

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. February 1, 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

1 reliance on computational techniques in all areas of scientific enquiry, en-
2 gineering design, and policy making.

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

- 8 • It is not always true that running the same program twice gives the
9 same results, even if the program is correctly written. On a computer,
10 the order in which operations are performed can make a difference
11 even if operations commute in theory. When using optimizing compil-
12 ers or parallel computers, the order of operations may change from
13 one run to another.
- 14 • Some programs cannot easily be run on a different computer than the
15 one where the original experiment was performed. This may be be-
16 cause of the use of proprietary or commercial software that cannot be
17 transferred, or the use of specialized hardware such as a massively-
18 parallel supercomputer.
- 19 • Even if the same result is always obtained when running the program
20 repeatedly on a number of different computers, this does not guaran-
21 tee that the program is correct or that the result is meaningful. Nor
22 does it guarantee that other scientists can confirm that the program
23 faithfully implements the ideas contained in a publication or can build
24 on this work in future research.
- 25 • The program and input data may not be available at a later date, even
26 to the person who wrote it and originally performed the experiments.
27 Computer codes often evolve rapidly in the course of research and
28 are adapted to solve new problems without carefully documenting or
29 archiving the version of code and data that were used to obtain previ-
30 ous results.

31 Although the first two difficulties above should not be overlooked, the
32 term “reproducible” in computational science generally means much more
33 than simply getting the same result in a dependable manner when the same
34 program is run repeatedly. (This more limited version of reproducibility is
35 sometimes called “replicable” or “repeatable” to make this distinction clear.)
36 Reproducibility also does not directly address the correctness of computer

37 code for solving the target problem; see the entries on *Validation* and *Veri-*
1 *fication* for that topic.

2 The remainder of this article addresses the difficulties inherent in archiv-
3 ing and publishing computer codes and data, and some tools that are cur-
4 rently used to facilitate this. Approaches and methods are rapidly evol-
5 ving and interested readers should search the literature for further develop-
6 ments. See [1] or [2] for some further references.

7 **2.1 Version control**

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

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

23 More recently, *distributed version control systems* have become more
24 popular, in which every “clone” of the repository contains the entire history
25 and developers can work independently but easily merge changes between
26 repositories when convenient. Popular examples include *Mercurial*, *Git*,
27 and *Bazaar*.

28 **2.2 Web-based repositories**

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

33 Although version control is extremely useful even when practiced by a
34 lone researcher on an isolated computer, for collaboration it is often conve-
35 nient to use repositories that are hosted on websites such as `bitbucket.org`

36 or `github.org` that can be used for a “master copy” of a shared repository
1 and to host the issue tracker. Public repositories are frequently used for
2 open source software projects that allow anyone to download code, and
3 can be a valuable component in reproducibility when used to host code
4 associated with a journal publication. Many institutions also maintain insti-
5 tutional repositories that can be used to archive the code or data used in
6 publications, generally without version control.

7 **2.3 Related ideas**

8 **2.3.1 Data provenance**

9 The term “provenance” refers to the documentation of the complete his-
10 tory of an object and its ownership, and was originally used primarily for
11 works of art. Since scientific results now frequently depend on data that
12 has been collected from numerous sources, or is generated or processed
13 by computer programs that may change or be run with different choices of
14 parameters, the issue of *data provenance* is an important aspect of repro-
15 ducibility.

16 **2.3.2 Literate programming**

17 The term “literate programming” was coined by the computer scientist Don-
18 ald Knuth, who developed a system to combine computer code with its own
19 description and documentation. Several other approaches have been de-
20 veloped since that also assist in writing self-documented code. These sys-
21 tems can be a useful component in reproducible research, and can greatly
22 assist in deciphering code written by someone else or in the distant past.

23 **2.3.3 Scientific workflow systems**

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

29

2.3.4 Virtualization

1
2
3
4
5
6
7
8
9

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*. This machine can then be run on any computer (with the required player) in order to produce an emulator of the original environment.

10

References

11
12
13
14
15
16
17

- [1] REPRODUCIBLERESEARCH.ORG, *Links to resources*.
http://reproducibleresearch.net/index.php/RR_links, 2012.
- [2] YALE LAW SCHOOL ROUNDTABLE ON DATA AND CODE SHARING, *Reproducible research: Addressing the need for data and code sharing in computational science*, Computing in Science and Engineering, 12 (2010), pp. 8–13. <http://doi.ieeecomputersociety.org/10.1109/MCSE.2010.113>.