Reimplementation offers additional benefits beyond its theoretical contributions to theory testing. Solving a computational problem with a novel approach is also useful for detecting problems or bugs in a previous implementation. For example, if we thought that people optimized a given behavior, but discovered that only one kind of optimization algorithm worked well, we could discover that the real underlying process is not about optimization itself but about some incidental properties of one particular algorithm. Or, if two implementations gave two different answers, we might discover that there is a bug in one (or both) models, or that some of the assumptions underlying one (or both) implementations are incorrect. The same logic holds true of statistical analyses as well. Analogously, if a Pearson correlation is statistically reliable but a non-parametric one is not, that could suggest that the Pearson correlation is driven by an outlier. The ability of reimplementation to detect bugs becomes even more powerful when applied to complex models with many moving parts. Successful implementation requires all of the parts to work together. Further, writing code is usually easier and more pleasant than reading code, especially if you are reading specifically for bug identification. Reimplementation is a far more fun and fast way of checking computational ideas.

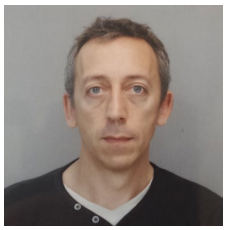
Marr, D. (1982), Vision: A Computational Approach, San Francisco, Freeman & Co

Diversity in Reproducibility

Olivia Guest

Department of Experimental Psychology University College London, United Kingdom

o.guest@ucl.ac.uk



Nicolas P. Rougier

INRIA Bordeaux Sud-Ouest, Institut des Maladies Neurodégénératives, Université Bordeaux, Bordeaux, France

nicolas.rougier@inria.fr

In our previous contribution, we proposed computational modelling-related definitions for replicable, i.e., experiments within a model can be recreated using its original codebase, and reproducible, i.e., a model can be recreated based on its specification. We stressed the importance of specifications and of access to codebases. Furthermore, we highlighted an issue in scholarly communication—many journals do not require nor facilitate the sharing of code. In contrast, many third-party services have filled the gaps left by traditional publishers (e.g., Binder, 2016; GitHub, 2007; Open Science Framework, 2011; ReScience, 2015). Notwithstanding, journals and peers rarely request or expect use of such services. We ended by asking: are we ready to associate codebases with articles and are we prepared to ensure computational theories are well-specified and coherently implemented?

Scope of Evaluation

Dialogue contributions include proposals for: intermediate levels between replicability and reproducibility (Crook, Hinsen); going beyond reproducibility (Kidd); encompassing computational science at large (Gureckis & Rich, Varoquaux); and addressing communities as a function of expertise (French & Addyman). On the one hand, some replies discuss evaluation more broadly, empirical data collection, and software engineering. On the other hand, some delve into the details of evaluating modelling accounts. We will discuss the former first.

In Varoquaux's contribution, reproducibility includes replicability and code rot (e.g., in fMRI: Eklund, Nichols, & Knutsson, 2016). However, the titular computational reproducibility is orthogonal to maintaining a re-usable codebase. Software and hardware inevitably go out of fashion meaning codebases expire. Nevertheless, the overarching theory encapsulated by modelling software could withstand the effects of entropy if specified coherently, e.g., early artificial neural network codebases are not required to understand nor reproduce these models. Indubitably, there is a balance to be struck between reimplementation and re-use.

In contrast, Gureckis and Rich extend their scope to the empirical replication crisis in psychology. They mention that implicit knowledge often goes unpublished and thus only fully automated on-line experiments are computationally reproducible psychology.

Epistemically, empirical and software replication and reproduction are distinct from their modelling-related counterparts — they are six related endeavours. The difference between software for science (e.g., a statistical test) and science that is software (e.g., a cognitive model) is an important one to underline. In the former case the code is a tool, in the latter it constitutes an experiment. Notwithstanding, all such evaluations have scientific merit.

Levels of Evaluation

We mentioned two of the levels in which modelling work is evaluated. Unanimity is reached on replication as a minimum check, however some dialogue contributions go further. To wit, Hinsen separates this endeavor into three steps. Specifically we must check

that a model is: bug-free; reproducible as presented; congruent with empirical data. These roughly map onto the levels of talking about modelling work more generally, as Kidd notes (Marr, 1982).

Implementation Level
With respect to the implementation level, as Crook explains, re-running code both within a lab and by others allows for checking for bugs and, importantly, if assumed-to-be-irrelevant variables, e.g., the random seed, are not driving the results. This also ensures documentation is appropriate. Success at this level indicates a model is replicable.

Model Level
To evaluate the quality of the specification, we may rewrite, i.e., reproduce, the model from scratch. This provides evidence for or against depending on the reimplementation's success. As Kidd mentions, and as we discovered (Cooper & Guest, 2014), this process allows us to: discern when implementation details must be elevated to the theory level and vice versa; evaluate the specification; and uncover bugs.

Theory Level
Many methods exist for testing theories. One such method involves computationally implementing a theory—another is to test predictions by gathering empirical data. As Crook points out, such data is also used to evaluate models and should be associated with the original article and codebase. In such cases, empirical data requires re-collecting. This is because if the phenomenon to-be-modelled, Hinsen warns, does not occur as described by the overarching theoretical account, then both theory and model are brought into question. "A* is a model of [...] A to the extent that [we] can use A* to answer questions [...] about A." (Minsky, 1965, p. 426)

Conclusions

Even though definitions for terms across the replies do not fully converge[1], all contributors agree that change is needed and imminent. A notable divergence of opinion can be found in the reply by French and Addyman, who believe specifications are less vital than we do. Importantly, we agree on some fundamentals: sharing codebases; linking articles with codebases; and reproducing models (e.g., ReScience, 2015).

In response to our question: Hinsen proposes modellers include a specific article section on evaluation; while Crook lists community-driven initiatives for sharing codebases and specifications. Crook hopes, as we do, for topdown publisher-enforced sharing of resources in partially-centralised repositories. However, this does not preclude, and may in fact require, grassroots demands. If the scientific community rightfully yearns for change, we are required to act to make this happen.

Binder. (2016). mybinder.org. Retrieved 2017-01-02, from http://mybinder.org/
Cooper, R. P., & Guest, O. (2014). Implementations are not specifications: Specification, replication and experimentation in computational cognitive modeling. Cognitive Systems Research, 27, 42-49. doi: 10.1016/j.cogsys.2013.05.001
Eklund, A., Nichols, T. E., & Knutsson, H. (2016). Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. Proceedings of the National Academy of Sciences, 113(28), 7900-7905. Retrieved from http://www.pnas.org/ content/113/28/7900.abstract doi: 10.1073/ pnas.1602413113
GitHub. (2007). GitHub. Retrieved 2017-01-02, from http://github.com/
Marr, D. (1982). Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Freeman.
Mesnard, O., & Barba, L. A. (2016, May). Reproducible and replicable CFD: it's harder than you think. ArXiv e-prints.
Minsky, M. (1965). Matter, mind and models. In International federation of information processing congress.
Open Science Framework. (2011). OSF. Retrieved 2017-01-02, from http://osf.io/
Patil, P., Peng, R. D., & Leek, J. (2016). A statistical definition for reproducibility and replicability. bioRxiv. Retrieved from http://biorxiv.org/ content/early/2016/07/29/066803 doi: 10 .1101/066803
ReScience. (2015). The ReScience Journal. Retrieved 2017-01-02, from http://rescience.github.io/

1. We do not wish to prescriptively enforce our terms and definitions—and we are open to suggestions, especially based on the use of such terms by computationally-based disciplines (e.g., Mesnard & Barba, 2016; Patil, Peng, & Leek, 2016).