PROJECT DESCRIPTION

The Impact of Intensive Software Skills Training on Students' Scientific Careers

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

1 The Problem: Elusive Computational Competence in Science

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.

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, making 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.

Studies have repeatedly shown that most researchers learn what they know about computing by word of mouth [6], but this approach is failing to meet present-day needs: most faculty would agree that today's graduates are no more able to use computing and the web *in their research* than they were a generation ago. Attempts to integrate more training in basic computing skills into undergraduate education have largely failed to take root 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. *Cultural difference.* Scientists and software developers have different priorities and different approaches to problem solving, which often impedes collaboration and knowledge transfer [15].

One final 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 for this new kind of 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.

2 Proposed Work: Leveraging Proven Curriculum to Promote Computational Competence in Science

This proposal builds on the success to date of the Software Carpentry workshops (Section 3.3), a proven curriculum of essential software skills that enhance the productivity of graduate students, post-docs, and faculty. We propose 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 widely.

We will run software-skills workshops for undergraduate students taking part each year in summer research opportunities such as the NSF's Research Experience for Undergraduates (REU) program, at or near the start of those students' time in the lab. We expect this training will help them be more productive during their research (graduate-level participants in our existing workshops typically report that what we teach saves them a day per week), and also prepare them to work in a world where all aspects of science are increasingly dependent on computing.

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. In order to conduct this study, we will hire an expert in educational assessment, whose full-time work for the duration of the project will be to explore the effects of the training on workshop participants.

2.1 Workshops: A Distributed Model for National Impact

We will run two-day workshops at a steadily increasing number of sites each year for five years, timed to coincide with the start of the summer influx of undergraduate research students. Each workshop will be offered to a minimum of 40 learners per site (giving us a target study population of at least 2200 students by year 5). 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. By design, it is straightforward to adapt workshop materials and contribute changes back to the Software Carpentry organization. These features enhance the portability and flexibility of the workshops, and increase the likelihood of wide dissemination beyond this project.

The home sites for investigators named in this proposal (George Washington University, Michigan State University, University of California – Berkeley, and University of Wisconsin – Madison, and Utah State University) will run workshops in each of those years. Other sites will be added each year, increasing the total to 15 in year 5, which increases the size of our study population. We will focus expansion on NSF REU sites [11], but as detailed in Section 2.3, we will also offer some workshops to other communities.

One set of possible sites for expansion are those campuses included in the "Condo of Condos" consortium, recently recommended for funding by the National Science Foundation. The Software

Carpentry workshops proposed here will be valuable to that consortium in meeting its goals of increasing the number and diversity of researchers using advanced cyberinfrastructure, and of developing data science practitioners.

2.2 Curriculum: From Tools to Techniques to Concepts

While there is considerable scope for customizing workshops to accommodate learners' prior experience and discipline-specific needs, a typical workshop devotes roughly half a day to each of these four topics:

- the Unix shell;
- version control with Git;
- programming in Python or R; and
- using SQL with databases.

What these workshops actually seek to convey, though, is the best practices a researcher needs to be *computationally competent* [19]:

- how to automate repetitive tasks;
- how to track and share work over the web;
- how to grow a program in a modular, testable, reusable way; and
- how to create, use, and share structured data.

All of the workshop instructors will have been trained and certified by Software Carpentry, and will have had experience teaching this material prior to engaging in these particular workshops. Just as importantly, all instructors will themselves be working scientists who use these skills and concepts daily in their own research, and are therefore able both to serve as role models and to deal with unanticipated questions or challenges based on personal experience (e.g., [14]).

2.3 Increasing Diversity: Changing the Odds for Underrepresented Scientists

In order to increase the diversity of the study population, at least one workshop in each year will be aimed specifically 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 workshops in years 2 through 5 specifically aimed at students from minority groups that are underrepresented 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 College) in early 2014, and we expect to have significantly expanded these efforts by year two of this project.

2.4 Formative and Summative Assessment to Maximize Learning and Impact

We will employ an expert in educational assessment full-time for five years to monitor and compare undergraduate participants in these software carpentry skills building workshops, participants in a subset of our regular (graduate-level) workshops, and non-participants (as a control population). As part of their work, this person will be responsible not only for collecting and analyzing data, but also for refining and extending the methods and measures used to gauge impact.

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 computationally 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. Do outcomes differ between women and underrepresented minorities on one hand and non-underrepresented minorities and men on the other? If so, in what ways, and what steps are effective in correcting for these differences?
6. 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 Software Carpentry's work to date (Section 3.3), 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, and any new methods or measures developed, will be shared with other researchers through publication in peer-reviewed journals and high-profile conferences (Section 2.5).

2.5 Community Building: Supporting Computational Competence

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 (Section 3.4). 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 to help them 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 the local graduate student 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/or on their engagement with open and web-enabled science in particular.

While it will not be feasible to bring all of the students participating in a given year's workshops together physically, we will organize and run virtual conferences toward the end of their research term to give them an opportunity to present their work to one another, discuss what they have learned, and build peer-to-peer connections. These conferences will also provide an opportunity to introduce participants to new forms of scientific "publishing", including blogging, the creation of screencasts and demonstration videos, and other methods that may not yet exist.

2.6 Curriculum Development and Dissemination: Expanding the Impact

We will employ one instructional designer part-time throughout the project 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. Specifically, we plan to publish articles in:

- *Science Education*
- *Physics Education*
- *Journal of Computers in Mathematics and Science Teaching*
- *American Scientist*

and at the following conferences:

- the Special Interest Group on Computer Science Education (SIGCSE) of the Association for

Computing Machinery, Inc. (ACM)
- the International Conference on Physics Education (ICPE)
- the Association for Biology Laboratory Education (ABLE)
- the Society for Industrial and Applied Mathematics (SIAM)'s education session at the Joint Mathematics Meeting
- the Education Training and Workforce Development Division's track at the American Nuclear Society (ANS)'s annual meeting

The dissemination of this project's curriculum has strong potential to be high. Workshop materials are all open access and flexible, thus they can be readily adopted by others. Adapting workshop materials is low cost and does not require special equipment. Workshop materials are structured such that they can scale to the size and application of interest to a particular group. The Software Carpentry infrastructure provides support in the form of materials and people. Anyone using workshop materials can directly contribute changes and feedback, which both increases buy-in and improves the materials organically. And finally, local chapters of The Hacker Within create a natural ecosystem of support for workshop participants, their peers, and faculty.

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.

2.7 Project Management

Prof. Slaybaugh will be responsible for overall project management and reporting. The educational assessment expert hired by this project will report directly to her. Prof. Slaybaugh will be assisted by Dr. Huff, who will manage and coordinate the graduate student assistants at each site. Dr. Huff will also be responsible for organizing the workshops aimed at female students.

Profs. Barba, Brown, White, and P. Wilson will be responsible for coordinating workshops and for recruiting and supervising the graduate student assistant at their respective institutions.

Dr. G. Wilson and the half-time instructional designer hired by this project will be responsible for preparation and publication of learning materials. Dr. G. Wilson will also provide instructional training for the graduate student assistants and other participants in the project on an ongoing basis, and will be responsible for organizing the workshops aimed at students from underrepresented minorities.

Workshop operations (such as finding instructors and arranging their travel) will be handled by Mozilla staff who are performing these duties for the Software Carpentry program more generally. These staff will be supervised by Ms. Thaney, who will also be responsible for connecting the other PIs and the graduate student assistants with other open and web-enabled science groups.

3 Related Work

3.1 Theoretical Positioning

Our theory of action is straightforward: if students are explicitly taught software skills in a way that makes them seem both useful and important, they are likely to begin using them in day-to-day work, which will create a positive feedback cycle leading to them acquire more (and more advanced) skills on their own. This positive feedback cycle will in turn result in the students being more likely to engage in open and web-enabled scientific activities that would otherwise have been unknown, incomprehensible, or out of reach.

Using the terminology of [10], our work is primarily *design and development research*. We plan to design and develop solutions related to student engagement and mastery of specific skills, drawing on existing evidence from Software Carpentry's graduate-level workshops, and investigating their impact and effectiveness. Further, we further plan to design and iteratively develop interventions. We are ready to begin collecting data on the feasibility of implementing solutions in typical delivery settings.

3.2 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 haphazardly and through word of mouth [6, 13]. 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 [3], or the other foundational skills laid out in [19]; and/or
- jumps to advanced topics such as parallel computing before scientists have mastered the foundations. (Most research on scientific computing, such as [8], does the same.)

On the other hand, studies of how people in general learn to program, and of how effective different approaches to teaching them are, have made significant strides in the past decade. In particular, our work is informed by the long-running research program of Guzdial et al. at Georgia Tech, who have found that a "media first" introduction to computing outperforms more conventional alternatives [5], and that it is possible to assess the extent to which programming concepts, rather than merely the syntax of a particular programming language, have been mastered [17].

Others (e.g., [12]) have demonstrated that peer instruction is a significantly better way to teach introductory programming than conventional classroom approaches. As discussed in the section below, we are already working to incorporate these evidence-based approaches into our teaching, and will accelerate these efforts within the scope of this award.

3.3 *Software Carpentry*

Software Carpentry [2, 18] is the largest effort to date to address issues surrounding inadequate software carpentry skill training for students. 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.

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 [1] introduces participants to a variety of modern teaching techniques (e.g., peer instruction, active learning, and understanding by design), to concepts underlying these techniques (e.g., cognitive load theory), and to topic-specific work by computing education researchers (see [4], [7], and the first third of [16] for overviews.) One example of how they translate theory into practice is their 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 funding 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).

Dr. Aranda surveyed and interviewed participants, observed a workshop, and analyzed screencasts of participants working through a programming assignment. He found significant increases in participants' understanding and use of shell commands, version control tools, Python, and testing techniques. Perhaps more importantly, participants reported better proficiency with software tools; greater concern for issues of provenance and code quality; better strategies to approach software development; and new research questions that have become accessible thanks to an increase in participants' software development skills.

Prof. Libarkin performed a more detailed assessment of participants in a workshop held there, which was attended remotely by students from the University of Texas at Austin. Eighty-five percent of participants reported that they learned what they hoped to learn, 81% changed their computational understanding, and 96% said they would recommend the workshop to others.

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.

3.4 *The Hacker Within*

The Hacker Within (THW) was founded by graduate students in nuclear engineering at the University of Wisconsin – Madison to provide a forum for sharing scientific computing skills and best practices with their peers [9]. As THW matured as a student organization, it attracted students from many scientific disciplines and academic levels. THW conducted bi-weekly seminars, and developed a series of short courses addressing the programming languages C++, Python, and Fortran; best practices such as version control and test-driven code development; and basic skills such as UNIX mobility. This curriculum was delivered primarily as interactive short workshops

on campuses and during scientific conferences. Many previous founders of the Hacker Within have since become instructors with Software Carpentry, and a new generation of THW graduate students has begun to emerge in their place. In 2013, new branches of THW were initiated at the University of Melbourne and the University of California - Berkeley.

3.5 Condo of Condos

The "Condo of Condos" consortium, led by Clemson University and including the University of Wisconsin – Madison and four other campuses during its pilot phase, has recently been recommended for funding by the National Science Foundation. The consortium's primary task is to build a network of advanced cyberinfrastructure research and education facilitors (ACI-REFs), with goals that include increasing the diversity of researchers using advanced cyberinfrastructure on each campus and developing data science practitioners. The Software Carpentry workshops being designed under this proposal will serve those goals directly. As an investigator on both that project and this proposal, Prof. P. Wilson will engage the network of ACI-REFs to share this curriculum with both the initial consortium institutions, and any institutions who are able to join in the future.

4 Broader Impact

We believe this work will have significant impact in several areas beyond directly improving the computational science skills of workshop participants.

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 prepared 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 underrepresented 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 cycle: 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.

5 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 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 with whom to discuss computational careers, 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.

6 Results from Prior NSF Support

6.1 Titus Brown

- Award number: NSF 09-23812 (PI Brown)
- Total Award Amount: $50,000
- Starting Date: 01/01/10; Ending Date: 12/31/13
- Project title: Symbiont Separation and Investigation of the Novel Heterotrophic Osedax Symbiosis using Comparative Genomics
- Summary of results: This project was a collaborative project with Dr. Shana Goffredi at Occidental College, where we analyzed sequence from MDA-amplified metagenomic samples taken from an Osedax bone eating worm. We produced the first high-quality assemblies of two symbiont genomes, giving us a whole-genome perspective on symbiont metabolism.
- Publications: ISME J in November 2013 (PubMed ID 24225886)

6.2 Lorena Barba

To be added.

References Cited

[1] Software Carpentry Instructor Training web site. http://teaching.software-carpentry.org. Accessed: 2014-01-23.

[2] Software Carpentry web site. http://software-carpentry.org. Accessed: 2014-01-23.

[3] P. F. Dubois. Maintaining correctness in scientific programs. *Computing in Science & Engineering*, 7(3):80–85, May-June 2005.

[4] 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.

[5] Mark Guzdial. Exploring hypotheses about media computation. In *Proc. Ninth Annual International ACM Conference on International Computing Education Research*, ICER'13, pages 19–26. ACM, 2013.

[6] 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.

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

[8] L. Hochstein, J. Carver, F. Shull, S. Asgari, V. R. Basili, J. Hollingsworth, and M. Zelkowitz. Parallel programmer productivity: A case study of novice HPC programmers. In *Proceedings of Supercomputing 2005 (SC05)*, 2005.

[9] Kathryn Huff, A.M. Scopatz, N.D. Preston, and P.P.H. Wilson. Rapid peer education of a computational nuclear engineering skill suite. In *Transactions of the American Nuclear Society*, volume 104, pages 103–104. American Nuclear Society, June 2011.

[10] Institute of Education Sciences, U.S. Dept. of Education, and the National Science Foundation. Common guidelines for education research and development. http://www.nsf.gov/pubs/2013/nsf13126/nsf13126.pdf, August 2013. Accessed: 2014-01-23.

[11] NSF. Search results for REU sites. http://www.nsf.gov/crssprgm/reu/list_result.jsp. Accessed: 2014-01-23.

[12] Leo Porter, Mark Guzdial, Charlie McDowell, and Beth Simon. Success in introductory programming: What works? *Communications of the ACM*, 56(8), 2013.

[13] 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.

[14] Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source Code for Biology and Medicine*, 8(1):7, 2013.

[15] Judith Segal. When software engineers met research scientists: A case study. *Empirical Software Engineering*, 10(4):517–536, 2005.

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

[17] Allison Elliott Tew and Mark Guzdial. The FCS1: A language independent assessment of CS1 knowledge. In *Proc. 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE'11, pages 111–116. ACM, 2011.

[18] Greg Wilson. Software Carpentry: Lessons learned. *pre-print*, 2013. arXiv:cs.GL/1307.5448.

[19] 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.