Deliberate Individual Change Framework for Understanding Programming Practices in Four Oceanography Groups

Kateryna Kuksenok[1], Cecilia Aragon[2], James Fogarty[3], Charlotte P. Lee[2], Gina Neff[4]
[1]Hasso-Plattner Institute, Potsdam, Germany
[2]University of Washington, Human Centered Design & Engineering, Seattle, US
[3]University of Washington, Computer Science & Engineering, Seattle, US
[4]University of Oxford, Oxford Internet Institute, Department of Sociology, Oxford, UK
kateryna.kuksenok@hpi.de, aragon@uw.edu, jfogarty@cs.washington.edu, cplee@uw.edu, gina.neff@oii.ox.ac.uk

Abstract. Computing affects how scientific knowledge is constructed, verified, and validated. Rapid changes in hardware capability, and software flexibility, are coupled with a volatile tool and skill set, particularly in the interdisciplinary scientific contexts of oceanography. Existing research considers the role of scientists as both users and producers of code. We focus on how an intentional, individually-initiated but socially-situated, process of uptake influences code written by scientists. We present an 18-month interview and observation study of four oceanography teams, with a focus on ethnographic shadowing of individuals undertaking code work. Through qualitative analysis, we developed a framework of deliberate individual change, which builds upon prior work on programming practices in science through the lens of sociotechnical infrastructures. We use qualitative vignettes to illustrate how our theoretical framework helps to understand changing programming practices. Our findings suggest that scientists use and produce software in a way that deliberately mitigates the potential pitfalls of their programming practice. In particular, the object and method of visualization is subject to restraint intended to prevent accidental misuse.

Introduction

Higher-resolution, greater-coverage data increasingly becomes available for scientific research. Programming tools and skills enable scientists to make use of datasets that exceed by orders of magnitude what had previously been available. Newly available data, enabled by hardware improvements, is made useable and useful by a corresponding shift in coding practices. Software is one of many activities that scientists undertake driven by knowledge acquisition needs (Kelly, 2015). Code is co-produced with data, as a coupled articulation of scientific expectations and assumptions (Paine and Lee, 2014). Doing the work of making software a re-usable resource is a secondary, socially-motivated concern (Trainer et al., 2015). Increased use of programming in ecology, where established practice involves laboratory or fieldwork, corresponds to vocational and identity shifts (Jackson and Barbrow, 2013). We contribute a framework to understand the temporal, sociotechnical process of uptake at the level of small, daily choices of individual oceanographers who code. A simplifying framework is necessitated by the diversity of the object of our research, and arose from iterative qualitative analysis of interview and observational data.

Programming in science is in “permanent beta” (Kelly, 2015), and subject to internal and external temporal processes (Chen et al., 2016; Steinhardt, 2016). The past is connected through data integration and compatibility with existing systems, or with collaborators' infrastructures. The future is connected by way of promising new research group members, and new scientific areas to contend with. Scientists who code (cf., Fig. 1) must constantly re-evaluate which of a growing set of available technologies is worth investigating. Then, they must determine which is worth the integration and collaboration costs involved in subsequent uptake. Scientific programming may be relatively more “risk-averse” due to long lifetimes of code in science and the perceived “cradle-to-grave” responsibility of maintenance (Kelly, 2015). Carver et al. (2007) also describe risk-averse behavior among scientific programmers with respect to the use of external components, which may have a short expected lifespan. Although new technologies are not quickly or easily adopted, uptake and adaptation are commonplace processes over time in activities related to code. Such activities include collaborative software development, writing ephemeral scripts, and using complex software. All these activities demand, or at least benefit from, an increased familiarity with the machine. By uptake, we refer to the process of increasing familiarity through concept learning, skills development in existing technology, and active choice to adopt different tools over time.

Approaching programming in science through the sociotechnical infrastructure lens enables the study of a variety of coders doing a variety of code work. Suchman (2002) shifts from the divisions of “user” and “designer” roles instead toward “an extended field of alliances.” This supports the “view of systems development as entry into the networks of working relations … that make

technical systems possible" (Suchman, 2002). Pipek and Wulf (2009) highlight the flexibility of software, and the reflexive capacity of working environments to be re-constituted over time. This, in turn, allows for small and large software alike to constitute a technical element of a sociotechnical infrastructure (Pipek and Wulf, 2009).

We focus on oceanography, which is affected in a particularly interesting way by hardware advances and corresponding methodological shifts in computational practice. The study of the ocean and its processes spans a variety of disciplines, approaches, questions, applications, geographic areas, and periods of time (Kunzig, 2010). It is also rich in both new and established programming practice applied to many research questions across many temporal and spatial scales (ibid.). In the study of the ocean, methods in general - not just those involving coding - are varied and "radically unstandardized" (Steinhardt and Jackson, 2015). Modeling oceanographers draw on a long tradition of computational practice which poses significant legacy and integration challenges (Edwards, 2010). Observational oceanographers, like ecologists, face a shifting vocational landscape, where daily work moves from the field to the computer screen (Jackson and Barbrow, 2013).

Shifting coding practices in oceanography contribute to bridging the historical divide between the modeling and the observational oceanographers. This divide, in turn, affects uptake of programming practices. Numerical and computer modelling of the oceans has been a major part of knowledge production in the field for nearly half a century. Scientists integrate fieldwork and observational data into the development and parameterization of models as well as interpretation of their results (Edwards, 2010). Data standardization and integration have long been a focus of study and engineering, in service of scientific research and collaboration. In oceanography, internal cultivation and application of programming skills is the norm (Kunzig, 2009; Edwards, 2010). Researchers adopt and adapt aspects of professionalized software engineering terms of art, tools, and best practices (e.g., Mislan et al., 2016; Wilson et al., 2014; Sletholt et al., 2011). The cultivation process includes debate, reflection, piecewise adoption, and adaptation of ideas from the methods discipline. Uptake, the object of our study, is an observable outcome of constructive internal critique.

The visual representation of data and concepts is central in oceanography (Gilbert, 2005). An oceanography lecture typically features an array of diagrams, photographs, and figures using familiar metaphors to express the context and outcome of study. In their daily work, oceanographers create and contemplate many image representations, produced computationally as well as on whiteboards or with pencil and paper. Visualization also plays an interesting role in uptake and mediation of code work. Easterbrook and Johns (2009) describe ways in which climate scientists use visualization as a form of continuous integration. "Continuous integration" refers, here, to a software engineering term applied by

Easterbrook and Johns to actions of the oceanographers they studied. In particular, they found that the scientists routinely tested the entire experiment code as a whole (ibid.). In a literature survey of software engineering research claims about scientific programming process, Heaton and Carver find that design, debugging, and testing are not distinguishable processes with respect to the code itself, which is may result in code more difficult to debug and maintain (Heaton and Carver, 2015). The processing and transformation of data to make visual representation and comparison possible is a major, unsolved, research and design challenge (e.g., Howe et al., 2008, Young and Lutters, 2015). An understanding of uptake and code work must account for the privileged role of visual artifacts. Can such an understanding also provide some predictions of how visual practice in oceanography might be expected to mediate uptake?

We conducted an empirical, descriptive study of programming practices in four oceanography teams over 18 months. We focused on scientists who code (see Fig. 1), interviewing and observing individual undertaking many small uptake actions throughout daily work routines. The inclusion criterion was willingness to build familiarity with the machine, as a means to achieve scientific goals. A total of 46 informants were involved in various social events studied. Of these, 21 individuals were actively incorporating new mechanisms into their scientific practice. These individuals were post-doctoral researchers, PhD students, research scientists and scientific programmers (see Fig. 1). The data were analyzed qualitatively using iterative memos (Miles et al., 2013), to answer the following:

Research Question: How does the intentional, individually-initiated but socially-situated, process of uptake influence code written by scientists?

This paper is structured as follows. The *Background* section provides a theoretical basis for the framework and reviews related work on research on software and programming in science. The *Methods* section describes the site and the informant population, and the qualitative data collection and analysis methods employed. In the *Findings & Framework* section, we present the components of the analytic framework. The findings are reported using "qualitative vignettes" (Barter and Renold, 2000) which are listed in Table 1. The framework enables a comparable description of incremental change through many daily opportunities, influenced by subjective evaluations of mechanisms in the working environment relative to a vision of a "perfect world". In the *Discussion* section, we apply the framework to understand restraint in visualization object and method as it impacts programming practice uptake. Both the *Findings & Framework* section and the *Discussion* section make use of vignettes, but in different ways. The former set of vignettes were selected to introduce the elements of the framework. The latter set, on the other hand, are complex examples to which the framework was applied to

understand visualization during code work. While the *Findings* section presents and justifies the framework and its components, the *Discussion* section explores these through additional vignettes. We outline our findings and offer some future directions in the *Conclusions and Future Work* section.

	Topic		**Section**
Vignette 1	Version control concepts	Findings	Mechanisms, Working Environment
Vignette 2	Switching between set-ups		Evaluation of Mechanisms Relative to the Perfect World
Vignette 3	The moving target		
Vignette 4	Recovery from setbacks		Applying the Framework
Vignette 5	"Just" a visualization	Discussion	Object of Visualization
Vignette 6	"Real" enough to look at		
Vignette 7	Uptake without adoption		Means of Visualization

Table 1. Overview of vignettes in this paper. The framework introduced in this article is demonstrated, applied, and used to derive insights in the following qualitative vignettes. Each can be found in the indicated section in this paper.

Background

Programming practice is subject to individual and social assessments in relation to time availability. Chen et al. (2016) identify differences in human and technical time in a high-performance computing context, particularly in terms of human intervention needs or costs. Button and Sharrock (1994) identify temporal influence as constraint that results in "postponing" better process. Although this would produce better software, it is foregone "in recognition of overriding organizational realities" (ibid., p.221). Steinhardt and Jackson (2015) conducted an ethnographic study of a large-scale cyberinfrastructure project involving oceanography. They identify "anticipation work" as "the mundane, local, and sometimes highly personal accommodations to the future" (ibid.). Programming in science is in "permanent beta" (Kelly, 2015) while being strongly situated in context-specific technical heritage.

Climatologists in the 1950s used many heterogeneous mechanisms to cope with an "explosion of data" (Edwards, 2010). The computational approaches of decades-old data practices affect contemporary attempts to build integrative analytic systems. This subject is explored by Young and Lutters (2015) in the process of designing of a tool that supports synthetizing datasets in Land Change Science. Steinhardt (2016) further describes the "care work of breaking down," and identifies how interpersonal relationships are built up during this breakdown process. In the wake of breakdown, or "frustration" of otherwise "seamless" work, human agents undertake subtle acts of repair work (Jackson, 2014). Lee et al.

(2006) identify “human infrastructure of cyberinfrastructure” as “multimorphous” and “dynamic,” changing more rapidly than physical infrastructure.

We interpret code and code work through the sociotechnical infrastructures lens to capture both relative fluidity and interdependence of technical artifacts. Star and Ruhleder (1996) identify eight aspects of sociotechnical infrastructure. One of these aspects concerns having an installed base. In other words, dependence on existing technical and social structures, previous infrastructures, and systems of support, funding, training, and expertise. Pipek and Wulf (2009) build up the notion of infrastructuring and work environment that can be either collective or individual. This allows for infrastructure objects at different scales: "the concept of infrastructure remains useful regardless of the artifact’s sheer size" (ibid.). Suchman (2002) describes a “dense and differentiated layering of people, activities and things, each operating within a limited sphere of knowing and acting that includes variously crude or sophisticated conceptualizations of the others.” Orlikowski’s (2006) concept of scaffolding can help articulate how technological mechanisms can act simultaneously as constraints and resources. Scaffolds are both dangerous (“as temporary, emergent, and rapidly constructed assemblages, they are vulnerable to breakdown and failure”) and generative (“serve as a basis of other (creative) work”), as well as emergent (“erected over time and changing in form and function” (ibid.).

Pipek and Wulf (2009) identify actors “involved in these processes [of work infrastructure] to … perform deliberate, creative activity directed toward what they consider a lasting improvement.” One friction that arises during the process of developing innovative cyberinfrastructure involves a mismatch in vision. Scientific users are unable to articulate realistic, yet still novel, needs (Ribes and Finholt, 2009). Button and Sharrock (1994) distinguish “how things were” from “how they were to be” (p.225), in the context of evaluating (or strategically postponing) preferable, in favor of practical, programming processes. This is consistent with Suchman’s argument for recognizing “reality of partial translation in place of claims for universality” (2002).

Furthermore, Suchman (2002) challenges the “the vision of a single technology that subsumes all others” with the idea of a “continued existence of hybrid systems composed of heterogeneous devices.” Deliberate, explicit adoption of practices from software engineering can be partial, patchwork hybrids. Sletholt et al. (2011) found partial adoption in a set of case studies evaluating the use of Agile programming practices as reported in scientific articles. Piecewise adoption is not unique to scientific code work. Truex et al. (2000) articulate two extreme narratives of “amethodical” and “methodical” systems development, which are each present to an extent in real-world software engineering groups. Social mechanisms (Trainer et al., 2015) and personal taste and aesthetics (Leach, 2009) motivate production and inform prioritization.

Scientists who program have been characterized as professional end-user developers (Segal, 2007). They do not self-identify as "real" programmers (Kelly, 2015). End-user development is defined as the “set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact” (Lieberman et al., 2006). Because of the extent to which scientists “create, modify, or extend the software artefact,” some studies have considered scientists as developers. Programming practices in scientific contexts recombine or partially adopt software engineering practices (Heaton and Carver, 2015). Some such adaptations may inhibit effective production of high-quality code, because they diverse from software engineering best practices (ibid.). Others have argued that scientific programming is systematic, but in a way distinct from software engineering industry contexts (Kelly, 2015; Easterbrook and Johns, 2009).

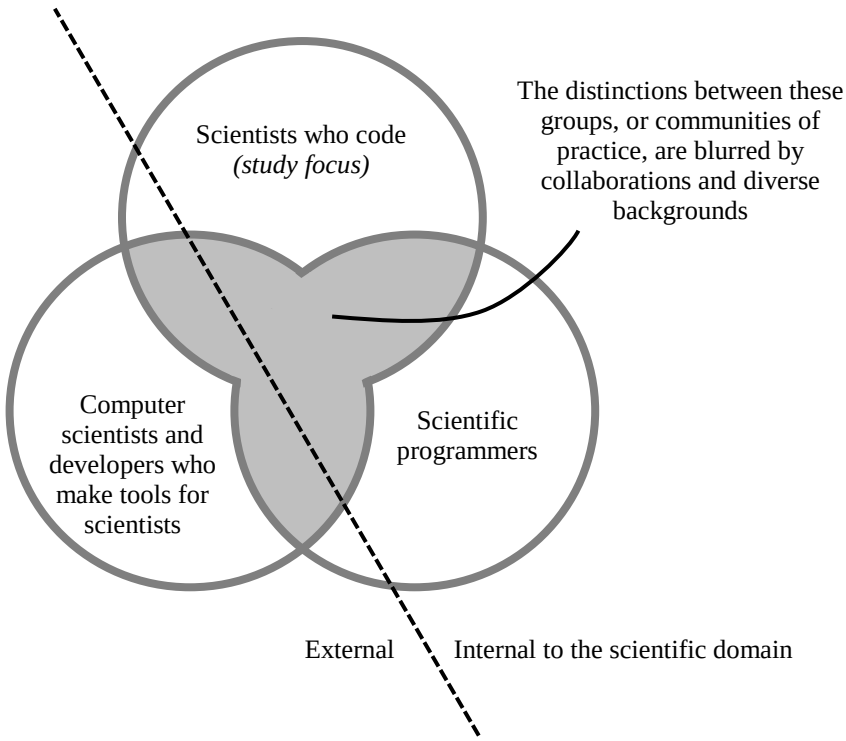


Figure 1. Who does code work in science? We focus primarily on scientists who code. Prior work has focused on scientific developers (e.g., Kelly, 2015; Heaton and Carver, 2015) and computer scientists working in collaboration with scientists (e.g., Ribes and Finholt, 2009; Howe et al., 2008). Recent increasing abundance of data, due to hardware improvement, has resulted in scientists not formally trained to program, and not occupying a formal programming post, to undertake code work (e.g., vocational shifts in oceanography - Jackson and Barbrow, 2013).

Orlikowski develops the notion of scaffolding to describe temporary sociomaterial structuring in knowledge production (2006). We relate the notion of scaffolding to apply to temporary, or intermediate, visualizations. Code may be in the process of being socially vetted through visual intermediaries. Visualization mediates code work in oceanography as a means for testing and validation (Easterbrook and Johns, 2009). Planning and visual artifacts make breakdowns in software visible. Rönkkö et al. (2005) build on the concepts of articulation work and due process in the study of software plans. They define a mediating role of Implementation Plans in software production, which make breakdown visible (ibid.). De Souza et al. (2005) develop a visual artifact to identify the social and socio-technical dependencies in code, using a social-call graph. Based on interviews with biologists, Yeh and Klemmer (2004) find patterns of using physical notebooks that encode relationships to code comprehension: "whereas handwritten data is unprocessed, needing to be transcribed for further analysis, printed data may contain fully 'debugged' lab procedures that have been used with success." These notebooks contain, in addition to printouts, lists, and sketches blank spaces "pre-allocated for future work" (ibid.).

In Kelly's (2015) knowledge acquisition model, software production is driven by scientific knowledge acquisition needs. Desire for more, not less visibility is reported in in the Land Change Science data synthesis system (Young and Lutters, 2015). Chen et al. also found this preference toward transparency in their study of a high-performance computing process management system (2016). Easterbrook and Johns describe a "continuous integration testing" process (2009) neither uses an established Agile method, nor is recognized as such by the scientists. Rather, it is "part of the business of 'doing science' [to] continually experiment with the software itself to improve their understanding" (ibid.). Expertise and familiarity with the machine arises in connections between moral judgments about better process and material judgments about better outcome (Leach, 2009). On the other hand, unfamiliarity results in under-utilization in absence of familiarity of "capabilities and how to exploit them" (Suchman, 2002). Computational skills among scientists are subject to change with respect to professional trends and scientific utility (Steinhardt and Jackson 2015).

Methods

Data collection was conducted over the course of 18 months from April 2014 to December 2015. Our study design accommodated many types of software adoption and adaptation, as well as individual and interpersonal narratives of code-related decisions. Initial interviews with Software Carpentry workshop participants informed the recruitment and study of 4 specific oceanography teams, which are summarized in Table 2. Semi-structured interview protocols and observation approach were informed by prior research and data from a similar

site. Multi-year interview data gathered by Charlotte P. Lee's group regarding an oceanographic collaboration was used to create initial study materials.

Recruitment took place at a variety of events, including workshops, skill-shares, and group meetings. Initial observation also took place at some of these events. Access to two of the four oceanography teams occurred through some members of those groups participating in SWC events attended or observed. The key inclusion criterion for this study was a group's interest in improving their coding skills for scientific work. This willingness was not tied to a specific research agenda or computational tool stack. Therefore, our findings concern tools of varied degree of complexity and abstraction. Groups' interest did not necessarily stem from an appreciation of intrinsic qualities of computational approaches. Neither did it stem from a desire to pursue an expansion of capabilities. Instead, by "interest" and "willingness" we refer to a collective sense of a changing methodological landscape in oceanography.

Not all informants were unequivocally enthusiastic about the increasing amount of programming competence expected or required for professional success. Some expressed enthusiasm about computing methods for their own sake, citing greater computational capability. Some described experiencing a satisfaction with solving the many mundane puzzles of programming, or with engaging with elegant code. Even in the cases of great enthusiasm about programming for its own sake, science was unequivocally the primary endeavor. The time spent programming, therefore, was pulled in pieces out of busy schedules. Initial exploration, or prior experience, was a prerequisite to forming a practical vision for the application of the tool. Both exploration and the maintenance of experience demanded time, as do the immediate needs of defects or experimental requirements. "Willingness" refers also the purposeful and sustained prioritization of non-urgent tasks in scientific programming over the more urgent ones.

Our findings are based on observation of individual scientists doing code work, and semi-structured and unstructured interviews. Observation of workshops and meetings helped to situate individual work in its social and scientific context. All data were recorded in the form of interview transcripts or typed notes, which were then analyzed using iterative coding and memo-writing. The first author performed data collection and preparation of anonymized vignettes. The other authors were engaged in the discussion of these vignettes, their relationships, and connections to literature, as part of analysis. In the following sections, we describe the research site and groups, and the data collection and analysis methods.

Site Description

Four different oceanography groups were studied. To protect the anonymity of these groups, only a minimal relevant background is provided about their work. Each group or lab was led by a Principal Investigator (PI), and included graduate

students and post-doctoral researchers. Some of the groups had additional positions. Both modeling groups had a scientific programmer position, occupied by an experienced programmer who had been with the group longer than most other members. The groups with labs also had lab-specific positions, such as a lab manager, and research scientists who worked on enabling computational methods.

Group Label	**BioGeoChem**	**CustomInstr.**	**Omics**	**RegionalNowcast**
Model/Lab?	Model	Lab	Lab	Model
				Number of Participants, esp. Scientists who code
Total	11	15	9	11
Focus (of total)	7	4	4	6
				Interest in Programming: Scientific Challenges to be Addressed through Code
Analysis	Transformation, comparison	Faster analysis, more data	Effective automation	More features
Visualization	Interactivity in analysis	Real-time & archive website	Sequencing big data	Daily forecast website
Big Data	Access	Storage	Handling	Access
				Additional personnel influencing code work
Programmers	1	1	0	1
Collaborators	1	5	1	1

Table 2: Summary of interest in code work among oceanography participant population, categorized by group. In addition to participant count (total and in the study-focus group), this table provides a summary of the scientific challenges that the teams have in mind as they explore various interests in programming. The name of each group highlights its distinguishing characteristic relative to the other groups. These distinctions may not reflect precisely the way the groups would position themselves in their scientific contexts.

The groups studied dynamics, processes, and small particles and micro-organisms in the ocean. Ecology, genomics, biogeochemistry, physics, biology, and biochemistry research questions were represented across the four groups. Common aspects of analysis involved changes of concentration of nutrients over time and the movement of organisms and particles as affected by eddies and currents. Similarity in data processing and presentation operations placed comparable demands on software functionality and transparency. Many diverse projects could be characterized as study of changing quantities over time while accounting for complex mobility processes. Temperature and salinity were

common, and variations between night and day, or across seasons, were typically taken into account both in observational and modelling contexts.

All groups worked in different sets of offices, through two were situated on different floors of the same building. Despite not spending much time with water samples or the ocean, both modelling groups were personally connected to their surrounding environments. The physicists in our study expressed enthusiasm about and closeness to the outdoors in general. This contrasts the high-energy particle physicists whose research location was purposely at odds with its environment (Traweek, 2009). Parts of this may have reflected the outdoors ethos of the Pacific Northwest at large. Both modelling groups had at least one member who had never been to sea, who was interested in the experience. The Omics-Lab, like CustomInstrument-Lab, had a laboratory space in addition to computer workstations, where members of each spent considerable uninterruptible time. During routine group meetings, certain members were occasionally justifiably absent and "under the hood" (referring to a fume hood in a chemical lab). Computer work was done in shared offices or labs, and short interruptions to ask colleagues for ideas or help were commonplace. Two members of the CustomInstrument-Lab reflected on how their facilities reflected the changing role of computation. The space had been refurbished less than a decade ago. The preferred side of the building which had a good view was occupied by the laboratory space, to purposely accommodate the expected typical workspace. However, in practice, members this group faced data challenges that required extensive computer time. Therefore, the set of offices on the side opposite to the view received more use than expected.

Data Collection and Analysis

The data gathered spanned transcribed interviews and extensive notes from observation of group meetings, other social events, and, primarily, shadowing individuals doing routine code work. Of the 46 individuals in the four oceanography teams, 21 had been shadowed and interviewed during some process of change. The others either did not use computational tools, did not undertake any related changes, or were too busy to participate in the study further. Most of the 300+ hours of observations involved shadowing each of the core informants. A shadowing session lasted between several hours and multiple days during negotiated times. This allowed access to moments throughout the day relevant to the research question. Shadowing was supplemented by unstructured and semi-structured interviews in order to build an understanding of daily work.

Unstructured and semi-structured interview tactics were embedded in a shadowing activity. The semi-structured approach relied on asking deeper follow-up questions about meaning and process based on answers to more specific questions. Some formal interviews were used: scheduled, private, audio-recorded meetings in order to get reflections on some particular event or dynamic. Most of

the reflection and quotes came from countless more unstructured interview opportunities. Interview probes followed particular moments of natural pause in code work. For example, leaning back in the chair and sighing, or expressing a moment of victory. These moments provided opportunity to ask about the challenge and the solution, as well as the current scientific interpretation.

Toward the end of the study, the follow-up questions asked became purposely more speculative. For example, during one brown-bag lunch break, a discussion of a recently-encountered bug took place among two post-docs and the observer. the observer asked the informants to critique an idea for interactive visualization for debugging that built on prior observations. Such probes were able to broach in a concrete way the broader topics of interest, like trust and code comprehension. As in the prior approach of follow-up questions, the goal was to ask about salient topics concretely first, and then abstractly. The probe regarding a visualization for debugging led the informants to describe a prior attempt in the group to make something similar. That attempt had been abandoned, because visual cues seemed too precise, and created an impression of certainty which was dangerously misleading. The exchange revealed an attitude of reluctance which, upon further investigation, resulted in the synthesis presented in the *Discussion* section.

The analysis was carried out by all authors throughout the study, iterated and pooled together by the first author. Subsets of the authors participated in discussions of particular anonymized case study write-ups and preliminary theoretical framings throughout the theory development process (Miles et al., 2013). Three of the authors, including the first author, have a computer science and software engineering background. This enabled our technical focus and interpretation. The data collection and analysis were carried out by the first author, who was able to ask technical follow-up questions.

The outcome of the analysis was the framework, which is presented here. It was developed to reflect a diversity in attitudes towards the introduction of new computational tools. The phrasing and framing remain close to the terms and meanings of the informants. While our framework constitutes an *etic* form of knowledge, the language and concepts are *emic*, derived from the world of the participants themselves. For example, the “ideal” or “perfect world” as an explicit articulation of the direction of desired progress. Evaluation, or normative framing arises in native descriptions of standards or norms: the “shoulds” of their work. These descriptions arise in internal discourse and rhetoric, as arguments or suggestions: “we should really use version control.”

Limitations

Our study was limited to a particular set of individuals, pursuing specific research in oceanography, and is not necessarily representative of other scientific contexts. Our inclusion criterion necessitated an attitude of willingness toward uptake of new technologies. Participant recruitment began at sites of expression of this

willingness: local workshops and tutorials on code in science. As a result of the inclusion criterion, study spanned a wide range of programming background, motivations, interests, and tools adopted (and abandoned). Willingness did not necessarily imply enthusiasm, so reluctance and mistrust could also be captured. This diversity limited the extent to which we could compare software practices.

Furthermore, all Principal Investigators (PIs) in our study encouraged group members to invest time and energy in exploring new technologies. All the groups were successful constituents of high-profile departments. We did not include those oceanographers not holding the view that their scientific landscape of was undergoing a significant computational methods shift. We cannot claim any generalization of our findings to groups more reluctant to changing methodologies. Likewise, these findings may be overshadowed by financial constraints in a less professionally secure or supportive environment. We only conducted the study for 18 months, which allowed us a detailed view on small-scale deliberate change but not on longer-scale changes and drift.

Findings & Framework

In this section, we focus on defining the framework that emerged from this study. The terminology used in this framework is selected to mirror the terminology of the participants, as described in the *Methods* section. The structure of this framework builds on the related and background work. This section presents qualitative vignettes of increasing complexity using concepts form the framework, and addressing our question: *how does the intentional, individually-initiated but socially-situated, process of uptake influence code written by scientists?*

Mechanisms in the Working Environment

Thus far, we have described a study site characterized by heterogeneity of technology and its uses (Suchman, 2002; Steinhardt and Jackson, 2015). In this section, we build up the definition of a working environment comprised of mechanisms. Both concepts include not only the technological function of a mechanism, but also social and personal. The technical function of a mechanism is connected to "getting set up" for new contributors to a joint scientific project that has a coding component. A mechanism can act as a social resource by supporting communication, decision-making, goal-setting, and reflection with respect to a project. Lastly, the dimension of personal skill and knowledge can enable an unused resource to have influence. Consider Vignette 1, which provides an overview of different roles that GitHub, as a mechanism, can play from the perspective of an individual.

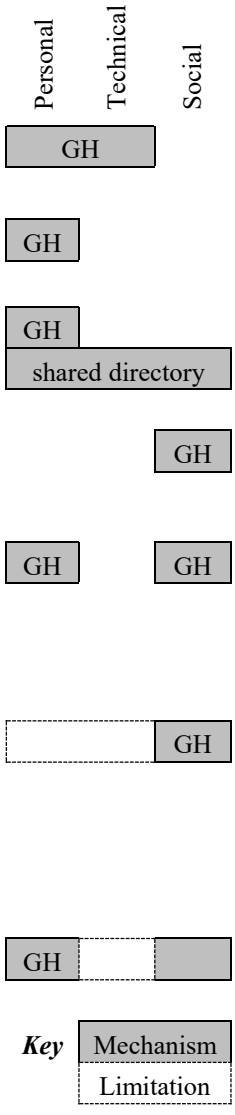
We consider GitHub a mechanism, despite the variety the roles it plays, capabilities it entails, and interpretations it inspires. Scientist express their

willingness to build familiarity with the machine to achieve scientific goals. One such means for expression is attending Software Carpentry (SWC), an international organization and movement to teach basic computing skills to researchers. There are other workshops of this type, but SWC workshops have been offered since 1998 and have been in increasing demand in recent years. GitHub is one of the four components of the two-day curriculum, and the subject of lectures advocating for better programming practices in science. These workshops and lectures are delivered not only by external advocates but by members of the domain discipline. In Vignette 1, we highlight also how miscommunication about the nature of GitHub impacts uptake.

Vignette 1: Version Control Concepts

GitHub (GH) is a web-based service for hosting code version-controlled through *git*, a command-line tool that keeps track of changes to code over time to make recovery easier and more clearly show collaborative contributions. Version control with GitHub is a core part of the Software Carpentry (SWC) curriculum and other introductory resources.

How do scientists understand GitHub after participating in SWC?



Specific technical tools and operations: "merge," "branch," "clone." The use of these terms suggest familiarity with the technology.

However, familiarity does not imply active use. Not using GitHub is not experienced as a problem, but as a context-specific decision.

An alternative to GH may be a shared directory on a common server, or in a file-sharing application like Dropbox. If an individual's colleagues use the shared directory, the individual need not apply her familiarity with GH.

An individual need not be familiar with or use GH to benefit from it as a social resource: his other colleagues can set up helpful email notifications of updates.

GH is tied to a broader professional and scientific discourse: "accountability and transparency," "open source software" "standards in my new lab," "collaboration with other institutions," "online professional portfolio." These are not tied to the technical utility of GH in managing code. This association helps SWC motivate the inclusion of GH in its curriculum.

This can lead to confusion, however. Of 10 SWC attendees interviewed after participation regarding uptake, 3 reported that they did not want to use version control because they did not want their code to be public. This prevents the tool from being useful on the basis of a miscommunication. Version control tools, *git* hosted on GH included, do not necessitate public access.

Sometimes, GH – a hosting service – may be down, or *git* – a command line tool – may struggle with merging two colleagues' work on the same file. In the event of a temporary technical breakdown, the individual can leverage their personal skill/knowledge and social resources (colleagues and StackExchange) to address the breakdown while potentially strengthening those resources by doing so.

The landscape of technology occupies many different scales. A single software "tool" might be seen as a collection of distinct programs, or as a component of a larger system. Our concept of the mechanism discretizes the technological landscape on the basis of emic narratives expressed by study informants. Despite the broad definition, GitHub is a mechanism and not a working environment. We define the working environment as a personal, contextualized collection of mechanisms and supports technical work activity in its entirety.

Evaluation of Mechanisms Relative to the Perfect World

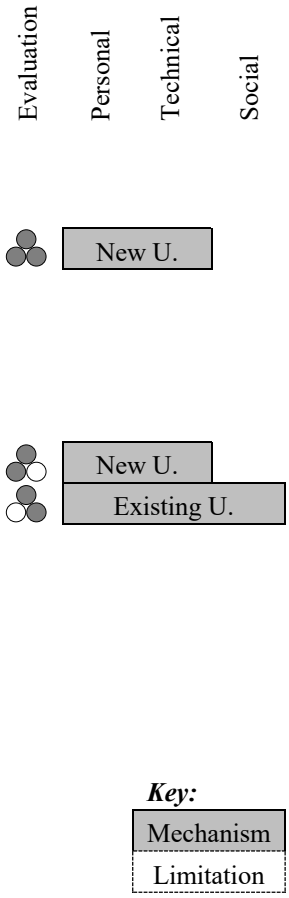
A working environment is subject to reflexive change (Pipek and Wulf, 2009). This section provides the definition and justification for our model of how individuals enact change. The evaluation of a mechanism in the working environment is subjective, and roots the mechanism in the scientific use case. In the “perfect world,” tools are (1) understandable, (2) perform the necessary task, and (3) persist over time. As Figure 2 illustrates, these properties interact with the limitations and bottlenecks in the personal, social, or technical resources of a mechanism. Even if no one particular mechanism taken alone is difficult to evaluate, understand, use, or improve, the combination of mechanisms over time exacerbates integration and compatibility challenges. Our framework enables a specificity and comparability in revealing these challenges.

The relevance of these three properties (***functionality, understandability, sustainability***) is revealed through internal discourse and critique of technology. During almost every explanation of the current working environment, the primary scientific programmer in the BioGeoChem-Model group called it “not ideal” while providing a historic or domain-specific reason for it being so. Another scientific programmer in the CustomInstrument-Lab team noted a shortcoming in the data collection: “in a perfect world, there would be a satellite for data transfer during cruise, but this is not that world, so data is transferred once [on shore].” Another member in the BioGeoChem-Model team described the difference between two major programs: both are large and have many constituent packages, but one is more “conservative” in terms of including new packages, which makes it generally more “reliable.” Furthermore, neither is especially “user-friendly” because the “programmers don’t want people to get too complacent.” The barrier to the perfect world is described as not currently surmountable: historical constraints in a large codebase; procedural or technical constraints in cruises; and the potential complacency of users.

Our model of change centers prioritization, and one important embodiment of prioritization rests in the ***role of the software advisor***. We identify this role in our data based on whether non-professional developer scientists consistently sought, and acted on, the advice of particular individuals. Formal positions enable individuals to undertake this role, but are not a pre-requisite. Table 2 lists these for each group studied as "additional personnel influencing code work." The role

of the software advisor is sustained by others' actions toward them. Colleagues' requests are filtered through desire for sustainability in the working environment.

Vignette 2: The Set-Ups



When does the evaluation of a mechanism deteriorate?

Erin was a post-doctoral fellow in the RegionalNowcast-Model group who worked with both observational data and models. She had two different "set-ups" for tasks in these historically distinct areas in oceanography.

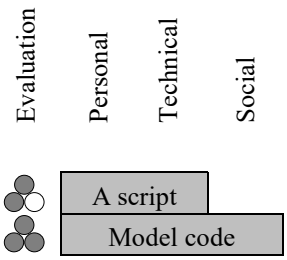
The "set-ups" were distinct enough in her mind that, in one observation session, she accidentally re-implemented a small utility (*New U.*) because she forgot that it already existed in the other, inactive environment.
The diagram to the left highlights this new mechanism, which is evaluated to be overall acceptable.

When Erin began to explain to the observer what she had been working on, the similarity of this utility to an existing one (*Existing U.*) occur to her. Duplication of code, and lack of code reuse, was a violation of the "best practices" which she considered important.
The evaluation of New U. now excludes "sustainability," which is undermined by duplication of the existing, shared functionality in Existing U.; meanwhile, Existing U. does not have the desired functionality "out of the box" and needs additional action, even though it is preferable overall.

Scientists chose which problems warranted the attention of the software advisor. For one informant, the approach of having a specialized tool for every case is an impractical ideal: "well, it would be wonderful to have the exact right custom solution for every problem! But that is not the reality." This kind of customization and maintenance would place overwhelming demands on the limited time on Andrew, an in-house software advisor in the CustomInstrument-Lab group. Two oceanography post-docs (including Mallory, cf. Vignette 4) were cognizant of Andrew's limited time and self-censored which problems to bring to him: "we can't [meaning, should not] be walking down the hall asking [Andrew] to do things all the time," despite the sense that "well, he can just [solve any problem] in the best way." Minor shortcomings and setbacks were routine, and many more than could be addressed by the software advisor. Study participants did most of their code work on the assumption of being personally responsible for maintenance and communication.

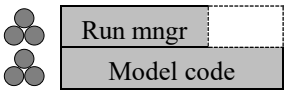
The need to understand is a pre-requisite for using a software tool or code for a scientific task. For example, the software advisor for the RegionalNowscast-Model described the experience of providing technical explanation to the domain scientists he works with: “I can see when their eyes glaze over. [The PI] is really good at pushing me on it: ‘do I - as a scientist - really need to know this?’” However, what a scientist might “really need to know” varied widely depending on the scientific and social context. One researcher in the CustomInstrument-Lab group expressed frustration over the efforts of a programmer to make a tool too user-friendly. The effort backfired: the mechanism was not clear to the users, as one post-doctoral fellow remarked: “We are scientists! Looking inside and figuring it out is what we do!” A member of the BioGeoChem-Model team also joked about how sometimes it was difficult to get the software advisor to explain his actions: “he doesn't like talking, just fixes all our problems super-fast.”

Vignette 3: The Moving Target

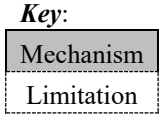


Evaluation from the perspective of a software advisor.

The part-time scientific programmer who was working with members of the RegionalNowcast-Model team had explained in the beginning of the study that many of the group’s Python scripts use a large model code with many constituent packages and many diverse users internationally.
The diagram to the left shows a combination of an ephemeral, unsustainable script used in combination with a shared model code.



Unsatisfied with this, he described a hypothetical program which would initiate a run with a single command (*Run mngr*). It would carry out many mechanical but essential processes, like creating symbolic links and adding jobs to the queue manager in the computing resource, behind the scenes. Several months later, he unveiled such an implementation to the four students and post-docs who comprised its current users. His demo was filled with mentions of the many ways in which the command would be improved "ideally". These comments provided an equally specific but different vision of a technology that would support scientific work. Notably, the revised vision expanded no only to include more features, but also to account for a wider and more diverse audience. In the next iteration, it would also account for as-yet-unspecified but expected and desired new additions to the group. *The diagram on the left is updated to reflect the better sustainability of the run manager script, but highlighting its limitation in supporting new group members.*



Functional: *the mechanism performs the task*
Understandable: *the individual understands it*
Sustainable: *the individual expects it to work in the future*

In Vignettes 2 and 3, we map an individual's subjective experience. Over time, the individual's evaluation of the working environment degrades. Greater awareness or self-initiated improvement highlights new shortcomings or limitations. These form a perceived demand for action, or opportunity. The action then serves to bring the mechanism closer to the "perfect world," by recovering from the shortcoming. Vignette 2 illustrates how the evaluation of a mechanism adjusts on the basis of the broader working environment context. Vignette 3 describes the moving target of the "perfect world" that adjusts to new limitations when previous shortcomings are addressed. In both cases, the adjustment of evaluation is internal in that it is initiated by the individual. It can also be seen as external, however, in that it is triggered by social interaction, either with the observer or with the colleagues. Furthermore, when the barriers are overcome in the pursuit of a vision, the vision changes.

An individual's imagination is informed in successive acts of trial and error, and through interactions with the software advisor/advocate. The perfect world is unattainable as long as there are resource constraints, and while external resources and internal functionality changes over time. Action is targeted at the adoption of an identifiable mechanism. A localized attempt to enact that change can impact the arrangement and relationships of other components, even if the mechanism is not ultimately adopted. In this way, it is possible for partial adoption, or non-adoption, to nevertheless change working environment. Under this framework, all changes are deliberately oriented toward the vision of a perfect world.

Our framework identifies stages of change that include attitudinal stance rather than uptake or adoption directly. In this way the framework accounts for the connection between external influences and internal direction experienced by people. The preconditions for action with respect to a mechanism are: (1) awareness; (2) normative framing; and (3) opportunity. For example, a local free workshop on a technology that has been in the normative framing stage provides an opportunity. In the case of the Omics-Lab group, this is how they described their trigger to attend the workshop. The normative framing stage involves the recognition of shortcoming in the evaluation relative to the perfect world. Direct external influences can also help establish initial awareness, such as through talks contributing to awareness of useful mechanisms. Externally-triggered breakdowns can both contribute to evaluation adjustment and provide opportunity for action.

Applying the Framework

We introduced a model of incremental change through many daily opportunities, influenced by subjective evaluations of mechanisms in the working environment relative to a vision of a perfect world. Figure 2 provides a fictional example that illustrates how the notation supports expressing a wide range of subjective evaluations of mechanisms.

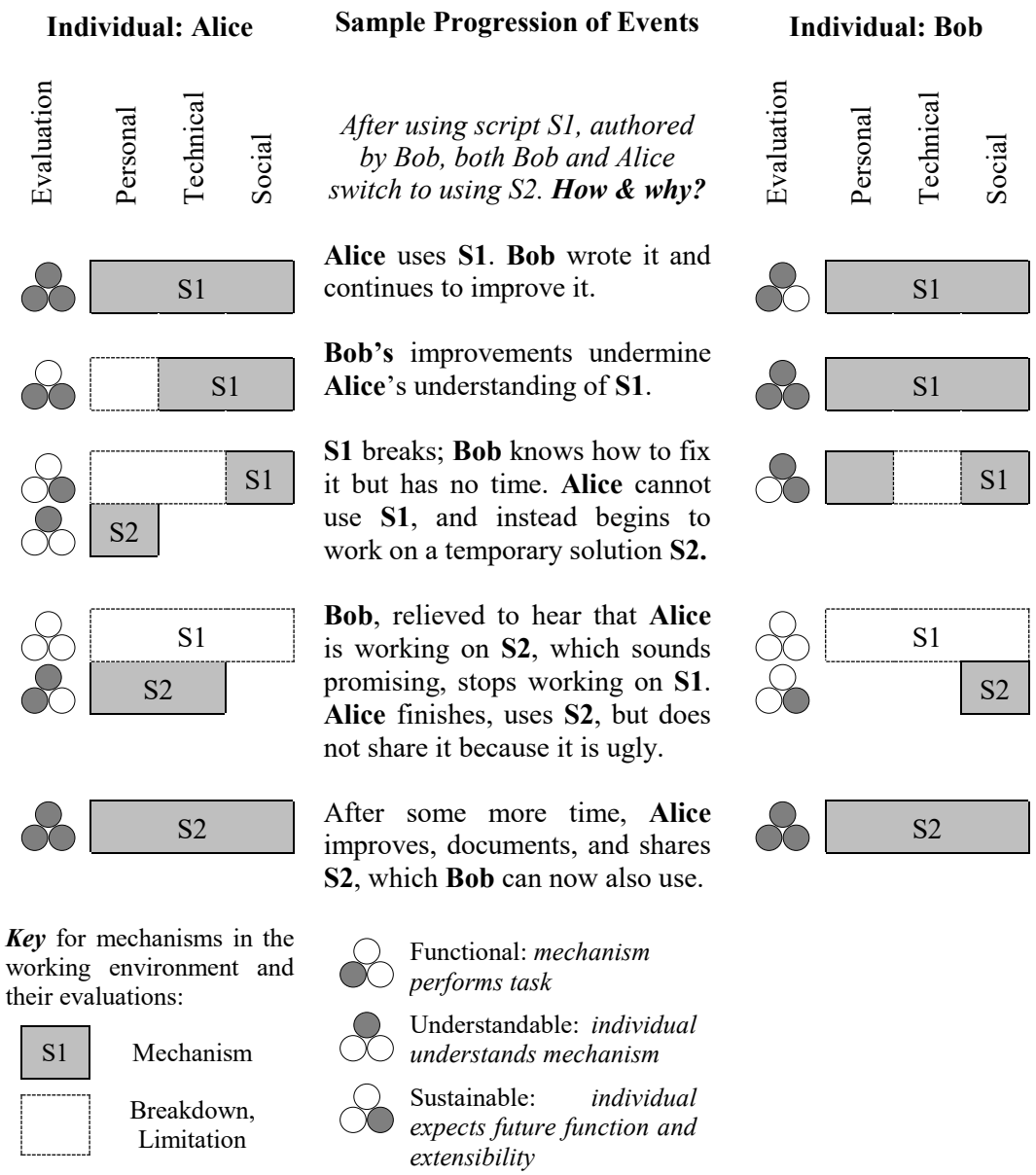
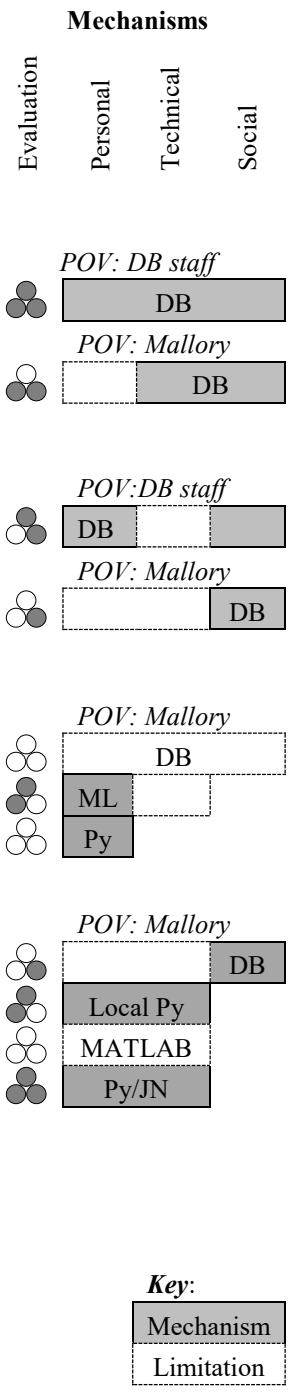


Figure 2. Notation for the deliberate individual change framework, demonstrated in context of a fictional narrative. Evaluations are not always aligned between the individuals, even when they use the same mechanism. The combination of mechanism breakdown in part of the working environment, and its evaluation, informs action. Individuals strive to recover from breakdown in the direction of what they experience to be functional, understandable, and sustainable mechanisms. The qualitative notation highlights tensions and work completed at cross purposes by reducing complex variables 6 dimensions. The notation for evaluation relative to the perfect world is binary (presence/absence) with respect to 3 properties. The notation for how a mechanism fits into the working environment is ternary (presence/absence/breakdown) with respect to 3 resources. These subjective properties are justified in the *Findings and Framework* section.

In Vignette 4, we consider a post-doc, Mallory, recovering from two technological setbacks. We use the framework to formulate hypotheses explaining why her experience of the two technologically-similar setbacks is so different.



Vignette 4: Recovery from Setback

Two different setbacks, occurring to the same person, require outdated scripts to be fixed and re-run. The individual experiences both setbacks, and their respective recovery processes, differently. One is "devastating" and the other is a "time investment." Why?

Mallory, a post-doc in the CustomInstrument-Lab, was using a database (DB) maintained as part of a collaborative project. As an experimental tool, working with it required close collaboration with computing experts. This at times left Mallory wanting a deeper understanding of the system.

A hard-write failure corrupted some of the data. The computing researchers involved in the collaboration wiped the database clean, without affecting the original data. From their perspective this was unfortunate, but unsurprising in an experimental context. Mallory commented that, "in a sense nothing is lost, but nothing is in a format I could do anything with." She described the event as a "major set-back," "devastating," because she had "nothing to show" for her scientific work.

A prior colleague who was most knowledgeable about how to get the data into the system had recently left. Recovery from the breakdown involved also retracing past actions, at times under-documented, with new personnel. Meanwhile, her MATLAB license had expired, forcing her to make a decision: pay for the license, or re-implement her visualization code in Python.

Regarding the visualizations, she decided in favor of reimplementation using open-source, free components, which would be more sustainable. This involved a concerted effort to go through tutorials and learn how to use a library that she had been "meaning to" learn. Mallory viewed this activity as a "time investment". Regarding the database event, she continued to work with the new DB staff, improving communication with the possibility of future work. In the meantime, she was able to use familiar, local tools on her machine (Python and Jupyter Notebook) to run smaller-scale but still fruitful analyses.

One explanation would cite only the scale of the setback: the data loss problem took longer to solve, at least in part because it was a bigger breakdown and involved more people. Another explanation would distinguish the former action as reinstating something previously-available and suddenly frustrated. The latter action, of reimplementation, was enabling something new to the working environment. Both variants distinguish the two situations through individuals' relationship to time. Using the concepts of our framework, we can formulate more precise observations. In the case of the database, there were two breakdowns, in the social and technical aspects of the working environment. These exacerbated the already-imperfect evaluation. Recovery action required simultaneous maintenance of two alternative mechanisms, both with shortcomings. In the case of the visualization, the setback was localized to ne particular problem, and the recovery action addressed it well relative to all relevant dimensions.

Discussion

At the end of the *Background* section we noted the importance of visualization in supporting both code work and scientific work, and mediating code use in oceanography (e.g., Easterbrook and Johns, 2009). In this section, we provide one answer to the question, *how does the intentional, individually-initiated but socially-situated, process of uptake influence code written by scientists?* The process of answering this question is aided by the framework introduced previously, which is applied to more complex vignettes and related to existing scholarship. The answer that is produced is intended to shed light on the question, as well as to offer a discussion of how our framework can be used to gain insight. This answer returns to the centrality of visualization, specifically to restraint in scientific code work (cf., risk-averse programming practice, Kelly, 2015). We consider the object and means of visualization as targets of intentional restraint. This restraint is intended to prevent accidental misuse or complacent carelessness.

Creating and considering visual representations of data constitutes a set of common activities that cross-cut personal, social, and technical resources of each individual's working environment. The practice of interpreting visualization is both a personal and a social resource. The individual capacity to interpret data using consistent visual metaphors is cultivated socially. Technical steps that lead to a visual representation of data are triggered by intuiting something "weird" or "interesting" enough to investigate. The outcome arrives, through iterative subsequent visualization, at either an error or a finding. This discourse interrogates the visualization and its scientific implications. It supports the social construction of visualization literacy. It also structures, and is structured by, further technical exploration and uptake.

To contextualize the discussion of visualization, we briefly describe the role of visualization in daily oceanographic work. Printouts of charts can be physically

arranged; a robust working environment may provide rapid, interactive exploration of data. The observations of Easterbrook and Johns (2009) with regards to oceanographic visualizations used for testing in code align with our own. Gathering around a monitor was the most common social practice observed. An interactive environment allowed the generation of new images on demand as part of a process of rapid social exploratory analysis. Solitary generation of a wide array of exploratory figures in an interactive environment, notably MATLAB or Jupyter Notebook, was the most common visual practice overall. An individual typically continued to be the only one physically interacting with the workstation as one or more colleagues joined to discuss. This imposed no requirements on the specific technologies used to produce a picture. Printing and laying out the charts, or projecting during a group meeting, was robust to major differences between individuals' working environments. Print-outs have the added affordance of being rearranged into pairs or sets, which was an important capability for one post-doc in the BioGeoChem-Model group. Even without physical printouts, it was rare for a single chart to be the subject of attention. Multiple charts were discussed, either in context of making a comparison or in triangulating a process/feature of interest across multiple slices through space or time.

The Object of Visualization: Restraint toward Representation

Visualization is central in sense-making and relies upon a common visual language specific to the scientific community. In the below Vignettes 5 and 6, we explore restraint towards representation. In both instances, a less experienced contributor encounters a situation where a quantitative measure ought not to be embedded visually. One interpretation is that an insufficiently “real” (e.g., Vignette 6) representation may undermine an individual’s personal meaning-making process. Which measure to plot from a list of available options is a technologically arbitrary decision becomes socially meaningful. Vignette 5 exemplifies a breakdown resulting from unawareness of this meaning.

Both vignettes illustrate how the realness of data intersects with its interpretability. In the examples, the relationship of the data to the phenomenon studied was that of systematic measurement bias. The PI of the BioGeochem-Model group explains that models represent “different worlds.” The things that happen in models “are real,” but how or whether they match the observed world remains to be articulated explicitly. Ideally, they have an understandable relationship to the elements affecting the natural phenomenon of interest.

The use of different bathymetry files in modelling research helps exemplify these different, real worlds. The ocean, both in reality and in representation, is bounded by both the surrounding land topography and underwater topography. Bathymetry is the study of underwater topography. To run a model, the researcher needs - among other things - a file that contains information about depths, or a "bathymetry file." A post-doc in the RegionalNowcast-Model group compared

several such files to determine which was "best." She looked for one that preserved the high-level patterns relevant to claiming a link between the model and the observed world. Of these "real" worlds, the most preferable one is not necessarily the most accurate in terms of depth. Rather, it is one that produces more realistic results in the model relative to the dynamic being studied. Visualization was the primary means for making this assessment.

Vignette 5. "Just" a Visualization. Melissa is a computer science student who at one point had been part of the CustomInstrument-Lab collaborative project. Her involvement with the group had wound down, and she was describing some not-too-promising attempts to make connections with other groups. She had recently reached out to some researchers working with Argo floats[1]. She expressed difficulty finding physical oceanographers to collaborate with: "I just hoped they would be more excited, but they were getting caught up on things that I didn't think were important." She had sent them a plot based on the data, and received in response an explanation for how a particular column ought not be used for analysis, but only for quality control.

The column names in the standardized format (NetCDF) were not self-explanatory, so the common format did not save an enthusiastic student from violating unstated assumptions. This was frustrating for her, because her intention to demonstrate a visualization process had backfired in a way that she perceived as disproportionate to the technical issue itself: "this is a *database*, we can filter that out later! I was *just* plotting something!" Her working environment lacked the social resources in the oceanographic domain, but she evaluated her visualization as understandable, functional, and extensible. In particular, she understood how to extend the solution to include other data to address functional needs. Those who received it, however, did not share this interpretation.

Vignette 6: "Real" Enough to Look at. The amount of matter in a biochemical context can be measured in several ways. Counting discrete instances, such as organisms, is another measure of amount. Alternative to counting, weigh of a substance in grams may be used. A mole is a unit for measuring the amount of a substance relative to its chemical makeup. A femtomole (*fmol*) is 10-15th of a mole. A mole is measure used in chemistry to reason about the quantities of molecules as they react to form other molecules. In the equation $2H2+O2 \rightarrow 2H2O$, measuring in weight directly would require further calculation, since weight is related to reactive amount via molecular weights.

Colleen is a post-doctoral fellow whose doctoral work is from a neighboring discipline. She used *fmoles* in a weekly group presentation. The PI stopped her mid-presentation in a welcome, teaching, but uncompromising manner: "fmoles are not real." The PI explained that the distinction between counting and weight produces "totally different shapes" for two outputs that are both "about amount". The reference to "shapes" here refers to plots with the measure in question on the Y-axis, and various substances along the X.

---

[1] "Part of the integrated global observation strategy," a project that now offers 15 years of global float data. See http://www.argo.ucsd.edu/

Measurements depend on their context of use and influence on interpretation. In the above example, one common, relatively direct measure of quantity is less "real". Realness refers less to representational capability than to consistency of visual language in the scientific community. Other common examples of the context of use affecting the interpretation of data include the use of relative measures. Especially in the study of earth systems (Edwards, 2010), model-laden data and data-laden models, dual interaction. In field data collection, as the many discussions we observed revealed, many biases and errors may influence an absolute measure. However, repeated measurement, either in the model or the observational sense, may be used to produce relative measures. For example, the count of the number of a certain kind of organism is difficult to estimate. If the estimation is biased in the same way at different time points, organism birth, death, or reproduction rates can nevertheless be studied.

Restraint toward what is visually represented is of central importance to understanding code work in science. Visualizations are subject to a variety of social and scientific requirements, with regard to consistency and to accurately representing a wealth of heterogeneous data.

The Means of Visualization: Restraint toward Affordance

The groups we studied shared an attitude of interest, or at least willingness, toward adoption of new technologies and learning new technological concepts. However, they were relatively reluctant when faced with specific tools or interventions. Especially in the area of interactive visualization, which was a subject of potential collaboration discussions for multiple groups. The perfect world involves improvement, but not if it means sacrificing the flexibility and social precision of existing practices.

Vignette 6 provides an example of scientists prioritizing measures prior to representation, demonstrating care in the work of crafting what can be seen as inscription devices (Latour and Woolgar, 1979). The conversation from this vignette occurred in a weekly meeting of the Omics-Lab group, in the context of exploring an interactive exploratory visualization software. The story of uptake of this software is explored in Vignette 7.

Viewing static images or animations side by side was one of the most common activities throughout observation of individuals and groups. Common visual representations are taught through instruction and practice to understand. They are crucial in mitigating the many sources of error in such a wide-ranging discipline that integrates may difficult sources of data. The uptake, pedagogy, rhetoric, and discourse of code work is subject to pervasive processes uptake, pedagogy, rhetoric, and discourse of visual language.

Vignette 7: Uptake without Adoption. Tableau is a desktop Graphical User Interface (rather than a Command Line Interface) application for interactive visualization. Members of the Omics-Lab group try it out. They do not adopt it, in the sense of reorganizing their working environments to replace prior tooling (Excel) with the new tooling. However, the experience has an impact.

The PI was aware of Tableau through email announcements of trainings and collaboration opportunities. She prioritized building awareness of available tools that may be useful in the changing methodological landscape to prepare the lab members professionally. In this group, the adoption of Tableau would impact parts of the working environment that were relying on Excel and, in some cases, a Python script written by Lana, a graduate student. The opportunity to try out Tableau was created inadvertently by the observer, who was asked whether she knew how to use Tableau and responded in the affirmative. In the next shadowing observation session, Colleen, a post-doc, commented that it was helpful to have a dedicated time to try it out, and someone to provide some guidance. This interaction occurred prior to data analysis and theory development.

A great deal of initial excitement about Tableau dampened upon deeper contact: it was not the solution to all problems, despite certain desirable features. Within a few weeks of the shadowing session where Colleen tried out Tableau, an email thread preceded the weekly lab meeting, demonstrating the excitement. The first email was sent by the PI asking if the meeting is necessary and whether it should be rescheduled. Colleen replied with an excited message about having Tableau charts to show, which roused the PI's enthusiasm.

Several months after the Tableau meeting, a graduate student in the Omics-Lab group, Alice, used Tableau to make some plots with an exciting set of data after a data-collection cruise. The observer became aware of this having encountered the PI after a guest lecture, when the PI expressed her excitement: “ask [Alice] about her beautiful chart! She made it in R!” Showing her results, Alice explained that she tried out Tableau (which has a Graphical User Interface, as previously noted), but ultimately chose to switch to R (which has a Command Line Interface) because Tableau did not offer enough control over formatting and other visual aspects of the chart that would be necessary for it to be a publication-ready visual artifact. In the final R chart, differently-sized and differently-colored disks were arranged to follow the path of the cruise on a map of the region of interest. A complementary figure featured a series of pie-charts arranged in a table varying by size and composition. Both used visual vocabulary native to Tableau but uncommon in oceanography talks, so the observer asked the inspiration behind the choice of visual mapping, and Alice confirmed that she had re-implemented the Tableau variant in R. In an indirect way, it had an impact. However, even if it did not, it needed to be tried in order to be rejected for offering insufficient control over output relative to professional needs.

An affordance of an object refers to the subjective interpretation of possible action made possible by this object. Vignette 6 reports on a discussion about which measure should or should not be plotted. This occurrence was in context of trying out an interactive visualization software that made it very easy to make potentially misleading charts. The technical object affords this action; the PI of the group can be said to be concerned about the accidental misuse of the

affordance. Restraint toward affordance can also be seen in how Caleb, a post-doctoral fellow in the BioGeoChem-Model team, described the difference between two major programs. Both are large and have many constituent packages, but one is more “conservative” in terms of including new packages, which makes it generally more “reliable.” Neither is especially “user-friendly” because the “programmers don’t want people to get too complacent.”

We take "complacency" to refer to accidental or careless execution of actions that a mechanism affords, but which are undesirable in the scientific context. This interacts with the three qualities of the perfect world, especially sustainability. Post-docs in the CustomInstrument-Lab and the RegionalNowcast-Model teams would sometimes avoid external libraries. This anticipated of hypothetical future colleagues working with the code and being unable to use or maintain it correctly. Re-use of some code was replaced by re-implementation, or more localized re-use. Lack of re-use is problematized in software engineering accounts of scientific programming practice, as well as in internal critique that adapts software engineering rhetoric. Our framework serves to provide a reason for this action that is not related to lack of interest, willingness, or awareness. Instead, motivated scientists enthusiastic about best practices choose to adapt them to safeguard the scientific usefulness of the code they work on.

Conclusions and Future Work

We studied code work practices, building on existing work in CSCW and Software Engineering research. The first contribution of this article is an analysis of an 18-month qualitative study in the uniquely illustrative scientific code context of interdisciplinary oceanographic research. We asked the following question: *How does the intentional, individually-initiated but socially-situated, process of uptake influence code written by scientists?* We argued that restraint with respect to the object and means of visualization is intended to reduce potential misuse and misunderstanding. This argument made use of the deliberate individual change framework, which we introduce as a second contribution.

Prior scholarship has considered scientific code through a sociotechnical lens (Pipek and Wulf, 2009) and as situated in time (Chen et al., 2016; Steinhardt, 2016). Risk-averse behavior among scientific programmers, regarding uptake, has also been explored (Carver et al., 2007; Kelly, 2015). Our framework allows for additional technical detail in analyzing risk-averse uptake situated in time. This framework can be used in future studies by structuring interview design and observation. Our work, like many studies of computing in science, is motivated by an experience of change. The current collective understanding is based on in-depth analyses of groups or projects (e.g., Steinhardt, 2016), or broad snapshots (e.g., our study). Quantifying states and transitions in this process would help study change in both breadth and depth.

The framework we presented (1) focuses on the individual, (2) highlights sociotechnical mechanisms that have a discursive or rhetorical basis meaningful to that individual, and (3) enables articulating subjective evaluations of those mechanisms in the current working environment relative to an imagination of the perfect world. The working environment is the set of mechanisms available to an individual, composed of technical, social, and personal resources. Personal resources refer to skills, knowledge, and awareness, through which an individual can inform her current work by the experience gained from previous work.

Each mechanism (identifiable tool or protocol) is subject to evaluation and change relative to an imagination of the perfect world. Deliberate change, informed by this evaluation, is possible throughout many small daily opportunities for prioritization of one activity over another. In the scientific context, access to experts in both computational methods and domain-specific application is scarce. Scientists who code therefore prioritize not only their work, but the work with which they appeal to such experts, which we refer to as software advisors. For an individual to choose to pursue a change in the working environment, it is necessary to have awareness of a mechanism. Then, the individual evaluates the future possibility as being closer to the imagined perfect world. Finally, the individual can be moved to action through an opportunity affording a clear course of action. Triggers for change arise from the social and scientific context.

This framework was developed in a grounded, iterative manner from our qualitative data, and refined through discussion and connections to existing theory. In addition to providing examples from our data, we presented a qualitative notation (see Figure 2) that may help to condense complex accounts and compare subjective experiences over time.

We reviewed related work spanning CSCW and Software Engineering (SE) research on scientific programming practice. CSCW research focuses on human activity and human relationships around the code or mediated by the code. Meanwhile, SE research is chiefly concerned with applying a critical technical lens to induce improvement through the development of tools or advocacy of practices (e.g., Wilson et al., 2014; Heaton and Carver, 2015). The goal of CSCW is aligned with the improvement mandate some of the time, but focuses more on the interpersonal or contextual barriers to uptake than on technological correctness or advocacy. Our aim has been to provide an initial understanding of the internal technical criticism undertaken *within a scientific domain* which impacts uptake. This has not been recently investigated in either, because it necessitates a critical technical lens, but without the improvement imperative.

Future work can extend the understanding of change in scientific programming practice. This framework can be applied in qualitative and quantitative study contexts. In a perfect world, future work would improve and condense this framework. Ideally, it would become an understandable, functional, and sustainable mechanism for a variety of scholars and engineers.

Acknowledgments

This work was supported by grants from the Gordon and Betty Moore Foundation and the Alfred P. Sloan Foundation awarded to Professor Cecilia Aragon; and by NSF Graduate Research Fellowship and AT&T Graduate Research Fellowship awarded to Kateryna Kuksenok. NSF awards IIS-0954088 and NSF ACI-1302272 awarded to Prof. Charlotte P. Lee supported supplementary data used for the beginning of the study. We would like to thank Emmerline Wu and Madeline Wood, who assisted with gathering observational data from meetings. We would also like to thank the study participants for their time, energy, and goodwill.

References

Barter, Christine; and Emma Renold (2000). 'I wanna tell you a story': Exploring the Application of Vignettes in Qualitative Research with Children and Young People. *International Journal of Social Research Methodology,* vol. 3, no. 4, pp. 307-323.

Button, Graham; and Wes Sharrock (1994). Occasioned Practices in the Work of Software Engineers. In M. Jirotka and J. A. Goguen (eds.): *Requirements Engineering*. pp. 217-240. San Diego, California: Academic Press Professional, Inc.

Chen, Nan-Chen; Sarah Poon; Lavanya Ramakrishnan; and Cecilia R. Aragon (2016). Considering Time in Designing Large-Scale Systems for Scientific Computing. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, San Francisco, CA, USA, February 27 – March 02, 2016*. ACM Press, pp. 1535-1547.

Carver, Jeffrey C.; Richard P. Kendall; Susan E. Squires; and Douglass E. Post (2007). Software Development Environments for Scientific and Engineering Software: A Series of Case Studies. In *29th International Conference on Software Engineering, 2007. ICSE 2007, Minneapolis, MN, May 19 – May 27, 2007*. IEEE, pp. 550-559.

Easterbrook, Steve M.; and Timothy C. Johns (2009). Engineering the Software for Understanding Climate Change. *Computing in Science & Engineering*, vol. 11, no. 6, pp. 64-74.

Edwards, Paul N. (2010). *A Vast Machine: Computer Models, Climate Data, and the Politics of Global Warming.* Cambridge: The MIT Press.

Gilbert, John K. (2005). Visualization: A Metacognitive Skill in Science and Science Education. In J. K. Gilbert (ed): *Visualization in Science Education*, pp. 9-27. Dordrecht: Springer Netherlands.

Hannay, Jo Erskine; Carolyn MacLeod; Janice Singer; Hans Petter Langtangen; Dietmar Pfahl; and Greg Wilson (2009). How do Scientists Develop and Use Scientific Software?. In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, 2009. SECSE'09, May 23, 2009.* IEEE, pp. 1-8.

Heaton, Dustin; and Jeffrey C. Carver (2015). Claims About the Use of Software Engineering Practices in Science: A Systematic Literature Review. *Information and Software Technology*, vol. 67, no. C, pp. 207-219.

Howe, Bill; Peter Lawson; Renee Bellinger; Erik Anderson; Emanuele Santos; Juliana Freire; Carlos Scheidegger; António Baptista; and Cláudio Silva (2008). End-to-end eScience: Integrating Workflow, Query, Visualization, and Provenance at an Ocean Observatory. In *IEEE Fourth International Conference on eScience, Indianapolis, Indiana, USA, December 10 – 12, 2008.* IEEE, pp. 127-134.

Jackson, Steven J.; and Sarah Barbrow (2013). Infrastructure and Vocation: Field, Calling and Computation in Ecology. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Paris, France, April 27 – May 2, 2013.* ACM Press, pp. 2873-2882.

Jackson, Steven J. (2014). Rethinking Repair. In T. Gillespie, P. J. Boczkowski and K. A. Foot (eds.): *Media Technologies: Essays on Communication, Materiality, and Society,* pp. 221-39. Cambridge: The MIT Press.

Kelly, Diane (2015). Scientific Software Development Viewed as Knowledge Acquisition: Towards Understanding the Development of Risk-Averse Scientific Software. *Journal of Systems and Software*, vol. 109, no. C, pp. 50-61.

Kunzig, Robert (2000). *Mapping the Deep: The Extraordinary Story of Ocean Science.* New York: WW Norton & Company.

Latour, Bruno; and Steven Woolgar (1979). *Laboratory Life: The Construction of Scientific Facts*. Princeton: Princeton University Press.

Leach, James; Dawn Nafus; and Bernhard Krieger (2009). Freedom Imagined: Morality and Aesthetics in Open Source Software Design. *Ethnos*. vol. 74, no. 1, pp. 51-71.

Lee, Charlotte P.; Paul Dourish; and Gloria Mark (2006). The Human Infrastructure of Cyberinfrastructure. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, Banff, Canada, November 6- 8, 2006*. ACM Press, pp. 483-492.

Lieberman, Henry; Fabio Paternò; Markus Klann; and Volker Wulf (2006). End-User Development: An Emerging Paradigm. In H. Lieberman, F. Paternò, Fabio, V. Wulf (eds.): *End user development*, pp. 1-8. Dordrecht: Springer Netherlands.

Miles, Matthew B.; A. Michael Huberman; and Johnny Saldaña (2013). *Qualitative Data Analysis: A Methods Sourcebook.* Washington DC: SAGE Publications.

Mislan, K. A. S.; Jeffrey M. Heer; and Ethan P. White (2016). Elevating the Status of Code in Ecology. *Trends in Ecology & Evolution*, vol. 31, no. 1, pp. 4-7.

Orlikowski, Wanda J (2006). Material Knowing: The Scaffolding of Human Knowledgeability. *European Journal of Information Systems*, vol. 15, no. 5, p. 460.

Paine, Drew; and Charlotte P. Lee (2014). Producing Data, Producing Software: Developing a Radio Astronomy Research Infrastructure. In *IEEE 10th International Conference on e-Science, October 20 – 24, 2014*. IEEE, vol. 1, pp. 231-238.

Pipek, Volkmar; and Volker Wulf (2009). Infrastructuring: Toward an Integrated Perspective on the Design and Use of Information Technology. *Journal of the Association for Information Systems*, vol. 10. no. 5, pp. 447-473.

Ribes, David; and Thomas A. Finholt (2009). The Long Now of Technology Infrastructure: Articulating Tensions in Development. *Journal of the Association for Information Systems*, vol. 10, no. 5, pp. 375-398.

Rönkkö, Kari; Yvonne Dittrich; and Dave Randall (2005). When Plans Do Not Work Out: How Plans are Used in Software Development Projects. *Computer Supported Cooperative Work (CSCW)*, vol. 14, no. 5, pp. 433-468.

Sletholt, Magnus Thorstein; Jo Hannay; Dietmar Pfahl; Hans Christian Benestad; and Hans Petter Langtangen (2011). A Literature Review of Agile Practices and Their Effects in Scientific Software Development. In *Proceedings of the 2011 ICSE Workshop on Software Engineering for Computational Science and Engineering, Waikiki, Honolulu, HI, USA, May 21 – 28, 2011*. ACM Press, pp. 1-9.

Trainer, Erik H.; Chalalai Chaihirunkarn; Arun Kalyanasundaram; and James D. Herbsleb (2015). From Personal Tool to Community Resource: What's the Extra Work and Who Will Do It?. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, Vancouver, BC, Canada, March 14 – 18, 2015.* ACM Press, pp. 417-430.

De Souza, Cleidson; Jon Froehlich; and Paul Dourish (2005). Seeking the Source: Software Source Code as a Social and Technical Artifact. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work, Sanibel Island, FL, USA, November 06 - 09, 2005*. ACM Press, pp. 197-206.

Traweek, Sharon (2009). *Beamtimes and Lifetimes.* Cambridge: Harvard University Press.

Truex, Duane; Richard Baskerville; and Julie Travis (2000). Amethodical Systems Development: The Deferred Meaning of Systems Development Methods. *Accounting, Management and Information Technologies,* vol. 10, no. 1, pp. 53-79.

Segal, Judith (2007). Some Problems of Professional End User Developers. In P. Cox and J. Hosking (eds.): *Visual Languages and Human-Centric Computing, Coeur d'Alene, ID, USA, September 23 – 27, 2007,* pp. 111-118. IEEE.

Star, Susan Leigh; and Karen Ruhleder (1996). Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces. *Information Systems Research,* vol. 7, no. 1, pp. 111-134.

Steinhardt, Stephanie B.; and Steven J. Jackson (2015). Anticipation Work: Cultivating Vision in Collective Practice. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, Vancouver, BC, Canada, March 14 – 18, 2015*. ACM Press, pp. 443-453.

Steinhardt, Stephanie B. (2016). Breaking Down While Building Up: Design and Decline in Emerging Infrastructures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 07 – 12, 2016*. ACM Press, pp. 2198-2208.

Suchman, Lucy (2002). Located Accountabilities in Technology Production. *Scandinavian Journal of Information Systems*, vol. 14, no. 2, p. 7.

Wilson, Greg (2013). Software Carpentry: Lessons Learned. *F1000Research, Faculty of 1000 Ltd*.

Wilson, Greg; Dhavide A. Aruliah; C. Titus Brown; Neil P. Chue Hong; Matt Davis; Richard T. Guy; Steven H. D. Haddock; Kathryn D. Huff; Ian M. Mitchell; Mark D. Plumbley; Ben Waugh; Ethan P. White; and Paul Wilson (2014). Best Practices for Scientific Computing. *PLoS Biology*, vol. 12, no. 1, e1001745.

Yeh, Ron B.; and Scott Klemmer (2004). Field Notes on Field Notes: Informing Technology Support for Biologists. Stanford University, California: Stanford InfoLab.

Young, Alyson L.; and Wayne G. Lutters (2015). (Re) defining Land Change Science through Synthetic Research Practices. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing, Vancouver, BC, Canada, March 14 – 18, 2015*. ACM Press, pp. 431-442.