Comment

https://doi.org/10.1038/s41562-026-02570-w

Improving computational reproducibility in the social sciences

Michael V. Reiss, Susanne J. Adler, Christopher Barrie, Jana B. Berkessel, Johannes Breuer, Abel Brodeur, Garret Christensen, Tobias Dienlin, Jeremy Freese, Ben Greiner, Mario Haim, Tom E. Hardwicke, Magnus Johannesson, Gary King, Frauke Kreuter, Christopher Lucas, Ana Martinovici, Christophe Pérignon, Marko Sarstedt, Felix D. Schönbrodt, Oleg Urminsky, Wouter van Atteveldt, Simine Vazire, Lars Vilhuber, Florian von Wangenheim, Eric-Jan Wagenmakers, Nicolas Pröllochs, Claire Robertson, Felix Holzmeister, Stefan Feuerriegel & Hauke Roggenhamp

Code underlying published findings in the social sciences often fails to reproduce reported results when others re-run it on the original data. This Comment proposes four recommendations to strengthen computational reproducibility, facilitating the trustworthiness and reliability of research.

Wherever empirical findings rely on code and data, computational reproducibility is a basic requirement for their credibility. Given the code, data and documentation that authors share, an independent researcher should be able to run the code on the shared data and obtain the same results. In practice, however, computational reproductions often fail: reproducibility rates range from below 20% of papers if the absence of available code and data is counted as a failed reproduction[1], to over 85% of claims at journals that mandate code and data sharing[2]. While higher rates partly reflect improvements in data-sharing policies and their enforcement, failures are documented even under the most favourable conditions. Each failure to reproduce results can create undesirable uncertainty regarding the credibility of the original findings[1], and future research risks building on unstable foundations[3].

Computational reproducibility often breaks not because of major errors, but because of routine omissions that researchers may not realize are problematic: for example, an analytical step performed by hand with no record left behind, a package that has since been updated, or a data file that was never labelled. Many such failures persist because the practices that prevent them are not yet widespread. This Comment describes four such practices and explains why their adoption matters.

Why computational reproducibility is important

We consider two checks to evaluate a finding's credibility: replications where researchers test whether an original finding can be repeated using new data, and computational reproductions where researchers test whether an original finding can be repeated using the original code, data and documentation[4] (Fig. 1). Whereas replicability has attracted much attention in the behavioural and social sciences over the past decade[5], we argue that computational reproducibility is also important, as findings that cannot be computationally reproduced impose costs on both the scientific community and individual researchers.

At the collective level, reproducible workflows mean that errors are more likely to be detected early and less likely to spread through the literature. Without this foundation, computational errors can go unnoticed, be cited and even be included in meta-analyses. When such errors surface, authors must issue corrections or papers are retracted, which erodes trust in science and wastes limited resources on work that may have no meaningful empirical foundation[3]. Reproductions, therefore, constitute a sensible first step in the process of verifying scientific claims.

For individual researchers, computational reproducibility makes research easier to manage, verify and extend. An analysis that neglects principles of reproducibility can quickly become difficult to interpret, even for its original author, making it harder to respond to reviewer requests months or even years later. Without a reproducible workflow, authors are also less likely to catch errors before submission, increasing the risk of corrections once findings are published and the reputational consequences these entail. Yet much of this is avoidable, as reproducibility lies largely within the author's control. With journals increasingly implementing reproducibility checks to bolster the credibility of published findings, authors face growing incentives to assume that responsibility.

Four recommendations for reproducible workflows

From our collective experience conducting and advocating reproducibility checks across various capacities and disciplines, we offer four recommendations to authors: script, encapsulate, explain and deposit (SEED). Table 1 summarizes concrete practices across these four recommendations to prevent common sources of failure. In their simplest form, our recommendations translate into a clear project structure in which raw and processed data, analysis scripts and outputs are stored in dedicated directories, and where a README file explains how everything fits together. A useful way to approach these recommendations is to write code and documentation as if someone unfamiliar with the project would need to run it — whether a reviewer, another researcher building on the work, a future collaborator, or even the original author returning to the project later.

Script the entire analysis pipeline. Computational reproducibility improves when the entire analysis, from reading raw data to producing the final figures and tables, is expressed as executable code, ideally involving no manual steps. Clicking through interfaces that leave no record of user actions, filtering data in a spreadsheet before import, and renaming files by hand all make it more difficult to verify the work and to distinguish deliberate choices from errors. A fully scripted workflow

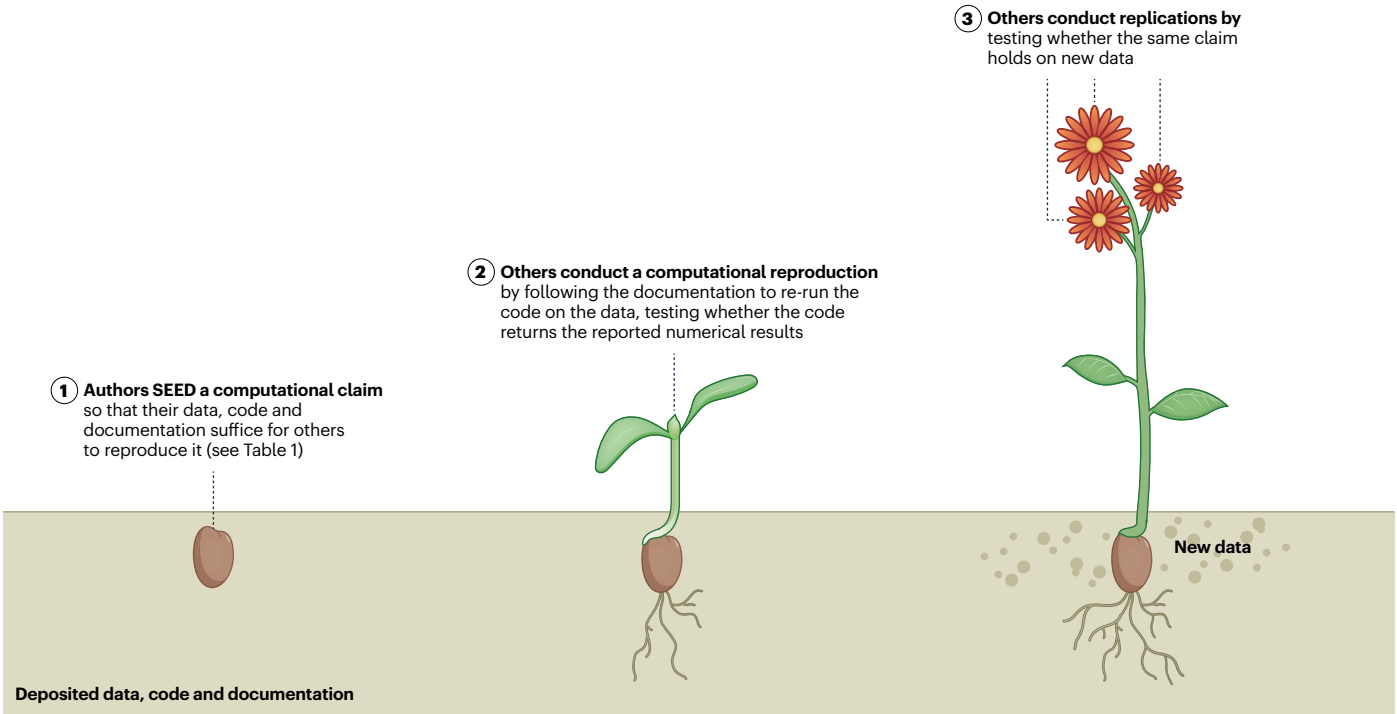

Fig. 1 | Growing credibility from data, code and documentation. A computational claim is the seed, and the data, code and documentation that authors deposit are the substrate that must sustain it. (1) Authors SEED a computational claim (script, encapsulate, explain, deposit; Table 1), and these practices determine whether the deposited materials suffice for someone else to computationally reproduce the reported results; that is, to obtain them by re-running the deposited code on the deposited data. (2) An independent computational reproduction tests whether the materials suffice for reproduction. When it succeeds, a claim's credibility can grow. (3) Replications test the same claim against new data where feasible and provide independent evidence that may strengthen the claim further.

(from data retrieval to final analysis results) avoids this by making every step explicit. Graphical user interfaces (for example, IBM SPSS Statistics, jamovi or JASP) that export a human- and machine-readable syntax file or action log can meet this standard as well.

We highlight two practical suggestions in this context. First, the analysis should have a single, clear entry point: a controller script that calls all other scripts, functions or code chunks in the correct order. This removes any ambiguity about the sequence in which the analysis is executed. Second, analyses that involve stochastic processes (such as bootstrap resampling, cross-validation or simulations) should set and report a fixed random seed to permit reproducibility (Table 1). Without a seed, each run of the code produces different results, making verification more ambiguous.

Encapsulate the computational environment. Code that runs on one computer may break on another. Two common reasons for this are differences in file paths and in software versions that exist only on the author's computer. Hard-coded, absolute paths are not only frequent but also avoidable: replacing them with paths relative to the project root is a simple way to make an analysis more portable. A related challenge is that an analysis usually requires a variety of software packages that the author has installed on their system. However, the same analysis might break on a different computer if one of these packages has changed since the original analysis. Each additional package multiplies this risk, so minimizing dependencies improves portability. For those packages that remain, tools such as renv or groundhog in R, or pip freeze in Python, capture exact version information in a lockfile that others can use to recreate it on a different computer. More advanced tools such as containers go further and capture the full computational environment, including the operating system and system libraries[6].

Whichever approach is used, authors should test it by running the analysis on a separate machine or cloud instance (for example, Binder, Google Colab or Posit Cloud), or by asking a colleague to reproduce it from scratch to catch issues they would otherwise miss.

Explain code and data for human readers. A scripted and encapsulated analysis pipeline reflects machine-readable records, but authors should also provide a human-readable complement that explains what the materials contain and how to run the analysis. A study of over 1,000 computational reproducibility tests found that inadequate documentation was the most commonly cited obstacle to successful reproduction[7]. Hence, authors should provide a README file that clearly states how each dataset can be accessed (in particular when data are not included, for example, under access agreements for confidential data), the software and hardware requirements, the expected running time, the analysis procedure and outputs, and any idiosyncrasies (for example, required API keys or manual steps that cannot be scripted).

Furthermore, linking code to manuscript results through inline comments (for example, "`# This block generates Table 2`") can make the code more accessible and results more traceable by allowing others to quickly identify which part of a script corresponds to which output.

Table 1 | Concrete practices to improve computational reproducibility

Script	
Automate analysis pipeline	Script all analysis steps from data retrieval to final analysis results, eliminate GUI interactions that leave no record, and provide a single entry point that runs the full pipeline in order (such as a `run_all.R` or `main.py` file that sources all scripts in sequence). Tools such as Snakemake can assist managing such pipelines.
Set random seeds	Set and document a seed in every script that uses randomization (for example, `set.seed(20261305)` in R or `np.random.seed(20261305)` in Python, placed at the top of each script).
Protect raw data	Store raw data in a dedicated read-only folder and save all processed versions separately (for example, originals in `data/raw/` and cleaned data in `data/processed/`).
Keep intermediate outputs	Save computationally expensive intermediate outputs so downstream scripts can load them.
Encapsulate	
Minimize dependencies	Add a package only when it provides clear value over base language functions (prefer base R or the standard library in Python for simple operations).
Use relative paths	Use paths relative to the project root and avoid OS-specific commands.
Capture and document the environment	Record all package and language versions in a lockfile and note the operating system and critical dependencies in the README file. Use, for example, `groundhog` or `renv` in R or `conda/pip` in Python.
Containerize when warranted	For complex projects, provide a container capturing the full system environment[6]. Docker bundles code and system libraries into a single portable image.
Test before sharing	Run all code in a fresh environment before sharing (ask a colleague to reproduce your results without further instructions or use a cloud instance such as Binder, Google Colab or Posit Cloud).
Explain	
Write a README file	Create a human-readable file, for example, by completing the Social Science Data Editors' template README. State how each dataset can be accessed (in particular, for confidential or restricted data), its provenance (sources, collection or retrieval method, dates and licensing), the software and hardware requirements, environment setup, execution order and expected runtime. See also figure 4 in ref. 7.
Link code to manuscript	Add comments indicating which outputs correspond to tables, figures and other elements in your paper (for example, `# Creates` Table 1 placed directly above the relevant code block).
Create a data dictionary	Document each variable's name, type, values and units (for example, age_years, numeric, 18–65, years). Follow field-specific metadata standards.
Write readable code and comment decisions	Use variable names that make sense to humans. Comment and explain non-obvious choices, transformations and exclusion criteria (for example, `# Log-transform: right-skewed distribution; # Exclude ID=047: failed pre-registered attention check`).
Deposit	
Organize for others	Confirm that the folder structure is intuitive and follows field-specific standards, if available; that file names reflect their contents and position in the workflow; and that variable names are descriptive and consistent for someone unfamiliar with the project.
Deposit in a persistent repository	Archive code, data and documentation in a repository that provides a persistent DOI (such as Dataverse, ResearchBox and Zenodo; see also re3data.org).
Handle restricted data	When data cannot be shared publicly, document how others can access or reproduce them, arrange verifier access at the design stage (for example, in data-sharing agreements or ethics committee applications), consider data enclaves, non-consumptive environments or synthetic data, and include log files from code runs on the original data. Share analysis code regardless.
Declare licence	Add a LICENSE file specifying terms of reuse for code and data separately (see choosealicense.com for code licences such as MIT and creativecommons.org for data licences such as CC-BY-4.0).

Practices are organized under four recommendations: script, encapsulate, explain and deposit (SEED).

Deposit in a dedicated research repository. Researchers frequently state that their code and data are available upon request, yet in practice, such requests often go unanswered or unfulfilled[1,2]. Making materials publicly and permanently available at the time of publication can help to avoid this dependence on author response. Dedicated repositories such as Dataverse, ResearchBox and Zenodo make it straightforward to archive code, data and documentation in ways that others can access and build upon.

Yet data cannot always be shared publicly because of legitimate privacy or legal constraints. When data access is restricted, researchers should document how others can obtain it. For data governed by non-disclosure agreements or collected under ethics committee protocols, researchers should anticipate access to verifying parties (such as data editors) at the design stage: data-sharing agreements and ethics committee applications can explicitly permit a designated verifier to access restricted materials, making confidential reproduction possible even if public release is precluded. Secure research data systems around the world (such as the Secure ANalysis Environment (SANE) in the Netherlands or the Centre d'accès sécurisé aux données (CASD) in France, which are controlled environments in which analysts

run code or queries with no or restricted access to the underlying records) offer a further option for sensitive or copyrighted data[8]. In some cases, authors may be able to generate synthetic data that preserve the statistical properties of the original data, an approach long used to broaden access to sensitive microdata[9]. Notably, these constraints do not justify withholding analysis code, which is almost never subject to the same restrictions.

Practical considerations. Exact numerical reproduction, while desirable, might not always be achievable. Computationally intensive procedures and non-deterministic processes can make it impractical. For example, outputs from proprietary large language models (LLMs) can vary across repeated runs even in their most deterministic setting. In our view, such cases should be documented and, where possible, mitigated: authors could specify an acceptable range of results for non-deterministic analyses and share processed data to enable reproduction from an intermediate stage.

At the same time, not every project requires the same investment in computational reproducibility. A simple regression on a public dataset demands different infrastructure than a multi-stage simulation pipeline on sensitive data. For more complex projects, researchers may even need tools that capture the full computational environment, such as packaging code together with its software dependencies[6].

Our experience suggests that longer-term gains often outweigh initial costs, regardless of project complexity. Well-structured, reproducible workflows can benefit collaboration and make analyses easier to revisit, extend and verify, reducing ad hoc fixes and repeated reconstruction and leading to more efficient, scalable and reliable research over time. Integrating practices that improve computational reproducibility from the beginning, therefore, also benefits individual productivity throughout all stages of a research project.

LLMs and agentic coding systems (for example, Claude Code, Gemini CLI or OpenAI Codex) are an increasingly prominent resource in academic research and may help to improve computational reproducibility. Their value is most apparent on well-defined tasks with clear success criteria, where the author can readily detect errors and hallucinations. Such tasks include generating a README file from a structured template, identifying hard-coded file paths that prevent portability and flagging undeclared software dependencies. Our early experience with such use cases is promising. However, given the rapid evolution of these tools and a lack of systematic evidence on their use at scale, transparent use and close author review remain essential, with ultimate responsibility resting with the authors.

Making reproducibility routine

As institutional expectations around computational reproducibility rise and the infrastructure to support it becomes more accessible, the barriers that once made reproducibility difficult have largely been removed. What remains is mostly within each author's control: building awareness of these practices and making them routine. The SEED practices can be adopted incrementally, and each step makes findings easier to verify, thereby contributing to more reliable science.

Michael V. Reiss[1], Susanne J. Adler[2], Christopher Barrie[3], Jana B. Berkessel[4,5], Johannes Breuer[6,7], Abel Brodeur[8,9], Garret Christensen[10], Tobias Dienlin[11], Jeremy Freese[12], Ben Greiner[13], Mario Haim[1], Tom E. Hardwicke[14], Magnus Johannesson[15], Gary King[16], Frauke Kreuter[17,18,19], Christopher Lucas[20], Ana Martinovici[21], Christophe Pérignon[22,23], Marko Sarstedt[2,24], Felix D. Schönbrodt[25], Oleg Urminsky[26], Wouter van Atteveldt[27], Simine Vazire[28], Lars Vilhuber[29], Florian von Wangenheim[30], Eric-Jan Wagenmakers[31], Nicolas Pröllochs[32], Claire Robertson[33], Felix Holzmeister[34], Stefan Feuerriegel[18] & Hauke Roggenhamp[30]

[1]Department of Media and Communication, LMU Munich, Munich, Germany. [2]Munich School of Management, LMU Munich, Munich, Germany. [3]Department of Sociology, New York University, New York, NY, USA. [4]Mannheim Centre for European Social Research, University of Mannheim, Mannheim, Germany. [5]Open Science Office, University of St. Gallen, St. Gallen, Switzerland. [6]Research Data and Methods, Center for Advanced Internet Studies (CAIS), Bochum, Germany. [7]Department of Social Sciences, University of Duisburg-Essen, Essen, Germany. [8]Department of Economics, University of Ottawa, Ottawa, Ontario, Canada. [9]Institute for Replication, Ottawa, Ontario, Canada. [10]Berkeley Initiative for Transparency in the Social Sciences, University of California, Berkeley, Berkeley, CA, USA. [11]Department of Communication, University of Zurich, Zurich, Switzerland. [12]Department of Sociology, Stanford University, Stanford, CA, USA. [13]Department of Strategy and Innovation, WU Vienna University of Economics and Business, Vienna, Austria. [14]School of Psychology, University of Sydney, Sydney, New South Wales, Australia. [15]Department of Economics, Stockholm School of Economics, Stockholm, Sweden. [16]Institute for Quantitative Social Science, Harvard University, Cambridge, MA, USA. [17]Institute for Statistics, LMU Munich, Munich, Germany. [18]Munich Center for Machine Learning, LMU Munich, Munich, Germany. [19]University of Maryland, College Park, MD, USA. [20]Department of Political Science, Washington University in St. Louis, St. Louis, MO, USA. [21]Department of Marketing Management, Rotterdam School of Management, Erasmus University, Rotterdam, Netherlands. [22]Department of Finance, HEC Paris, Paris, France. [23]Cascad, Paris, France. [24]Faculty of Economics and Business Administration, Babeș-Bolyai University, Cluj-Napoca, Romania. [25]Department of Psychology, LMU Munich, Munich, Germany. [26]Booth School of Business, University of Chicago, Chicago, IL, USA. [27]Vrije Universiteit Amsterdam, Amsterdam, Netherlands. [28]Melbourne School of Psychological Sciences, University of Melbourne, Melbourne, Victoria, Australia. [29]Department of Economics, Cornell University, Ithaca, NY, USA. [30]Department of Management, Technology, and Economics, ETH Zurich, Zurich, Switzerland. [31]Department of Psychology, University of Amsterdam, Amsterdam, Netherlands. [32]Department of Business and Economics, JLU Giessen, Giessen, Germany. [33]Department of Psychology, Colby College, Waterville, ME, USA. [34]Department of Economics, University of Innsbruck, Innsbruck, Austria.

✉e-mail: hauke.roggenkamp@mtec.ethz.ch

Published online: 01 September 2026

References

1. Miske, O. et al. *Nature* **652**, 126–134 (2026).
2. Brodeur, A. et al. *Nature* **652**, 151–156 (2026).
3. King, G. *PS Polit. Sci. Polit.* **28**, 444–452 (1995).
4. Dreber, A. & Johannesson, M. *Econ. Inq.* **63**, 338–356 (2025).
5. Tyner, A. H. et al. *Nature* **652**, 143–150 (2026).
6. Boettiger, C. *ACM Oper. Syst. Rev.* **49**, 71–79 (2015).
7. Pérignon, C. et al. *Rev. Financ. Stud.* **37**, 3558–3593 (2024).
8. van Atteveldt, W., Althaus, S. & Wessler, H. *Polit. Commun.* **38**, 192–198 (2021).
9. Drechsler, J. & Haensch, A. C. *Stat. Sci.* **39**, 221–242 (2024).

Funding
F.K. and S.F. are part of the National Research Data Infrastructure (NFDI) from the German Research Foundation (NFDI 27/1-2026, 460037581).

Competing interests
O.U. is Editor-in-Chief of the *Journal of Consumer Research*, S.V. is Editor-in-Chief of *Psychological Science* and F.v.W. is Editor-in-Chief of the *Journal of Service Research*. L.V. is the Data Editor of the American Economic Association's journals, a compensated position; the views expressed here are his own and do not reflect AEA policy. T.E.H. is Senior Editor for Statistics, Transparency, and Rigor at *Psychological Science*. A.B. leads the Institute for Replication. C.P. is a co-founder of Cascad, a certification agency for scientific code and data. E.-J.W. is a developer of JASP, an open-source, non-commercial, publicly funded software package for Bayesian statistics, and the CEO of JASP Services B.V., a company that supports organizations adopting JASP, with a focus on applications in quality control. The remaining authors declare no competing interests.