





Temporality in Data Science Education: Early Results from a Grounded Theory Study of an NSF-Funded CyberTraining Workshop

Elliott Hauser¹(✉)  and Will Sutherland² 

¹ The University of North Carolina at Chapel Hill, Chapel Hill, NC, USA
eah13@email.unc.edu

² University of Washington, Seattle, WA, USA
willsk88@uw.edu

Abstract. Interest in data science, especially within the context of graduate education, is exploding. In this study we present initial results from an ongoing qualitative study of an interdisciplinary cyberinfrastructure-focused NSF-funded graduate data science education workshop hosted at an iSchool in the US. The complexity of the workshop curriculum, the participants' and instructors' disparate disciplinary backgrounds, and the technical tools employed are particularly suited to qualitative methods which can synthesize all of these aspects from rich observational, ethnographic, and trace data collected as part of the authors' role on the grant's qualitative evaluation team. The success of the workshop in equipping participants to do reproducible computational science was in part due to the successful acculturation process, whereby participants comprehended, altered, and enacted new norms amongst themselves. At the same time, we observed potential challenges for data science instruction resulting from the rhetorical framing of the technologies as inescapably new. This language, which mirrors that of a successful grant proposal, tends to obscure the deeply embedded and contingent history of the command-line technologies required to preform computational science, many of which are decades old. We conclude by describing our ongoing work, future theoretical sampling plans from this and future data, and the contributions that our findings can provide to graduate data science curriculum development and pedagogy.

Keywords: Data science education · Temporality · Grounded theory

1 Introduction

Data science education is a pressing concern for funding agencies, universities around the world, and members of the iSchools caucus. Recent work has explored

This work was partially supported by National Science Foundation grant 1730390.

© Springer Nature Switzerland AG 2020

A. Sundqvist et al. (Eds.): iConference 2020, LNCS 12051, pp. 536–544, 2020.

https://doi.org/10.1007/978-3-030-43687-2_43

the role that iSchools might play in data science education, both as a mechanism for modernizing the iSchool curriculum and a form of university service [9]. Data science is commonly presented as a ‘cross-cutting’ field which can be applied to any ‘domain’ [13]. In this way it is similar to many of the fields that preceded it, such as statistics or, most pertinently, the so-called ‘metadiscipline’ of information science [3]. As such, the iSchools and the broad fields of information studies have much to bring to the initiatives, funded research, and curriculum updates.

The iSchools are particularly well-suited to contribute a socially informed and yet technically rigorous perspective to data science education. While some recent work has sought to clarify the role that iSchools might play in data science education [9], more work is clearly needed. Other disciplines are bringing their disciplinary norms and perspectives to bear upon this problem [6], and it is critical that the information fields do so as well so that we remain full partners in the transformations under way. While some of these contributions should come from applications of existing literature or lessons learned from analogous transitions in the field’s past, there is still much that is unique to the field of data science that we don’t fully understand.

The present study is intended as a contribution in the latter vein. Our broader study applies disciplinary perspectives from information studies to a qualitative analysis of an NSF-funded data science workshop focused on cyberinfrastructure skills at a US-based iSchool. Here we present some initial results of our ongoing work, based on a grounded theory analysis of interview, survey, observational, and digital trace data sources. We consider data science education a sociotechnical phenomenon, where neither social or technical methods alone can adequately explain what is observed [14]. iSchools initiatives such as the Data Science Education Committee seek to help define the iSchools’ role in higher education’s data-scientific future. Studies which apply the insights of information studies to rich data collected at the sites of ongoing efforts at transformation will be well-positioned to help the iSchools not only find their way in this future but do so in a manner that is consistent with the long history of holistic sociotechnical innovation and research in the field.

2 Theoretical Methods

Given the nascence of data science methods in the sciences, we avoid applying a ready-made frame, and instead rely on a grounded theory approach to sensitize ourselves to the critical points of engagement emerging between data science and scientific practice. Grounded theory has a range of different traditions, idioms, and disciplinary adaptations [2]. For this work, the authors primarily rely on Charmaz [4]. Grounded theory has a history of application in information studies [15], and has been widely used to study both software development [1, 10, 11] and educational issues in technical and scientific education [7, 8]. Though qualitative work commonly relies upon ethnographic observation as a primary data source, Charmaz emphasizes that “All is Data” in grounded theory. This study takes advantage of grounded theory’s data agnosticism to combine traditional qualitative data sources such as ethnographic observation, participatory interaction,

semi-structured interviews, and surveys with trace ethnography of the extensive digital artifacts generated at the study site by both participants and instructors [5]. While trace ethnography has traditionally applied to the large scale data such as server logs, our application of it here applies it to small scale but rich digital artifacts created by the participants, described in detail in Sect. 3. Treating these as material artifacts with specific situated histories [12], they became essential supplementary pieces of our constant comparative analysis of our data.

3 Study Setting and Data Collection Methods

The site of the study was a workshop aimed at instructing scientists and engineers from a wide variety domains in using computational and data management tools in their work. The workshop was held at an information school at a large US public university, and hosted doctoral and postdoctoral participants from a large number of institutions across the US. Instructors were professors in the fields of information or computer science, research scientists, and research staff from NSF-funded cyberinfrastructure projects. A summary of the participants and some descriptive information is provided in Table 1.

The workshop was designed as a two-week, intensive introduction to reproducible computational science. The workshop’s class sessions lasted most of the day each day, and breakfast and lunch were provided on-site. The first week of the workshop consisted of class sessions, including lecture as well as significant hands-on work. In the second week, the participants worked on a variety of group projects, which involved applying computational methods, such as machine learning, or reproducing scientific processing pipelines (in some cases the participants’ own).

Table 1. Instructors ($n = 17$) and participant ($n = 21$) career stages. All staff, including those who did not formally teach sessions, are listed as instructors.

	PhD students	Postdocs	Asst. prof.	Assoc. prof.	Full prof.	Staff	Industry
Instructors	3	–	–	2	2	8	2
Participants	13	8	–	–	–	–	–

3.1 Data Collection, Analysis, and Theoretical Sampling

The study utilizes a variety of data sources and formats. Surveys, conducted after the first and second week of the workshop, provided more broadly comparable textual responses to the course content. Participants provided anonymous feedback by filling out sticky note responses identifying what went well and what did not go well about each session, which were collected and transcribed. Direct

observation was carried out as the authors participated in the workshop as mentors, helping participants with technical breakdowns and project work throughout the workshop and over the intervening weekend. Informal interactions during meals and breaks supplemented formal observation and helped develop rapport. One of the authors taught two sessions on project collaboration tools Git and GitHub during the workshop, by request of the organizers. In addition to the above, the authors had access to and reviewed a large amount of digital artifacts generated by instructors and/or participants. These include:

- Group Slack chats
- Collaborative notetaking via HackMD
- GitHub code repositories and documentation created by participants
- Instructor presentations
- Participant final presentations

Interviews were conducted in the second half of the second week of the workshop, lasting from 20–60 min each. Interviews contained a structured common core of questions and were supplemented by a changing list of topics which had emerged from the current state of our constant comparative analysis and theoretical sampling process. For participants, these included:

- The nature of technical difficulties encountered by the participants.
- Problems participants had conceptualizing the tools and procedures being taught.
- The participants’ experience and history with computational tools.
- The relevance of the workshop content to problems they were facing in their work.
- The relevance of the workshop content to their careers as researchers.

Finally, the authors completed many of the workshop activities themselves, producing field notes from this process. Constant comparative analysis has continued with digital traces and field notes after the event, and theoretical sampling has guided the researchers’ engagement with the voluminous amount of digital trace data.

4 Initial Results

The initial results presented here all deal with the construction of temporality in the workshop: where the present is situated in relation to the past, and what value the past might have to understanding the present. These topics were chosen for their theoretical saturation, coherence with each other, and as examples of the insights we are seeking to generate with our work.

4.1 “Five Years Ago”: The Temporal Framing of Computational Tools

Multiple instructors framed the importance of their subjects by emphasizing the differences between the present and “five years ago.” A variety of tool names, usually proper nouns, were cited as evidence for this, tools that either had changed

scientific practice during that time or, sometimes, were invented during this time. Rhetorically, this places the learner at the cusp of a new and exciting technological world, one which the instructor is familiar with. The names of particular tools stood in for acquired or desired skillsets (“machine learning with Keras”, “reproducibility with Docker”, “workflows with Snakemake”, etc.), and, on a larger scale, the broad application or literacy with these tools established a temporal framing in which a participant, or even a scientific field might be “behind” or “ahead” others in adopting computational methods.

An implication of this construction is that the currently used technologies will themselves be obsolete five years hence, an implication conspiratorially acknowledged by Instructor 10: “of course, no one wants to think about that the technologies we’re learning today will be obsolete in five years’ time, but that’s another story.” This implication is held up as justification for the need for constant training, and perhaps to make learners glad that they are getting caught up now. Participant 3 described his motivation for attending the workshop as, “I think I fell back a bit so I need to keep up and learn these new technologies”. This sea of constant change is presented as an easy-to-deny but ultimately undeniable fact, and one that will allow learners to separate themselves from their peers and maintain their professional relevance.¹

4.2 The Obscured Past

When the workshop participants sat down to learn these cutting edge computational methods, they immediately stumbled upon an array of older, prerequisite technologies. In order to learn cloud systems participants had to struggle with accessing remote machines using SSH, and editing files with command line editors like nano or vim. Furthermore, where participants were able to quickly understand the tools being presented, it was often because they had encountered older technologies that were analogous in some way. Some participants reported picking up the concept of containers more easily, for instance, because of their prior experience with virtual machines.

These encounters with invisible old technologies highlighted a disconnect in the narrative of 5-year technological churn. That narrative obscured the fact that many, if not most, of the technologies, operating systems, platforms, and protocols used during the workshop were comparatively ancient. Command line utilities like ssh, vi, and bash have existed for decades and have a deep history. Version control and social collaboration site GitHub, cited by many participants as one of the most revolutionary tools encountered during the workshop, was founded in 2007 as an easier way to host git repositories. Open source version control software git, first developed in 2005 by Linux inventor Linus Torvalds,

¹ Professional relevance in this context is not limited to academic science. A major theme we will address in future work is the role and involvement of industry in scientific training. Genomics researchers we talked to, for instance, noted that many of their peers completed their doctorate and went to work for, for instance, social media or finance companies.

uses the Vim command-line text editor by default. Vim was first released in 1991.² Git and GitHub both make use of SSH (the binary program and the file transfer protocol), first released in 1995. The historical contingencies and mutual dependencies of these technologies extend in all directions. We cannot give a complete account of them here but rather seek to place the notion of “five years ago” into the context that it helps obscure. Five years hence, it is very likely that GitHub, git, Vim, and SSH will all still be in use. The technologies developed in this span will most likely be as deeply imbricated with current-day technologies as each of these is with the then-current technologies at the time of its initial development.

4.3 “Freezing” the Past: Versions, Tags, and Names

The complex relationship with the past informs the nexus of innovation and preservation constituted by practices of unambiguous naming in software development. Software is deeply embedded within a complex network of historically contingent binaries, programming languages, protocols, interfaces, and idioms. And yet there is immense pressure to collapse this complexity into comprehensible concepts, like “deep learning,” “cloud,” “container,” or “workflow”. This pressure is in one respect cognitive, making ‘hooks’ for understanding. In another respect it is a commodification, an implied equivalence that allows this cloud platform to be substituted for that one, this container to be equivalent from that one (provided they were built from the same image), and this workflow to generate the same results. Technological labels such as Python, Docker, or Ubuntu encapsulate a range of potential versions of their type: Python 3.7, Ubuntu 14.04 LTS, etc. They allow someone to say “I can code Python” or “I know Docker” even as the precise referent of these statements changes over time, sometimes markedly.³ The version name, the commit, and the tag play dual roles in this process, marking innovation and enabling preservation. Version control technology marks a potential boundary between cultures which seek to innovate, those that seek to preserve, and those that are negotiating the relative value of each.

4.4 Honoring Legacy Code: Resisting Invisibility

The constructions of temporality we observed were not univocal, or even necessarily consistent. Participants and instructors also utilized constructions of

² Vim was a clone of and improvement on vi, first released in 1976 as a visual mode improvement on the ex line editor program for UNIX systems. Vim itself was based upon the 1988 C code of an Amiga port of STEVIE (ST Editor for vi Enthusiasts), a 1987 vi clone for the Atari ST. Development of Vim has been near-constant since the 1990s, and new versions are released every few months.

³ An example of this is the discontinuities between Python 2 and 3, which played a role in the workshop as a Participant 7 updated his old processing pipeline from Python 2 to Python 3 as part of the workshop’s project phase.

temporality that placed their work in dialog with technologies and computational artifacts from the past. In many cases these constructions emphasized the scientific relevance of work completed in the past. Instructor 3 framed her lecture around a story that many participants could relate to: when she started her PhD, she inherited code from an outgoing student and then spent almost a year trying to get it working, before being able to get started on any “real” science. The point of her framing was to help stoke participants’ interest in the principles of reproducibility, and it was successful in doing so. During this lecture we learned that, in addition to the instructor, two other participants had used code written in FORTRAN 77 that was critical to their research. FORTRAN 77 was finalized in 1978 and was heavily used in scientific research for decades, but obscure and rarely used today.⁴ This instructor defined “legacy code” as old software that does its job well but is hard to run on modern computers. This is a non-idiomatic usage of a term originally from software engineering. Legacy code in the software industry is any old software that is difficult to work with, prone to breakage, and expensive to maintain. It remains merely because it is too expensive or impractical to replace it with something better. For Instructor 3, legacy code represented something deserving of respect: a valuable, ‘validated’ computational artifact, a literal legacy left to the field by prior researchers. The history it represents is a connection with disciplinary expertise and scientific values.

5 Discussion

The construction of temporality in the workshop we observed placed the participants at the culmination of the past five years of development, and situated the course content as the cusp of innovation in scientific methodology. While this may be an appealing rhetorical framing for funding agencies, deans, and other units of the university, when applied too forcefully it can obscure the deeply historical and embedded nature of the command-line tools used in data science.

We observed participants encounter and successfully utilize many kinds of software tools for the first time. Some, like Singularity, were recently developed, while others, like Git, vi, SSH, or bash, were decades old. All of the software used existing protocols, libraries, conventions, and data formats in ways that were not novel and many activities would have been technically feasible five years ago. What this suggests to us is that computational tools do not travel alone. They embody a history of technological accretion, contain an array of literal and figurative dependencies, and embed practices which give them scientific meaning and value. This has strong implications for how researchers learn data science and scientific software development, but also for the changes that might emerge in various disciplines as computational methods are adopted. The negotiated temporality we observed in our workshop, simultaneously emphasizing

⁴ FORTRAN stands for Formula Translator and was intended for easily translating mathematical formulas, which many scientists were familiar with, into compiled machine code, which they were not.

novelty and mobilizing the past to inform the present, is perhaps a microcosm of larger processes that are playing out in all disciplines which must contend with the adoption and integration of computational methods into their practices and norms.

Our findings show this dynamic at work for participants and instructors alike. On the one hand they were committed to the new cutting edge methods of data science which are redefining their work. On the other hand they were committed to the meticulous preservation activities of reproducibility, through which they work to construct continuity with the past. These combined as a motivation to use tools for computational reproducibility, as a way of making their present work valuable to a hypothetical future.

6 Future Work

The authors continue to study data science workshops as site to further develop this work. More broadly, though, the authors would like to expand out methods to include other data science education formats. Many data science curriculum initiatives are underway, including semester-length NSF CyberTraining programs at several institutions in the US. The authors hope to select one or more of these longer format classes as a future site. The less intensive data collection schedule should allow for a more thorough constant comparative analysis than the opportunistic short-term access utilized in this study can provide.

As data science becomes integrated more completely into curricula at iSchools and beyond, shaping what happens in the classroom and its effects upon subsequent research will become all the more important. The authors hope to employ insights from what is working at their study sites to the semester-length graduate data science curriculum at an iSchool. The resulting work will contribute to the ongoing conversation about the role the iSchools can play in the future of data science education.

References

1. Adolph, S., Hall, W., Kruchten, P.: Using grounded theory to study the experience of software development. *Empir. Softw. Eng.* **16**(4), 487–513 (2011)
2. Apramian, T., Cristancho, S., Watling, C., Lingard, L.: (Re)Grounding grounded theory: a close reading of theory in four schools. *Qual. Res. (QR)* **17**(4), 359–376 (2016)
3. Bates, M.J.: The invisible substrate of information science. *J. Am. Soc. Inf. Sci.* **50**(12), 1043–1050 (1999)
4. Charmaz, K.: *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*. Sage Publications, London (2006)
5. Geiger, R.S., Ribes, D.: Trace ethnography: following coordination through documentary practices. In: 2011 44th Hawaii International Conference on System Sciences, pp. 1–10, January 2011
6. Hicks, S.C., Irizarry, R.A.: A guide to teaching data science. *Am. Stat.* **72**(4), 382–391 (2018)

7. Kampov-Polevoi, J., Hemminger, B.M.: A curricula-based comparison of biomedical and health informatics programs in the USA. *J. Am. Med. Inform. Assoc. (JAMIA)* **18**(2), 195–202 (2011)
8. Kinnunen, P., Simon, B.: My program is ok - am I? Computing freshmen's experiences of doing programming assignments. *Comput. Sci. Educ.* **22**(1), 1–28 (2012)
9. Ortiz-Repiso, V., Greenberg, J., Calzada-Prado, J.: A cross-institutional analysis of data-related curricula in information science programmes: a focused look at the ischools. *J. Inf. Sci. Eng.* **44**(6), 768–784 (2018)
10. Pang, A., Anslow, C., Noble, J.: What programming languages do developers use? A theory of static vs dynamic language choice. In: 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), pp. 239–247, October 2018
11. Prechelt, L., Schmeisky, H., Zieris, F.: Quality experience: a grounded theory of successful agile projects without dedicated testers. In: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), pp. 1017–1027, May 2016
12. Ribes, D.: Materiality methodology, and some tricks of the trade in the study of data and specimens. In: Vertesi, J., Ribes, D. (eds.) *digitalSTS*, pp. 43–60. Princeton University Press, Princeton (2019)
13. Ribes, D., Hoffman, A.S., Slota, S.C., Bowker, G.C.: The logic of domains. *Soc. Stud. Sci.* **49**(3), 281–309 (2019)
14. Sawyer, S., Jarrahi, M.: Sociotechnical approaches to the study of information systems. In: Topi, H., Tucker, A. (eds.) *Computing Handbook*, vol. 19, 3rd edn, pp. 5-1–5-27. Chapman and Hall/CRC, London (2014)
15. Star, S.L.: Living grounded theory: cognitive and emotional forms of pragmatism. In: Bowker, G.C., Timmermans, S., Clarke, A.E., Balka, E. (eds.) *Boundary Objects and Beyond: Working with Leigh Star*, pp. 121–142. MIT Press, Cambridge (2015)