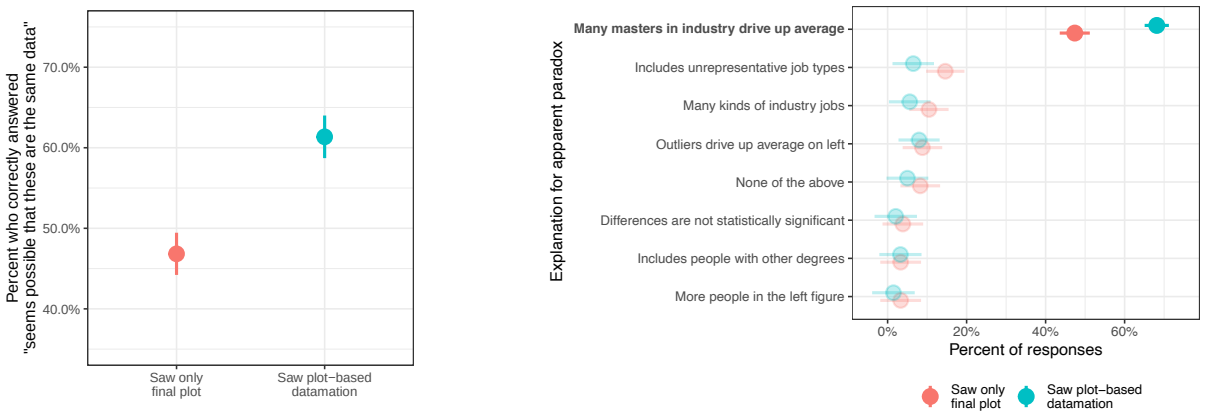
- In the right chart, there are many kinds of industry jobs, but fewer kinds of academic jobs.
- The differences in the left chart are not statistically significant and therefore could be due to chance.
- Due to outliers, the master's point on the left chart can be higher than both master's points on the right chart.
- The chart on the left includes salaries from people who work in neither industry nor academia and who are unrepresentative of the general trend.
- The left chart shows data from more respondents than the right chart, so it is not appropriate to compare the two charts.
- None of the above explain the difference.

The correct answer is the first answer we show here, however the order of all of the answer choices (except for "None of the above") were randomized for each participant. We derived these answer choices from a pilot study we conducted using Amazon Mechanical Turk workers where we solicited free response explanations for what could be causing the apparent paradox and coded them into these eight categories.

After submitting their answers participants were asked to indicate whether they preferred static charts or datamations, and they were encouraged to explain their preference in free response text.

6.5.3 Participants

We recruited participants from Amazon Mechanical Turk and, after excluding those who had taken part in any pilots of this study, randomly assigned 368 participants into the condition that only saw static plots and 340 participants into the condition that saw datamations. Each task was available to Turk workers with an approval rating greater than or equal to 99%, and to



(a) **(b)**

Figure 6.6. Main results of Experiment 1, comparing static plots to plot-based datamations. The left panel shows the fraction of participants who were correctly able to resolve Simpson's paradox in each condition, stating that it was not impossible that the two different plots they were shown could have come from the same dataset. The right panel shows the explanation participants chose to resolve the paradox from a multiple choice list. The top option (in bold) is the correct one. Error bars in both plots show one standard error on the estimated mean.

be eligible to participate the worker had have previously completed at least 100 tasks on AMT. Workers were paid a one-time fee of $1.50.

6.5.4 Results

After collecting responses from all participants, we conducted the analyses specified in our preregistration plan. Accordingly, we removed participants who finished the experiment too quickly (five participants who finished in under 45 seconds for the static condition, and one who finished in under 90 seconds for the datamations condition). This left us with 363 participants who saw only the final static plots and 339 participants who saw datamations. Median completion time for the experiment was 6.9 minutes.

Acknowledging that a reversal is possible. We first looked at participants' ability to correctly identify that a reversal in average earnings between workers with masters and PhD is possible based on whether one conditions on the work setting or not. For each condition we

computed the fraction of participants who correctly stated that "it seems possible" that results could come out this way. As shown in Figure 6.6a, we find that that while only 47% of participants who saw only the final plots answered this question correctly, 61% of participants who saw plot-based datamations did so, a statistically significant ($\chi^2(1, N = 702) = 14.30, p < .001$, one-tailed) and sizeable difference of 14 percentage points (Cohen's $h = 0.29$).

These results support our hypothesis that plot-based datamations can improve people's ability to understand data analyses over static data visualizations alone. Why might this be? While our experiments were not designed to uncover a precise mechanism to explain our results, we can offer a guess. In this case, participants saw two sets of datamations: one for salaries split only by degree type and one split on degree type and work setting. While the final plots showed different directional trends that might be puzzling on their own, *the initial frames of each of these datamations are identical*, and the animations showed smooth transitions from these identical initial frames to the final ones. Although this does not prove to the reader that both datasets are identical or that the reversal is possible, it might nonetheless increase the chances that participants find it plausible that this is the case, leading to a boost in correct answers to this question.

This potential explanation relies on a simple strength of datamations over static plots: they offer strictly more information about the underlying analyses than plots alone. This of course could be done by making a series of plots for different stages of an analysis with text explaining the relationship between these plots, but this can quickly become cumbersome for authors to produce and for readers to consume. Datamations, in contrast, offer an easy and compelling way for this information to be shared with readers.

Identifying the correct explanation. After answering the identification question all participants were told that both figures they saw were in fact from the same underlying dataset and asked to choose one of eight multiple choice explanations for how this could be the case. Figure 6.6b shows the distribution of explanations chosen within each of the two conditions, ranked from

Figure 6.7. For our second experiment we tested whether tabular datamations help participants to understand data analysis pipelines. Participants were randomly assigned to see either two images of tables (shown in this figure) or a animation (illustrated in Figure 6.8).

most to least commonly chosen. The answer on top—that "Most people with a master's degree work in industry, which pays more and drives up the average master's salary in the left plot"—is the correct one. Comparing the top-most blue and red points, we again see a large (21 percentage point, Cohen's $h = 0.42$) and statistically significant ($\chi^2(1, N = 702) = 30.05, p < .001$, one-tailed) improvement between those who saw datamations and those who did not. Even after being told that these were in fact the same dataset, under half (47%) of people who saw only the final plots answered this question correctly, whereas more than two thirds (68%) of people who saw datamations did so.

The distribution over the remaining (incorrect) answers provides some additional insights as to why datamations might help readers understand data analyses. The two most frequent incorrect responses to this question for participants who saw only the final plots were explanations that involved heterogeneity in or representativeness of the data. The next involved the presence of outliers. As discussed with the previous question, one explanation consistent with these results is that participants who saw datamations were shown smooth transitions between identical initial frames and final results, reducing the chances that they thought differences in the datasets were responsible for the reversal.

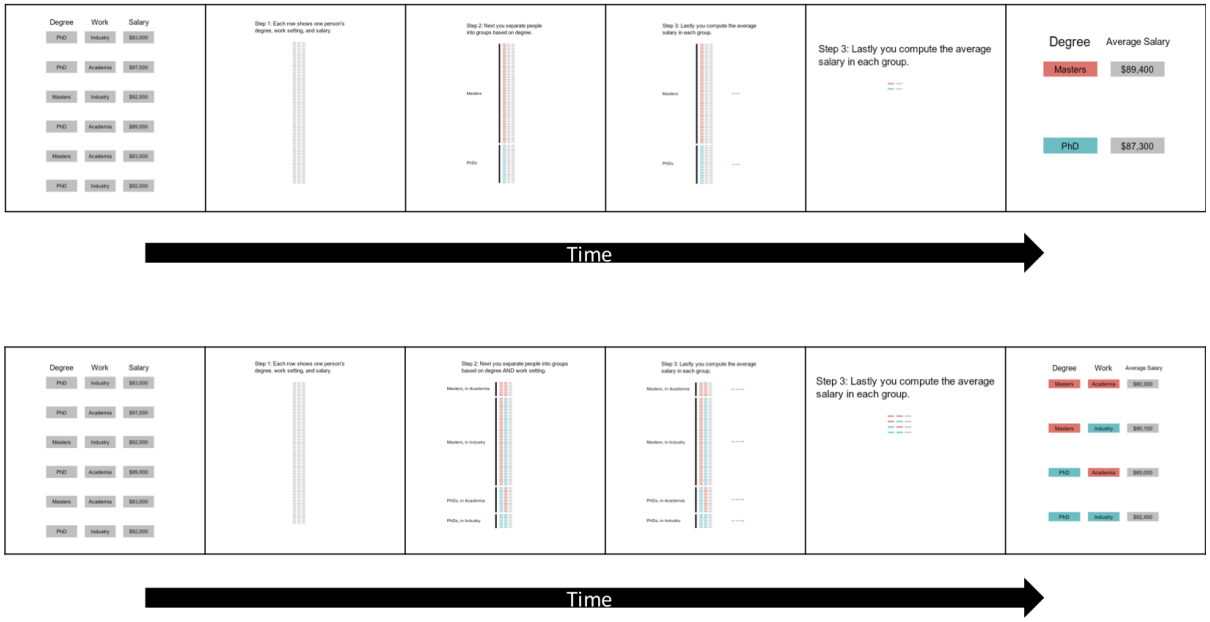


Figure 6.8. This timeline illustrates the tabular datamations in our second experiment with two series of animations that were shown to participants in sequence. In the top datamation every survey respondent is represented by a row, where the first column in that row shows their degree, the second column shows whether they work in academia or industry, and the third column shows their salary. The datamation then zooms out to show the entire table, and the table cells in the first column are colored in depending on which degree an individual has. New, summarized values representing the average salary of each group then appear to the right of the table. The original table then disappears and the table of summarized average values is centered. Finally, the datamation zooms back in to show the summarized table with the average value for each type of degree. The mechanics of the bottom datamation are similar, however both the columns representing degree and work environment are colored in, and groups of cells are distinguished by the four possible combinations of degrees and work settings. Finally the final table shows average salaries for each pair of degree type and work setting. In the last frame we can see that the trend from the first datamation is reversed: PhD holders make more money on average than people with master's degrees in both academia and industry.

6.6 Experiment 2: Table-based Datamations

This experiment mirrored our first experiment, but was designed for settings where people are presented with tables instead of plots. As many of the details are the same, here we simply discuss the changes that were made from the plot- to table-based setting.

6.6.1 Stimuli

The first stimulus for this experiment, seen in Figure 6.7, shows two tables that could be computed at the end of a relatively small data analysis pipeline. The first table shows the average salary for workers according to what kind of degree they have, while the second table shows the average salary for combinations of degree and work setting. These tables and the following datamations were created using the same synthetic salary survey data from the first experiment.

The second stimulus for this experiment, which is illustrated in Figure 6.8, is another looping animated GIF that highlights the transitions and transformations required to go from the raw table of survey data to a summarized table that contains average salaries for each group. Both datamations start with the same raw table of survey data, but then they differentiate depending on how many subgroups are required by the analysis: there are only two subgroups when calculating the average salary across two different types of degrees, however four subgroups are highlighted when calculating average salary across degree types and work settings. Once the average salary values have been calculated the raw values fade away and the summarized values take focus. The two static tables are the end frames for the two datamations: both tables are transformed differently depending on how they are grouped.

6.6.2 Procedure

The experiment for tabular datamations proceeded similarly to the experiment for plot based datamations. Participants received the same information about our study up front and were presented with the same consent form. In the first section of the study they were presented

with the same prompt (*Imagine that you are an analyst working for a think tank...*) and after acknowledging they had read that information they were in either shown static images (featured in Figure 6.7) or a series of tabular datamations, illustrated in Figure 6.8.

As in the first experiment, participants who saw datamations were presented with six datamations in sequence, with each datamation corresponding to a step in the data analysis pipeline. Each of these datamations played on a loop. Participants in this condition then saw the datamations pieced together. Finally both the complete datamations and the final static images (as in Figure 6.4) were shown to the participants in the datamation condition, whereas participants in the other condition saw only the final images.

Participants were asked the same question they were asked in the first experiment (*Does it seem impossible to you that the results would come out this way?*) except the language was changed slightly to discuss tables instead of plots. After submitting their response participants were provided with the correct answer and then asked to select one of eight multiple choice explanations from the first experiment except with different language talking about tables instead of plots where appropriate. The order of answer choices was randomized for each participant. Finally participants were asked to indicate whether they preferred static figures or datamations, and they could provide a written explanation.

6.6.3 Participants

As in the first experiment we recruited participants from Amazon Mechanical Turk and randomly assigned 298 participants into the condition that only saw static tables and 267 participants into the condition that saw tabular datamations. Workers were held to the same standard as they were in the first experiment: they had to have 99% or better approval ratings, experience completing 100 or more tasks, and they could not have participated in our previous pilots. Workers were paid a one-time fee of $1.50.

6.6.4 Results

After collecting responses from all participants, we again removed participants who finished the experiment too quickly (four participants who finished in under 45 seconds for the static condition, and one who finished in under 90 seconds for the datamations condition). This left us with 294 participants who saw only the final static tables and 266 participants who saw datamations. Median completion time for the experiment was 6.9 minutes.

Acknowledging that a reversal is possible. As with the previous experiment, we first looked at people's ability to correctly identify that a reversal is possible. As shown in Figure 6.9a, we see a similar boost in ability to answer this question correctly for those who saw datamations compared to those who saw only final tables, supporting our first hypothesis. While only 52% of participants who saw static tables recognized that the reversal was possible, 60% of participants who saw datamations did, a statistically significant ($\chi^2(1, N = 560) = 2.80, p = .05$, one-tailed) 8 percentage point difference (Cohen's $h = 0.15$).

One explanation consistent with these results is that improvements are due to the increased transparency offered by datamations: participants saw identical starting frames of the animations, which could have increased the chances that they would recognize these results as possibly coming from the same dataset. That said, in these table-based datamations, only a subset of the actual salary values are visible, as it quickly becomes prohibitive to display hundreds of values in plain text such that readers can actually see and process them. As a result, this leaves reasonable ambiguity as to whether the datasets in the initial frames of the datamations are indeed identical. Nonetheless, we see that table-based datamations can help readers recognize that Simpson's reversal is indeed possible.

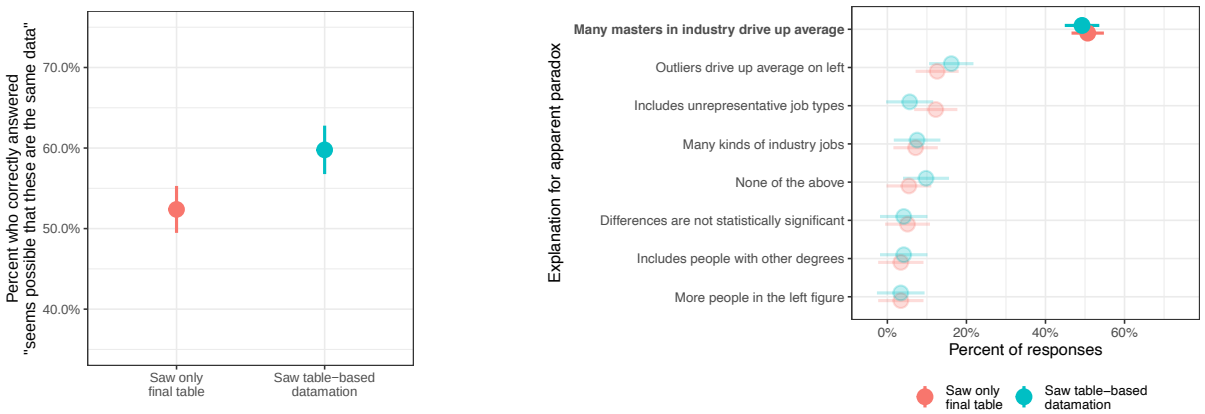
Identifying the correct explanation. After all participants were told that the datasets were in fact identical and the reversal was possible, they again chose one of eight explanations for

why the reversal occurs. Figure 6.9b shows the distribution of responses within each condition, ranked from most to least popular, with the correct answer in bold on the top. Comparing the the top-most blue and red points, we see a lack of support for our second hypothesis, as we find no evidence of a statistically significant difference ($\chi^2(1, N = 560) = 0.06, p = .60$, one-tailed) between the proportion of participants who chose the right answer (51% for those who saw only final tables versus 49% for those who saw datamations).

It is difficult to explain why we see that datamations boost ability to identify that the reversal is possible, but see no evidence for an improvement in ability to explain why the reversal occurs. That said, we can offer two potential explanations. The first is simply that laypeople may be less comfortable or facile with tabular representations of data. The second is based on the same ambiguity mentioned above: while table-based datamations show the relative sizes of different subgroups in the data, they may be unable to clearly convey all of the underlying individual salary information. The cells can simply become too small to insert the salaries as text when several hundred rows are shown, and even if they were visible they would likely be difficult for participants to process holistically. If this is correct, it may be the case that table-based datamations are good for some users (e.g., data scientists) and not others, or may be more helpful for certain types of data analysis operations than others (e.g., joins or data reshaping) not explored here.

6.7 Future preferences and feedback on datamations

As mentioned above, the final page in both experiments asked participants for their future preferences about datamations. Regardless of which condition participants were assigned to, everyone was shown both static figures (either plots or tables, depending on the experiment) and the corresponding datamations, and asked whether they would like to see future figures presented as either a) only static or b) static "accompanied by animations like these". As shown in Figure 6.10, close to two thirds of participants who saw plot-based datamations expressed that



(a) **(b)**

Figure 6.9. Main results of Experiment 2, comparing static tables to table-based datamations. The left panel shows the fraction of participants who were correctly able to resolve Simpson's paradox in each condition, stating that it was not impossible that the two different figures they were shown could have come from the same dataset. The right panel shows the explanation participants chose to resolve the paradox from a multiple choice list. The top option (in bold) is the correct one. Error bars in both plots show one standard error on the estimated mean.

they would like to see these animations in the future, while almost half of participants who saw table-based did.

After making this choice, participants were asked to provide free text feedback about whether these animations were helpful or not. While a good deal of the feedback was generally supportive of the goal behind datamations and in line with insights from our main analyses ("It makes me think more about the data rather than the static charts", "It gives a good visual of how the change takes place when averaging on the chart", "Since it shows exactly how the charts are made it gives a little bit more information and understanding to the reader"), there were also informative critiques that offered insights for how datamations could be improved.

One fairly common piece of feedback was that refined timing of the animations and the ability to control playback would be very helpful ("It is easier for me to read the tables rather than animations like these because I can read at my own pace and not have to feel rushed to obtain the information", "I would have preferred to be able to see each static slide as well, or to have some control over the animation (pause, slow down, rewind, etc.)", "I think they're helpful,

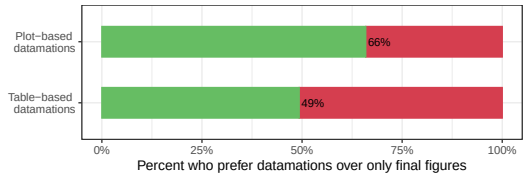


Figure 6.10. People's preferences for seeing datamations in the future for both the plot-based and table-based experiments.

but I would like to be able to pause them so I could read and compare them").

In designing these experiments, we initially considered including controls for playback with the datamations, but ultimately decided against them to keep the experiment as tightly controlled as possible. This provided all participants with the same experience and avoided concerns about endogeneity, but it would be interesting to conduct further studies on how datamations with playback controls perform in terms of both the main hypotheses we studied and people's preferences going forward. Another issue with timing may be that participants were Mechanical Turk workers who are often trying to complete as many microtasks as possible in a given time frame to earn the highest hourly wage possible. As such, it is likely the case that our participants had less tolerance for watching animations play out than, say, a reader who opts into viewing an article containing a datamation because they are inherently interested in the topic it concerns.

Finally, there were several participants who pointed out issues in reading details of the table-based datamations ("I like the animation, but it is very small and hard to read so would prefer the static tables", "I liked the tables and animation, but they were a little small"). While we believe that some of these concerns could be addressed with different design choices in our stimuli, we also recognize that as datasets grow in size, it becomes increasingly difficult to create datamations that contain all of the underlying data, especially in the table-based animations. In these settings we might consider aggregation or sampling to reduce the amount of information shown to readers, and leave this as future work.

6.8 Discussion

In this chapter we have described an opportunity to improve how data analyses are communicated to readers through the use of datamations, which are animations that reveal details of the data analysis pipeline that led to a given plot or table. We presented a modular framework that allows analysts to programmatically translate data analysis code into datamations that attempt to explain that code. We then used this framework to generate datamations and ran experiments comparing them to static plots and tables.

The results of our experiments show that datamations, in most cases, improved people's performance on test items about a subtle data analysis pipeline. Specifically, through two, large-scaled pre-registered experiments we found evidence that both the plot-based and table-based datamations we showed people offered sizable increases on comprehension questions about an instance of Simpson's paradox. Furthermore, many participants reported gaining important insights from these datamations as well as a preference for seeing animations like these in the future. Participants also provided helpful critiques of the specific implementations of datamations we showed them and how they could be improved. Key insights are that the timing of the animations and ability to control their playback are important for an enjoyable and informative user experience. Our experiments also surfaced the insight that table-based datamations can become difficult to read with medium to large datasets due to constraints in displaying many small table cells at once.

There are, of course, many limitations to this work, as we have explored only a small subset of possible research directions related to the task of communicating and explaining data analysis pipelines. Specifically, we looked at how datamations help one population (laypeople being paid to participate in a lab experiment) to perform one task (resolve Simpson's paradox) with a particular visual implementation (self-playing GIFs with the visual transitions we tested). These experiments have helped us answer some questions, but raise many others.

First, even in this particular setting there are several additional questions one might ask.

For instance, did participants truly "understand" the datamations they saw? Unfortunately our experiments cannot answer this question, as comprehension is abstract construct that cannot be directly measured or assessed. One could, however, imagine more detailed studies designed to assess what participants do and don't take away from datamations using techniques such as think-aloud protocols and in-depth interviews. Likewise, what is the precise mechanism responsible for results we saw, and are there alternative, possibly simpler, interventions for communicating data analysis pipelines that could be just as effective as datamations? Again, our experiments cannot answer these questions, as they were designed to compare the status quo (static plots and tables) to what we thought was a viable, practical alternative (animated GIFs) for the purpose of detecting if an effect exists at all. Now that we know this is the case, one could conduct more tightly controlled experiments to isolate effects and identify underlying mechanisms. For instance, is it crucial that datamations are animated, or even visual? Or could a series of static panels depicting a data analysis pipeline or a simple paragraph of descriptive text be just as effective? And what future design choices could make datamations even more effective than those tested here? Some of these questions are notoriously difficult to answer [216], but might be appealing subjects for future research.

Second, beyond the setting we studied here there are several ways this work could be extended by testing datamations on different audiences, for different purposes, and with different implementation details. For instance, it would be interesting to see how datamations could be used to teach students learning data analysis about different concepts, or to see how seasoned data scientists make use of datamations for understanding and debugging code they are writing. One could also evaluate datamations for a host of other types of data analysis pipelines—for instance involving more complex operations like data joins, reshaping, modeling, statistical estimation, etc.—and investigate for which settings they do (and don't) provide value over the status quo. Likewise, there are many opportunities to add to, experiment with, and optimize the visual elements and transitions used in datamations, akin to the research done in the HCI community to improve the details of static visualizations. Finally, there is the opportunity to

develop software packages that make it easier for developers to explain their data analysis pipelines to their audiences. We have created one such implementation based on the framework described here, but there are many ways it can be extended in the future. It is currently designed to work for R's tidyverse, but we hope to see it extended to other programming languages and packages. We view this as the first of many steps towards developing more tools to explain data analysis pipelines and their results to students, analysts, and their audiences.

6.9 Acknowledgements

Chapter 6, in part, is a reprint of the material as it appears in the proceedings of the 2021 CHI Conference on Human Factors in Computing Systems as as Datamations: Animated Explanations of Data Analysis Pipelines. Xiaoying Pu, Sean Kross, Jake Hofman, Daniel Goldstein. 2021. The dissertation author was one of the primary investigators and author of this paper.

Chapter 7

Discussion and Conclusions

7.1 Reflections on the Contributions of this Dissertation

This dissertation offers several contributions, many of which support my thesis, in addition to multiple contributions that add to the understanding of data science at large:

Chapter 3 presents findings from 20 interviews with professional data scientists who are also data science instructors. The design and subject matter of this study was influenced by two fundamental questions: "What is data science?" and "Who are data scientists?" Chapter 3 offers one perspective on those fundamental questions by investigating social and technical challenges faced by data science practitioner-instructors. I found that the participants were trying to create both computing and learning environments that approximate the conditions of real-world data science work. This held true across the varied settings where these practitioner-instructors teach. Chapter 3 found that people seeking data science education are motivated by ends, like understanding data they possess, or wanting to know answers to scientific questions, rather than means, like how data structures work or how to write code with the best programming style practices. Additionally, Chapter 3 finds that teaching authentic programming workflows is a priority for practitioner-instructors. These workflows heavily borrow tools from software engineering, however they have to be adapted to serve the needs of data scientists. Software engineering tools are not built for data scientists, therefore making these adaptations introduces uncertainty and complexity into data science work. Finally, this chapter finds that novice data

scientists must learn to navigate the entire technology stack from high level machine learning libraries to kernel-level virtualization infrastructure. These findings highlight many of the seams that exist in data science work and point to opportunities for tools that center the data scientist's programming experience.

Chapter 4 explores how a group of academic data scientists with little financial or organizational support were able to build an educational program to reach adult learners from historically underrepresented groups in computing. They could only develop this program because of their deep expertise in data science oriented programming. This chapter finds that data scientists can adapt their tools for purposes they were never intended for: tools originally designed for creating reproducible scientific reports were repurposed to make lecture notes, slides, and videos, while frameworks for testing data science code were repurposed for deploying and checking the correctness of course infrastructure. Chapter 4 offers a case study of how data scientists repurpose their existing tools and build their own tools to meet their needs. Therefore Chapter 4 provides insight into what data scientists believe is missing from their experience of doing data science programming.

Chapter 5 characterizes the entire cycle of work that data scientists traverse in their consulting relationships with clients. This chapter zooms out from the core of the data science programming cycle to show the nuanced and essential steps that have to be taken for data science work to be successful. This chapter finds there is an outer loop of data science collaboration that contains other nested loops, including the back-and-forth between problem framing and bridging knowledge gaps, and how analysis results drive the next analysis. Additionally this chapter illuminates the relationships that data scientists have to manage as a part of executing an analysis, and it describes the emotional work that data scientists have to engage with. Chapter 5 shines a light on the decision points that happen during a data analysis process, and it highlights the absence of tools for data scientists that would help them integrate these decisions or the interests of their collaborators into their programming work. The final finding of this chapter is the articulation of an approach to collaborative data science work that shares characteristics

with design work: data scientists must reconcile competing interests, limited resources, and mathematical and ethic rigor while still fulfilling their client's needs.

Chapter 6 contributes two new representations of data science workflows: Datamations, a software library for creating animations from data analysis code, and Tidy Data Tutor, a web application that visualizes and annotates every step in a data analysis pipeline. This chapter is an exploration of what data scientists should be able to get "for free" (without doing any work other than their typical programming work). These tools bring transparency to abstractions in a data scientist's work by helping them to see data analysis pipelines in new ways. This can aid a data scientist who is trying to understand code they are encountering for the first time, it could help data scientists find bugs in code they believe could be improved, and it can help novice data scientists understand how different operations affect data as it moves through the pipeline. Both of the systems in Chapter 6 are publicly available with daily novice and professional users from all over the world.

The following sections expand upon the contributions of the dissertation with a discussion of future work and potential implications.

7.2 DataFrame-Oriented Programming

Chapter 3 and Chapter 6 highlight the importance of data frames in data science programming workflows. Data frames are built into R, and they are the backbone of the Pandas framework in Python. One of the most common data science tasks is importing, cleaning, and arranging a single data frame. Sometimes this one data frame is augmented by a few other data frames that are incorporated via joins with corresponding columns.

Pipe operators, similar to those found in command line languages like Bash, have made reading and writing the code for data frame programming workflows significantly easier, as seen in Figure 7.1. The mechanism for the pipe operator is straightforward: the value on the left-hand-side of the pipe becomes the first argument of the function on the right-hand-side of

```r
mtcars |>
  select(cyl, mpg, hp) |>
  filter(hp > 90) |>
  group_by(cyl) |>
  summarize(Mean_MPG = mean(mpg))

mtcars_cmh <- select(mtcars, cyl, mpg, hp)
mtcars_gt90 <- filter(mtcars_cmh, hp > 90)
mtcars_grouped <- group_by(mtcars_gt90, cyl)
summarize(mtcars_grouped, Mean_MPG = mean(mpg))
```

Figure 7.1. Different blocks of R code that produce the same results: lines 1 through 4 use the pipe operator (`|>`) at the end of each line. Line 1 starts with the variable name of a data frame (`mtcars`) and then lines 2 through 5 begin with the name of a function that applies a transformation to the data frame. Each of these functions returns the transformed data frame which is then piped to the next function. This allows code writers and readers to see data frame transformations listed as a series of steps. Compare this with lines 7 through 10, which produces the same result, however several variables are created and the code is harder to read.

the pipe. This operator creates multiple affordances: 1. It allows multiple transformations of a data frame to be chained together in a series. 2. It eases debugging since the state of the data as it moves through the pipeline can be evaluated at any step. 3. The pipeline and can be easily reused and recomposed, especially if the columns of the input data frame are stable. 4. Exploratory programming within the pipeline is easy since a programmer can see how changing a single step in the pipeline can cause cascading effects. 5. Big chunks of the pipeline can be commented out, which makes it easy to compare different versions of the pipeline.

The utility of these affordances led to the pipe operator being added recently as a builtin operator in R, and this pipeline style of programming has become ubiquitous in the Python Pandas framework via the similar "dot" operator. The success of the pipe operator and the data analysis pipelines it affords highlights the possibilities of an up-and-coming programming paradigm: **dataframe-oriented programming**. In dataframe-oriented programming the central goal of the programmer is to build up one or few data frames. This data frame can be valuable on its own, or the purpose of creating the data frame can be to pass it along to some other

system, like a modeling framework or a visualization library. Dataframe-oriented programming is a focused paradigm that borrows concepts from object-oriented programming, functional programming, and dataflow programming. However, unlike object-oriented programming the programmer is usually only concerned with a few similarly structured objects (mostly data frames and functions) that they can reason about in their working memory. And unlike dataflow programming, dataframe-oriented programing is done via manipulating text-based code as opposed to direct manipulation of objects on a screen.

Despite the fact that this kind of programming has been taking place for years, the innovation of the pipe operator for dataframe-oriented programming has only become mainstream recently. If researchers and practitioners invested more time and thought, we might discover what other valuable programming innovations, affordances, and operators could benefit people who do dataframe-oriented programming. My contribution to this idea has been new ways of visually representing the data pipelines that are built from dataframe-oriented code (Datamations and Tidy Data Tutor). In the future I hope to work on, and I hope to inspire others to imagine new programming languages, development environments, and data interchange formats that center the experience of the dataframe-oriented programmer.

7.3 The Future of Data Science Education

7.3.1 Integrating Statistics and Data Science

I taught data science mostly to adults and graduate students before starting work on this dissertation, during graduate school I taught data science to undergraduate students for the first time, and I expect to continue teaching data science after graduate school. In different settings over the years I have been encouraged to teach data science without mentioning concepts from the field of statistics. These courses are usually focused on teaching programming just above an introductory level, where students understand the basic semantics of a language, but they are trying to get more exposure to libraries that are specifically used for data science.

As discussed in Chapter 3, there are dozens of important data science packages and it has been very rewarding to have the opportunity to guide students through learning how to use that kind of software. However, teaching students to import and clean data is not very motivating where there are no goals beyond the data and the code meeting a certain specification. Data science is ultimately about organizing knowledge, sense-making, and attempting to find insight about the real world. Statistics is one "way of knowing" that is befitting of the goals of data science: when data are gathered together one can use statistics to ask questions about what the data represent.

Finding answers to questions in data seems to be an incredible motivation to learn data science, but not being able to talk about measures of central tendency, regression, or basic modeling hinders teaching and learning efforts. However, I also recognize that teaching both statistics and data science simultaneously is a daunting challenge. Data science and statistics are like Orpheus and Eurydice: star-crossed lovers, but my hope is that future data science learning environments [159] will help to scaffold an easier integration of data science and statistics instruction.

7.3.2 Motivating Data Science Education

Just as having tools like statistics at students' disposal helps to motivate the possibilities of data science, relevant data and programming tasks also help to motivate students. Consider the stereotypical introduction to computer science course: after learning the basic syntax, students often work on projects like building small video games, implementing a system for keeping track of employees at a company, or coding a building block for another computer system like a network. The problem with these prompts is that they are all only interesting to students if they are interested in building computer systems.

Consider the alternative, an introduction to a data science course: after learning the basic syntax, students can choose to work with data from a cancer research study, climate data collected from all over the world, or data obtained via the Freedom of Information Act about

automated prison sentencing. Students who are interested in subjects beyond learning to build computer systems have a wide range of topics they can explore, and they still get an opportunity to learn the full stack of data science technologies.

Chapter 3 discusses how important it is for instructors to be able to meet the diverse needs and educational backgrounds of data science students. This thought experiment describes an ideal situation, however Chapter 3 also discusses the challenges that instructors face finding datasets that are relevant to their students. There are still ample opportunities for helping instructors find and create their own datasets. While those innovations are being developed, it is also worth exploring the possibility that data science education could motivate the most students to learn computer programming. This raises the question: Does the future of computing education start with data science?

7.3.3 Teaching Data Science as a Design Discipline

Mastery of the technical skills required to do data science work is of course important, however data science is rarely done alone in a vacuum. This dissertation stands alongside a deluge of research that shows how tightly integrated data scientists are into other teams, and how closely data scientists collaborate with their colleagues.

Taking this into consideration, there is a surprising gap in most undergraduate and masters-level data science programs, since they rarely teach novice data scientists about the most common type of interactions they can expect to have with their colleagues. There are valuable "capstone" style courses that allow data science students to work on real-world problems, and these experiences can seriously benefit a student's understanding of how to translate classroom skills into the professional world. However courses that help students think through how their relationships with their colleagues are integral parts of the systems they work inside of are not as widespread as they are valuable.

One way to remedy this situation is to bring lessons from the field of design into data science classrooms. Data scientists frequently engage in the same kinds of negotiation and

decision making processes as designers, and Chapter 5 shows how many of these experiences are shared between data scientists. Chapter 5 illustrates one model for how to think about the interactions between data scientists and their colleagues, and how these interactions reflect the experience of designers. My hypothesis is that teaching students about how to think about their interactions with colleagues through the lens of design work can help them act deliberately to achieve day-to-day success in their careers.

7.4 New Representations Enabling Better Data Science and Design

Design principles do not just need to be taken up in data science classrooms, but they also need to be integrated into the workflows of professional data scientists. Designers of products, buildings, and websites have the luxury of a shared understanding with their clients about the materiality of their goals: a product is manufactured, a building is erected, or a website is launched. In each of these examples, clients themselves are usually not experts in any of these fields of design, but they still have an intuitive understanding of the goals and outputs of the design process.

Unfortunately, data scientists rarely have the same luxury. The goals and outputs of the data science process are often much more abstract: statistical models that must be kept up to date with changing trends, reports full of measurements that represent the health of a business, or conclusions about whether a potential medical intervention is helpful or harmful. Clients often do not have an intuitive understanding of any of the steps in the process including how the data is collected, why a particular data analysis plan is appropriate, how that plan is translated into code, how the code itself works, or what action should be taken given a computed output. Consider the example of evaluating the data scientific conclusion of the analysis of a medical intervention, clients may need to know to ask: 1. Are the differences between two treatments significant? 2. Is the amount of improvement between the two treatments clinically relevant? 3. Can the treatment

be scaled in a way that is sustainable? The answer to each of these questions can theoretically be answered with data, but each question requires the integration of many kinds of expertise that go beyond the knowledge of one individual.

Chapter 6 in this dissertation tries to assuage this situation by offering new visual representations of data analysis pipelines that are implemented to help programmers understand their own data science code. Ultimately, I believe that more kinds of representations will be needed to make data science work more concrete, which will help integrate design principles into data science work. One can imagine explorable visualizations of how a data collection process was performed, or how medical treatments might affect different patients. These representations could be based on code that written to organize the data collection, or based on the resulting data. I am hopeful that a rich set of tools for transforming the artifacts that data scientists have already worked hard to build, like code they have written and data they have curated, is in our near future. These new representations have the potential to bring more people into a deeper understanding of foundational scientific investigations.

7.5 Epilogue

Each of these chapters tries to shine a light on a problem that if solved, I believe would have a large positive effect on people doing data science work, and on the quality of scientific work overall. In this sense each chapter is a prototype: a set of ideas, a little universe where I hope you can see why these problems are important and how solutions could be reached. The most exciting thing to me about the future of data science is not the data necessarily, but the science that data scientific methods will enable. The coming developer experience for data scientists will in some ways look like the software development experience: there will be more coherent, and if we are lucky, intelligent, computing environments that are aware of the specific technical needs of data scientists. But I am also hopeful that those same environments will go beyond scaffolding programming work to providing a foundation for doing scientific work,

where we can build on the discoveries of the past as easily as we build on the software of the past. This is the environment where many of our great problems are going to be solved, and it is our responsibility to build the tools and the communities that will provide the greatest discoveries we can imagine about our world and about each other.

Bibliography

[1] Amazon polly: Turn text into lifelike speech using deep learning. https://aws.amazon.com/polly/. Accessed: 2019-05-01.

[2] Black girls code: Imagine. build. create. http://www.blackgirlscode.com/. Accessed: 2019-05-01.

[3] Code2040: Join the largest racial equity community in tech. http://www.code2040.org/. Accessed: 2019-05-01.

[4] Girls who code: Join the girls who code movement! https://girlswhocode.com/. Accessed: 2019-05-01.

[5] Historic East Baltimore Community Action Coalition (HEBCAC): Thriving Baltimore Communities. https://hebcac.org/. Accessed: 2019-05-01.

[6] Leanpub: publish early, publish often. https://leanpub.com/. Accessed: 2019-05-01.

[7] R Markdown: Analyze. share. reproduce. https://rmarkdown.rstudio.com/. Accessed: 2019-05-01.

[8] Welcome to RStudio Cloud: Do, share, teach and learn data science with R. https://rstudio.cloud/. Accessed: 2019-05-01.

[9] The 50 most popular moocs of all time. http://www.onlinecoursereport.com/the-50-most-popular-moocs-of-all-time/, 2017.

[10] The complete list of data science bootcamps & fellowships. http://www.skilledup.com/articles/list-data-science-bootcamps, 2017.

[11] Ffmpeg: A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/, 2017.

[12] Project jupyter. http://jupyter.org/, 2017.

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

[14] Datacamp: Learn r, python & data science online. https://www.datacamp.com/, 2018. Accessed: 2018-09-20.

[15] Dataquest: Learn data science with python and r projects. https://www.dataquest.io/, 2018. Accessed: 2018-09-20.

[16] ggplot2 is a system for declaratively creating graphics, based on the grammar of graphics. https://ggplot2.tidyverse.org/, 2018. Accessed: 2018-09-20.

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

[18] R markdown: Analyze. share. reproduce. https://rmarkdown.rstudio.com/, 2018. Accessed: 2018-09-20.

[19] Rstudio for the enterprise. https://www.rstudio.com/products/rstudio-server-pro/, 2018. Accessed: 2018-09-20.

[20] Tpi: Teaching perspectives inventory. http://www.teachingperspectives.com/tpi/, 2018. Accessed: 2018-09-20.

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

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

[23] Sara Alspaugh, Nava Zokaei, Andrea Liu, Cindy Jin, and Marti A. Hearst. Futzing and moseying: Interviews with professional data analysts on exploration practices. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):22–31, January 2019.

[24] Richard B. Anderson, Laura Marie Leventhal, Don C. Zhang, Jr Daniel Fasko, Zachariah Basehore, Christopher Gamsby, Jared Branch, and Timothy Patrick. Belief bias and representation in assessing the Bayesian rationality of others. *Judgment and Decision Making*, 14(1):1–10, 2019.

[25] Ruth E. Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. A data programming cs1 course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 150–155, New York, NY, USA, 2015. ACM.

[26] Craig Anslow, John Brosz, Frank Maurer, and Mike Boyes. 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, pages 615–620, New York, NY, USA, 2016. ACM.

[27] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. GraphDiaries: Animated Transitions andTemporal Navigation for Dynamic Networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, May 2014.

[28] Stefan Milton Bache and Hadley Wickham. magrittr: A forward-pipe operator for r. manual, RStudio, 2014.

[29] Emily Badger, Claire Ann Miller, Adam Pearce, and Kevin Quealy. Income Mobility Charts for Girls, Asian-Americans and Other Groups. Or Make Your Own. *New York Times*, -(-):–, March 2018.

[30] A. Balalaie, A. Heydarnoori, and P. Jamshidi. Microservices architecture enables devops: Migration to a cloud-native architecture. *IEEE Software*, 33(3):42–52, May 2016.

[31] Austin Cory Bart, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 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, pages 1029–1034, New York, NY, USA, 2018. ACM.

[32] Austin Cory Bart, Ryan Whitcomb, Dennis Kafura, Clifford A. Shaffer, and Eli Tilevich. 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, pages 57–62, New York, NY, USA, 2017. ACM.

[33] Klara Benda, Amy Bruckman, and Mark Guzdial. When life and learning do not fit: Challenges of workload and communication in introductory computer science online. *Trans. Comput. Educ.*, 12(4):15:1–15:38, November 2012.

[34] Enrico Bertini, Michael Correll, and Steven Franconeri. Why Shouldn't All Charts Be Scatter Plots? Beyond Precision-Driven Visualizations. *arXiv:2008.11310 [cs]*, -(-):–, August 2020. arXiv: 2008.11310.

[35] Nick Bertoni and Scott Keeter. 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/, Mar 2018. Accessed: 2018-09-20.

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

[37] Johan Kaj Blomkvist, Johan Persson, and Johan Åberg. *Communication through Boundary Objects in Distributed Agile Teams*, page 1875–1884. Association for Computing Machinery, New York, NY, USA, 2015.

[38] Bootstrap. Data science curriculum (spring 2018 edition). http://www.bootstrapworld.org/materials/spring2018/courses/data-science/english/, 2018. Accessed: 2018-09-01.

[39] Christian Bors, Theresia Gschwandtner, and Silvia Miksch. Capturing and Visualizing Provenance From Data Wrangling. *IEEE Computer Graphics and Applications*, 39(6):61–75, 2019.

[40] Indranil Bose and Radha K. Mahapatra. Business data mining — a machine learning perspective. *Information & Management*, 39(3):211 – 225, 2001.

[41] Lori Breslow, David E Pritchard, Jennifer DeBoer, Glenda S Stump, Andrew D Ho, and Daniel T Seaton. Studying learning in the worldwide classroom: Research into edX's first MOOC. *Research & Practice in Assessment*, 8, 2013.

[42] Bo Brinkman and Amanda Diekman. Applying the communal goal congruity perspective to enhance diversity and inclusion in undergraduate computing degrees. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 102–107, New York, NY, USA, 2016. ACM.

[43] Irwin D. J. Bross. The role of the statistician: Scientist or shoe clerk. *The American Statistician*, 28(4):126–127, 1974.

[44] Robert J. Brunner and Edward J. Kim. Teaching data science. *Procedia Comput. Sci.*, 80(C):1947–1956, June 2016.

[45] Margaret M. Burnett and Brad A. Myers. Future of end-user software engineering: Beyond the silos. In *Proceedings of the on Future of Software Engineering*, FOSE 2014, pages 201–211, New York, NY, USA, 2014. ACM.

[46] Ricardo Caceffo, Steve Wolfman, Kellogg S. Booth, and Rodolfo Azevedo. Developing a computer science concept inventory for introductory programming. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 364–369, New York, NY, USA, 2016. ACM.

[47] Amira Chalbi, Jacob Ritchie, Deokgun Park, Jungu Choi, Nicolas Roussel, Niklas Elmqvist, and Fanny Chevalier. Common Fate for Animated Transitions in Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1–1, 2019.

[48] Fanny Chevalier, Pierre Dragicevic, and Steven Franconeri. The Not-so-Staggering Effect of Staggered Animated Transitions on Visual Tracking. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2241–2250, December 2014.

[49] Fanny Chevalier, Nathalie Henry Riche, Catherine Plaisant, Amira Chalbi, and Christophe Hurter. Animations 25 Years Later: New Roles and Opportunities. In *Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '16*, pages 280–287, Bari, Italy, 2016. ACM Press.

[50] Parmit K. Chilana, Rishabh Singh, and Philip J. Guo. Understanding conversational programmers: A perspective from the software industry. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1462–1472, New York, NY, USA, 2016. ACM.

[51] Joohee Choi and Yla Tausczik. Characteristics of collaboration in the emerging practice of open data analysis. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17, page 835–846, New York, NY, USA, 2017. Association for Computing Machinery.

[52] Herbert H. Clark and Susan E. Brennan. Grounding in communication. In L.B. Resnick, J.M. Levine, and S.D. Teasley, editors, *Perspectives on Socially Shared Cognition*, pages 127–149. American Psychological Association, 1991.

[53] William S. Cleveland and Robert McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. *Journal of the American Statistical Association*, 79(387):531–554, September 1984.

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

[55] Nigel Cross. *Design Thinking: Understanding How Designers Think and Work*. Bloomsbury, 2011.

[56] Nigel Cross. *Expertise in Professional Design*, page 372–388. Cambridge Handbooks in Psychology. Cambridge University Press, 2 edition, 2018.

[57] Zachary T Cutler, Kiran Gadhave, and Alexander Lex. Trrack: A Library for Provenance Tracking in Web-Based Visualizations. In *OSF Preprints*, page 5, -, July 2020. Open Science Framework.

[58] Sarah Dahlby Albright, Titus H. Klinge, and Samuel A. Rebelsky. A functional approach to data science in cs1. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, pages 1035–1040, New York, NY, USA, 2018. ACM.

[59] Sayamindu Dasgupta and Benjamin Mako Hill. Scratch community blocks: Supporting children as data scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3620–3631, New York, NY, USA, 2017. ACM.

[60] James Densmore. There are two types of data scientists – and two types of problems to solve. https://medium.com/@jamesdensmore/there-are-two-types-of-data-scientists-and-two-types-of-problems-to-solve-a149a0148e64, 2017. Accessed: 2020-10-10.

[61] Conor Dewey. An ode to the type a data scientist. Towards Data Science – https://towardsdatascience.com/ode-to-the-type-a-data-scientist-78d11456019, 2018. Accessed: 2020-10-10.

[62] Betsy DiSalvo, Mark Guzdial, Amy Bruckman, and Tom McKlin. Saving face while geeking out: Video game testing as a justification for learning computer science. *Journal of the Learning Sciences*, 23(3):272–315, 2014.

[63] Betsy DiSalvo, Mark Guzdial, Charles Meadows, Ken Perry, Tom McKlin, and Amy Bruckman. Workifying games: Successfully engaging african american gamers with computer science. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 317–322, New York, NY, USA, 2013. ACM.

[64] Betsy DiSalvo, Sarita Yardi, Mark Guzdial, Tom McKlin, Charles Meadows, Kenneth Perry, and Amy Bruckman. African american men constructing computing identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2967–2970, New York, NY, USA, 2011. ACM.

[65] David Donoho. 50 years of data science. *Journal of Computational and Graphical Statistics*, 26(4):745–766, 2017.

[66] Brian Dorn and Mark Guzdial. Graphic designers who program as informal computer science learners. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 127–134, New York, NY, USA, 2006. ACM.

[67] Brian Dorn and Mark Guzdial. Discovering computing: Perspectives of web designers. In *Proceedings of the Sixth International Workshop on Computing Education Research*, ICER '10, pages 23–30, New York, NY, USA, 2010. ACM.

[68] Brian Dorn and Mark Guzdial. 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, pages 703–712, New York, NY, USA, 2010. ACM.

[69] Paul Dourish. Process descriptions as organisational accounting devices: The dual use of workflow technologies. In *Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work*, GROUP '01, page 52–60, New York, NY, USA, 2001. Association for Computing Machinery.

[70] Pierre Dragicevic, Anastasia Bezerianos, Waqas Javed, Niklas Elmqvist, and Jean-Daniel Fekete. Temporal distortion for animated transitions. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, page 2009, Vancouver, BC, Canada, 2011. ACM Press.

[71] Matt Dray. Fix leaky pipes in R, April 2019.

[72] Ian Drosos, Titus Barik, Philip J. Guo, Robert DeLine, and Sumit Gulwani. Wrex: A unified programming-by-example interaction for synthesizing readable code for data scientists. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[73] Steven M. Drucker and Roland Fernandez. A Unifying Framework for Animated and Interactive Unit Visualizations. Technical report, Microsoft, 2015.

[74] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. Devops. *IEEE Software*, 33(3):94–100, May 2016.

[75] Tommy Ellkvist, David Koop, Erik W. Anderson, Juliana Freire, and Cláudio Silva. Using provenance to support real-time collaborative design of workflows. In *International Provenance and Annotation Workshop*, pages 266–279, Berlin, Heidelberg, 2008. Springer.

[76] Barbara Ericson, Shelly Engelman, Tom McKlin, and Ja'Quan Taylor. Project rise up 4 cs: Increasing the number of black students who pass advanced placement cs a. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 439–444, New York, NY, USA, 2014. ACM.

[77] Virginia Eubanks. *Automating inequality: How high-tech tools profile, police, and punish the poor*. St. Martin's Press, 2018.

[78] Fabian Fagerholm and Jürgen Münch. Developer experience: Concept and definition. In *2012 International Conference on Software and System Process (ICSSP)*, pages 73–77, 2012.

[79] Mohammed F. Farghally, Kyu Han Koh, Jeremy V. Ernst, and Clifford A. Shaffer. Towards a concept inventory for algorithm analysis topics. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 207–212, New York, NY, USA, 2017. ACM.

[80] Michael Fernandes, Logan Walls, Sean Munson, Jessica Hullman, and Matthew Kay. Uncertainty Displays Using Quantile Dotplots or CDFs Improve Transit Decision-Making. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 144:1–144:12, New York, NY, USA, 2018. ACM. event-place: Montreal QC, Canada.

[81] D. J. Finney. The questioning statistician. *Statistics in Medicine*, 1(1):5–13, 1982.

[82] Danyel Fisher. Animation for visualization: Opportunities and drawbacks. In *Beautiful Visualization*, page 24. O'Reilly Media, E-book, 2010.

[83] Denae Ford and Chris Parnin. Exploring causes of frustration for software developers. In *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE '15, page 115–116. IEEE Press, 2015.

[84] Merrick Furst, Charles Isbell, and Mark Guzdial. Threads™: How to restructure a computer science curriculum for a flat world. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '07, pages 420–424, New York, NY, USA, 2007. ACM.

[85] Mirta Galesic, Rocio Garcia-Retamero, and Gerd Gigerenzer. Using Icon Arrays to Communicate Medical Risks: Overcoming Low Numeracy. *Health Psychology*, 28(2):210–216, 2009. ISBN: 0278-6133.

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

[87] T. Ge, Y. Zhao, B. Lee, D. Ren, B. Chen, and Y. Wang. Canis: A High-Level Language for Data-Driven Chart Animations. *Computer Graphics Forum*, 39(3):607–617, June 2020.

[88] Gerd Gigerenzer and Ulrich Hoffrage. How to improve Bayesian reasoning without instruction: Frequency formats. *Psychological Review*, 102(4):684–704, 1995. ISBN: 0033-295X\r1939-1471.

[89] Cristina Gallego Gómez and Consuelo Puchades Ruiz. The inclusion of methodologies user experience in the consulting industry: An approach to the experience of capgemini. In *Proceedings of the XVII International Conference on Human Computer Interaction*, Interacción '16, New York, NY, USA, 2016. Association for Computing Machinery.

[90] Alicia A Grandey. Emotion regulation in the workplace: a new way to conceptualize emotional labour. *Journal of Occupational Health Psychology*, 5:95–100, 2000.

[91] Philip J. Guo. *Software Tools to Facilitate Research Programming*. PhD thesis, Stanford University, May 2012.

[92] Philip J. Guo. Online python tutor: Embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. Association for Computing Machinery.

[93] Philip J. Guo. Older adults learning computer programming: Motivations, frustrations, and design opportunities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 7070–7083, New York, NY, USA, 2017. ACM.

[94] Philip J. Guo, Sean Kandel, Joseph M. Hellerstein, and Jeffrey Heer. Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 65–74, New York, NY, USA, 2011. Association for Computing Machinery.

[95] Yue Guo, Carsten Binnig, and Tim Kraska. What you see is not what you get! Detecting Simpson's Paradoxes during Data Exploration. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, HILDA'17, pages 1–5, New York, NY, USA, May 2017. Association for Computing Machinery.

[96] Mark Guzdial. Exploring hypotheses about media computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 19–26, New York, NY, USA, 2013. ACM.

[97] Mark Guzdial. Limitations of moocs for computing education- addressing our needs: Moocs and technology to advance learning and learning research (ubiquity symposium). *Ubiquity*, 2014(July):1:1–1:9, July 2014.

[98] Mark Guzdial. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8(6):1–165, 2015.

[99] Olaf A. Hall-Holt and Kevin R. Sanft. Statistics-infused introduction to computer science. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, pages 138–143, New York, NY, USA, 2015. ACM.

[100] John D. Hansen and Justin Reich. Democratizing education? examining access and usage patterns in massive open online courses. *Science*, 350(6265):1245–1248, 2015.

[101] 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. Data science in statistics curricula: Preparing students to "think with data". *The American Statistician*, 69(4):343–353, 2015.

[102] Sarah T. Hawley, Brian Zikmund-Fisher, Peter Ubel, Aleksandra Jancovic, Todd Lucas, and Angela Fagerlin. The impact of the format of graphical presentation on health-related knowledge and treatment choices. *Patient Education and Counseling*, 73(3):448–455, 2008. ISBN: 0738-3991 (Print)\r0738-3991 (Linking).

[103] Bob Hayes. Who does the machine learning and data science work? customer think – https://customerthink.com/who-does-the-machine-learning-and-data-science-work/, 2020. Accessed: 2021-01-10.

[104] Andrew Head, Elena Glassman, Gustavo Soares, Ryo Suzuki, Lucas Figueredo, Loris D'Antoni, and Björn Hartmann. 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, pages 89–98, New York, NY, USA, 2017. ACM.

[105] J. Heer and G. Robertson. Animated Transitions in Statistical Data Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, November 2007.

[106] Jeffrey Heer. An Illustrated Guide to Arquero Verbs, September 2020.

[107] Jeffrey Heer and Michael Bostock. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, page 203, Atlanta, Georgia, USA, 2010. ACM Press.

[108] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE transactions on visualization and computer graphics*, 14(6):1189–1196, 2008. Publisher: IEEE.

[109] Daniel Hellmann, Carleen Maitland, and Andrea Tapia. Collaborative analytics and brokering in digital humanitarian response. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, CSCW '16, page 1284–1294, New York, NY, USA, 2016. Association for Computing Machinery.

[110] Stephanie C. Hicks and Rafael A. Irizarry. A guide to teaching data science. *The American Statistician*, 0(ja):00–00, 2017.

[111] Stephanie C. Hicks and Roger D. Peng. Elements and principles for characterizing variation between data analyses, 2019.

[112] C. Hill, R. Bellamy, T. Erickson, and M. Burnett. Trials and tribulations of developers of intelligent systems: A field study. In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 162–170, 2016.

[113] Suz Hinton. 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, 2017.

[114] Arlie Russell Hochschild. *The Managed Heart: Commercialization of Human Feeling*. University of California Press, 1 edition, 2012.

[115] Youyang Hou and Dakuo Wang. Hacking with npos: Collaborative analytics and broker roles in civic data hackathons. *Proc. ACM Hum.-Comput. Interact.*, 1(CSCW), December 2017.

[116] J. Hullman, S. Drucker, N. Henry Riche, B. Lee, D. Fisher, and E. Adar. A Deeper Understanding of Sequence in Narrative Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2406–2415, December 2013.

[117] Jessica Hullman, Paul Resnick, and Eytan Adar. Hypothetical Outcome Plots Outperform Error Bars and Violin Plots for Inferences about Reliability of Variable Ordering. *PLOS ONE*, 10(11):e0142444, November 2015.

[118] Christopher D. Hundhausen, Sarah A. Douglas, and John T. Stasko. A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, June 2002.

[119] Daniela Huppenkothen, Anthony Arendt, David W. Hogg, Karthik Ram, Jacob T. VanderPlas, and Ariel Rokem. Hack weeks as a model for data science education and collaboration. *Proceedings of the National Academy of Sciences*, 2018.

[120] Marina Jirotka, Charlotte P. Lee, and Gary M. Olson. Supporting scientific collaboration: Methods, tools and concepts. *Comput. Supported Coop. Work*, 22(4–6):667–715, August 2013.

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

[122] Nazanin Kadivar, Victor Chen, Dustin Dunsmuir, Eric Lee, Cheryl Qian, John Dill, Christopher Shaw, and Robert Woodbury. Capturing and supporting the analysis process. In *2009 IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138, Atlantic City, New Jersey, 2009. IEEE.

[123] A. Kale, F. Nguyen, M. Kay, and J. Hullman. Hypothetical Outcome Plots Help Untrained Observers Judge Trends in Ambiguous Data. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):892–902, January 2019.

[124] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372, New York, NY, USA, 2011. ACM.

[125] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, December 2012.

[126] Matthew Kay, Tara Kola, Jessica R Hullman, and Sean A Munson. When (ish) is My Bus?: User-centered Visualizations of Uncertainty in Everyday, Mobile Predictive Systems. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*, pages 5092–5103, New York, NY, USA, 2016. Association for Computing Machinery.

[127] Caitlin Kelleher, Randy Pausch, and Sara Kiesler. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1455–1464, New York, NY, USA, 2007. ACM.

[128] Mary Beth Kery, Amber Horvath, and Brad Myers. Variolite: Supporting exploratory programming by data scientists. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 1265–1276, New York, NY, USA, 2017. ACM.

[129] Mary Beth Kery and Brad A. Myers. Exploring exploratory programming. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 25–29, Oct 2017.

[130] Meraj Khan, Larry Xu, Arnab Nandi, and Joseph M. Hellerstein. Data tweening: incremental visualization of data transforms. *Proceedings of the VLDB Endowment*, 10(6):661–672, February 2017.

[131] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. The emerging role of data scientists on software development teams. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 96–107, New York, NY, USA, 2016. ACM.

[132] Y. Kim and J. Heer. Gemini: A Grammar and Recommender System for Animated Transitions in Statistical Graphics. *IEEE Transactions on Visualization and Computer Graphics*, -(-):1–1, 2020. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[133] Younghoon Kim, Michael Correll, and Jeffrey Heer. Designing Animated Transitions to Convey Aggregate Operations. *Computer Graphics Forum*, 38(3):541–551, June 2019.

[134] Younghoon Kim and Jeffrey Heer. Assessing Effects of Task and Data Distribution on the Effectiveness of Visual Encodings. *Computer Graphics Forum*, 37(3):157–167, 2018.

[135] Donald E. Knuth. Literate programming. *Comput. J.*, 27(2):97–111, May 1984.

[136] Amy 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. The state of the art in end-user software engineering. *ACM Comput. Surv.*, 43(3):21:1–21:44, April 2011.

[137] Amy J. Ko, Brad A. Myers, and Htet Htet Aung. Six learning barriers in end-user programming systems. In *Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing*, VLHCC '04, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

[138] Sean Kross and Philip J. Guo. Practitioners teaching data science in industry and academia: Expectations, workflows, and challenges. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 263:1–263:14, New York, NY, USA, 2019. ACM.

[139] Sean Kross, Roger D. Peng, Brian S. Caffo, Ira Gooding, and Jeffrey T. Leek. The democratization of data science education. *The American Statistician*, 74(1):1–7, 2020.

[140] Daniel Kunin. Seeing Theory, 2020.

[141] Sam Lau, Ian Drosos, Julia M. Markel, and Philip J. Guo. The design space of computational notebooks: An analysis of 60 systems in academia and industry. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, VL/HCC '20, Aug 2020.

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

[143] Katherine A. Lawrence. Walking the tightrope: The balancing acts of a large e-research project. *Comput. Supported Coop. Work*, 15(4):385–411, August 2006.

[144] C Lecher. What happens when an algorithm cuts your health care. *The Verge*, 2018.

[145] Diane Lending and Thomas W. Dillon. Identifying skills for entry-level it consultants. In *Proceedings of the 2013 Annual Conference on Computers and People Research*, SIGMIS-CPR '13, page 87–92, New York, NY, USA, 2013. Association for Computing Machinery.

[146] Jessica Lingel and Tim Regan. "it's in your spinal cord, it's in your fingertips": Practices of tools and craft in building software. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 295–304, New York, NY, USA, 2014. Association for Computing Machinery.

[147] Steve Lohr. Where the stem jobs are (and where they aren't). New York Times, Nov 2017.

[148] Julia S Stewart Lowndes, Benjamin D Best, Courtney Scarborough, Jamie C Afflerbach, Melanie R Frazier, Casey C O'Hara, Ning Jiang, and Benjamin S Halpern. Our path to better science in less time using open data science tools. *Nature ecology & evolution*, 1(6):0160, 2017.

[149] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1039–1065, 2006.

[150] Willam Lurie. The impertinent questioner: The scientist's guide to the statistician's mind. *American Scientist*, 46(1):57–61, 1958.

[151] Jock Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics (TOG)*, 5(2):110–141, April 1986.

[152] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: Urban youth learning programming with scratch. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 367–371, New York, NY, USA, 2008. ACM.

[153] Gideon Mann and Cathy O'Neil. Hiring algorithms are not neutral. Harvard Business Review, December 2016.

[154] Yaoli Mao, Dakuo Wang, Michael Muller, Kush R. Varshney, Ioana Baldini, Casey Dugan, and Aleksandra Mojsilović. How data scientists work together with domain experts in scientific collaborations: To find the right answer or to ask the right question? *Proc. ACM Hum.-Comput. Interact.*, 3(GROUP), December 2019.

[155] Jane Margolis. *Stuck in the shallow end: Education, race, and computing*. MIT Press, 2010.

[156] Geraldine Mason and Annette Jinks. Examining the role of the practitioner-teacher in nursing. *British Journal of Nursing*, 3(20):1063–1072, 1994. PMID: 7827455.

[157] Justin Matejka and George Fitzmaurice. 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, pages 1290–1294, New York, NY, USA, 2017. ACM.

[158] Pietro Mazzoleni, Sweefen Goh, Richard Goodwin, Manisha Bhandar, Shyh-Kwei Chen, Juhnyoung Lee, Vibha Singhal Sinha, Senthil Mani, Debdoot Mukherjee, Biplav Srivastava, Pankaj Dhoolia, Elad Fein, and Natalia Razinkov. Consultant assistant: A tool for collaborative requirements gathering and business process documentation. In *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, OOPSLA '09, page 807–808, New York, NY, USA, 2009. Association for Computing Machinery.

[159] Amelia Ahlers McNamara. *Bridging the gap between tools for learning and for doing statistics*. PhD thesis, UCLA, 2015.

[160] André N. Meyer, Thomas Fritz, Gail C. Murphy, and Thomas Zimmermann. Software developers' perceptions of productivity. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 19–29, New York, NY, USA, 2014. Association for Computing Machinery.

[161] Hui Miao, Ang Li, Larry S. Davis, and Amol Deshpande. Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 571–582, 2017.

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

[163] Wendy Moncur. *The Emotional Wellbeing of Researchers: Considerations for Practice*, page 1883–1890. Association for Computing Machinery, New York, NY, USA, 2013.

[164] D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):438–448, January 2019.

[165] Michael Muller, Ingrid Lange, Dakuo Wang, David Piorkowski, Jason Tsay, Q. Vera Liao, Casey Dugan, and Thomas Erickson. How data science workers work with data: Discovery, capture, curation, design, creation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. Association for Computing Machinery.

[166] Brad A. Myers, Amy J. Ko, Thomas D. LaToza, and YoungSeok Yoon. Programmers are users too: Human-centered methods for improving programming tools. *Computer*, 49(7):44–52, jul 2016.

[167] Lijun Ni. *Building Professional Identity As Computer Science Teachers: Supporting High School Computer Science Teachers Through Reflection and Community Building*. PhD thesis, Atlanta, GA, USA, 2011. AAI3500584.

[168] Lijun Ni and Mark Guzdial. 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, pages 499–504, New York, NY, USA, 2012. ACM.

[169] NIST.gov. Engineering statistics handbook: Measures of skewness and kurtosis. https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm, 2018. Accessed: 2018-09-20.

[170] Safiya Umoja Noble. *Algorithms of oppression: How search engines reinforce racism*. NYU Press, 2018.

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

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

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

[174] Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover, Carole Goble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(10):1067–1100, 2006.

[175] Gary M. Olson and Judith S. Olson. Distance matters. *Hum.-Comput. Interact.*, 15(2):139–178, September 2000.

[176] Gary M. Olson, Ann Zimmerman, and Nathan Bos. *Scientific Collaboration on the Internet*. The MIT Press, 2008.

[177] Judea Pearl. Simpson's Paradox: An Anatomy. Technical report, UCLA, Los Angeles, CA, 2011.

[178] Judea Pearl. Understanding Simpson's Paradox. *The American Statistician*, 68(1):8–13, January 2014.

[179] Roger Peng. How data scientists think - a mini case study. Simply Stats blog – https://simplystatistics.org/2019/01/09/how-data-scientists-think-a-mini-case-study/, 2019. Accessed: 2020-10-10.

[180] Roger Peng. The tentpoles of data science. Simply Stats blog – https://simplystatistics.org/2019/01/18/the-tentpoles-of-data-science/, 2019. Accessed: 2020-10-10.

[181] Roger Peng and Hilary Parker. Not so standard deviations podcast, episodes on design thinking (episodes 63–69). https://nssdeviations.com/63-book-club-part-1, 2018. Accessed: 2020-10-10.

[182] Roger D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[183] pig. ILLUSTRATE, 2017.

[184] Kathleen H. Pine and Max Liboiron. *The Politics of Measurement and Action*, page 3147–3156. Association for Computing Machinery, New York, NY, USA, 2015.

[185] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. Success in introductory programming: What works? *Commun. ACM*, 56(8):34–36, August 2013.

[186] Leo Porter and Beth Simon. Retaining nearly one-third more majors with a trio of instructional best practices in cs1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 165–170, New York, NY, USA, 2013. ACM.

[187] ProjectPro. Type a data scientist vs. type b data scientist. https://www.dezyre.com/article/type-a-data-scientist-vs-type-b-data-scientist/194, 2020. Accessed: 2020-10-10.

[188] E. D. Ragan, A. Endert, J. Sanyal, and J. Chen. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):31–40, January 2016. Conference Name: IEEE Transactions on Visualization and Computer Graphics.

[189] Bina Ramamurthy. 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, pages 169–174, New York, NY, USA, 2016. ACM.

[190] Noopur Raval and Paul Dourish. Standing out from the crowd: Emotional labor, body labor, and temporal labor in ridesharing. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, CSCW '16, page 97–107, New York, NY, USA, 2016. Association for Computing Machinery.

[191] Samuel A. Rebelsky, Janet Davis, and Jerod Weinman. Building knowledge and confidence with mediascripting: A successful interdisciplinary approach to cs1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 483–488, New York, NY, USA, 2013. ACM.

[192] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. Scratch: Programming for all. *Commun. ACM*, 52(11):60–67, November 2009.

[193] Evangeline "Gina" Reynolds. flipbookr: Parses code, creates partial code builds, delivers code movie, 2020. Note: R package version 0.1.0.

[194] Reudismam Rolim, Gustavo Soares, Loris D'Antoni, Oleksandr Polozov, Sumit Gulwani, Rohit Gheyi, Ryo Suzuki, and Björn Hartmann. Learning syntactic program transformations from examples. In *Proceedings of the 39th International Conference on Software Engineering*, ICSE '17, pages 404–415, Piscataway, NJ, USA, 2017. IEEE Press.

[195] Adam Rule, Aurélien Tabard, and James D. Hollan. Exploration and explanation in computational notebooks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 32:1–32:12, New York, NY, USA, 2018. ACM.

[196] E Sands. How to build great data products. *Harvard Business Review.[Online]. Available from: https://hbr. org/2018/10/how-to-build-great-data-products*, 2018.

[197] Arvind Satyanarayan and Jeffrey Heer. Lyra: An interactive visualization design environment. In *Proceedings of the 16th Eurographics Conference on Visualization*, EuroVis '14, page 351–360, Goslar, DEU, 2014. Eurographics Association.

[198] Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & quantity*, 52(4), 2018.

[199] Christopher Scaffidi. Counts and earnings of end-user developers – https://www.linkedin.com/pulse/counts-earnings-end-user-developers-chris-scaffidi/. September 2017.

[200] Christopher Scaffidi, Mary Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. In *Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, VLHCC '05, pages 207–214, Washington, DC, USA, 2005. IEEE Computer Society.

[201] Carlos E. Scheidegger, Huy T. Vo, David Koop, Juliana Freire, and Claudio T. Silva. Querying and re-using workflows with VisTrails. In *SIGMOD '08*. ACM, 2008.

[202] Judith Segal. Some problems of professional end user developers. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, VLHCC '07, pages 111–118, Washington, DC, USA, 2007. IEEE Computer Society.

[203] Yedendra Babu Shrinivasan and Jarke J. van Wijk. Supporting the analytical reasoning process in information visualization. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1237–1246, New York, NY, 2008. Association for Computing Machinery.

[204] Claudio T. Silva, Juliana Freire, and Steven P. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science & Engineering*, 9(5):82–89, 2007. Publisher: IEEE Computer Society.

[205] Petr Slovák and Geraldine Fitzpatrick. Teaching and developing social and emotional skills with technology. *ACM Trans. Comput.-Hum. Interact.*, 22(4), June 2015.

[206] Kent Smith. A brief history of ncbi's formation and growth. https://www.ncbi.nlm.nih.gov/books/NBK148949/, 2013. Accessed: 2018-09-20.

[207] Susan Stager. The consultant as collaborator: The process facilatator model. *SIGUCCS Newsl.*, 16(2):22–26, June 1986.

[208] Sarah L.R. Stevens, Mateusz Kuzak, Carlos Martinez, Aurelia Moser, Petra M. Bleeker, and Marc Galland. Building a local community of practice in scientific programming for life scientists. *bioRxiv*, 2018.

[209] Victoria Stodden, Marcia McNutt, David H Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A Heroux, John PA Ioannidis, and Michela Taufer. Enhancing reproducibility for computational methods. *Science*, 354(6317):1240–1241, 2016.

[210] Sara Stoudt, Váleri N. Vásquez, and Ciera C. Martinez. Principles for data analysis workflows. *PLOS Computational Biology*, 17(3):1–26, 03 2021.

[211] Lucy Suchman. Do categories have politics? the language/action perspective reconsidered. In *Proceedings of the Third Conference on European Conference on Computer-Supported Cooperative Work*, ECSCW'93, page 1–14, USA, 1993. Kluwer Academic Publishers.

[212] Hanxin Tang. The building of trust in client-consultant relationships and its influence on data protection in consulting. In *Proceedings of the 2019 2nd International Conference on Information Management and Management Sciences*, IMMS 2019, page 75–79, New York, NY, USA, 2019. Association for Computing Machinery.

[213] Allison Elliott Tew and Mark Guzdial. The fcs1: A language independent assessment of cs1 knowledge. In *Proceedings of the 42Nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, pages 111–116, New York, NY, USA, 2011. ACM.

[214] Allison Elliott Tew, W. Michael McCracken, and Mark Guzdial. Impact of alternative introductory courses on programming concept understanding. In *Proceedings of the First International Workshop on Computing Education Research*, ICER '05, pages 25–35, New York, NY, USA, 2005. ACM.

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

[216] BARBARA Tversky, JULIE BAUER Morrison, and MIREILLE Betrancourt. Animation: can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, October 2002.

[217] Alexa Vanhooser. Uc berkeley announces data science pipeline program for students. The Daily Californian, Sep 2018.

[218] 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. Best practices in bioinformatics training for life scientists. *Briefings in Bioinformatics*, 14(5):528–537, 2013.

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

[220] April Y. Wang, Ryan Mitts, Philip J. Guo, and Parmit K. Chilana. 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, pages 511:1–511:13, New York, NY, USA, 2018. ACM.

[221] Dakuo Wang, Justin D. Weisz, Michael Muller, Parikshit Ram, Werner Geyer, Casey Dugan, Yla Tausczik, Horst Samulowitz, and Alexander Gray. Human-ai collaboration in data science: Exploring data scientists' perceptions of automated ai. *Proc. ACM Hum.-Comput. Interact.*, 3(CSCW), November 2019.

[222] Zezhong Wang, Jacob Ritchie, Jingtao Zhou, Fanny Chevalier, and Benjamin Bach. Data Comics for Reporting Controlled User Studies in Human-Computer Interaction. *IEEE Transactions on Visualization and Computer Graphics*, -(-):1–1, 2020.

[223] H. Wickham, D. Cook, H. Hofmann, and A. Buja. Graphical inference for infovis. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):973–979, November 2010.

[224] Hadley Wickham. Tidy data. *Journal of Statistical Software*, 59(1):1–23, 2014.

[225] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. dplyr: A grammar of data manipulation, 2020.

[226] Hadley Wickham and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc., 1st edition, 2017.

[227] Leland Wilkinson. *The Grammar of Graphics*. Statistics and Computing. Springer-Verlag, New York, 2005.

[228] Karlijn Willems. Data scientist vs. data engineer. https://www.datacamp.com/community/blog/data-scientist-vs-data-engineer, 2017. Accessed: 2020-10-10.

[229] G. Wilson. Software carpentry: Getting scientists to write better code by making them more productive. *Computing in Science Engineering*, 8(6):66–69, Nov 2006.

[230] Greg Wilson. End-user teachers. http://third-bit.com/2018/06/20/end-user-teachers.html, Jun 2018. Accessed: 2018-09-01.

[231] David Wolber, Harold Abelson, and Mark Friedman. Democratizing computing with app inventor. *GetMobile: Mobile Comp. and Comm.*, 18(4):53–58, January 2015.

[232] Kanit Wongsuphasawat, Yang Liu, and Jeffrey Heer. Goals, process, and challenges of exploratory data analysis: An interview study, 2019.

[233] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2016.

[234] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. 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, pages 1906–1917, New York, NY, USA, 2015. ACM.

[235] Amy X. Zhang, Michael Muller, and Dakuo Wang. How do data science workers collaborate? roles, workflows, and tools. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW1), May 2020.

[236] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, 2017.

[237] L. Zhu, L. Bass, and G. Champlin-Scharff. Devops and its practices. *IEEE Software*, 33(3):32–34, May 2016.

[238] Douglas Zongker and David H. Salesin. *Creating animation for presentations*. PhD Thesis, Citeseer, 2003.