Reproducibility: Methods

Randall J. LeVeque

Department of Applied Mathematics, University of Washington

DRAFT of an article in preparation for the Encyclopedia of Applied and Computational Mathematics, to be published by Springer. All rights reserved. April 22, 2012.

1 Summary

The term "reproducible research" in scientific computing and computational mathematics, science or engineering generally refers to the archiving and/or publication of all computer codes and data necessary to later reconstruct research results.

2 Description

The requirement of reproducibility of experimental results has long been an integral part of the "scientific method". To the extent possible, researchers are expected to repeat carefully controlled experiments in order to insure that observed results are not the result of flawed experimental procedure or external influences. Experimental scientists are expected to keep careful laboratory notebooks documenting all steps of experiments, including those that fail to support the desired result. Such notebooks have a legal standing in issues of intellectual property rights or investigations of research falsification, and are critical in facilitating future research by the same scientist or by new personnel joining an established laboratory. Publications that result from experimental research are expected to contain a detailed description of the procedures and materials used in the experiments. These descriptions are often used by other researchers to independently verify the results presented, or as a basis for new research that builds on the published work.

Similar standards are generally not the norm in computational research, but the development of such standards and tools to facilitate reproducibility is an active area of research. There is growing concern regarding the reproducibility of computational experiments, particularly with the increasing use of computer simulation to replace physical experiments and the increased reliance on computational techniques in all areas of scientific enquiry, engineering design, and policy making.

At first glance it may seem that a computational experiment is much more easily repeatable than a physical experiment: running the same program a second time might be expected to give the same results as the first time, even if run on a different computer. However, in practice there are several challenges:

- It is not always true that running the same program twice gives the same results, even if the program is correctly written. On a computer, the order in which operations are performed can make a difference even if operations commute in theory. When using optimizing compilers or parallel computers, the order of operations may change from one run to another.
- Some programs cannot easily be run on a different computer than the one where the original experiment was performed. This may be because of the use of proprietary or commercial software that cannot be transferred, or the use of specialized hardware such as a massively-parallel supercomputer.
- Even if the same result is always obtained when running the program repeatedly on a number of different computers, this does not guarantee that the program is correct or that the result is meaningful. Nor does it guarantee that other scientists can confirm that the program faithfully implements the ideas contained in a publication or can build on this work in future research.
- The program and input data may not be available at a later date, even to the person who wrote it and originally performed the experiments. Computer codes often evolve rapidly in the course of research and are adapted to solve new problems without carefully documenting or archiving the version of code and data that were used to obtain previous results.

Although the first two difficulties above should not be overlooked, the term “reproducible” in computational science generally means much more than simply getting the same result in a dependable manner when the same program is run repeatedly. (This more limited version of reproducibility is sometimes called “replicable” or “repeatable” to make this distinction clear.) Reproducibility also does not directly address the correctness of computer code for solving the target problem; see the entries on *Validation* and *Verification* for that topic.

The remainder of this article addresses the difficulties inherent in archiving and publishing computer codes and data, and some tools that are currently used to facilitate this. Approaches and methods are rapidly evolving and rather than citing specific tools currently in use, it is recommended that interested readers search the literature for the latest developments using some of the terms introduced below.
See [Yale Law School Roundtable on Data and Code Sharing(2010)] or [reproducibleresearch.org(2012)] for some further references.

2.1 Version control

A technique that is well established in software development communities (and increasingly among computational scientists) is the use of a *version control system (VCS)* to track changes to source code and perhaps data. Once a file is under version control, a modified version can be "committed" and the system will keep track of the difference between this version and the previous version. Only differences are stored, which greatly reduces the storage required to track large numbers of changes, but any previous version of a file or the entire code base can be automatically regenerated with a few commands.

Popular version control systems include *CVS* and its successor *Subversion*. These are examples of the *client-server* model of version control, in which a master repository exists on a server that contains the full history. All developers commit changes to this repository and must have access to the repository (often via the internet) in order to commit changes or reconstruct previous versions.

More recently, *distributed version control systems* have become more popular, in which every "clone" of the repository contains the entire history and developers can work independently but easily merge changes between repositories when convenient. Popular examples include *Mercurial*, *Git*, and *Bazaar*. A good introduction to version control can be found in [Sink(2011)].

2.2 Web-based repositories

Most version control systems have associated web-based tools to assist in the exploration of past versions and changes between versions. These tools typically also provide “issue tracking” facilities to keep track of bug reports and proposals for enhancements to the code.

Although version control is extremely useful even when practiced by a lone researcher on an isolated computer, for collaboration it is often convenient to use repositories that are hosted on websites such as `bitbucket.org` or `github.org` that can be used for a “master copy” of a shared repository and to host the issue tracker. Public repositories are frequently used for open source software projects that allow anyone to download code, and can be a valuable component in reproducibility when used to host code associated with a journal publication. Many institutions also maintain institutional repositories that can be used to archive the code or data used in publications, generally without version control.

2.3 Related ideas

2.3.1 Data provenance

The term “provenance” refers to the documentation of the complete history of an object and its ownership, and was originally used primarily for works of art. Since scientific results now frequently depend on data that has been collected from numerous sources, or is generated or processed by computer programs that may change or be run with different choices of parameters, the issue of *data provenence* is an important aspect of reproducibility.

2.3.2 Literate programming

The term “literate programming” was coined by the computer scientist Donald Knuth [Knuth(1984)], who developed a system to combine computer code with its own description and documentation. Several other approaches have been developed since that also assist in writing self-documented code. These systems can be a useful component in reproducible research, and can greatly assist in deciphering code written by someone else or in the distant past.

2.3.3 Scientific workflow systems

A workflow management system designed to build up and keep track of a sequence of computational steps and their data is often called a *scientific workflow system*. Their use can aid in preserving a complete record of all computations performed in the course of a research project and the provenance of the associated data.

2.3.4 Virtualization

Often having the computer program that generated results is insufficient to replicate the same results later, since subtle changes in compilers, visualization tools, or other software used by the program can change the results. With the passage of time it may not be possible to run the code at all on a newer operating system. One approach to archiving or sharing codes is to use virtualization, in which the entire operating system and software environment is preserved in a *virtual machine* (VM). This machine can then be run on any computer (with an appropriate player) in order to emulate the original environment. This approach has become even more convenient recently with the growth of commercial cloud computing: a VM can be created and archived on a public cloud computing platform in such a way that it can be run by anyone who purchases sufficient computing time (typically at a rate of pennies per CPU hour as of this writing). Publicly funded cloud computing platforms, free for use in scientific research, are also being deployed, and open source alternatives to commercial cloud platforms provide comparable capabilities.

References

[Knuth(1984)] Knuth DE (1984) Literate programming. The Computer Journal 27:97–111

[reproducibleresearch.org(2012)] reproducibleresearchorg (2012) Links to resources.
http://reproducibleresearch.net/index.php/RR_links

[Sink(2011)] Sink E (2011) Version control by example. http://www.ericsink.com/vcbe/

[Yale Law School Roundtable on Data and Code Sharing(2010)] Yale Law School Roundtable on Data and Code Sharing (2010) Reproducible research: Addressing the need for data and code sharing in computational science. Computing in Science and Engineering 12:8–13, http://doi.ieeecomputersociety.org/10.1109/MCSE.2010.113