Setting the Default to Reproducible

Reproducibility in Computational and Experimental Mathematics

Developed collaboratively by the ICERM workshop participants[1]

Compiled and edited by the Organizers

V. Stodden, D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider, and W. Stein

Abstract

Science is built upon foundations of theory and experiment validated and improved through open, transparent communication. With the increasingly central role of computation in scientific discovery this means communicating all details of the computations needed for others to replicate the experiment, i.e. making available to others the associated data and code. The "reproducible research" movement recognizes that traditional scientific research and publication practices now fall short of this ideal, and encourages all those involved in the production of computational science – scientists who use computational methods and the institutions that employ them, journals and dissemination mechanisms, and funding agencies – to facilitate and practice really reproducible research.

This report summarizes discussions that took place during the ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, held December 10-14, 2012. The main recommendations that emerged from the workshop discussions are:

1. It is important to promote a culture change that will integrate computational reproducibility into the research process.
2. Journals, funding agencies, and employers should support this culture change.
3. Reproducible research practices and the use of appropriate tools should be taught as standard operating procedure in relation to computational aspects of research.

The workshop discussions included presentations of a number of the diverse and rapidly growing set of software tools available to aid in this effort. We call for a broad implementation of these three recommendations across the computational sciences.

Introduction

The emergence of powerful computational hardware, combined with a vast array of computational software, presents unprecedented opportunities for researchers in mathematics and science. Computing is rapidly becoming the backbone of both theory and experiment, and essential in data analysis, interpretation, and inference.

Unfortunately the scientific culture surrounding computational work has evolved in ways that often make it difficult to verify findings, efficiently build on past research, or even to apply the basic tenets of the scientific method to computational procedures. Bench scientists are taught to keep careful lab notebooks documenting all aspects of the materials and

[1] For a list of participants see Appendix H or the workshop webpage http://icerm.brown.edu/tw12-5-rcem.

Version of February 2, 2013.

methods they use including their negative as well as positive results, but computational work is often done in a much less careful, transparent, or well-documented manner. Often there is no record of the workflow process or the code actually used to obtain the published results, let alone a record of the false starts. This ultimately has a detrimental effect on researchers' own productivity, their ability to build on past results or participate in community efforts, and the credibility of the research among other scientists and the public [6].

There is increasing concern with the current state of affairs in computational science and mathematics, and growing interest in the idea that doing things differently can have a host of positive benefits that will more than make up for the effort required to learn new work habits. This research paradigm is often summarized in the computational community by the phrase "reproducible research." Recent interest and improvements in computational power have led to a host of new tools developed to assist in this process. At the same time there is growing recognition among funding agencies, policy makers, and the editorial boards of scientific journals of the need to support and encourage this movement. [2]. A number of workshops have recently been held on related topics, including a Roundtable at Yale Law School [14] a workshop as part of the Applied Mathematics Perspectives 2011 conference [2, 7], and several minisymposia at other conferences, including SIAM Conferences on Computational Science and Engineering 2011 and ICIAM 2011.

The ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, held December 10-14, 2012, provided the first opportunity for a broad cross section of computational scientists and mathematicians, including pure mathematicians who focus on experimental mathematics or computer-assisted proofs, to discuss these issues and brainstorm ways to improve on current practices. The first two days of the workshop focused on introducing the themes of the meeting and discussing policy and cultural issues. In addition to introductory talks and open discussion periods, there were panels on funding agency policies and on journal and publication policies. The final three days featured many talks on software tools that help achieve reproducibility and other more technical topics in the mornings. Afternoons were devoted to breakout groups discussing specific topics in more depth, which resulted in recommendations and other outcomes. Breakout group topics included: reproducibility tools, funding policies, publication policies, numerical reproducibility, taxonomy of terms, reward structure and cultural issues, and teaching reproducible research techniques. [3] We also held a tutorial on version control the day before the official start of the workshop. [4]

Both in the workshop and in this report the terms "reproducible research" and "reproducibility" most often refer to the ability to recreate computational results from the data and code used by the original researcher [11]. This is related to but distinct from both the notions of "numerical reproducibility" of computational results, referring to when the same program may give different results due to hardware or compiler issues, particular in the context of parallel computing, and "repeatability," when an experiment is conducted independently from first principles. A taxonomy of reproducibility concepts is developed in Appendix A and a discussion of numerical reproducibility appears in Appendix B.

[2] See National Science Foundation Data Management Plan http://, ACM Publications Policy http://

[3] See the workshop program at http://icerm.brown.edu/tw12-5-rcem.

[4] http://icerm.brown.edu/tw12-5-rcem-tutorial.

About this document

This document reports on the three main recommendations emerging from the workshop discussions:

1. It is important to promote a culture change that will integrate computational reproducibility into the research process.
2. Journals, funding agencies, and employers should support this culture change.
3. Reproducible research practices and the use of appropriate tools should be taught as standard operating procedure in relation to computational aspects of research.

The recommendations are each discussed in turn in the three sections of this document, and we include five appendices that develop important topics in further detail. Besides the appendices mentioned above on taxonomy and numerical reproducibility, there are appendices on best practices for publishing research, the state of reproducibility in experimental mathematics, and tools to aid in reproducible research. An initial draft of this document was presented to participants and discussed on the final day of the workshop, and participants were able to give input on the final draft before submission.

In addition to this document, a number of other products emerged from the workshop. Video of the talks is available at http://icerm.brown.edu/video_archive, and numerous topical references were collected on the workshop wiki [5]. The workshop webpage and the wiki also contain participant thought pieces, slides from the talks, and breakout group reports. Readers are invited to contribute to the wiki. A snapshot of the wiki is appended at the end of the report as Figure 1.

1. Changing the Culture and Reward Structure

For reproducibility to be fostered and maintained, workshop participants agreed that cultural changes need to take place within the field of computationally based research that instill the open and transparent communication of results as a default. Such a mode will increase productivity — less time wasted in trying to recover output that was lost or misplaced, less time wasted trying to double-check results in the manuscript with computational output, and less time wasted trying to determine whether other published results (or even their own) are truly reliable. Open access to any data used in the research and to both primary and auxiliary source code also provides the basis for research to be communicated transparently creating the opportunity to build upon previous work, in a similar spirit as open software provided the basis for Linux. Code and data should be made available under open licensing terms as discussed in Appendix F. [9] This practice enables researchers both to benefit more deeply from the creative energies of the global community and to participate more fully in it. Most great science is built upon the discoveries of preceding generations and open access to the data and code associated with published computational science allows this tradition to continue. Researchers should be encouraged to recognize the potential benefits of openness and reproducibility.

[5] Available at http://is.gd/RRlinks, see Figure 1.

It is also important to recognize that there are costs and barriers to shifting to a practice of reproducible research, particularly when the culture does not recognize the value of developing this new paradigm or the effort that can be required to develop or learn to use suitable tools. This is of particular concern to young people who need to earn tenure or secure a permanent position. To encourage more movement towards openness and reproducibility, it is crucial that such work be acknowledged and rewarded. The current system, which places a great deal of emphasis on the number of journal publications and virtually none on reproducibility (and often too little on related computational issues such as verification and validation), penalizes authors who spend extra time on a publication rather than doing the minimum required to meet current community standards. Appropriate credit should given for code and data contributions including an expectation of citation. Another suggestion is to instantiate yearly award from journals and/or professional societies, to be awarded to investigators for excellent reproducible practice. Such awards are highly motivating to young researchers in particular, and potentially could result in a sea change in attitudes. These awards could also be cross-conference and journal awards; the collected list of award recipients would both increase the visibility of researchers following good practices and provide examples for others.

More generally, it is unfortunate that software development and data curation are often discounted in the scientific community, and programming is treated as something to spend as little time on as possible. Serious scientists are not expected to carefully test code, let alone document it, in the same way they are trained to properly use other tools or document their experiments. It has been said in some quarters that writing a large piece of software is akin to building infrastructure such as a telescope rather than a creditable scientific contribution, and not worthy of tenure or comparable status at a research laboratory. This attitude must change if we are to encourage young researchers to specialize in computing skills that are essential for the future of mathematical and scientific research. We believe the more proper analog to a large scale scientific instrument is a supercomputer, whereas software reflects the intellectual engine that makes the supercomputers useful, and has scientific value beyond the hardware itself. Important computational results, accompanied by verification, validation, and reproducibility, should be accorded with honors similar to a strong publication record [7].

Several tools were presented at the workshop that enable users to write and publish documents that integrate the text and figures seen in reports with code and data used to generate both text and graphical results, such as IPython, Sage notebooks, Lepton, knitr, and Vistrails. Slides for these talks are available on the wiki [1] and Appendix E discusses these and other tools in detail.

The following two sections and the appendices outline ideas from the workshop on ways in which journals, funding agencies, and employers can support reproducibility.

2. Funding Agencies, Journals, Employers Should Support This Change

Incentives in scholarly research are influenced by three main sources, the funding agency, dissemination processes such as journals, and employers such as those on tenure committees and lab managers. The workshop discussions mentioned the role of each of them in shifting to a culture of reproducible computational research.

The Role of Funding Agency Policy

Workshop participants suggested that funding sources, both government agencies and private foundations, consider establishing some reasonable standards for proposals in the arena of mathematical and computational science. If such standards become common among related agencies this would significantly simplify the tasks involved in both preparing and reviewing proposals, as well as supporting a culture change toward reproducible computational research.

For example, workshop participants recommend that software and data be "open by default" unless it conflicts with other considerations. Proposals involving computational work might be required to provide details such as:

- Extent of computational work to be performed.
- Platforms and software to be utilized.
- Reasonable standards for dataset and software documentation, including reuse (some agencies already have such requirements [8]).
- Reasonable standards for persistence of resulting software and dataset preservation and archiving.
- Reasonable standards for sharing resulting software among reviewers and other researchers.

In addition, we suggest that funding agencies might add "reproducible research" to the list of specific examples that proposals could include in their requirements such as "Broader Impact" statements. Software and dataset curation should be explicitly included in grant proposals and recognized as a scientific contribution by funding agencies. Templates for data management plans could be made available that include making software open and available, perhaps by funding agencies, or by institutional archiving and library centers. [6]

Participants also suggested that statements from societies and others on the importance of reproducibility could advance the culture change. In addition, funding agencies could provide support for training workshops on reproducibility, and cyberinfrastructure for reproducibility at scale, for both large projects and long-tail research efforts. Funding agencies are key to the promotion of a culture that embraces reproducible research, due to their central importance in the research process. We turn to journals next, and then employers.

The Role of Journal Policy

There is a need to produce a set of "best practices" for publication of computational results i.e. any scientific results in which computation plays a role, for example in empirical research, statistical analysis, or image processing. We recommend that a group representing several professional societies in the mathematical sciences be formed to develop a set of best practices for publication of research results. Such guidelines would be useful

[6] For examples see http://scholcomm.columbia.edu/data-management/data-management-plan-templates/, http://www2.lib.virginia.edu/brown/data/NSFDMP.html

to the editorial boards of many journals, as well as to authors, editors, and referees who are concerned about promoting reproducibility. Best practices may be tailored to different communities, but one central concern, for which there was almost unanimous agreement by the workshop participants, is the need for full disclosure of salient details regarding software and data use. This should include specification of the dataset used (including URL and version), details of the algorithms employed (or references), the hardware and software environment, the testing performed, etc., and would ideally include availability of the relevant computer code with a reasonable level of documentation and instructions for repeating the computations performed to obtain the results in the paper. [7].

There is also a need for better standards on how to include citations for software and data in the references of a paper, instead of inline or as footnotes. Proper citation is essential both for improving reproducibility and in order to provide credit for work done developing software and producing data, which is a key component in encouraging the desired culture change [7].

Workshop participants agreed that it is important that a set of standards for reviewing papers in the computational arena be established. Such a set of standards might include many or all of the items from a "best practices" list, together with a rational procedure for allowing exceptions or exclusions. Additionally, provisions are needed to permit referees to obtain access to auxiliary information such as computer codes or data, and the ability to run computational tests of the results in submitted papers, if desired.

Different journals may well adopt somewhat different standards of code and data disclosure and review [12], but it is important that certain minimal standards of reproducibility and rigor be maintained in all refereed journal publications. Along these lines, it may be desirable for the computational claims of a manuscript to be verifiable at another site such as RunMyCode.org, or on another computer system with a similar configuration.

Some related issues in this arena include: (a) anonymous versus public review, (b) persistence (longevity) of code and data that is made publicly available, and (c) how code and data can be "watermarked," so that instances of uncited usage (plagiarism) can be detected and provenance better established (d) how to adjudicate disagreements that inevitably will arise.

Very rigorous verification and validity testing, along with a full disclosure of computational details, should be required of papers making important assertions, such as the computer-assisted proof of a long-standing mathematical result, new scientific breakthroughs, or studies that will be the basis for critical policy decisions [13].

Proper consideration of openness constraints can enable a larger community to participate in the goals of reproducible research. This can include issues such as copyright, patent, medical privacy, personal privacy, security, and export issues. This is discussed further in Appendix F.

It was recognized that including such details in submitted manuscripts (or, at the least, in supplementary materials hosted by the journal) is a significant departure from established practice, where few such details are typically presented. But these changes will be required if the integrity of the computational literature is to be maintained. Computational approaches have become central to science and cannot be completely documented and

[7] See for example http://software.ac.uk/so-exactly-what-software-did-you-use

transparent without the full disclosure of computational details. Appendix D contains the full list of workshop suggestions.

The Role of Employers and Research Managers

The third source of influence on the research process stems from employers – tenure and promotion committees and research managers at research labs. Software and dataset contributions, as described in the previous two subsections, should be rewarded as part of expected research practices. Data and code citation practices should be recognized and expected in computational research. Prizes for reproducible research should also be recognized in tenure and promotion decisions.

Institutional libraries can also play a role in supporting a culture change toward reproducible research. As mentioned above, they can and do provide template data management plans, but they are also highly experienced in archiving, stewardship and dissemination of scholarly objects. Greater coordination between departments and the institute's library system could help provide the support and resources necessary to manage and maintain digital scholarly output, including datasets and code [4].

3. Teaching Reproducibility Skills

Proficiency in the skills required to carry out reproducible research in the computational sciences should be taught as part of the scientific methodology, along with teaching modern programming and software engineering techniques. This should be a standard part of any computational research curriculum, just as experimental or observational scientists are taught to keep a laboratory notebook and follow the scientific method. Adopting appropriate tools (see Appendix E) should be encouraged, if not formally taught, during the training and mentoring of students and postdoctoral fellows. Without a change in culture and expectations at this stage, reproducibility will likely never enter the mainstream of mathematical and scientific computing.

We see at least five separate ways in which these skills can be taught: full academic courses, incorporation into existing courses, workshops and summer schools, online courses or self-study materials, and last but certainly not least, teaching-by-example on the part of mentors.

Although a few full-scale courses on reproducibility have been attempted (see the wiki for links), we recognize that adding a new course to the curriculum or the students' schedules is generally not feasible. It seems more effective as well as more feasible to incorporate teaching the tools and culture of reproducibility into existing courses on various subjects, concentrating on the tools most appropriate for the domain of application. For example, several workshop participants have taught classes in which version control is briefly introduced and then students are required to submit homework by pushing to a version control repository as a means of encouraging this habit.

A list of potential curriculum topics on reproducibility are listed in Appendix G. Ideally, courseware produced at one institution should be shared with others under an appropriate open license.

Conclusion

The goal of computational reproducibility is to provide a solid foundation to computational science, much like a rigorous proof is the foundation of mathematics. Such a foundation permits the transfer of knowledge that can be understood, implemented, evaluated, and used by others. This reports discusses the efforts of participants and organizers of the ICERM workshop on "Reproducibility in Computational and Experimental Mathematics" to formulate steps toward the ideal of reproducible computational research. We identified three key recommendations emerging from workshop discussions, calling for a culture change toward reproducible research, mapping roles for funding agencies, journals, and employers to support this change, and emphasizing that methods and best practices for reproducible research must be taught. We also include detailed appendices on related issues that arose in the workshop discussions, including a taxonomy of terms, numerical reproducibility, best practices for publishing reproducible research, a summary of the state of experimental mathematics, and tools to aid in reproducible research. To capture the phenomenal level of engagement by workshop participants, we collate further information, including their talk slides, thought pieces, and further references on the workshop wiki.

References

[1] Applied mathematics perspectives 2011 workshop on reproducible research: Tools and strategies for scientific computing. http://wiki.stodden.net/AMP2011/, 2012.

[2] Applied mathematics perspectives 2011 workshop on reproducible research: Tools and strategies for scientific computing. http://stodden.net/AMP2011/, 2009.

[3] David H. Bailey, Roberto Barrio, and Jonathan M. Borwein. High precision computation: Mathematical physics and dynamics. *Applied Mathematics and Computation*, 218:10106–10121, 2012.

[4] Christine Borgman. Research data, reproducibility, and curation. In *Digital Social Research: A Forum for Policy and Practice, Oxford Internet Institute Invitational Symposium*, 2012.

[5] Jonathan M. Borwein and David H. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A K Peters (Taylor and Francis), Natick, MA, 2008.

[6] David Donoho, Arian Maleki, Morteza Shahram, Victoria Stodden, and Inam Ur Rahman. Reproducible research in computational harmonic analysis. *Computing in Science and Engineering*, 11, January 2009.

[7] Randall LeVeque, Ian Mitchell, and Victoria Stodden. Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science and Engineering*, pages 13–17, July 2012.

[8] Brian Matthews, Brian McIlwrath, David Giaretta, and Ester Conway. The significant properties of software: A study. http://www.jisc.ac.uk/media/documents/programmes/preservation/spsoftware_report_redacted.pdf, 2008.

[9] Victoria Stodden. Enabling reproducible research: Licensing for scientific innovation. *International Journal of Communications Law and Policy*, pages 1–25, 2009.

[10] Victoria Stodden. The legal framework for reproducible scientific research. *Computing in Science and Engineering*, 11, January 2009.

[11] Victoria Stodden. Trust your science? open your data and code. *Amstat News*, 2011.

[12] Victoria Stodden, Peixuan Guo, and Zhaokun Ma. How journals are adopting open data and code policies. In *The First Global Thematic IASC Conference on the Knowledge Commons: Governing Pooled Knowledge Resources*, 2012.

[13] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Steven H. D. Haddock Richard T. Guy, Katy Huff, Ian M. Mitchell, Mark Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing.

[14] Yale law school, data and code sharing roundtable. http://www.stanford.edu/~vcs/Conferences/RoundtableNov212009/, 2009.

Appendices

These appendices contain some additional material arising from workshop discussions. We have avoided including long lists of references in these appendices. Instead, many links have been collected and categorized on the workshop wiki, which can be referred to for more examples, additional tools, articles, editorials, etc.[8]

A Terminology and Taxonomy

The terms "reproducible research" and "reproducibility" are used in many different ways to encompass diverse aspects of the desire to make research based on computation more credible and extensible. Lively discussion over the course of the workshop has led to some suggestions for terminology, listed below. We encourage authors who use such terms in their work to clarify what they mean in order to avoid confusion.

There are several possible levels of reproducibility, and it seems valuable to distinguish between the following:

- *Reviewable Research.* The descriptions of the research methods can be independently assessed and the results judged credible. (This includes both traditional peer review and community review, and does not necessarily imply reproducibility.)
- *Replicable Research.* Tools are made available that would allow one to duplicate the results of the research, for example by running the authors' code to produce the plots shown in the publication. (Here tools might be limited in scope, e.g., only essential data or executables, and might only be made available to referees or only upon request.)
- *Confirmable Research.* The main conclusions of the research can be attained independently without the use of software provided by the author. (But using the complete description of algorithms and methodology provided in the publication and any supplementary materials.)
- *Auditable Research.* Sufficient records (including data and software) have been archived so that the research can be defended later if necessary or differences between independent confirmations resolved. The archive might be private, as with traditional laboratory notebooks.
- *Open or Reproducible Research.* Auditable research made openly available. This comprised well-documented and fully open code and data that are publicly available that would allow one to (a) fully audit the computational procedure, (b) replicate and also independently reproduce the results of the research, and (c) extend the results or apply the method to new problems.

Other terms that often arise in discussing reproducibility have specific meanings in computational science. In particular the widely-used acronym V&V (verification & valida-

[8] For the wiki see http://icerm.brown.edu/tw12-5-rcem-wiki.php or http://is.gd/RRlinks

tion) makes it difficult to use “verify” or “validate” more generally. These terms are often defined as follows:

- *Verification.* Checking that the computer code correctly solves the mathematical problem it claims to solve. (Does it solve the equation right?)
- *Validation.* Checking that the results of a computer simulation agree with experiments or observations of the phenomenon being studied. (Does it solve the right equation?)

The term “Uncertainty Quantification (UQ)” is also commonly used in computational science to refer to various approaches to assessing the effects of all of then uncertainties in data, models, and methods on the final result, which is often then viewed as a probability distribution on a space of possible results rather than a single answer. This is an important aspect of reproducibility in situations where exact duplication of results cannot be expected for various reasons.

The *provenance* of a computational result is a term borrowed from the art world, and refers to a complete record of the source of any raw data used, the computer programs or software packages employed, etc. The concept of provenance generally includes a record of changes that the dataset or software has undergone.

B Numerical Reproducibility

Numerical round-off error and numerical differences are greatly magnified as computational simulations are scaled up to run on highly parallel systems. As a result, it is increasingly difficult to determine whether a code has been correctly ported to a new system, because computational results quickly diverge from standard benchmark cases. And it is doubly difficult for other researchers, using independently written codes and distinct computer systems, to reproduce published results.

One solution is to utilize some form of higher precision arithmetic, such as Kahan’s summation or “double-double” arithmetic. In many cases, such higher precision arithmetic need only be used in global summations or other particularly sensitive operations, so that the overall runtime is not greatly increased. Such measures have dramatically increased reproducibility in various codes, ranging from climate simulations to computational physics applications [3].

But it is clear that this solution will not work for all applications. Other approaches include interval arithmetic (which potentially can provide provable bounds on final results), affine arithmetic (which works better than interval arithmetic in some cases), and also some proposed new tools, currently under development at U.C. Berkeley, that can pin down numerically sensitive sections of code and take corrective actions. in any event, additional study and research is in order. Certainly the available tools for high-precision computation need to be significantly refined so as to be usable and highly reliable for a wide range of users.

It is clear that these issues must be addressed with greater diligence by authors of all manuscripts presenting results of numeric computations. They must be more careful

to state exactly what levels of numeric precision (32-bit, 64-bit or higher precision) have been used, and to present evidence that their selected precision is adequate to achieve a reasonable level of reproducibility in their results.

One of the foundations of reproducibility is how to deal with (and set standards for) difficulties such as numerical round-off error and numerical differences when a code is run on different systems or different numbers of processors. Such difficulties are magnified as problems are scaled up to run on very large, highly parallel systems.

Computations on a parallel computer system present particularly acute difficulties for reproducibility since, in typical parallel usage, the number of processors may vary from run to run. Even if the same number of processors is used, computations may be split differently between them or combined in a different order. Since computer arithmetic is not commutative, associative, or distributive, achieving the same results twice can be a matter of luck. Similar challenges arise when porting a code from one hardware or software platform to another.

The IEEE Standards for computer arithmetic resulted in significant improvements in numerical reproducibility on single processors when they were introduced in the 1970s. Some work is underway on extending similar reproducibility to parallel computations, for example in the Intel Mathematics Kernel Library (MKL), which can use used to provide parallel reproducibility for mathematical computations.

Additional issues in this general arena include: (a) floating-point standards and whether they being adhered to on the platform in question, (b) changes that result from different levels of optimization, (c) changes that result from employing library software, (d) verification of results, and (e) fundamental limits of numerical reproducibility, what are reasonable expectations and what are not.

The foundation of numerical reproducibility is also grounded in the computing hardware and in the software stack. Studies on silent data corruption (SDC) have documented SDC in field testing, as discussed in some of the references on the wiki.

Field data on supercomputer DRAM memory failures have shown that advanced error correcting codes (ECC) are required and technology roadmaps suggest this problem will only get worse in the coming years. Designing software that can do some or all of identification, protection, and correction will become increasingly important. Still, there is much work being done to quantify the problem on current and next generation hardware and approaches to addressing it. Several United States and international governmental reports have been produced on the need for, outlining ongoing research in, and proscribing roadmaps.

These foundational components set a limit to the achievable reproducibility and make us aware that we must continually assess just how reproducible our methods really are.

C The State of Experimental Mathematics

Automatic theorem proving has now achieved some truly impressive results such as fully formalized proofs of the Four color theorem and the Prime number theorem. While such tools currently require great effort, one can anticipate a time in the distant future when all

truly consequential results are so validated.

The emerging discipline of experimental mathematics, namely the application of high-performance computing technology to explore research questions in pure and applied mathematics, raises numerous issues of computational reproducibility [5]. Experimental mathematics research often press the state-of-the-art in very high precision computation (often hundreds or thousands of digits), symbolic computation, graphics and parallel computation. There is a need to carefully document algorithms, implementation techniques, computer environments, experiments and results, much as with other forms of computation-based research. Even more emphasis needs to be placed on aspects of such research that are unique to this discipline: (a) Are numeric precision levels (often hundreds or even thousands of digits) adequate for the task at hand? (b) What independent consistency checks have been employed to validate the results? (c) If symbolic manipulation software was employed (e.g., Mathematica or Maple), exactly which version was used?[9] (c) Have numeric spot-checks been performed to check derived identities and other results? (d) Have symbolic manipulations been validated, say by using two different symbolic manipulation packages?

Such checks are often required, because even the best symbolic and numeric computation packages have bugs and limitations, which bugs are often only exhibited when doing state-of-the-art research computations. Workshop participants identified numerous instances of such errors in their work, underscoring the fact that one cannot place unquestioned trust in such results.

D Best Practices for Publishing Research

Publishing can take many forms – traditional journal publication is one avenue but other electronic options are increasingly being used. Traditional publications are also frequently complemented by "supplementary materials" posted on a journal's website or in other archival-quality data or code repositories.

A number of suggestions were made regarding best practices for publications of research results. To aid in reproducibility, the available materials should ideally contain:

- A precise statement of assertions to be made in the paper.
- A statement of the computational approach, and why it constitutes a rigorous test of the hypothesized assertions.
- Complete statements of, or references to, every algorithm employed.
- Salient details of auxiliary software (both research and commercial software) used in the computation.
- Salient details of the test environment, including hardware, system software and the number of processors utilized.

[9] Indeed, one needs to know which precise functions were called, with what parameter values and environmental settings?

- Salient details of data reduction and statistical analysis methods.
- Discussion of the adequacy of parameters such as precision level and grid resolution.
- Full statement (or at least a valid summary) of experimental results.
- Verification and validation tests performed by the author(s).
- Availability of computer code, input data and output data, with some reasonable level of documentation.
- Curation: where are code and data available? With what expected persistence and longevity? Is there a site for site for future updates, e.g. a version control repository of the code base?
- Instructions for repeating computational experiments described in the paper.
- Terms of use and licensing. Ideally code and data "default to open", i.e. a permissive re-use license, if nothing opposes it.
- Avenues of exploration examined throughout development, including information about negative findings.
- Proper citation of all code and data used, including that generated by the authors.

Several publications have adopted some requirements for reproducibility (e.g., Biostatistics, TOMS, IPOL, or conferences such as SIGMOD). In addition to those discussed in the main article, some other recommendations arose in discussions and break-out groups to change the culture in relation to reproducibility in publications. Journals or other publications could offer certifications of reproducibility that would kite-mark a paper satisfying certain requirements, as done by the journal Biostatistics, for example. Certification could also come from an independent entity such as RunMyCode.org. Journals could also create reproducible overlay issues for journals that collect together reproducible papers. Linking publications to sites where code and data are hosted will help shift toward reproducible research. For example, the *SIAM Journal on Imaging Science* provides cross-referencing with the peer-reviewed journal *Image Processing On Line (IPOL)* and encourage authors to submit software to IPOL. Other sites such as RunMyCode.org or Wakari might be used in a similar way. Finally, all code and data should be labeled with author information.

E Tools to aid in reproducible research

A substantial portion of the workshop focused on tools to aid in replicating past computational results (by the same researcher and/or by others) and to assist in tracking the provenance of results and the workflow used to produce figures or tables, along with discussion of the policy issues that arise in connection with this process.

Some tools are aimed at easing literate programming and publishing of computer code, either as commented code or in notebook environments. Other tools help capture the provenance of a computational result and/or the complete software software environment

used to run a code. Version control systems have been around for decades, but new tools facilitate the use of version control both for collaborative work and for archiving projects along with the complete history. Collaborative high performance computational tools, while still infrequently used, now allow researchers at multiple locations to explore climate or ocean flow models in real time. Less sophisticated but instructive applets generated in geometry or computer algebra packages can easily be shared and run over the internet. We gives an overview of tools in these various categories. A list of links to these tools and many others can also be found on the wiki.

Literate programming, authoring, and publishing tools. These tools enable users to write and publish documents that integrate the text and figures seen in reports with code and data used to generate both text and graphical results. In contrast to notebook-based tools discussed below, this process is typically not interactive, and requires a separate compilation step. Tools that enable literate programming include both programming-language-specific tools such as WEB, Sweave, and knitr, as well as programming-language-independent tools such as Dexy, Lepton, and noweb. Other authoring environments include SHARE, Doxygen, Sphinx, CWEB, and the Collage Authoring Environment.

Tools that define and execute structured computation and track provenance. Provenance refers to the tracking of chronology and origin of research objects, such as data, source code, figures, and results. Tools that record provenance of computations include VisTrails, Kepler, Taverna, Sumatra, Pegasus, Galaxy, Workflow4ever, and Madagascar.

Integrated tools for version control and collaboration. Tools that track and manage work as it evolves facilitate reproducibility among a group of collaborators. With the advent of version control systems (e.g., Git, Mercurial, SVN, CVS), it has become easier to track the investigation of new ideas, and collaborative version control sites like Github, Google Code, BitBucket, and Sourceforge enable such ideas to be more easily shared. Furthermore, these web-based systems ease tasks like code review and feature integration, and encourage collaboration.

Tools that express computations as notebooks. These tools represent sequences of commands and calculations as an interactive worksheet with pretty printing and integrated displays, decoupling content (the data, calculations) from representation (PDF, HTML, shell console), so that the same research content can be presented in multiple ways. Examples include both closed-source tools such as MATLAB (through the publish and app features), Maple, and Mathematica, as well as open-source tools such as IPython, Sage, RStudio (with knitr), and TeXmacs.

Tools that capture and preserve a software environment. A major challenge in reproducing computations is installing the prerequisite software environment. New tools make it possible to exactly capture the computational environment and pass it on to someone who wishes to reproduce a computation. For instance, VirtualBox, VMWare, or Vagrant can be used to construct a virtual machine image containing the environment. These images are typically large binary files, but a small yet complete text description (a recipe to create the virtual machine) can be stored in their place using tools like Puppet, Chef, Fabric, or shell scripts. Blueprint analyzes the configuration of a machine and outputs its text description. ReproZip captures all the dependencies, files and binaries of the experiment, and also creates a workflow specification for the VisTrails system in order to make

the execution and exploration process easier. Application virtualization tools, such as CDE (Code, Data, and Environment), attach themselves to the computational process in order to find and capture software dependencies.

Computational environments can also be constructed and made available in the cloud, using Amazon EC2, Wakari, RunMyCode.org and other tools. VCR, or Verifiable Computational Research, creates unique identifiers for results that permits their reproduction in the cloud.

Another group are those tools that create an integrated software environment for research that includes workflow tracking, as well as data access and version control. Examples include Synapse/clearScience and HUBzero including nanoHUB.

Interactive theorem proving systems for verifying mathematics and computation. "Interactive theorem proving", a method of formal verification, uses computational proof assistants to construct formal axiomatic proofs of mathematical claims. Examples include coq, Mizar, HOL4, HOL Light, ProofPowerHOL, Isabelle, ACL2, Nuprl, Veritas, and PVS. Notable theorems such as the Four Color Theorem have been verified in this way, and Thomas Hales's Flyspeck project, using HOL Light and Isabelle, aims to obtain a formal proof of the Kepler conjecture. Each one of these projects produces machine-readable and exchangeable code that can be integrated in to other programs. For instance, each formula in the web version of NIST's authoritative Digital Library of Mathematical Functions may be downloaded in TeX or MathML (or indeed as a PNG image) and the fragment directly embedded in an article or other code. This dramatically reduces chances of transcription error and other infelicities being introduced.

While we have organized these tools into broad categories, it is important to note that users often require a collection of tools depending on their domain and the scope of reproducibility desired. For example, capturing source code is often enough to document algorithms, but to replicate results on high-performance computing resources, for example, the build environment or hardware configuration are also important ingredients. Such concerns have been categorized in terms of the depth, portability, and coverage of reproducibility desired.

The development of software tools enabling reproducible research is a new and rapidly growing area of research. We think that the difficulty of working reproducibly will be significantly reduced as these and other tools continue to be adopted and improved. The scientific, mathematical, and engineering communities should encourage the development of such tools by valuing them as significant contributions to scientific progress.

F Copyright and licensing

The copyright issue is pervasive in software and can affect data, but solutions have been created through open licensing and public domain dedication. Copyright adhere to all software and scripts as an author types, and care must be taken when sharing these codes that permission is given for others to copy, reproduce, execute, modify and otherwise use the code. For reproducibility of scientific findings an attribution-only license is recommended, such as the Apache, MIT, or Modified BSD license [10]. Copyright does not adhere to raw facts, and so the raw numbers in a dataset do not fall under copyright. But datasets can

have copyright barriers to reuse if they contain “original selection and arrangement” of the data, and for this reason dedication to the public domain is suggested using the Creative Commons CC0 license for example [9]. In addition, dataset authors can provide a citation suggestion for others who use the dataset. These steps will permit shared code and data to be copied, run on other systems, modified, and results replicated, and help encourage a system of citation for both code and data.

Limits to disclosure of data also include issues such as release of individual data for medical records, census data, and for example Google search data is not publicly shareable except in the aggregate. Of course “the aggregate” is defined differently in each domain. We also recognize that legal standards in different jurisdictions (e.g. European Union, United States, Japan) can vary and that each individual needs to apprise themselves of the most substantial differences.

The algorithms embodied in software can be patentable and the author or institution may choose to seek a patent. Patents create a barrier to access and it is recommended to license the software to commercial entities through a traditional patent, and permit open access for research purposes. If patents restrict access to code this can inhibit reproducibility, access to methods, and scientific progress. Within the commercial sphere, there is a need for avenues to allow audit such as non-disclosure agreements (NDA) and independent agents for auditing similar to financial audits. Public disclosure of algorithms and code can prevent patenting by others, and ensure that such scholarly objects remain in the public domain.

G The teaching and training of reproducibility skills

The breakout group on Teaching identified the following topics as ones that instructors might consider including in a course on scientific computing with an emphasis on reproducibility. Some subset of these might be appropriate for inclusion in many other courses.

- version control and use of online repositories,
- modern programming practice including unit testing and regression testing,
- maintaining “notebooks” or “research compendia”,
- recording the provenance of final results relative to code and/or data,
- numerical / floating point reproducibility and nondeterminism,
- reproducibility on parallel systems,
- dealing with large datasets,
- dealing with complicated software stacks and use of virtual machines,
- documentation and literate programming,
- IP and licensing issues, proper citation and attribution.

The fundamentals/principles of reproducibility can and should taught already at the undergraduate level. However, care must be taken to not overload the students with technicalities whose need is not clear from the tasks assigned to them. Collaborative projects/assignments can be a good motivation.

H Workshop Participants

Aron Ahmadia, Dhavide Aruliah, Jeremy Avigad, David Bailey, Lorena Barba, Blake Barker, Sara Billey, Ron Boisvert, Jon Borwein, Brian Bot, Andre Brodtkorb, Neil Calkin, Vincent Carey, Ryan Chamberlain, Neil Chue Hong, Timothy Clem, Noah Clemons, Constantine Dafermos, Andrew Davison, Nathan DeBardeleben, Andrew Dienstfrey, David Donoho, Katherine Evans, Sergey Fomel, Juliana Freire, James Glimm, Sigal Gottlieb, Josh Greenberg, Tom Hales, Nicolas Hengartner, David Ketcheson, Matt Knepley, David Koop, Randall LeVeque, Nicolas Limare, Elizabeth Loew, Ursula Martin, Bruce McHenry, Chris Mentzel, Sarah Michalak, Ian Mitchell, Victor Moll, Hatef Monajemi, Akil Narayan, Peter Norvig, Travis Oliphant, Peter Olver, Geoffrey Oxberry, Fernando Perez, Konrad Polthier, Bill Rider, Robert Robey, Todd Rosenquist, Michael Rubinstein, Thomas Russell, Fernando Seabra Chirigati, Li-Thiao-Te Sebastien, Benjamin Seibold, Loren Shure, Philip Stark, William Stein, Victoria Stodden, Benjamin Stubbs, Andrew Sutherland, Matthias Troyer, Jan Verschelde, Stephen Watt, Greg Wilson, Carol Woodward, Yihui Xie.

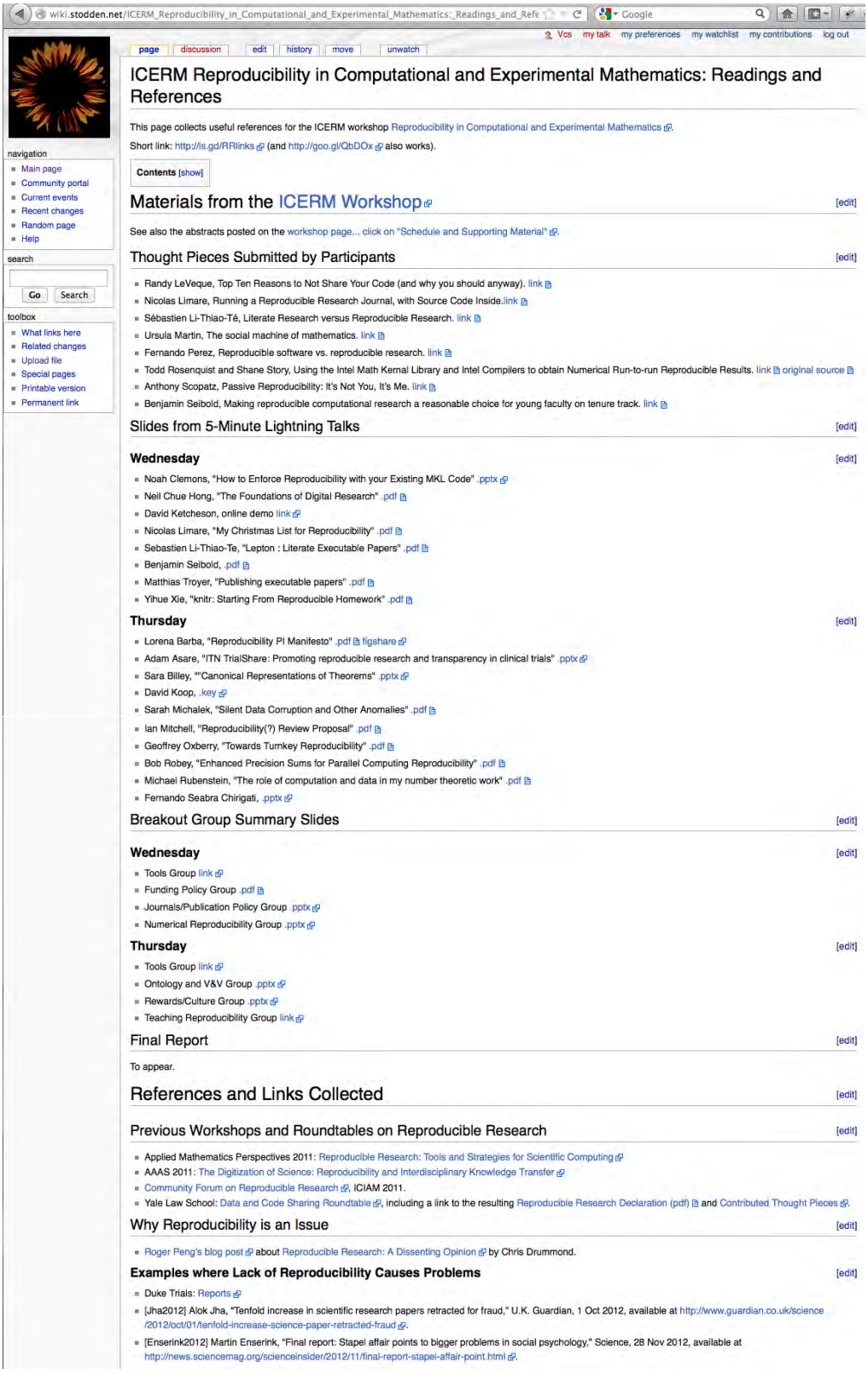

Figure 1: Snapshot of the Workshop Wiki as of January 20, 2013