Credible science is reproducible science

Hauke Roggenkamp[1], Michael V. Reiss[2,3], Susanne Adler[4], Stefan Feuerriegel[5], Nicolas Pröllochs[6], Claire Robertson[7] & Felix Holzmeister[8]

[1]University of St.Gallen, Institute of Behavioral Science and Technology
[2]Leibniz Institute for Media Research, Hans-Bredow-Institut
[3]Ludwig-Maximilians-Universität München, Institute for Communication Science and Media Research
[4]Ludwig-Maximilians-Universität München, Institute for Market-based Management
[5]Ludwig-Maximilians-Universität München, Institute of Artificial Intelligence in Management
[6]University of Giessen, Department of Business and Economics
[7]Colby College, Extremism and Polarization Lab
[8]University of Innsbruck, Department of Economics

Words: 2383

Reproducibility should be the easiest promise science keeps. When researchers share their data and code, others should be able to regenerate the reported results. This is not a high bar; it simply asks that the numbers in a paper come from the analysis described. Yet, we too often fail to meet even this minimal standard (Trisovic et al. 2022) such that doubts emerge: if we cannot regenerate the reported numbers from the shared materials, something is wrong, and in the worst case, that something might be the results themselves.

Levels of credibility

Transparency provides the foundation for reproducibility. When researchers share their code and data, others can verify results and advance scientific knowledge production (Hussey 2025). Without this foundation (depicted as G in Figure 1), verification requires a leap of faith, and the integrity of the scientific process may suffer (Munafò et al. 2017). We cannot build robust, credible findings on materials we cannot examine.

Verifiability takes different forms (Dreber and Johannesson 2025; Brodeur et al. 2024). *Computational reproducibility* (blue brick in Level 1) means that anyone who runs the authors' code on their data obtains the exact numerical values reported. This minimal requirement verifies whether the code maps to the reported results.

Recreate reproducibility (gray L-shaped brick in Level 1) demands more. It asks whether independent researchers could obtain identical results by recreating the analysis based solely on the authors' data and reported methods, without consulting the original code. This tests whether reported methods actually produce the results (Stark 2018), which matters because peer review typically examines what authors say they did, not what their code actually does.

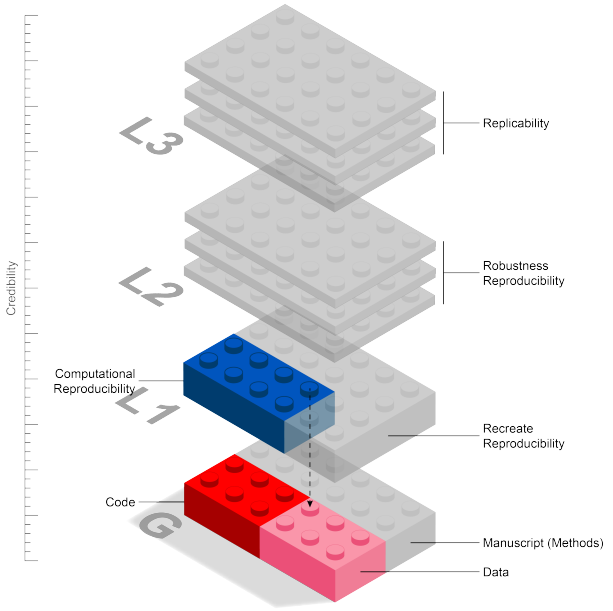

Figure 1. A pragmatic hierarchy for verifying research findings. Description follows.

These two forms of reproducibility should align but sometimes diverge. A project may pass computational reproducibility (the code runs and yields the reported numbers) yet fail recreate reproducibility (the methods description does not match what the code actually does). This divergence signals a problem: either the code contains errors that happen to produce the reported results, or the written methods misrepresent the actual analysis.

Only when both computational and recreate reproducibility are established does it make sense to invest in more demanding forms of verification. *Robustness reproducibility* tests whether results hold under alternative reasonable analytical decisions on the same data. *Replicability* requires the most resources, as it tests whether

findings hold with new data, though for many studies replication proves prohibitively expensive or impossible (consider historical events or rare observational opportunities).

Each level adds credibility, and researchers can conduct multiple robustness tests or replications to further strengthen confidence. Without the foundational layers, however, higher-order questions about robustness or generalizability remain premature.

Three barriers to reproducibility

Large-scale studies reveal the extent of reproducibility problems. When researchers attempted to execute over 9,000 R files from publicly available replication datasets, 74% failed to complete without error (Trisovic et al. 2022). Such failures likely reflect three intertwined obstacles: the curse of expertise (Hinds 1999), technological barriers, and misaligned incentives.

Original authors possess deep domain knowledge about their methods and data, which leads them to underestimate what others need to reproduce their work. What seems obvious to experts requires explicit documentation for others: authors may believe they have provided sufficient information when critical details remain implicit, undocumented, or buried in their tacit knowledge.

Technological barriers compound the problem. Hard-coded file paths work only on the original computer, software versions and package dependencies go undocumented, manual operations remain unrecorded, and analyses scatter across multiple software environments without clear sequencing. Even in computationally light projects, rapidly evolving software environments, complex dependency chains, and inconsistent standards make verification difficult (Epskamp 2019).

These barriers persist not because they are technically insurmountable, but because addressing them conflicts with how research is practiced and rewarded. Researchers operate under time constraints where publication speed often determines career advancement, making the (upfront) investment in reproducible workflows seem like a luxury they cannot afford. This perception is reinforced by institutional evaluation systems as well as the absence of reproducibility training in most graduate programs. This creates a vicious cycle where perceived time constraints prevent researchers from adopting workflows that would ultimately save time through reduced debugging, easier revisions, and less backwards engineering. Hence, the perceived trade-off between efficiency and reproducibility may be short-sighted and misleading.

Why reproducibility should be a priority

Reproducibility warrants attention even when perfect achievement seems impossible. Data sharing restrictions, proprietary software, or computational costs may create genuine barriers, but researchers should pursue reproducibility to the fullest extent feasible. Partial reproducibility provides more value than none, and the attempt itself often surfaces problems that would otherwise remain hidden.

Irreproducible research creates opportunity costs across the research community. In the best case, researchers lose time deciphering undocumented work, while students face barriers to extending published findings. At worst, vastly more time can be wasted in ultimately fruitless efforts to expand, extend, and build on a body of work that has no empirical foundation (King 1995). In contrast, reproducible workflows offer multiple benefits: they lower the threshold for peer review scrutiny, serve as educational resources for analytical methods, and can increase research impact when other researchers can easily adopt, adapt, and, ultimately, cite the analytical methods. Reproducible (i.e., well-documented, well-organized and intuitive) code could become an asset that provides a template for similar analyses while reducing duplicated effort across research groups.

Most persuasively, reproducibility represents an investment in personal research efficiency rather than an altruistic burden. The primary beneficiary is typically the original author returning to analyses months later, having forgotten crucial details. Reproducible practices help researchers discover errors and avoid painful reconstruction of analytical decisions (King 1995). This reframes reproducibility from an imposed obligation to a personal productivity strategy that happens to benefit others. Taken together, reproducibility enhances both individual research projects and the field's collective efficiency.

Building reproducibility into research workflows

Reproducibility starts with shared materials (see Figure 1). Without these materials, verification becomes impossible. Yet having the building blocks does not guarantee that others can assemble them. Like toy bricks that only fit together when their shapes match, code must work across different computing environments. Like instructions that specify the order of assembly, documentation must guide others through the analytical sequence. These requirements reflect the barriers we identified earlier: experts forget what novices need to know, technology creates friction, and rushed workflows skip documentation.

The practices below address these obstacles. We organize them into four categories. Availability and portability form the essential minimum. However, several journals now require peer reviewers to verify that code is not only functional but also appropriately presented and documented. Researchers emphasize that this approach encourages more readable code and increases trust in computational results (Nature Human Behaviour 2021). Organization and explainability enable this scrutiny, and

add efficiency when projects grow complex, when teams collaborate, or when methods require justification.

Not every project needs the same approach though: A simple two-condition experiment needs different infrastructure and requires less methodological explanation than a deep learning pipeline. Researchers should therefore match their effort to their context, disciplinary norms, and their target audience.

Availability: Making materials accessible

Sharing research materials once posed genuine logistical challenges. Today's infrastructure removes these obstacles. Persistent repositories with DOIs (OSF, Zenodo, Harvard Dataverse) provide stable, citable access to research materials at no cost. These platforms ensure materials remain available indefinitely. Yet not all sharing platforms offer the same guarantees. Popular code-sharing platforms like GitHub serve useful purposes during active development but lack the permanence that published research requires.

Unfortunately, situations arise where direct data sharing proves impossible due to ethical or legal constraints. In these cases, synthetic data that preserve key statistical properties or clear access instructions become essential. When projects use restricted data sources, researchers should document access procedures to help others who hold appropriate permissions verify the work.

Relatedly, authors should specify reuse permissions for their materials (Wilkinson et al. 2016). Licensing choices affect how others can use shared materials. Open licenses (CC-BY for data, MIT or GPL for code) remove ambiguity about reuse permissions. Without explicit licenses, legal uncertainty may prevent others from building on published work even when materials are technically available.

Portability: Ensuring code runs elsewhere

Even when researchers share data and scripts, the code often fails to run on other systems. These failures reflect several distinct portability challenges, each requiring attention.

Analyses must first be automated rather than manual. Statistical software that relies on point-and-click operations creates barriers for independent verification because these actions remain unrecorded. Scripting languages like R and Python automate analytical steps and create an executable record of all operations. This eliminates the need for others to reconstruct undocumented manual procedures, though it also introduces the question of whether those scripts will run reliably across different computing environments.

Even properly scripted analyses face basic portability issues. Hard-coded file paths like `C:/Users/Jane/data` work only on Jane's machine, while relative paths like `./data` can work anywhere. These simple failures prove easy to fix but often go unnoticed until someone else attempts to run the code.

More complex failures arise from software dependencies. Packages update, functions get deprecated, and dependency chains shift. What ran perfectly in January may break in June when a single package updates. Researchers should therefore minimize dependencies and prefer stable, well-maintained packages over unvetted alternatives. Especially when such dependencies are unavoidable, researchers must document software choices, including exact package versions.

However, these documentation requirements scale with project complexity. Simple analyses may require only a minimal README file that lists system requirements, installation instructions, and software dependencies (including package versions). More complex projects benefit from tools that automate environment recreation and manage dependencies systematically. Various tools exist to address these challenges (see, e.g., nix, binder, code ocean or docker), each offering different levels of reproducibility guarantees (Rodrigues and Baumann 2025; Perkel 2019). Researchers should adopt tools that match both their technical comfort and their project's complexity. Even though sophisticated analyses often justify investment in more comprehensive solutions, simple solutions are better than no solutions.

Randomized procedures require additional attention. Bootstrapping, cross-validation, and neural network training produce different numbers on each run. Setting random seeds (`set.seed(230691)` in R, `random_state=230691` in Python) fixes the sequence of pseudorandom numbers the algorithm produces and allows other researchers to reproduce the exact results. Testing code on a fresh installation verifies that documentation captures all necessary steps. Asking colleagues unfamiliar with the project to run the code on her computer or a clean cloud environment reveals undocumented dependencies and implicit assumptions that authors may overlook.

Organization: Creating traceable workflows

Reproducibility benefits when others to understand how files relate and how analysis proceeds. Clear, intuitive folder structures (`code/`, `data/`, `output/`) provide a foundation, with raw data kept read-only and separate from processed data. This separation prevents accidental modifications and allows others to verify all transformations from the original source.

The analytical workflow builds on this structure. Simple projects might use a single well-documented script, while more complex analyses benefit from a modular structure where different scripts serve different purposes (such as data cleaning, statistical analysis, and visualization). Within such a structure, sequential numbering clarifies the order in which scripts should run (`01_clean.R`, `02_analyze.R`). A main file (`main.R` or `run_all.py`) that executes the complete analysis as 'push-button' reproduction removes any uncertainty about the workflow.

The README file complements this organization by documenting the project structure, explaining execution sequence, and providing technical requirements for running the code (see Portability).

Furthermore, outputs should be traceable both to the code that generates them and to the manuscript that reports them. Comments within scripts can link code sections to results (‘`# Creates Table 1`’), while file names should correspond to manuscript elements (`figure_2.png` for Figure 2 in the paper).

Explainability: Making decisions transparent

Code that runs and produces correct numbers may still resist scrutiny if its logic remains opaque. Linking code to manuscript results explicitly (e.g., “# Creates Table 1”) connects analytical steps to reported findings. For straightforward analyses, inline comments that explain what each section does and why those choices were made may suffice. Comments should explain purpose rather than mechanics: why a variable received log-transformation, not that a log function was applied. For complex methods, literate programming approaches (Knuth 1984) like Quarto or Jupyter Notebooks that interweave narrative explanations of analytical choices with executable code enhance comprehensibility.

Finally, writing data dictionaries that describe all variables, units, and coding schemes helps others understand the data structure.

Adapting practices to context

These dimensions involve trade-offs. Literate programming that maximizes explainability through tutorial-style documentation may reduce portability by adding software dependencies. Publication-ready documents that maximize traceability may sacrifice explanatory depth. Researchers must balance these competing demands rather than optimize all dimensions simultaneously.

Systemic steps towards reproducible research

Individual researchers can implement the practices outlined above and incremental steps matter: even partial adoption of reproducible workflows yields benefits, both for individual researchers and for the field.

Nevertheless, reproducibility as a norm requires systemic change. A growing culture of reproduction has emerged across disciplines, with systematic verification of published findings now more common than ever (Brodeur et al. 2024). Several prominent journals support this shift through institutional reforms. Nature Human Behaviour, Psychological Science, Management Science, and journals published by the American Economic Association now implement reproducibility checks to test whether code executes successfully before publication. As more journals adopt such standards, reproducibility becomes normalized.

Still, some researchers worry that reproducibility requirements create unnecessary burdens. As discussed earlier, this concern often reflects a short-sighted view: reproducible workflows ultimately save time–especially when integrated from project inception rather than retrofitted during manuscript preparation. Moreover, emerging technologies now further reduce these barriers. Generative AI tools can support tedious tasks such as code documentation and codebook creation.

Graduate education represents another critical lever for change. Reproducibility training remains rare in doctoral programs despite its fundamental importance. When programs treat reproducible workflows as core methodological training rather than optional skills, researchers adopt these practices from the start of their careers. Requiring students to reproduce a key paper their dissertation builds on would teach the curse of expertise firsthand while developing practical skills. Reproducibility then becomes standard practice rather than an additional burden.

Behavioral patterns take time to shift, and the field will not transform overnight. But simply bringing the notion of reproducibility to the forefront and making it a routine will make a difference, eventually (Peng 2011).

References

Brodeur, Abel, Anna Dreber, Fernando Hoces de la Guardia, and Edward Miguel. 2024. “Reproduction and Replication at Scale.” Correspondence. *Nature Human Behaviour* 8 (January): 2–3. https://doi.org/10.1038/s41562-023-01807-2.

Dreber, Anna, and Magnus Johannesson. 2025. “A Framework for Evaluating Reproducibility and Replicability in Economics.” *Economic Inquiry* 63 (2): 338–56. https://doi.org/https://doi.org/10.1111/ecin.13244.

Epskamp, Sacha. 2019. “Reproducibility and Replicability in a Fast-Paced Methodological World.” *Advances in Methods and Practices in Psychological Science* 2 (2): 145–55. https://doi.org/10.1177/2515245919847421.

Hinds, Pamela J. 1999. “The Curse of Expertise: The Effects of Expertise and Debiasing Methods on Predictions of Novice Performance.” *Journal of Experimental Psychology: Applied* 5 (2): 205–21. https://doi.org/10.1037/1076-898X.5.2.205.

Hussey, Ian. 2025. “Data Is Not Available Upon Request.” *Meta-Psychology* 9. https://doi.org/10.15626/MP.2023.4008.

King, Gary. 1995. “Replication, Replication.” *PS: Political Science & Politics* 28 (3): 444–52. https://doi.org/10.2307/420301.

Knuth, D. E. 1984. “Literate Programming.” *The Computer Journal* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.

Munafò, Marcus R., Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. 2017.

"A Manifesto for Reproducible Science." *Nature Human Behaviour* 1 (1): 0021. https://doi.org/10.1038/s41562-016-0021.

Nature Human Behaviour. 2021. "Supporting Computational Reproducibility Through Code Review." *Nature Human Behaviour* 5 (8): 965–66. https://doi.org/10.1038/s41562-021-01190-w.

Peng, Roger D. 2011. "Reproducible Research in Computational Science." *Science* 334 (6060): 1226–27. https://doi.org/10.1126/science.1213847.

Perkel, Jeffrey M. 2019. "Containers in the Cloud." *Nature* 575: 247–48. https://doi.org/10.1038/d41586-019-03366-x.

Rodrigues, Bruno, and Philipp Baumann. 2025. "Nix for Polyglot, Reproducible Data Science Workflows." https://github.com/b-rodrigues/rix_paper.

Stark, Philip B. 2018. "Before Reproducibility Must Come Preproducibility." *Nature* 557 (May): 613. https://link.gale.com/apps/doc/A572639336/AONE?u=anon~cbc539bf&sid=googleScholar&xid=3384b2a4.

Trisovic, Ana, Matthew K. Lau, Thomas Pasquier, and Mercè Crosas. 2022. "A Large-Scale Study on Research Code Quality and Execution." Analysis. *Scientific Data* 9 (60). https://doi.org/10.1038/s41597-022-01143-6.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3 (1): 160018. https://doi.org/10.1038/sdata.2016.18.

A1 Appendix content here