

Top Tips to Make Your Research Irreproducible

Neil P. Chue Hong¹, Tom Crick², Ian P. Gent³, Lars Kotthoff⁴ and Kenji Takeda⁵

¹Software Sustainability Institute, University of Edinburgh, UK

²Department of Computing & Information Systems, Cardiff Metropolitan University, UK

³School of Computer Science, University of St Andrews, UK

⁴Insight Centre for Data Analytics, University College Cork, Ireland

⁵Microsoft Research, Cambridge, UK

¹n.chuehong@software.ac.uk <http://www2.epcc.ed.ac.uk/~neilc/>

²tcrick@cardiffmet.ac.uk <http://drtomcrick.com/>

³ian.gent@st-andrews.ac.uk <http://ian.gent>

⁴lars.kotthoff@insight-centre.org <http://4c.ucc.ie/~larsko/>

⁵kenji.takeda@microsoft.com

<http://research.microsoft.com/en-us/people/kenjitak/>

1 April 2015

We have noticed (and contributed to) a number of manifestos, guides and top tips on how to make research reproducible [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; however, we have seen very little published on how to make research *irreproducible*.

Irreproducibility is the default setting for all of science, and irreproducible research is particularly common across the computational sciences. The study of making your work irreproducible without reviewers complaining is a much neglected area; we feel therefore that by encapsulating some of our top tips¹ on irreproducibility, we will be filling a much-needed gap in the domain literature. By following our starter tips, you can ensure that if your work is wrong, nobody will be able to check it; if it is correct, you will make everyone else do disproportionately more work than you to build upon it. In either case you are the beneficiary.

It is an unfortunate convention of science that research should pretend to be reproducible; our top tips will help you salve the conscience of certain reviewers still bound by this fussy conventionality, enabling them to enthusiastically recommend acceptance of your irreproducible work.

1. **Think “Big Picture”.** People are interested in the science, not the dull experimental setup, so don’t describe it. If necessary, camouflage this absence with brief, high-level details of insignificant aspects of your methodology.
2. **Be abstract.** Pseudo-code is a great way of communicating ideas quickly and clearly while giving readers no chance to understand the subtle implementation details (particularly the custom toolchains and manual interventions) that actually make it work.
3. **Short and sweet.** Any limitations of your methods or proofs will be obvious to the careful reader, so there is no need to waste space on making them explicit². However much work it takes colleagues to fill in the gaps, you will still get the credit if you just say you have amazing experiments or proofs (with a hat-tip to Pierre de Fermat: “*Cuius rei demonstrationem mirabilem sane detexi hanc marginis exiguitas non caperet.*”).
4. **The deficit model.** You’re *the* expert in the domain, only you can define what algorithms and data to run experiments with. In the unhappy circumstance that your methods do not do well on

¹*N.B.* We are by no means claiming this is an exhaustive list for making your research irreproducible...

²Space saved in this way can be used to cite the critical papers in the field, i.e. those papers that will inflate your own (as well as potential reviewers’) h-index.

community curated benchmarks, you should create your own bespoke benchmarks and use those (and preferably not make them available to others).

5. **Don't share.** Doing so only makes it easier for other people to scoop your research ideas, understand how your code actually works³ instead of why you say it does, or worst of all to understand that your code doesn't actually work at all.

However, our most important tip is deceptively but beautifully simple: **to ensure your work is irreproducible, make sure that you cannot reproduce it yourself.** If you were able to reproduce it, there would always be the danger of somebody else being able to do exactly the same as you. Much else follows from this; for example, complete confidence in your own inability to reproduce work saves tedious time revising your work on advice from reviewers: if you are unable to browbeat the editor into accepting it as is, you can always resubmit elsewhere. A major advantage of this key insight is that no strict discipline is required to ensure self-irreproducibility: in our experience, irreproducibility can happily occur after only the tiniest amount of carelessness at one of any number of stages.

We make a simple conjecture: **an experiment that is irreproducible is exactly equivalent to an experiment that was never carried out at all.** The happy consequences of this conjecture for experts in irreproducibility will be published elsewhere, with extremely impressive experimental support.

We close with a mantra for scientists interested in irreproducibility:

After Publishing Research, Irreproducibility Lets False Observations Obtain Longevity!

References

- [1] Andreas Prlić and James B. Procter. Ten Simple Rules for the Open Development of Scientific Software. *PLoS Computational Biology*, 12(8), 2012.
- [2] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10), 2013.
- [3] Ian P. Gent. The Recomputation Manifesto. Available from: <http://arxiv.org/abs/1304.3674>, April 2013.
- [4] Lucas N. Joppa, Greg McInerny, Richard Harper, Lara Salido, Kenji Takeda, Kenton O'Hara, David Gavaghan, and Stephen Emmott. Troubling Trends in Scientific Software Use. *Science*, 340(6134):814–815, 2013.
- [5] Ian P. Gent and Lars Kotthoff. Recomputation.org: Experience of its first year and lessons learned. In *Recomputability 2014*, December 2014.
- [6] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best Practices for Scientific Computing. *PLoS Biology*, 12(1), 2014.
- [7] Carole Goble. Better Software, Better Research. *IEEE Internet Computing*, 18(5):4–8, 2014.
- [8] Tom Crick, Benjamin A. Hall, and Samin Ishtiaq. “Can I Implement Your Algorithm?”: A Model for Reproducible Research Software. In *Proceedings of 2nd International Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE2)*, 2014.
- [9] Tom Crick, Benjamin A. Hall, Samin Ishtiaq, and Kenji Takeda. “Share and Enjoy”: Publishing Useful (and Usable) Scientific Models. In *Proceedings of 1st International Workshop on Recomputability*, 2014.
- [10] Victoria Stodden and Sheila Miguez. Best Practices for Computational Science: Software Infrastructure and Environments for Reproducible and Extensible Research. *Journal of Open Research Software*, 2(1):1–6, 2014.

³An exemplary example: <http://www.phdcomics.com/comics.php?f=1689>