**Report on the ICERM Workshop**

# Reproducibility in Computational and Experimental Mathematics

December 10–14, 2012

Written collaboratively by workshop participants[1]

**Abstract.** Science is built upon foundations of theory and experiment engaged through open, transparent communication. If done correctly, scientific theory is testable in a reproducible manner. The “reproducible research” movement in computational science and mathematics encourages this community and all scientists who use computational tools to benefit from this age-old, proven methodology by adopting better work habits and publication standards.

This report summarizes discussions that took place during the ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, held December 10-14, 2012. The themes and conclusions of the workshop can be briefly summarized as:

1. To maximize future scientific progress, it is important to promote a culture change that will integrate computational reproducibility and a greater degree of openness into the research and publication process.
2. Journals, funding agencies, and employers should support this culture change.
3. A diverse set of tools are available to aid in this enterprise.
4. Reproducible research practices and the use of appropriate tools should be taught as standard operating procedure in relation to computational aspects of research.

## 1 Introduction

The emergence of powerful new computational hardware, combined with a vast array of computational software, presents unprecedented opportunities for researchers in mathematics and science. Computing is no longer merely the third leg in the stool of modern science. In addition to the potential for producing new results in its own right, it is often the backbone of both theory and experiment, and a critical component in data analysis and interpretation.

Unfortunately the scientific culture surrounding computational work has evolved in ways that often make it difficult to efficiently build on past research, or even to apply the basic tenets of the scientific method to computational procedures. Laboratory scientists are

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

Version of January 8, 2013.

taught to keep careful lab notebooks documenting all aspects of the materials and methods they use and 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 negative effect on researchers' own productivity, their ability to build on past results or participate in community efforts, and their credibility with other scientists and the public.

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". Of course some researchers have been doing this for years, but 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. A number of workshops have recently been held on related topics, including a Roundtable at Yale Law School [**?**] a workshop as part of the Applied Mathematics Perspectives 2011 conference [**?**], and several minisymposia at other conferences, including SIAM Conferences on Computational Science and Engineering and ICIAM 2011.

The ICERM Workshop on Reproducibility in Computational and Experimental Mathematics, held December 10-14, 2012, provided an opportunity for a broad cross section of mathematicians and computational scientists to discuss many of these issues and possible 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 panel discussions on funding agency policies and on journal and publication policies. The final three days featured many talks on 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 several sets of 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.

This document reports on some of the discussions and recommendations from the workshop. A set of appendices gives more details on some topics, and numerous references are collected on the wiki, which can be reached from the workshop webpage, `http://icerm.brown.edu/tw12-5-rcem`. This webpage and the wiki also contain slides from talks and breakout group reports. [TODO: include snapshot of wiki as an appendix?]

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 led to some suggestions for terminology, as discussed further in Appendix B. We encourage others who use such terms in their work to clarify what they mean in order to avoid confusion.

[TODO: How much discussion of taxonomy to put here vs. in appendix?]

Not all aspects of reproducibility could be discussed in the workshop, and much of the focus was on tools to aid in replicating past computational results (by the same researcher and/or by others) and for assisting in tracking the provenance of results and the workflow used to produce figures or tables, along with discussion of policy issues in this connection.

There was also discussion of the related but somewhat distinct topic of “numerical reproducibility” of computational results when the same program may give different results due to hardware or compiler issues, particular in the context of parallel computing. This is discussed in Appendix D.

## 2 Changing the culture and reward structure

Workshop participants agreed that cultural changes need to take place within the field of computationally based research if reproducibility is to be fostered and maintained. We should work towards a culture where the default mode of computational science is open and transparent, to encourage collaboration in the most effective manner possible.

Researchers need to be persuaded that their efforts to ensure reproducibility will reward them with increased productivity — less time wasted in trying to recover data that was lost or misplaced, less time wasted trying to double-check results in the manuscript with data in output files, and less time wasted trying to determine whether other published results (or even their own) are truly reliable. Open access to both primary and auxiliary source code provides the basis for research to be conducted transparently with the opportunity to build upon previous work, in the same manner as open software provided the basis for Linux. This practice enables researchers both to benefit fully from the creative energies of the global community and to participate fully in it. Most great science is built upon the discoveries of preceding generations and open computational science allows this tradition to follow this well-worn path of greatness. Researchers should be encouraged to recognize the potential benefits of openness and reproducibility.

It is also important to recognize that in the short term at least there are costs and barriers to working in this manner, 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 are concerned about earning tenure or securing a permanent position. To encourage more movement towards openness and reproducibility, it is crucial that such work be acknowledged and rewarded, or at least not penalized. 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.

Changes in the culture could be accelerated by finding ways to give public recognition to good work in this direction. One suggestion is that journals and/or professional societies institute a yearly award, 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. This award could also be a cross-conference and journal award; the collected list of award recipients would both increase the visibility of researchers following good practices and provide examples for others.

The following two sections and the appendices give some additional ideas for ways in which funding agencies and journals could help encourage and recognize reproducibility. While too many requirement imposed from above could have a negative effect on the community, we believe they can take positive steps to reward desirable behavior and help to change the culture and expectations.

[LeVeque: Does the following paragraph belong here? Maybe better to leave out of main report and find a place in the appendices?] More generally, it is unfortunate that software development is 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 spend time carefully testing code, let alone documenting 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 to doing science with it, and not worthy of tenure or comparable status at a research laboratory. This attitude must be changed in the community 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 reproducibilty, should be accorded with honors similar to a strong publication record [Meyer2009; Patterson1999].

## 3 Standards of publication and journal policies

We feel there is a need to produce a set of "best practices" for publication of computational results (or any scientific results in which computation plays a role, for example in data processing, statistical analysis, or image manipulation). These may need to 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 use. This should include 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 and data with a reasonable level of documentation and instructions for repeaing the computations performed to obtain the results in the paper. Appendix C contains a more complete list of suggestions.

### 3.1 Recognizing constraints and goals

It is recognized that including such details in submitted manuscripts (or, at the least, in supplementary materials hosted by the journal) will be a significant departure from established practice, where few such details are typically presented. But science is also changing and becoming more complex and 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 transparent without the full disclosure of computational details.

At the same time, workshop participants agreed that some flexibility must be exercised. 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. On the other hand, a somewhat more informal approach might be satisfactory when the results are not of such gravity, or where confidentiality of proprietary, medical, security or personal information must be maintained. In other words, exceptions to the above standards may be appropriate in certain cases, so long as such exceptions are clearly disclosed by the authors and agreed to by reviewers and editors.

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 unauthorized usage (plagiarism) can be detected and (d) how to adjudicate disagreements that inevitably will arise.

Proper consideration of openness constraints can enable a greater community to participate in the goals of reproducible research. This includes copyright, patent, medical privacy, personal privacy, security, and export issues.

The copyright issue is pervasive in software and data, but many methods and categories have been established for software licensing with some granularity such as open licenses, public domain dedication, ... [Stodden09]. The current explosion in patent conflicts requires the same developments in handling patent rights, including the assertion of prior art against predatory patents by public disclosure [cf. http://www.researchdisclosure.com/]. There is also the need for a fully commercial policy with avenues to allow audit such as non-disclosure agreement (NDA) and independent agent for auditing similar to financial audits. [LeVeque: Do we need this paragraph here or move to an appendix with more about copyright and licenses?]

Limits to disclosure of data also include issues such as release of individual data for medical records (HIPAA: [HIPAAref]), census data and even Google search data that limit data release except in the aggregate. Of course "the aggregate" is defined differently in each domain. We also recognize that cultural and legal standards in different jurisdictions (e.g. European Union, United States, Japan) are significantly different and that each individual needs to apprise themselves of the most substantial differences [IMU2010, Hodges2011].

In terms of reproducibility, openness is not the goal but the means. It is sometimes easiest if the entire process is open, but the quantity of material can be an obstacle in itself. Thus it may be better to identify the key parts of the research and fully disclose the parts critical to the scientific process.

## 3.2 Responsibilities of reviewers and journals

Workshop participants agreed that changes are needed in policies and procedures followed by journals, editors and reviewers. To begin with, 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 permitting exceptions or exclusions. Additionally, provisions are needed for referees to obtain access to auxiliary information such as computer codes and for obtaining assistance, if required, in running computational tests of the results in submitted papers.

Along these lines, it may be judged permissible for the computational claims of a manuscript to be verifiable at another site, suggested by the authors, or on another computer system with a similar configuration. Such provisions and regulations are among the many issues in this arena that deserve discussion in the community.

As emphasized in the previous section, some flexibility is in order. Different journals may well adopt somewhat different review standards. But it is important that certain minimal standards of reproducibility and rigor be maintained in all refereed journal publications.

[LeVeque: Perhaps mention movement towards a working group representing several mathematical sciences professional societies that might propose a set of best practices?]

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 important 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.

## 4 Funding agency policies

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 scientific computing. If such standards can be common among related agencies, or at least agree on some basic details, this would significantly simplify the tasks involved in both preparing proposals and reviewing proposals.

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 software documentation and reuse (some agencies already have such requirements `http://www.jisc.ac.uk/media/documents/programmes/preservation/spsoftware_report_redacted.pdf`.
- Reasonable standards for persistence of resulting software.
- 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 Broader Impact statement.

There were also suggestions that templates for data management plans could be made available that include making software open and available. [LeVeque: I think such things exist already... I'll look through what's available at http://guides.lib.washington.edu/content.php?pid=259952&sid=2660743]

Other suggestions from breakout group:

- Set default to open  encourage statements from societies and others;
- Fund training workshops on reproducibility;
- Fund cyberinfrastructure for reproducibility at scale, for both large projects and many long-tail research efforts.
- Give publicity and stories to provide clarity on RR concept.

[TODO: Mention tools that specifically help with issues of preservation and archiving of software/data  tools that aid funding agencies goals as weve outlined them.]

## 5 The teaching and training of 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 curriculum, just as experimental or observational scientists are taught to keep a laboratory notebook and follow the scientific method. Adopting appropiate 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 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. Moreover a course on reproducibility in isolation may not make sense. 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 F. Ideally, courseware produced at one institution should be shared with others under an appropriate public license (e.g. CC BY) and upgraded if multiple institutions adopt it.

## 6 Conclusions

Computational reproducibility attempts to provide a rock solid foundation to computational science, much like a rigorous proof is the foundation of mathematics. The best reason for going down this path is its connection to the scientific method in its numerous forms. Science is built upon foundations of theory and experiment engaged through open, transparent communication and, if done correctly, is testable in a reproducible manner. Computationally-enabled science could benefit more from this age-old proven methodology. The recommendations in this report and the appendices are intended to help move the community in this direction.

[TODO: This could be improved.]

# Appendices

These appendices contain some additional material developed during the workshop. Many links were also collected on the wiki, which should be referred to for more examples of the things discussed here, additional tools, articles, editorials, etc. The wiki can be reached from a link at the bottom of the main workshop webpage, `http://icerm.brown.edu/tw12-5-rcem`.

## A The importance of reproducibility

Recent exponential increases in computing capabilities present unprecedented challenges to certify that the results of computations are sound. There is little point in setting new records for performance if the results are suspect because of flaws in design or execution. Incorrect computations can potentially impede scientific progress rather than enhancing it, lead to bad policy, safety, or medical decisions with serious consequences, and provoke distrust and confusion among other scientists and the public.

Casadevall and Fang, two prominent biologists, recently wrote in [**?**]: "Although scientists have always comforted themselves with the thought that science is self-correcting, the immediacy and rapidity with which knowledge disseminates today means that incorrect information can have a profound impact before any corrective process can take place." Steen (2011) analyzed the cause of retraction for 788 retracted papers and found that error was responsible for 545 (69%) cases. It seems likely these were predominantly computational and statistical errors. [Yu2013]

[TODO: The scientific method in the computational age?]

## B 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.

[TODO: Decide what terms to list here...]

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

- *Reviewed Research.* The descriptions of the research methods have been 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. The archive might be private, as with traditional laboratory notebooks.
- *Open Research.* Well-documented and fully open tools are publicly available (e.g., all data and open source software) 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 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. [TODO: Improve this description!]

## C Best practices for publications

[TODO: Need to clean up this section.]

A number of suggestions were made regarding best practices for publications. To aid in reproducibility they should 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.
- Salient details of data reduction and statistical analysis methods.
- 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.
- Instructions for repeating computational experiments described in the paper.
- Licensing issues. Ideally all these items "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.

Referreeing could perhaps be "modernized" through the use of technology to allow faster communication like IM or direct email while maintaining the demands of anonymity. This would allow simple questions and answers and perhaps even real-time revision of articles. The point would be to increase quality and reduce time to publish, or at least unnecessary time lag associated with the current system. [Is this sufficiently relevant to reproducibility?]

Along this line, journals might follow the lead of certain forums, such as SIGMOD [Bonnet2011], Biostatistics, IPOL and others, that have adopted some requirements for reproducibility.

Journal recommendations from breakout group:

- Create reproducible overlay issues for journals
- awards for reproducible papers / certification

Mention IPOL / RunMyCode and other tools specifically aimed at journal publication.

Further suggested guidelines from breakout groups: [TODO: Need to clean up this list, elaborate?]

- put your name on everything, including code/data;
- bibliography includes all code/data used, even your own;

- recommend awards for software (and data) contributions;
- include software explicitly in grant proposals.
- include peer review to increase quality.
- Provenance/version control, issues of changing the data after its creation without a trail.
- Auditable, Reversible, data protection, attribution
- Persistence (data longevity)
- What is the difference between access and openness (do you have access to Matlab or a supercomputer?)
- Up/down votes by peers, curation
- Anonymous versus public review
- What are the requirements/attributes of shared Data, Source code

## D Numerical reproducibility

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. Increasingly, it is difficult to determine whether a code has been correctly ported to a new system, because computational results differ from standard benchmark cases.

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 Kernal Library (MKL), which can use used to provide parallel reproducibility for mathematical computations.

A number of interesting computational applications have recently been identified that require even more precision than the 15-digit (64-bit) IEEE arithmetic that is now standard on modern computer processors. Such applications pose even more demanding requirements for justification of the level of precision used, and for certification of the final results. One solution that was suggested at the workshop is to utilize some form of higher precision arithmetic, such as Kahans summation or double-double arithmetic [Bailey2012]. 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 expanded. One workshop participant reported (Robey2011) that this change has dramatically increased reproducibility in numerous computational physics applications at the Los Alamos and Livermore National Laboratories. But it is clear that this solution will not work for all applications, and, 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.

These issues must be addressed with deliberate care by authors of manuscripts. They must be careful to state in their paper what levels of numerical reproducibility are appropriate for their task, and then to provide detailed justification that the level of numeric precision (32-bit, 64-bit or higher precision) is appropriate.

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, e.g. [Constaninescu2000, Kola2005, Panzer-Steindel2007, Michalak2010, Li2010, Autran2010], testing with proton and neutron beams, e.g. [Michalak2012 and citations therein], and in other types of studies, e.g. [Constantinescu2005], with suggestions that SDC may be an increasing issue with new technologies [Borkar2012].

Field data on supercomputer DRAM memory failures have shown that advanced error correcting codes (ECC) are required [Sridharan2012] 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 [InterAgency2012, DarpaResilience2009, TowardsExascaleResilience2009, HECResilience2009].

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.

## E Tools to aid in reproducible research

A principle impediment to the practice of reproducible research has been the lack of tools that make it easy to work reproducibly. A number of tools that address this need have recently emerged. In this section, we briefly categorize and describe the tools available to enable replicable or reproducible research. This list is not meant to be exhaustive but rather to give a few examples in each category, and mainly represents tools that were discussed at the workshop. The wiki contains links to these and other tools.

[Also would be useful to derive useful/essential tool attributes for reproducibility in different settings]

Dozens of tools were mentioned in workshop presentations, ranging across different domains and addressing different aspects of reproducibility. They can be loosely organized into groups on authoring, provenance, computational environments, and theorem proving.

**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, Taverna, 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 and other tools. VCR, or Verifiable Computational Research, creates unique identifiers for results that permits their reproduction in the cloud. [code/data repos such as mloss.org , datahub.io, figshare.org?]

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 Flyspeck project, using HOL Light and Isabelle, aims to obtain a formal proof of the Kepler conjecture. [MathML, MetaMath? and Probabilistically checkable proofs and codes? The Open Theory Project?] [TODO: Can someone put these more into the context of reprodubility?]

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 [Freire2012].

The development of software tools enabling reproducible research is a new and rapidly growing area of research [cite http://stodden.net/AMP2011/ ]. 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 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.

[TODO: Other possible appendices? For example...

- Issues related to experimental math and/or theorem proving.
- Relation to V&V, UQ,
- Also include a snapshot of wiki as another appendix?

]