Assessing the Impact of Intensive Software Skills Training on Students' Scientific Careers

Rachel Slaybaugh, Kaitlin Thaney, Lorena Barba, C. Titus Brown, and Paul Wilson
with
Scott Collins, Ethan White, Tracy Teal, and Greg Wilson

January 19, 2014

1 Summary

FIXME: one-page summary

2 The Problem

Scientists and engineers invented electronic computers to accelerate their work, but two generations later, many researchers in science, technology, engineering, and mathematics (STEM) are still not *computationally competent*: they do repetitive tasks manually instead of automating them, develop software using a methodology best summarized as "copy, paste, tweak, and pray", and fail to track their work in any systematic, reproducible way.

And while the World-Wide Web was created by a scientist to help his peers share information, many still use it primarily as a way to find and download PDFs. Researchers may understand that open data can fuel new insights, but often lack the skills needed to create and provide a reusable data set. Equally, any discussion of changing scientific publishing, make research reproducible, or using the web to support "science as a service" must eventually address the lack of pre-requisite skills in the general STEM research community.

Since the mid-1980s (at least), proponents of computational science have taken an "if we build it, they will come" approach to this problem. It is clear now, though, that learning-by-osmosis has not worked, and is unlikely to in future for several reasons:

1. *The curriculum is full.* Undergraduate STEM programs already struggle to cover material regarded as core to their field. While many scientists would agree that more material on programming, reproducible research, or web-enabled science would be useful, there is no consensus on what to take out to make room.

2. *The blind leading the blind.* Many faculty lack computational skills themselves, and hence are unable to pass them on.

3. Scientists and software developers have different cultures, different priorities, and different approaches to problem solving, which often impedes collaboration and knowledge transfer [8].

4. *Difficulty of assessing impact.* It is easy to say, "This discovery could not have been made without use of that supercomputer." It is much harder to attribute specific advances in science to prior training in general computing skills.

The final, and possibly largest, issue is that *the rewards are unknown.* Open, web-based science is still in its infancy, so there is no general understanding of what people might need to know in order to incorporate it into

their research careers. Since it is hard to measure something if you don't know what to look for, or if it is so young that there hasn't actually *been* long-term impact, little systematic study has been done to date of whether early training in the skills needed to practice open, web-enabled science actually has an impact, and if so, how and how much. Without such feedback, there is no systematic way to improve the training programs that currently exist.

3 Related Work

3.1 Research

Studies of how scientists use computers and the web have found that most scientists learn what they know about developing software and using computers and the web in their research through osmosis and word of mouth [4, 7]. In our experience, most training meant to address this issue:

- does not target scientists' specific needs (e.g., is a general "Introduction to Computing" class shared with students majoring in other areas);
- only covers the mechanics of programming in a particular language rather than giving a complete picture including data management, web-enabled publishing, the "defense in depth" approach to correctness discussed in [2], or the other foundational skills laid out in [11]; and/or
- jumps to advanced topics such as parallel computing before scientists have mastered the foundations. (Most research on scientific computing, such as [6], does the same.)

3.2 Software Carpentry

Software Carpentry [1, 10] is the largest effort to date to address these issues. Originally created as a training program at Los Alamos National Laboratory in the late 1990s, it is now part of the Mozilla Science Lab's efforts to help scientists take advantage of ways in which the web can change the practice of science today, and invent new ways tomorrow. Over 100 certified volunteer instructors delivered two-day intensive workshops to more than 4200 people in 2013 alone. Their subject matter usually includes:

- the Unix shell,

- version control with Git,
- programming in Python or R, and
- using SQL.

What these workshops actually seek to convey, though, is:

- how to automate repetitive tasks,
- how to track and share work,
- how to grow a program in a modular, testable, reusable way, and
- the difference is between structured and unstructured data.

Software Carpentry's curriculum and teaching practices have been refined via iterative design, and are informed by current research on teaching and learning best practices. Its instructor training program introduces participants to a variety of modern teaching techniques (e.g., peer instruction), to concepts underlying these techniques (e.g., cognitive load theory), and to topic-specific work by computing education researchers (see [3], [5], and the first third of [9] for overviews.) One example of how we translate theory into practice is our insistence on live coding during teaching as a way of demonstrating and transferring authentic practice to learners.

Software Carpentry has been assessing learning outcomes and retention since the beginning of its Sloan Foundation-funded in January 2012. The first round of assessment included both qualitative and quantitative assessment by Dr. Jorge Aranda (then at the University of Victoria) and Prof. Julie Libarkin (Michigan State University). An attempt to scale this up in 2013 was set back by personnel changes, but systematic follow-ups with past participants in workshops have now been resumed, and we expect to be able to present preliminary results by mid-2014.

3.3 The Hacker Within

FIXME (PW/KH): Please fill in.

4 Proposed Work

This proposal builds on our success to date in enhancing the skills of graduate students, post-docs, and faculty. We designed it to:

1. conduct formative evaluation of the impact of software skills training for undergraduates likely to continue in research careers as they progress through the early stages of those careers;

2. conduct summative evaluation of the training's overall impact on a multi-year timescale in order to improve the content and presentation of the training; and

3. disseminate the resulting curriculum as widely as possible.

More specifically, we will run two-day software skills workshops each year for four years for undergraduate students taking part in that year in the NSF's Research Experience for Undergraduates (REU) program, at or near the start of those students' time in the lab. We believe this training will help them be more productive during their REU (graduate-level participants in our existing workshops typically report that what we teach saves them a day a week), and also prepare them to work in a world where all aspects of science are increasingly dependent on computing. More importantly, these undergraduates will serve as the treatment population for a five-year study of the impact of this training on their careers in general, and their involvement with open and web-enabled science in particular.

The sections below detail the specific activities we will undertake.

4.1 Workshops

We will run two-day workshops at a steadily increasing number of sites each year for four years, timed to coincide with the start of the summer REU influx. Each workshop will be offered to a minimum of 40 learners per site (giving is a target study population of 1440 students by year 4). The content will be tailored to meet local needs, but will be based on what is being used at that time by Software Carpentry and affiliated educational efforts. All of the workshop instructors will have been trained and certified by Software Carpentry, and will have had prior experience teaching this material.

The home sites for investigators named in this proposal (Michigan State University, Utah State University, George Washington University, University of New Mexico, University of California – Berkeley, and University of Wisconsin – Madison) will run workshops in each of those years. Two other NSF REU sites will be added each year, increasing the total to 12 sites by year 4, to increase the size of our study population.

In order to increase the diversity of the study population, we will additionally run at least one workshop in each of years 1-4 that is specifically

aimed at female students. Software Carpentry's first such workshop, held in Boston in June 2013, attracted 120 participants; its second is scheduled for Lawrence Berkeley National Laboratory in March 2014, and at least two more will be held by the time work on this project commences (one in the United States and one in Europe). This work will build on that experience, and draw on the pool of instructors who have gained mentoring experience through those specific workshops.

Finally, we will organize an equal number of workshops specifically aimed at students from minority groups that are under-represented in STEM. We are already in contact with the Computing Alliance for Hispanic-Serving Institutions (CAHSI), and with the Association of Public and Land-grant Universities' program for historically black colleges and universities (HBCUs); Software Carpentry is running its first workshop at an HBCU (Spelman) in early 2014, and we expect to have expanded these efforts by the start of this project.

4.2 Assessment

We will employ one full-time researcher for five years to monitor and compare undergraduate participants in these workshops, participants in a subset of our regular (graduate-level) workshops, and non-participants (as a control population). Assessment will focus particularly, but not exclusively, on the following questions:

1. Are students who receive this training more likely to continue to graduate school than their peers?

2. Are students who receive this training more likely than their peers to incorporate open science and/or web-enabled science tools and practices into their work?

3. Are students who receive this training more likely than their peers to choose computationally-oriented research topics and/or careers? Are those who do not choose computationall-oriented paths nevertheless more likely to incorporate the tools and practices mentioned above into their work?

4. Are students who receive this training more likely than their peers to develop new tools and practices, and/or become involved in outreach and education activities (i.e., are they more likely to become creators and leaders)?

5. In what ways does this training change students' outlook on the practice of science itself?

This researcher will also explore ways in which our engagement with students changes the outlook and work practices of their peers and faculty supervisors (i.e., whether there is knowledge transfer sideways and upward), and at the effectiveness of the community building and dissemination activities detailed in the next sections.

As with our work to date, assessment will use both qualitative and quantitative techniques. On the qualitative side, we will conduct a series of interviews over the five-year period of the study to see how attitudes, aspirations, and activities change. Quantitatively, we will measure uptake of key tools such as version control as a proxy for adoption of related practices, as well as exploring more traditional measures of research success such as progression to graduate school and publication/citation rates.

Our findings will be shared with other researchers through publication in peer-reviewed journals such as *Science Education* and *Physics Education*, and through presentations at conferences such as SIGCSE (the ACM's Special Interest Group on Computer Science Education) and smaller, domain-specific venues including the Association for Biology Laboratory Education (ABLE), the Society of Industrial and Applied Mathematics (SIAM)'s education session at the Joint Mathematics Meeting, and the tracks hosted by the Education Training and Workforce Development Division at the American Nuclear Society (ANS)'s annual meeting. We will also continue to share our ideas in publications such as *American Scientist*, which reach a broader cross-section of interested parties.

4.3 Community Building

We will employ one graduate student part-time at each participating site each year to provide technical support to workshop participants, and to act as an anchor for a Hacker Within-style grassroots group at that site. These community liaisons will not be study subjects, but will help us stay in touch with students who are (a key requirement for any longitudinal study).

Separately, the Mozilla Science Lab will focus part of its ongoing community engagement efforts on the students who have taken part in our workshops during both the remainder of their undergraduate careers and afterward in order to ensure that they become part of the broader open science community. This may include helping the students organize and run workshops of their own in subsequent years, connecting them with other open

science projects, introducing them to potential graduate supervisors who understand and value their new skills and outlook, etc.

As a subordinate part of their work, the researcher employed by this project will assess the effectiveness of these local organizers. In particular, they will explore whether seeding activity in this way leads to the formation of self-sustaining grassroots groups, and if so, what activities those groups develop on their own, how (and how effectively) they share discoveries with each other, the extent to which alumni of this program stay engaged with these groups, and whether the presence of these groups has a demonstrable impact on students' career paths in general, and their engagement with open and web-enabled science in particular.

4.4 Curriculum Development and Dissemination

We will employ one instructional designer part-time during each of the study's first four years to create new material, and to improve existing material based on feedback from workshop participants and the assessment program. Here, "creating material" may include both designing and implementing new domain-specific learning modules, and translating existing materials into new forms, such as video recordings of lectures or auto-graded quizzes for self-paced instruction. This work will be done in consultation with educators at participating institutions in order to encourage incorporation of those materials into existing curricula.

All of the materials produced by and for this project will be made freely available under the Creative Commons – Attribution (CC-BY) license. The instructional designer will work with the Mozilla Science Lab and affiliated groups to share these materials, and the results of our studies of the program's impact, through science education journals, conferences, and other channels.

As a subordinate part of their work, the researcher employed by this project will assess the extent to which curriculum developed during this program is taken up by other educators (particularly those who think of themselves as scientists first and computationalists second), and their perception of its utility. Mid-point results of this evaluation will be shared with the instructional designer in order to allow evidence-based improvement of the materials.

5 Broader Impact

We believe this work will have significant impact in several related areas.

1. *Enhance economic competitiveness.* Computing is no longer optional in any part of science: even scientists who don't think of themselves as doing computational work rely on computers to prepare papers, store data, and collaborate with colleagues. The better their computing skills are, the better able they will be to contribute to the research that underpins the nation's economic competitiveness.

2. *Improving STEM education for everyone, not just participants.* By creating and validating high-quality open access teaching materials, and the methods used to deliver them, this project will enable improvement in STEM education for everyone, everywhere, not just for participating students and participating institutions.

3. *Improving STEM education tomorrow, not just today.* As noted in the introduction of this proposal, most of today's efforts to transfer computational skills to STEM researchers and connect them with 21st Century innovations in how science is done are flying blind: there is effectively no feedback from long-term impact to instructional action. By creating and validating such a feedback loop—i.e., by showing scientists how to apply science to their teaching—this project will demonstrate how STEM education can be continuously improved.

4. *Improve participation in STEM by women and under-represented minorities.* The disproportionately low participation of women and some minority groups in STEM is well documented, as is the fact that computing is one of the least diverse fields within STEM. This second fact creates a vicious circle: people with weaker computing skills may be less competitive in research than their peers, which reduces their participation in activities viewed as non-core, which in turn results in them having weaker skills. This project will strive to break this cycle by giving at-risk students an opportunity to "level up" in a supportive environment, and by connecting them with mentors who can serve as role models.

6 Career Management Plan

The graduate students who are serving as mentors for the undergraduates at the different universities will each be paired with a local faculty mentor. The faculty mentor will meet regularly with the graduate student to discuss and problem solve any issues that the graduate student or undergraduates

are having, and to provide active mentoring on how to train students in computational approaches.

In addition to engaging with the graduate students on their mentoring of the undergraduates, the faculty mentors will also serve as a mentors for computational aspects of the graduate students' research and careers. In many areas of science, computationally-minded students are located in labs where the PIs do not have strong computational backgrounds. This means that they do not have a mentor to teach them about good computational practice in research. In addition, they do not have someone to discuss computational careers with, thus limiting their exposure to career paths outside of academia. Because the faculty mentors will have strong computational backgrounds, themselves they can fill this void for computationally-minded students.

7 Data Management Plan

FIXME (TT): Talk about curating our data.

References

[1] Software carpentry web site.

[2] P. F. Dubois. Maintaining Correctness in Scientific Programs. *Computing in Science & Engineering*, 7(3):80–85, May-June 2005.

[3] Mark Guzdial. Why is it so hard to learn to program? In Andy Oram and Greg Wilson, editors, *Making Software: What Really Works, and Why We Believe It*, pages 111–124. O'Reilly Media, 2010.

[4] Jo Erskine Hannay, Hans Petter Langtangen, Carolyn MacLeod, Dietmar Pfahl, Janice Singer, and Greg Wilson. How Do Scientists Develop and Use Scientific Software? In *Second International Workshop on Software Engineering for Computational Science and Engineering (SECSE09)*, 2009.

[5] Orit Hazzan, Tami Lapidot, and Noa Ragonis. *Guide to Teaching Computer Science: An Activity-Based Approach.* Springer, 2011.

[6] L. Hochstein, J. Carver, F. Shull, S. Asgari, V. R. Basili, J. Hollingsworth, and M. Zelkowitz. Parallel Programmer Productiv-

ity: A Case Study of Novice HPC Programmers. In *Proceedings of Supercomputing 2005 (SC05)*, 2005.

[7] Prakash Prabhu, Thomas B. Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick P. Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David I. August. A Survey of the Practice of Computational Science. In *Proceedings of the 24th ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, 2011.

[8] Judith Segal. When Software Engineers Met Research Scientists: A Case Study. *Empirical Software Engineering*, 10(4):517–536, 2005.

[9] Juha Sorva. *Visual Program Simulation in Introductory Programming Education*. PhD thesis, 2012.

[10] Greg Wilson. Software carpentry: Lessons learned, 2012.

[11] Greg Wilson, D.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. Best practices for scientific computing. *PLoS Biology*, 12(1):e1001745, January 2014.