A FAIR Case for a Live Computational Commons

Cyrus Omar
University of Michigan
Michigan, USA
comar@umich.edu

Michael Coblenz
University of California at San Diego
San Diego, USA
mcoblenz@ucsd.edu

Anil Madhavapeddy
University of Cambridge
Cambridge, United Kingdom
avsm2@cam.ac.uk

Abstract

Scientists increasingly write software as part of large-scale collaborative workflows, but current tools make it difficult to follow FAIR principles (findability, accessibility, interoperability, reusability) and ensure reproducibility by default.

This paper proposes Fairground, a computational commons designed as a collaborative notebook system where thousands of scientific artifacts are authored, collected, and maintained together in executable form in a manner that is FAIR, reproducible, and live by default. Unlike existing platforms, Fairground notebooks can reference each other as libraries, forming a single planetary-scale live program executed by a distributed scheduler.

We describe the design of Fair Python, a purely functional subset of Python, and a foreign function interface for interoperating with existing code. Through three interleaved research tracks focusing on language design, interoperability, and distributed execution, we aim to create a next-generation collaborative scientific workflow system that makes best practices the path of least resistance.

CCS Concepts: • **Human-centered computing** → **Scientific visualization**; **Collaborative interaction**; *Open source software*; • **Software and its engineering** → **Collaboration in software development**; **Interoperability**; *Functional languages.*

Keywords: python, reproducible, scientific computing, functional, visualization, fair, reusability

ACM Reference Format:
Cyrus Omar, Michael Coblenz, and Anil Madhavapeddy. 2025. A FAIR Case for a Live Computational Commons. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Programming for the Planet (PROPL '25), October 12–18, 2025, Singapore, Singapore.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3759536.3763802

PROPL '25, Singapore, Singapore

ACM ISBN 979-8-4007-2161-8/25/10
https://doi.org/10.1145/3759536.3763802

1 Introduction

Scientists increasingly write software as contributors to large-scale collaborative workflows. For example, data scientists write software to standardize and analyze data [17, 33, 41, 44] collected by field and experimental scientists. These analyses flow into software written by computational scientists to model natural phenomena, e.g. fish population dynamics [20, 50]. These models may then flow into integrative models, like those developed to predict the impact of ecological interventions and climate policies [13, 15].

Scientists engage in these collaborative computational workflows using a patchwork of:

- programming environments (e.g. spreadsheets, computational notebooks like Jupyter, and IDEs) for various programming languages (e.g. Python, R, Scala, Bash, C++, and Fortran) and libraries (e.g. pandas, tidyverse),
- code and data management systems (e.g. shared storage, databases, version control systems),
- hardware and cloud management systems (e.g. cluster job systems, cloud environments like AWS and Google Earth Engine),
- authoring tools (e.g. LaTeX, Word),
- communication tools (e.g. email, Slack),
- domain-specific graphical applications (e.g. GIS tools, illustration tools, and custom applications).

When using these tools, scientists must put in considerable extra effort to follow identified best practices for data and software in the context of large-scale collaborative workflows, including the **FAIR criteria** for data and software [6, 53] (**findability**, **accessibility**, **interoperability**, **reusability**) and **reproducibility**. By default, artifacts are not FAIR (causing scientists to waste time reinventing the wheel) and not reproducible [23, 32, 46].

1.1 The importance of reproducible open science

Reproducibility is a keystone of scientific discovery but unfortunately, many scientific disciplines are undergoing a replication crisis, including psychology [51], economics [7], and extending to the physical sciences: 70% of researchers reported that they failed to reproduce others' experiments [4]. In systems biology, 49% of published models were not reproducible [48]. A 2023 workshop at the National Center for Atmospheric Research explored the challenges of reproducibility in science that relies on computation, with presenters arguing that new techniques are needed to ensure that software-dependent results can be reproduced [32].

Open science, in which data and analytic methods are made publicly available, is a key technique for improving reproducibility. When scientific work is conducted openly, other scientists and members of the public can study the work, check the analysis for errors, and re-analyze the data using their own methods [53]. Scientists in many domains, including in biology [3], ecology [23], and at NASA [46] have argued that open science is a key technique for improving reproducibility of scientific work and addressing the reproducibility crisis.

1.2 Towards live collaborations

Many fields of science, such as ecology and astronomy, are also *live* collaborations: scientists continuously collect new data and refine modeling and analysis approaches. However, scientific models and analyses as reported in publications and scientific artifact repositories are frozen in time. Considerable extra effort and computational expertise possessed by few working scientists is needed to maintain **liveness** of analyses and models as new data are generated by sensors and experiments, and as upstream models and libraries improve. This leaves scientists and other stakeholders with a fragmented and outdated view of the state of the art [38]. We argue that *liveness*, in which the impact of program changes is visible immediately, promotes reuse because it enables programmers to more easily understand existing code and adapt it to their needs.

We envision an ambitious but practicable transition to a collaborative scientific workflow where thousands of scientists and other stakeholders, assisted in the future by safely sandboxed AI agents [39], work together within a *computational commons*—which we call Fairground—where thousands of scientific artifacts are authored, collected, and maintained together *in executable form* in a manner that is FAIR, reproducible, and live *by default*. This vision goes beyond prior efforts focused on simple archival of data and code to emphasize live code execution and the development of interoperable software components, and echoes other recent calls for integrative scientific workflow systems.

Fairground will take the form of a collaborative computational notebook system that is as easy to use as competing platforms like JupyterLab or Google Earth Engine. It will consist of a collection of *computational notebooks* and *data sources* (including uploaded datasets and references to external datasets and streams) grouped into *repositories* managed by participating individuals and organizations, much like on GitHub or GitLab (and we may indeed use a combination of these as our underlying version control platform).

Unlike competing platforms, Fairground notebooks will be able to reference one another as libraries, collectively forming *a single planetary-scale live program* being collaboratively edited in real-time or with optional lightweight version control, and executing live (i.e. as it is being edited and whenever upstream data or models change) by Fairground's distributed parallelizing scheduler [31], which we call the *planetary compute engine*. Computations will leverage distributed hardware resources semi-automatically, with only the heaviest computations needing manual configuration. Computational resources, including CPUs, GPUs, and storage, will be contributed by participants and sponsors for use according to lightweight access policies (e.g. reserving a scientist's own individual and institutional hardware for the computations they author).

Design research emphasizes the value of reasonable defaults in shaping user behavior [22]. The central design goal that distinguishes Fairground from competing systems is that we aim for artifacts authored in or imported into Fairground to be FAIR, reproducible, and live by default. Fairground is targeted toward the needs of a graduate student without substantial formal training in computing whose primary goal is to finish their next paper, rather than on one willing to invest substantial extra labor into achieving these criteria, which is rarely incentivized or rewarded. Let us give an overview of how the proposed design will achieve this critical goal.

2 Design Considerations for Fairground

Our vision is inspired by the success of other collaborative commons [42], particularly Wikipedia [52], GitHub [28], and Google Earth Engine [19].

2.1 Findability and Accessibility

On most existing platforms, notebooks are private by default, and substantial extra effort is needed to make them publicly findable and accessible by others who may benefit from their findings, functions, or pre-computed data (e.g. cleaned up versions of datasets). In Fairground, repositories will be public by default, closely following the example of GitHub, with opt-in access control to allow scientists to work with sensitive data. Each repository will receive a unique and persistent digital object identifier (DOI) [11] and be associated with its authors' ORCID identifiers [40] by default.

Current repositories, such as those for oceanography [49], often include only data, not code, leaving scientists to re-implement analyses. Even when code is available in a repository such as Zenodo [12], it can be difficult to find and integrate into new contexts (e.g. due to outdated dependencies [43]).

In Fairground, the entire system forms a single live program, enabling Fairground to offer *semantic search* affordances such as find-all-uses (e.g. of a dataset, so that a scientist can find others who have already cleaned up that dataset, or of a statistical function, so that a scientist can find examples of similar analyses) and find-all-implementations of a particular interface (e.g. a data schema or a statistical function signature).

In addition to being able to access Fairground resources through a web browser, Fairground will offer an API so external applications can also access computational results. This will be particularly useful because Fairground is a live system: many results on Fairground will update continuously as new data are collected. Live data and visualizations maintained on Fairground might feed into external, public-facing projects, e.g. news articles or websites for public interest organizations; these feeds have been found to promote innovation in news media [2]. Users of these systems will be able to follow these embedded results back to their sources on Fairground, potentially increasing public trust in the results by inviting public scrutiny [36].

2.2 Interoperability, Reusability, and Reproducibility

Computational notebooks typically operate as scripts; they cannot be understood as libraries that can be accessed from other notebooks (without wastefully re-executing them in their entirety in the new context). It requires enormous extra effort for a scientist to create reusable libraries and contribute them to public library repositories like pip [47] or CRAN [1]. Consequently, it is common for scientists to re-implement behavior when they need an analysis that is not already in public package repositories. Reimplementing complex techniques can be error-prone, especially for non-experts who may not appreciate subtleties related to numerical accuracy [18] or leave statistical assumptions unchecked [16]. It is common to then copy-and-paste this functionality into each subsequent notebook where it might be useful, leading to messy and difficult-to-maintain code: 70% of all code snippets in Jupyter notebooks on GitHub are copies of other snippets [25].

On Fairground, every notebook is by default a library that exports its top-level definitions, including data and functions. Because the entire system operates as a single live program, importing one library into another does not require re-execution. This eliminates the process of packaging functionality into explicit libraries and submitting them to repositories—everything is immediately available for reuse across Fairground.

To enable this sort of open reuse, Fairground cannot permit the importing notebook to modify the state of the imported notebook's analyses. Consequently, a central feature of Fairground is the development of Fair Python, a pure (i.e. immutable, functional) dataflow subset of Python that will serve as the native language of Fairground. Pure dataflow is fundamentally the same model used by spreadsheets: programs consist of elements that compute results, potentially on the basis of results produced by other parts of the program. Spreadsheets require that these elements be in a grid and restrict the kinds of computations that can be done; Fair Python will relax these constraints, allowing general-purpose computation. Pure dataflow programming is already in wide use in scientific code beyond spreadsheets, e.g. it is the computational model underlying libraries like `pandas` for Python [34], and most `pandas` code will be expressible in Fair Python without modification. By avoiding ephemeral state, we ensure that Fair Python programs have reproducible outputs.

Of course, we do not expect scientists to (re)write all code in a restricted subset of Python—in reality, scientific code is written in a wide variety of languages, including scripting languages like Python, R, and Julia and native languages like C, C++ and Fortran. While we will design Fair Python to be a highly usable language for new scientific developments, we do not expect scientists working in Fairground to port existing code *en masse* to Fair Python. Instead, we will develop a *pure foreign function interface* (FFI) for Fair Python that adds *foreign nodes* to the dataflow, which support executing code written in any language. To maintain liveness and interoperability, foreign nodes are restricted only in that they execute within a sandboxed environment and their inputs and outputs must be pure data. For example, a node might pass data and parameters from elsewhere on Fairground into an existing optimized numerical simulation package written in Fortran, then specify that the dataset generated when the simulation ends (e.g. saved to a file in the sandboxed file system) is an output value. Each sandboxed environment is configured using the Nix package manager, which produces reproducible execution environments. This solves the common problem scientists face of interoperating with code that requires old or conflicting versions of packages.

To simplify common use cases, Fairground will be able to import existing Jupyter notebooks directly without modification, allowing them to be edited using the Jupyter web interface and automatically provisioning the necessary foreign node internally. Top-level definitions will be exported as outputs, and to maintain reproducibility, Fairground will persistently store the ephemeral state of each notebook. For native packages (those compiled for specific configurations from languages such as C++ or Fortran), we will require Nix configurations, which are written in a simple syntax that is widely documented. These configurations will be part of Fairground, so they can be reused by other scientists.

2.3 Liveness

In traditional Python code, lines of code must be executed in order. This makes it difficult to leverage multicore hardware, which is ubiquitous even on laptops. Fairground eliminates these unnecessary sequential dependencies and automatically distributes computations, and live recomputations, for parallel execution across users' individual cores as well as computational resources contributed by participants. We call this system the Fairground *planetary compute engine* [15].

The commons will centralize thousands of data and code artifacts, which will have permanent identifiers that can be referenced from papers and other permanent artifacts, and

can be searched to find relevant code and data. It will be open-source, enabling long-term maintenance and transparency of policies and behavior. It will support a distributed storage model, enabling both centralized, persistent storage and local replicas for high performance and customized access when needed. It will interoperate with other sources of data, such as NASA's PO.DAAC [24] and support computation in a variety of different languages.

Because Fairground programs will be live, it will seamlessly support reusability; any computation can be run, modified, and re-used in new contexts, even by users who have very different local computational infrastructure.

2.4 Next Steps Towards Live Programming

In summary, we propose three interleaved tracks of research that, together, will result in an operational prototype of a next-generation collaborative scientific workflow system. The proposed research will, in the process, lead to significant advances in the field of human-centered programming language design and implementation. In the first track, we will focus on the design and implementation of Fair Python. In the second, we will focus on interoperability with external data sources and code written in existing languages. In the third, we will advance the underlying planetary compute engine so that Fairground can scale to planetary-size datasets and analyses. We have already developed an operational prototype of the web-based user interface in the Hazel research programming environment [37] and plan, as a plenary approach, to integrate the research advances from all three tracks into this prototype, evolving it directly into Fairground.

To ensure that Fairground is as usable as possible for scientists, all three tracks will follow a *participatory design process*, following best practices for language design in which we iteratively make and evaluate design choices [9] with the specific goal of improving real-world scientific workflows being performed by scientists.

We will leverage our existing connections to involve scientists from several disciplines, with an initial focus on oceanography. In particular, we plan to develop case studies from the work of collaborators in the fields of oceanography and biology, and co-author Anil Madhavapeddy at the University of Cambridge. We will start with laptop-scale case studies from domain-specific data science courses with minimal external dependencies and then continue with full-scale case studies drawn from scientific research papers, ongoing projects, and graduate courses.

3 Discussion and Related Work

Encouragingly, our research direction into FAIR and live planetary computing is complemented by several ongoing efforts across the world, which we will discuss next.

3.1 Future Directions

We are currently working on a working prototype of Fairground, validated through a series of increasingly complex case studies and user studies. We anticipate that Fairground will first be ready for use by early adopters in the scientific community. At this point, we hope to continue development of Fairground by pursuing funding to develop a non-profit governance organization, guided by the TRUST principles [26], that will support both further technical development and, most importantly, develop the community around Fairground. While our initial focus will be on oceanography [21], biodiversity [5, 14] and forest protection [45] related case studies, we will broaden the scope of the project to other scientific communities at this time (and potentially earlier if opportunities arise). Our goal is for Fairground to be the most usable platform for doing day-to-day open science.

We anticipate that AI systems will be capable of assisting in the large-scale ingestion of existing computational notebooks and datasets into the Fairground system toward the end of the funding period, which we believe will help us bootstrap it to the point that it is quickly useful to working scientists rather than just early adopters. We believe that the time is right for scientific workflows to move to a next-generation platform designed with both usability and the FAIR criteria in mind from the start.

3.2 Related Planetary Computing Infrastructure

Our approach is motivated by multidisciplinary calls to develop tools that make it easier to follow FAIR practices for large-scale collaborative science and handle planetary-scale data and computation. Co-author Anil Madhavapeddy and colleagues in conservation made a case for "*planetary computing* — infrastructure to handle the ingestion, transformation, analysis, and publication of global data products for furthering environmental science and enabling better informed policy-making" [15]. This article explains that scientists need access to large petabyte-scale input datasets consisting of:

- primary observation data from satellites that is petabyte-scale or direct ground measurements;
- derived sources from algorithmic transformation or AI-based inference;
- previous results derived by third parties or from earlier runs

Programmers then define computation over these datasets that: (i) is either algorithmic or machine learning-based, using a mix of CPUs and GPUs; (ii) needs to autoscale to permit local development followed by global analysis; and (iii) can be expressed by a non-CS expert, ideally with a visual interface.

The Global Biodiversity Information Facility observed: "There has been an explosion of data and information, and a concomitant paradigm shift to data-driven research and

specifically biodiversity research and its myriad applications. [...] Redesigned scientific data organizations [...] are developing approaches designed specifically to take advantage of the unprecedented data opportunities in a cooperative framework" [10].

Participants at NCAR's Workshop on Correctness and Reproducibility for Climate and Weather Software [32] sought solutions for reproducibility problems. Running analyses on multiple systems results in discrepancies; our approach enables scientists to reuse common computations and results. At the 1st Programming for the Planet workshop [27], Cimadevilla argued [8] that acquisition and processing of data from heterogeneous data sources, followed by reuse of model outputs, is a key component of many scientific tasks.

Large enterprises have also recognized the need for commonly available computational infrastructure for earth science. In particular, Google offers the Google Earth Engine product [19] and Microsoft offers the Microsoft Planetary Computer [29] product upon request to environmental scientists. These tools allow scientists to develop computational notebooks and execute them using cloud hardware that has access to a common repository of data sets and software packages, much like our proposed system. However, as explained above, notebooks in these tools are not **FAIR**, not **reproducible**, and not **live**. The research described here may help these organizations address these problems for their proprietary planetary compute engines. We will instead lay the intellectual groundwork for a fully open complement to these platforms. Encouragingly, we can build upon emerging open and federated standards being established by initiatives such as OpenEO [30], which defines APIs independent of any single cloud provider or organization for the processing of earth observation data.

The most closely related effort to build a computational commons for scientists is the Observable community [35], which hosts a computational commons with support for **live** evaluation within individual notebooks (though not across notebooks). Notebooks can refer to explicitly exported values in other notebooks, which must be pure. The analyses must be written in JavaScript and run in the user's browser. While there is some support for controlling side effects, these are opt-in, so the system lacks **reproducibility** and automatic scalability. Although Observable does not meet our criteria, it demonstrates that it is possible to attract communities of users to a well-designed and curated computational commons.

References

[1] Institute for Statistics and Mathematics. The comprehensive r archive network. https://cran.r-project.org, 2025.

[2] Aitamurto, T., and Lewis, S. C. Open innovation in digital journalism: Examining the impact of open apis at four news organizations. *New media & society 15*, 2 (2013), 314–331.

[3] Allen, C., and Mehler, D. M. Open science challenges, benefits and tips in early career and beyond. *PLoS biology 17*, 5 (2019), e3000246.

[4] Baker, M. 1,500 scientists lift the lid on reproducibility. *Nature 533*, 7604 (2016).

[5] Ball, T., Dales, M., Eyres, A., Green, J., Madhavapeddy, A., Williams, D., and Balmford, A. Quantifying the impact of the food we eat on species extinctions, feb 2025.

[6] Barker, M., Chue Hong, N. P., Katz, D. S., Lamprecht, A.-L., Martinez-Ortiz, C., Psomopoulos, F., Harrow, J., Castro, L. J., Gruenpeter, M., Martinez, P. A., and Honeyman, T. Introducing the FAIR Principles for research software. *Scientific Data 9*, 1 (2022), 622.

[7] Camerer, C. F., Dreber, A., Forsell, E., Ho, T.-H., Huber, J., Johannesson, M., Kirchler, M., Almenberg, J., Altmejd, A., Chan, T., Heikensten, E., Holzmeister, F., Imai, T., Isaksson, S., Nave, G., Pfeiffer, T., Razen, M., and Wu, H. Evaluating replicability of laboratory experiments in economics. *Science 351*, 6280 (2016), 1433–1436.

[8] Cimadevilla, E. The programming challenges of climate data analysis. In *Programming for the Planet* (2024).

[9] Coblenz, M., Kambhatla, G., Koronkevich, P., Wise, J. L., Barnaby, C., Sunshine, J., Aldrich, J., and Myers, B. A. PLIERS: A process that integrates user-centered methods into programming language design. *ACM Trans. Comput.-Hum. Interact. 28*, 4 (jul 2021).

[10] CODATA, t. C. o. D. o. t. I. S. C., Pfeiffenberger, H., Uhlir, P., and Hodson, S. Twenty-Year Review of GBIF, May 2020.

[11] DOI Foundation. What is a DOI. https://www.doi.org/the-identifier/what-is-a-doi/, 2022.

[12] European Organization For Nuclear Research, and OpenAIRE. Zenodo, 2013.

[13] Eyres, A., Ball, T., Dales, M., Swinfield, T., Arnell, A., Baisero, D., Durán, A. P., Green, J., Green, R. E., and Madhavapeddy, A. LIFE: A metric for quantitively mapping the impact of land-cover change on global extinctions. *Cambridge Open Engage* (2024).

[14] Eyres, A., Ball, T. S., Dales, M., Swinfield, T., Arnell, A., Baisero, D., DurÃ¡N, A. P., Green, J. M. H., Green, R., Madhavapeddy, A., and Balmford, A. LIFE: A metric for mapping the impact of land-cover change on global extinctions. *Philosophical Transactions of the Royal Society B: Biological Sciences 380*, 1917 (jan 2025), 1–13.

[15] Ferris, P., Dales, M., Jaffer, S., Holcomb, A., Scott, E. T., Swinfield, T., Eyres, A., Balmford, A., Coomes, D., Keshav, S., and Madhavapeddy, A. Planetary computing for data-driven environmental policy-making, 2024.

[16] Gardenier, J., and and, D. R. The misuse of statistics: Concepts, tools, and a research agenda. *Accountability in Research 9*, 2 (2002), 65–74.

[17] Goble, C. Better software, better research. *IEEE Internet Computing 18*, 5 (2014), 4–8.

[18] Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM computing surveys (CSUR) 23*, 1 (1991), 5–48.

[19] Google. Google earth engine, 2024.

[20] Guibourd de Luzinais, V., du Pontavice, H., Reygondeau, G., Barrier, N., Blanchard, J. L., Bornarel, V., Büchner, M., Cheung, W. W. L., Eddy, T. D., Everett, J. D., Guiet, J., Harrison, C. S., Maury, O., Novaglio, C., Petrik, C. M., Steenbeek, J., Tittensor, D. P., and Gascuel, D. Trophic amplification: A model intercomparison of climate driven changes in marine food webs. *PLOS ONE 18*, 8 (08 2023), 1–23.

[21] Haine, T. W. N., Gelderloos, R., Jimenez-Urias, M. A., Siddiqui, A. H., Lemson, G., Medvedev, D., Szalay, A., Abernathey, R. P., Almansi, M., and Hill, C. N. Is computational oceanography coming of age? *Bulletin of the American Meteorological Society 102*, 8 (Aug. 2021), E1481–E1493.

[22] Jachimowicz, J. M., Duncan, S., Weber, E. U., and Johnson, E. J. When and why defaults influence decisions: A meta-analysis of default effects. *Behavioural Public Policy 3*, 2 (2019), 159–186.

[23] Jenkins, G. B., Beckerman, A. P., Bellard, C., Benítez-López, A.,

Ellison, A. M., Foote, C. G., Hufton, A. L., Lashley, M. A., Lortie, C. J., Ma, Z., Moore, A. J., Narum, S. R., Nilsson, J., O'Boyle, B., Provete, D. B., Razgour, O., Rieseberg, L., Riginos, C., Santini, L., Sibbett, B., and Peres-Neto, P. R. Reproducibility in ecology and evolution: Minimum standards for data and code. *Ecology and Evolution 13*, 5 (2023), e9961.

[24] Jet Propulsion Laboratory. Physical oceanography distributed active archive center, 2025.

[25] Källén, M., and Wrigstad, T. Jupyter notebooks on GitHub: Characteristics and code clones. *The Art, Science, and Engineering of Programming 5* (2021).

[26] Lin, D., Crabtree, J., Dillo, I., Downs, R. R., Edmunds, R., Giaretta, D., De Giusti, M., L'Hours, H., Hugo, W., and Jenkyns, R. The TRUST principles for digital repositories. *Scientific Data 7*, 1 (2020), 1–5.

[27] Madhavapeddy, A., and Orchard, D. Programming for the planet, 2024.

[28] Microsoft. GitHub, 2024.

[29] Microsoft. Microsoft planetary computer, 2024.

[30] Mohr, M., Pebesma, E., Dries, J., Lippens, S., Janssen, B., Thiex, D., Milcinski, G., Schumacher, B., Briese, C., Claus, M., et al. Federated and reusable processing of earth observation data. *Scientific Data 12*, 1 (2025), 194.

[31] Murray, D. G., Schwarzkopf, M., Smowton, C., Smith, S., Madhavapeddy, A., and Hand, S. CIEL: A universal execution engine for distributed data-flow computing. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)* (mar 2011).

[32] National Center for Atmospheric Research. Workshop on correctness and reproducibility for climate and weather software, 2023.

[33] Nguyen-Hoan, L., Flint, S., and Sankaranarayana, R. A survey of scientific software development. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA, 2010), ESEM '10, Association for Computing Machinery.

[34] NumFOCUS, Inc. pandas. https://pandas.pydata.org, 2025.

[35] Observable. Observable. https://www.observablehq.com.

[36] O'Hara, K. Transparency, open data and trust in government: shaping the infosphere. In *Proceedings of the 4th Annual ACM Web Science Conference* (New York, NY, USA, 2012), WebSci '12, Association for Computing Machinery, p. 223–232.

[37] Omar, C. Hazel. https://hazel.org, 2025.

[38] Omar, C., Aldrich, J., and Gerkin, R. C. Collaborative infrastructure for test-driven scientific model validation. In *Companion proceedings of the 36th international conference on software engineering* (2014), pp. 524–527.

[39] Omar, C., Ferris, P., and Madhavapeddy, A. Modularizing reasoning about AI capabilities via abstract dijkstra monads. In *the 12th ACM SIGPLAN Workshop on Higher-Order Programming with Effects (HOPE)* (sep 2024).

[40] Open Researcher and Contributor ID. ORCID. https://orcid.org, 2025.

[41] Prabhu, P., Jablin, T. B., Raman, A., Zhang, Y., Huang, J., Kim, H., Johnson, N. P., Liu, F., Ghosh, S., Beard, S., Oh, T., Zoufaly, M., Walker, D., and August, D. I. A survey of the practice of computational science. In *State of the Practice Reports* (New York, NY, USA, 2011), SC '11, Association for Computing Machinery.

[42] Rifkin, J. *The zero marginal cost society: The internet of things, the collaborative commons, and the eclipse of capitalism.* St. Martin's Press, 2014.

[43] Sauro, H. M. The practice of ensuring repeatable and reproducible computational models, 2021.

[44] Storer, T. Bridging the chasm: A survey of software engineering practice in scientific programming. *ACM Comput. Surv. 50*, 4 (aug 2017).

[45] Swinfield, T., Shrikanth, S., Bull, J., Madhavapeddy, A., and zu Ermgassen, S. Nature-based credit markets at a crossroads. *Nature Sustainability* (aug 2024), 1–4.

[46] The National Aeronautics and Space Administration. Open science at NASA, 2024.

[47] The Python Packaging Authority. pip 25.0.1. https://pypi.org/project/pip/, 2025.

[48] Tiwari, K., Kananathan, S., Roberts, M. G., Meyer, J. P., Sharif Shohan, M. U., Xavier, A., Maire, M., Zyoud, A., Men, J., Ng, S., Nguyen, T. V. N., Glont, M., Hermjakob, H., and Malik-Sheriff, R. S. Reproducibility in systems biology modelling. *Molecular systems biology 17*, 2 (2021), e9982.

[49] U.S. National Science Foundation. OCE data and sample repositories. https://www.nsf.gov/geo/oce/data-sample-repositories, 2025.

[50] van Denderen, D., Maureaud, A. A., Andersen, K. H., Gaichas, S., Lindegren, M., Petrik, C. M., Stock, C. A., and Collie, J. Demersal fish biomass declines with temperature across productive shelf seas. *Global Ecology and Biogeography 32*, 10 (2023), 1846–1857.

[51] Wiggins, B. J., and Christopherson, C. D. The replication crisis in psychology: An overview for theoretical and philosophical psychology. *Journal of Theoretical and Philosophical Psychology 39* (2019), 202–217.

[52] Wikipedia Contributors. Wikipedia, 2024.

[53] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., and Bourne, P. E. The FAIR guiding principles for scientific data management and stewardship. *Scientific data 3*, 1 (2016), 1–9.

Received 2025-07-08; accepted 2025-08-11