'closed' communities based on give-and-take structures. Instead, we advocate incentives to make data publicly available with full open access.

Credit for scientific research comes largely through acquiring citations. The number of citations received governs the success of individuals as much as journals. If we are serious about valuing data sharing, there must therefore be a benefit in terms of citations for those who produce high-quality data sets that are widely reused. Several journals now facilitate peer-reviewed publication of individual data sets in the context of data publications, such as *Scientific Data*, *Biodiversity Data Journal* or *Earth System Science Data*. These journals collaborate with data repositories, so that the journal publication links to the respective deposited data set, which hence becomes citable.

In parallel to emerging opportunities of data publications, it is important to ensure that all data sources are cited and indexed appropriately[2]. Because many journals restrict the length of reference lists, references to data sources are often moved to supplementary material that is not included in common indexing systems, like Web of Science, Scopus or Google Scholar. The introduction of 'data citations' as an additional category of citations to the usual reference list (as operated in *Scientific Data*[3]) resolves this situation: such a dedicated section allows any underlying data sources to be fully acknowledged.

It will be up to the scientific community to decide whether data citations are evaluated together with references to articles or separately. In a research article, the distinction between data citations and other references is useful to the reader, and it can also indicate the main area of contribution of a researcher or research group.

Nevertheless, we advocate that no distinction between data and paper citations should be made by indexing systems for the purposes of performance measures and impact factors. What we seek to measure is the influence of a researcher's work, and this can be mediated through their ideas or their data: both are equally important. Singling out data citations in a separate aggregated metric, such as a separate number of data citations or a 'data-h-index', carries the risk of making data provision a second class performance measure. Members of the data-producing community will likely object to that — and rightly so.

To achieve a step up in the popularity of data sharing, we need a well-structured system of opportunities and incentives. True benefits for researchers in return for the time-consuming task of making their data and metadata widely usable are beginning to be realized through the establishment of (curated) data repositories, data journals, data citations and their inclusion into common indexing systems and evaluation criteria.

The establishment of data publications and repositories in combination with the opportunity to appropriately cite data sources provides an effective system of incentives for sharing data on a voluntary basis. This not only has the potential to overcome the above mentioned stumbling blocks, but also provides motivation for the collection of high-quality data — an aspect so far neglected in the context of enforced data publication.

We are currently at an exciting turning point in science. High-quality primary data *per se* are beginning to be recognized as valuable raw material for scientific progress. It is high time that we give credit where credit belongs: to the researchers taking the original measurements[3–5]. ❐

Jens Kattge is at the Max Planck Institute for Biogeochemistry, Hans Knöll Str. 10, 07745 Jena, Germany and coordinates the TRY database of plant traits. Sandra Díaz is in Instituto Multidisciplinario de Biología Vegetal (IMBIV-CONICET) and FCEFyN at the Universidad Nacional de Córdoba, Av. Velez Sarsfield, Argentina and coordinates the International Research Network Nucleo DiverSus. Christian Wirth is at the Institute for Systematic Botany and Functional Biodiversity, University of Leipzig, Johannisallee 21 04103 Leipzig, Germany and the German Centre for Integrative Biodiversity Research (iDiv), Halle-Jena-Leipzig, Deutscher Platz 5e 04103 Leipzig, Germany.
e-mail: jkattge@bgc-jena.mpg.de

References

1. Michener, W. K., Brunt, J. W., Helly, J. J., Kirchner, T. B. & Stafford, S. G. *Ecol. Appl.* **7,** 330–342 (1997).
2. Kueffer, C. *et al. Trends Ecol. Evol.* **26,** 493–494 (2011).
3. Wilson, R. Endorsing the Joint Declaration of Data Citation Principles *Scientific Data Blog* (24 March 2014); http://go.nature.com/NHOqUp
4. Joint Declaration of Data Citation Principles; https://www.force11.org/datacitation
5. San Francisco Declaration on Research Assessment; http://am.ascb.org/dora/

Open code for open science?

Steve M. Easterbrook

Open source software is often seen as a path to reproducibility in computational science. In practice there are many obstacles, even when the code is freely available, but open source policies should at least lead to better quality code.

Poor code quality is endemic, and not just in scientific computation. It is always tempting to build something 'quick and dirty', under the assumption that it can be cleaned up later. This is especially true at the cutting edge of a field — why invest time writing beautifully engineered code from the outset, if you're not sure that what you're trying to do is even possible?

In software engineering, this is known as technical debt: by deferring issues such as code readability and maintainability, a debt is created that someone in the future might have to pay, in the extra effort needed to re-run or modify the code[1]. The point of the metaphor is not that debt is bad *per se*. After all, we frequently incur debt to obtain something of immediate value, for example, using a mortgage to buy a house. The point is that such debts have to be managed carefully, to prevent them spiralling out of control.

Open source policies in scholarly journals can help here. If journals ask for open code, they create a strong incentive for authors to clean up the code each time a paper is produced, rather than deferring such tasks indefinitely. As a second order effect, such policies should encourage more scientists to take the opportunity to improve their software-building skills, through courses such as Software Carpentry (http://software-carpentry.org/).

Box 1 | Barriers to sharing.

Portability. The huge variety of computing platforms and support software make portability a challenge. Complex models are often carefully optimized so that they run in reasonable time on one specific platform. Requiring them also to be portable represents an awkward trade-off. Portability will win only if there is a commitment to a broader user community, along with the resources to meet that commitment. Without this, optimization usually wins.

Configurability. Porting the code is only part of the battle — it may be just as hard to configure a model for a particular run. Even with a user manual and configuration scripts, figuring out which configuration was used to produce any particular result is hard, because there is no standard way to describe configurations of scientific models. In earth system modelling, a community effort is underway dedicated to creating a controlled vocabulary to help with this task[5], but even this does not contain enough detail to recreate a complete model configuration.

Entrenchment. In complex scientific models, many layers of decision-making go into the code[6]. In earth system modelling, these decisions often span several decades of development, as parts of the models have been under continual refinement for that long. These decisions are rarely documented comprehensively. Anyone outside the lab where the model was developed is unlikely to be able to understand the code in detail.

Model-data blur. Across the geosciences, it is increasingly hard to make meaningful separations between computational models and observational data[7]. What we call observational data typically encompasses processed data products, for example, from remote sensing. These have often been passed through a variety of algorithmic transformations, to derive the desired variables, and then to calibrate, homogenize and re-grid the data. Similarly, computational models are continually fed observational data in various forms, to correct biases, to parameterize processes that cannot be computed explicitly or to set boundary conditions. As a result, the validity of the code cannot be discussed without examining the data that went into it, and vice versa.

Provenance. Reproducing a set of results may depend on knowing all the computational steps that have been applied. But capturing the full provenance for highly processed, homogenized datasets remains a challenge, especially if some steps involve commercial software tools or heuristic techniques that are difficult to understand and to reproduce.

I argue that open source policies are unlikely to usher in an era of much greater sharing and reproducibility, because there are many barriers beyond the basic requirement of being able to read the code (see Box 1). Instead, such policies have an important role to play in improving the quality of scientific software by nudging scientists to manage their technical debt more carefully.

Repeat or reproduce?

In principle, by making models and data freely available, other scientists can perform their own analysis on the data, and can re-run the code to verify results[2]. Ideally, the end result is greater collaboration, wider acceptance of results and increased trust in the scientific endeavour. In practice, none of this comes easily.

First, there is some disagreement on what it means for research to be reproducible[3]. For example, repeatability and reproducibility are often conflated in the context of scientific computing. Repeatability means the ability to re-run the same code at a later time or on a different machine. Reproducibility means the ability to recreate the results, whether by re-running the same code, or by writing a new program (see Fig. 1).

It is possible to have each without the other. Repeatability without reproducibility — getting different results when re-running the code — can be a result of fragile code, combined with small changes in the hardware platform, the compiler or one of the ancillary tools. It is especially common in numerical simulations of chaotic phenomena, such as weather, where any change in how the code is compiled and optimized may lead to tiny rounding differences that rapidly multiply as the simulation proceeds. In meteorology and climate science, modellers handle this problem by using ensembles of runs to produce probabilistic results. Exact repeatability is extremely hard to maintain across platforms (see Box 1).

Reproducibility without repeatability — the confirmation of results using different code — is the computational equivalent of a replicated experiment, the bread-and-butter of doing science. Independently reproducing computational results is a creative process that can lead to the discovery of new approaches, and generates a stronger body of scientific evidence.

At the intersection, repeating a run to get the same results is rarely interesting. It may be an important testing strategy, for example when porting code, but it doesn't yield new scientific insights. Sometimes it can be useful for different labs to re-run each other's code to make detailed comparisons of their different approaches, but in practice, this rarely happens. In the climate sciences, for example, such inter-comparisons are achieved more easily without the need for code sharing, not least because of the technical difficulties that come with running complex code on another machine. Instead, each lab runs its own models on standard experiments and shares the model outputs via a community databank.

As a result, at least for more complex scientific software, it is not obvious that making the code associated with a specific scientific publication available will lead to significant advances in reproducibility or to significant new insights. A sharing strategy built around modular tools might be more useful than one based on the idea of repeating computations used in published papers.

The myth of many eyes

Journal policies to ensure code availability for published papers will not, on their own, create successful open source collaborations. In practice, making code truly open source — in the sense that it can be run, usefully, by a wide variety of people and on a wide variety of platforms — demands a commitment that few scientists are able to make. When building scientific software, researchers usually face the choice to build something that helps answer a specific problem, or to build a more generalized tool and invest in an effort to build a user community around it. These two goals tend to be mutually exclusive, and the latter approach is only worthwhile if there is a clear demand for the tool. Code that is created to produce an individual publication rarely qualifies.

One of the hidden secrets of open source is that the vast majority of software released as open source fails to attract any kind of

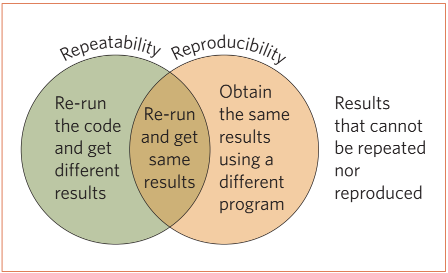
Repeatability
Reproducibility
Re-run the code and get different results
Re-run and get same results
Obtain the same results using a different program
Results that cannot be repeated nor reproduced

Figure 1 | Repeatability versus reproducibility.

a community at all. For example, detailed analyses of the projects on Sourceforge (http://sourceforge.net) demonstrate a clear power-law pattern of participation: a very small number of projects end up with a large number of participants, whereas the vast majority of projects end up with one participant or even none[4]. These results suggest that for the vast majority of scientists who release their code with a journal publication, there will probably be no uptake whatsoever. However, a small number of projects will attract a lot of attention — perhaps those with controversial results or exciting breakthroughs. Releasing the code is therefore a hopeful act — it allows for serendipitous discovery, even if we can't predict whether anyone will be interested.

On a different note, in the polarized context of climate research, making code available to public scrutiny holds the potential to improve trust. Certainly, if the code is not available, accusations that the science cannot be trusted are easy to make. But in reality, releasing the code makes little difference, as all but the simplest codes are impenetrable to non-experts. Unfortunately, trust via open source could also come at a price: climate scientists are often subject to politically-motivated attacks, and opening up access to source code brings the potential for 'denial of service' attacks on scientific labs. A science institution usually does not have the support staff to help resolve queries, if attempts to re-run the code fail. This puts scientists in a difficult position: if they do not respond, they can be exposed to pressure from the media; if they do, they could end up spending all their time resolving minor weaknesses in the code. Making code available can therefore only work on the understanding that it does not involve the obligation to support others in repeating the computations.

Realistic expectations

A growing number of scientific journals, including the *Nature* family, now encourage authors to share their software, and require them to include a statement in each paper about the availability of the code. Such policies are an important step forward for computational science, but the expectations around these policies should be realistic. Significant improvements in the sharing of software tools and in making computationally-based research reproducible require much more than merely making the code available. Nevertheless, even the short-term benefits of such policies are not negligible. Asking authors to open up access to their code is likely to lead to a rapid improvement in code quality: the mere possibility that someone could read the code is a strong incentive to make it presentable (even if nobody does read it in the end).

Journal efforts to move research communities towards a norm where all code is freely available, are only a first step. Building on such a culture of openness, an environment may eventually develop where small data sets and new software tools can be more readily discovered, and where reproducibility is achieved more easily. ❐

Steve M. Easterbrook is at the University of Toronto, 10 Kings College Rd, Toronto, Ontario M5S 2E4, Canada.
e-mail: sme@cs.toronto.edu

References

1. Lim, E., Taksande, N. & Seaman, C. *IEEE Software* **29,** 22–27 (2012).
2. Peng, R. D. *Science* **334,** 1226–1227 (2011).
3. Grubb, A. M. & Easterbrook, S. M. *PLoS ONE* **6,** e23420 (2011).
4. Xu, J., Gao, Y., Christley, S. & Madey, G. *Proc. 38th Annu. HICSS* 198a (2005); http://dx.doi.org/fqgn7p
5. Moine, M-P. *et al. Geosci. Model Dev*. **7,** 479–493 (2014).
6. Lenhard, J. & Winsberg, E. *Stud. Hist. Philos. Mod. Phys.* **41,** 253–262 (2010).
7. Edwards P. N. *A Vast Machine* (MIT Press, 2010).