We need to talk about the lack of investment in digital research infrastructure

Research software infrastructure is critical for accelerating science, and yet, these digital public goods are often unsustainably funded. Solving this problem requires an appreciation of the intrinsic value of research software outputs, and greater investment of time and effort into effectively funding maintenance of software at scale.

Rebecca Knowles, Bilal A. Mateen and Yo Yehudi

Foundational research software infrastructure, such as Matplotlib¹, NumPy², Pandas³, Machine Learning in R (MLR)⁴ and so forth, is critical for accelerating science. Yet, these digital public goods that make science possible are often unsustainably funded, and the communities of people who build and maintain them are under-represented when it comes to apportioning credit⁵.

When achieving the impossible isn't enough to justify funding

To genuinely understand what's at stake, we only need to look a few years into the past. Take the team who built the pipeline (based on several of the aforementioned Python packages) to stitch together images from the Event Horizon Telescope, resulting in the first ever image of a black hole in April 2019⁶. The image caught the attention of lay and scientific communities on a global scale, and the discovery itself was described as something "presumed impossible just a generation ago"⁷. However, few were interested in the software underpinning the image to the extent that, within five days of the image being breaking news, the US National Science Foundation denied them further funding to support, maintain and develop the software since it did not have a "sufficient impact"⁸. Fortunately, the team persisted, and their code appears to still be maintained⁹, but what happens when groups can't do this work pro bono (even if only for a short time until the next grant can be secured)?

In short, the answer is, research suffers. In 2015, two thirds of 3,900 genomic papers surveyed were using outdated software tools and databases that provided only 26% of the biological processes that an up-to-date resource would have held¹⁰. In 2016, thousands of developers, including scientific researchers, found their projects hamstrung after an eleven-line code package was withdrawn from distribution¹¹. There is arguably no better summary of how

precarious our current circumstances are than a recent XKCD comic pointing out that much of our current digital infrastructure has numerous dependencies on software developed by people with few resources available to update or maintain their work. In essence, the widespread use of high impact yet under-supported research software means that the entire scientific endeavor is perpetually balanced on a knife edge, which begs the question: are we at least doing enough to prevent a disaster?

Unfortunately, the academic funding model wasn't conceived to support software-based tools or to maintain digital infrastructure, and thus clearly needs substantial changes to acknowledge the computing-heavy nature of modern research. It isn't that our model is broken in some spectacularly obvious way: we aren't staring down the barrel of a gun or watching a train hurtle towards the end of a track. Instead, the problem is much more insidious. It comprises a thousand little injustices to which academics who pursue any form of software engineering have had to adapt. This is exemplified by the paucity of opportunities for maintenance funding, which forces academics to regularly reinvent their tools to compete for grant funding, or make trendy-sounding changes (for example, migrating from a relational database to a graph database) to justify further funding that can cover maintenance such as bug fixes, security fixes, and user support. In extremis, this has manifested as a framing of tools as disposable utensils by which to achieve an outcome, thereby perpetuating a wasteful cycle of having to rebuild the same thing multiple times.

Cows, code and free rides

Though this isn't a new problem, the overuse and exploitation of public goods (that is, 'commons') has been extensively studied. Arguably, the 'tragedy of the commons' is still best understood using the example that Hardin described in his seminal piece in

1968¹², in which a public resource (a field) is depleted by individuals acting independently according to their own self-interest (for example, cattle herders allowing their cows to graze in a limitless and unsustainable manner). Many argue that this is less relevant in the context of a digital commons where public goods are infinitely abundant and the use of a digital object does not result in its degradation (as occurs in Hardin's example)¹³. However, digital goods are often only useful while they are being supported, and thus there is a human cost implicit in their maintenance¹⁴.

Effectively, the community at large are free-riding on the time and resources invested by open-source developers to maintain critical digital tools. While many know the burden of maintenance to be considerable, as demonstrated by the fact that the vast majority of open-source codebases are abandoned¹⁵, quantifying it (whether in terms of academic, or financial opportunity cost) is non-trivial. This is because much of this work is undertaken as part of an informal labor market comprising small communities of practice that have coalesced around vital infrastructure. As such, although there are several back-of-a-napkin estimates for the true cost of open source in specific settings¹⁶, as far as we are aware, an authoritative estimate of the global contribution to open-source development is still lacking. Fortunately, several groups, such as the Core Infrastructure Initiative, have stepped in to address the aforementioned market failure¹⁷. However, there is still a lot left to do, in large part due to the ubiquity of open-source code; a recent report found that, of 1,250 commercial codebases audited, 99% contained at least one open-source component¹⁵.

Naming the beast

In short, research relies on software — seven out of ten researchers believe they would be unable to work without some

form of it¹⁸ — and yet, the systems in place to credit and support those who write the code and build the tool are insufficient. To make research software sustainable, we must adapt our credit and reward system, and ensure that we treat software as not only something that underpins research, but also as a first-class output. It needs to be funded, maintained and have viable career paths even if the researchers involved are writing more lines of computer code than lines in an academic manuscript. Researchers shouldn't have to choose between producing reproducible high-quality code and career progression. Moreover, the risk from continuing down this road doesn't only stem from continuing to disincentivize a necessary part of the modern scientific endeavor (that is, software development and coding), but rather from having vast swathes of the scientific literature potentially becoming even more unreproducible as the infrastructure that made the analysis possible falls into disrepair. Disaster doesn't require the identification of a major vulnerability¹⁹; the continued viability of any of these major resources is perpetually at risk by the mere fact that we (as a community) are largely relying on commercial entities or the goodwill of open-source volunteers to pick up the bill for maintenance. Until we collectively acknowledge the need for better support structures around computational research, the status quo will persist... so where do we go from here?

Fundamentally changing the narrative around digital infrastructure

The idea of reimagining the research landscape to recognize the importance of maintenance funding and to better apportion credit to software developers and engineers is not new²⁰. There are several other funders working in this setting (Chan-Zuckerberg Initiative²¹, Ford Foundation¹⁹ and so forth), as well as many individuals and academic groups that have acted as trailblazers by shining a light on this issue.

Drawing on these influences, below we describe two major changes to the landscape that funders can help realize to ensure the contributions of people who write code in an academic setting are recognized in similarly robust ways to wet-lab scientists:

1. Funders need to start by agreeing on the premise that these tools have intrinsic value — not necessarily because they are methodologically novel, but because they address a clear need in the community. But saying it isn't enough: while recognition is certainly the first

step, it is only through commitments to addressing the disproportionately small amount of funding that tools and software receive that we demonstrate our commitment to this principle. However, we appreciate that there are open questions that will need to be addressed as part of these efforts, such as which tools are 'worthy and/or deserving' of continued support. Although lots of progress has already been made on this topic, for example, by the Linux Foundation-funded Census Program¹⁷, a key outstanding challenge is how we meaningfully evaluate requests for maintenance funding alongside new research proposals, because ultimately, funders' resources are constrained, and hard decisions will need to be made — what we need to ensure is that these are the right decisions.

2. By and large, the answer to questions around long-term sustainability of research software has either focused on commercialization, or a roadmap to the next grant, thus exposing a fundamental tension that needs to be resolved. Namely, while funding of tools is currently limited to the duration of a specific grant, the maintenance implications for the developers extends far beyond this timescale (that is, maintenance doesn't end when the grant does). However, even if we all agree that maintenance is important, funding in perpetuity is unlikely to be feasible for most funding organizations. As such, if computational science is to be open and sustainable, then we (as funders) have a responsibility to help engineer alternative solutions to this issue — or we risk undermining the academic community every time a piece of vital research software infrastructure fails to attract additional funding.

In short, science has been too singularly focused on innovation, otherwise we would not see the maintenance of foundational digital infrastructure as a lesser task within the hierarchy of academic outputs. However, fundamentally changing the narrative around how funders support digital infrastructure requires us to do more than point out that there is a problem. In short, each and every funder needs to speak up and act, or we risk our silence being interpreted by a generation of data scientists and software engineers as an endorsement of the status quo.

At the Wellcome Trust, we intend to address this situation, starting by making sure that the tools we fund are open source, well documented, and sufficiently robust

that people can maintain and reuse them without restriction. As we begin to explore how to fund maintenance projects to make preexisting tools more sustainable, as well as codify a way of working that recognizes research software engineering as a discipline in its own right, we want to learn, fail and succeed out in the open. But we can't define success alone, nor can we achieve it without the support of the community at large. We hope that in writing this piece that we'll enfranchise members of the community to reach out and share with us their proposed solutions to the challenges we've articulated, and to help broaden our horizons by pointing out the emerging issues experienced in different parts of the vast computational science community that we may not know about. □

Rebecca Knowles ¹, Bilal A. Mateen ^{1,2,3} ✉ and Yo Yehudi ^{1,4,5}

¹The Wellcome Trust, Data for Science and Health, London, UK. ²The Alan Turing Institute, London, UK. ³University College London, Institute of Health Informatics, London, UK. ⁴University of Manchester, Department of Computer Science, Manchester, UK. ⁵Open Life Science.

✉e-mail: b.mateen@wellcome.org

Published online: 25 March 2021
<https://doi.org/10.1038/s43588-021-00048-5>

References

1. Hunter, J. D. *Comput. Sci. Eng.* **9**, 90–95 (2007).
2. Lang, M. et al. *J. Open Source Softw.* **4**, 1903 (2019).
3. Reback, J. et al. *Zenodo* <https://doi.org/10.5281/zenodo.3509134> (2020).
4. Harris, C. R. et al. *Nature* **585**, 357–362 (2020).
5. Casari, A. et al. *Nat. Comput. Sci.* **1**, 2 (2021).
6. Akiyama, K. et al. *Astrophys. J. Lett.* **875**, L3 (2019).
7. *Astronomers Capture First Image of a Black Hole* (National Science Foundation, 2019); https://www.nsf.gov/news/news_summ.jsp?cntn_id=298276
8. Nowogrodzki, A. *Nature* **571**, 133–135 (2019).
9. Chael, A. *GitHub* <https://github.com/achael/ehi-imaging> (2020).
10. Wadi, L., Meyer, M., Weiser, J., Stein, L. D. & Reimand, J. *Nat. Methods* **13**, 705–706 (2016).
11. Williams, C. *The Register* https://www.theregister.com/2016/03/23/npm_left_pad_chaos/ (2016).
12. Hardin, G. *J. Nat. Resour. Policy Res.* **1**, 243–253 (2009).
13. Nagle, F. *The Digital Commons: Tragedy or Opportunity? A Reflection on the 50th Anniversary of Hardin's Tragedy of the Commons* (Harvard Business School, 2018).
14. Greco, G. M. & Floridi, L. *Ethics Inform. Technol.* **6**, 73–81 (2004).
15. Synopsys <https://www.synopsys.com/software-integrity/resources/analyst-reports/2020-open-source-security-risk-analysis.html> (2020).
16. *The Hidden Costs of Open Source* (IBM, 2012).
17. Nagle, F., Wilkerson, J., Dana, J. & Hoffman, J. L. (2020). *Vulnerabilities in the Core: Preliminary Report and Census II of Open Source Software* (The Linux Foundation & The Laboratory for Innovation Science at Harvard, 2020).
18. Hettrick, S. et al. *Zenodo* <https://doi.org/10.5281/zenodo.1183562> (2014).
19. Eghbal, N. *Roads and Bridges: The Unseen Labor Behind our Digital Infrastructure* (Ford Foundation, 2016).
20. *Wellcome Science Review 2020* (Wellcome Trust, 2020).
21. *Essential Open Source Software for Science, Cycle 3* (Chan Zuckerberg Initiative, 2020).

Acknowledgements

The authors would like to thank T. Khokhar (head of the Data for Science and Health team at the Wellcome Trust)

for his support in formulating the ideas that informed this piece, as well as H.-R. Hotz and C. Mungall who assisted with the identification of references for outdated genomic software resources. There was no direct funding for this Comment; instead, the authors were financially supported via their status as employees of the Wellcome Trust. As a

fellow at the Alan Turing Institute, B.A.M. is also indirectly supported by the core EPSRC grant EP/N510129/.

Author contributions

B.A.M. conceptualized the piece and wrote the first draft. The draft was substantially revised by R.K. and Y.Y.

All authors reviewed and agreed to submit the final version of the manuscript.

Competing interests

The authors declare no competing interests.