Computational and Pedagogical Tools for Open Science: Commentary on Van Til et al. (2025)

Jeffrey M. Girard

Department of Psychology, University of Kansas

Author Note

Jeffrey M. Girard https://orcid.org/0000-0002-7359-3746

Supplemental materials are available from: https://github.com/jmgirard/cps2025

Correspondence concerning this article should be addressed to Jeffrey M. Girard, Department of Psychology, University of Kansas, 1415 Jayhawk Blvd (Room 426), Lawrence, KS 66045, USA, Email: jmgirard@ku.edu

Abstract

Clinical psychology has lagged in adopting open science practices, despite student interest and widespread recognition of their value. Building on Van Til et al. (2025)'s findings of limited exposure in doctoral training, I argue that transparency requires more than brief exposure; it demands scaffolded, intentional instruction. Integrating preregistration, code sharing, and reproducible workflows early in the curriculum is essential. I describe how computational tools can support this shift, highlight institutional and pedagogical barriers, and call for faculty development and program-level support to ensure open science becomes a shared norm, not just an ideal.

Keywords: commentary, open science, graduate training, research methods

Computational and Pedagogical Tools for Open Science: Commentary on Van Til et al. (2025)

Open science offers powerful tools for transparency and rigor, yet clinical psychology has lagged behind other subfields in embedding them in graduate training. Much of this lag reflects distinct headwinds: handling sensitive patient data, coordinating with clinics and IRBs, and teaching in highly individualized settings (Tackett et al., 2017). Still, these factors do not fully explain the gap: core practices such as preregistration, code sharing, and reproducible workflows can be taught and adopted despite them.

Drawing on syllabus audits and graduate-student surveys, Van Til et al. (2025) document that open-science training is present in some U.S. programs but uneven overall. Readings and occasional assignments appear, yet hands-on opportunities to practice core behaviors are scarce; students report interest, but unfamiliarity with practices like registered reports remains common.

Writing as a clinical psychologist with a strong computational focus, I take an opinionated stance: open science will not advance through exposure to principles alone. Graduate programs should provide sustained, scaffolded, and hands-on training that turns open-science ideals into routine habits and integrates modern computational tools. My aim in this commentary is to advance that agenda and spark discussion.

The Training Students Deserve

Despite widespread agreement that transparency and rigor are critical to the future of clinical science (cite), these values are not yet embedded in the day-to-day realities of graduate training. As Van Til and colleagues show, most students are not receiving sustained, structured exposure to open science practices. When these ideas do appear (e.g., as brief modules), they often lack opportunities for hands-on engagement. Exposure is not the same as training, and aspiration is not the same as implementation.

If clinical psychology programs are serious about producing scientifically rigorous researchers, open science should be woven into the curriculum from the beginning. The first semester of a Ph.D. program sets the tone for a student's identity as a researcher. Early methods

and statistics courses shape habits around study design, data analysis, error correction, and interpretation (cite). Just as importantly, they establish norms around transparency, collaboration, and research quality. Institutional messages during this time are especially powerful in shaping what students perceive as rigorous and responsible science. Delaying open science instruction, or treating it as an optional add-on, suggests it is peripheral rather than central.

Ideally, training should be scaffolded across courses and developmental stages. A first-semester methods course might introduce the rationale for open science and include preregistration exercises using structured templates. These help students practice articulating plans in advance and distinguishing between exploratory and confirmatory approaches. The goal is not to discourage exploration, but to encourage clear labeling and thoughtful reflection on how analytic choices relate to inferential claims. In a statistics course, students could analyze real or simulated datasets using reproducible code, gaining experience with documentation and producing transparent outputs. These settings can also model a more deliberate, theory-driven approach to analysis, emphasizing model comparison and resistance to selective reporting.

By the second year, students can begin applying these practices to research questions grounded in their own interests. A second-year (e.g., thesis) project offers a chance to design a study, preregister hypotheses, analyze original data, and share materials in an open repository. These projects reinforce open science practices and show students how rigor and transparency enhance work they care about and intend to build upon.

By the time students begin their dissertation, they should be familiar with core elements of open research workflows: preregistration, code sharing, and reproducible reporting. Some may also explore more advanced practices, such as preparing a registered report, conducting a replication, or using version control in collaborative projects (see below). At this stage, the goal is to reinforce good habits and provide tools to support transparency and reuse.

This kind of integrated training requires planning and commitment. Open science cannot be relegated to a single lecture or siloed unit; it needs to be woven into research design, measurement, statistics, and writing. It also requires mentorship and modeling. Faculty need not

be experts in open science tools to model transparency. Discussing how they document decisions or reflecting openly on project challenges can all serve as powerful signals. When instructors and research mentors make their reasoning visible, especially in moments of ambiguity, they give students permission to try, iterate, and learn.

While these training aspirations—and often the courses needed to realize them—are not unique to clinical psychology, the field occupies a distinctive position within the broader discipline. Many of the skills associated with open and reproducible science are equally relevant across psychological subfields. However, clinical psychology is unusual in being subject to formal accreditation, which confers both constraints and opportunities. Accrediting bodies such as the American Psychological Association and the Psychological Clinical Science Accreditation System wield considerable influence over program priorities. By setting expectations for competencies and curricular content, they can either facilitate or hinder the integration of open science principles into graduate education. This makes clinical psychology a particularly important proving ground for embedding transparency and rigor within professional training standards.

Transparent research practices are not something students can adopt in isolation. They require supportive structures, peer modeling, and, most of all, instructors who show that rigor and transparency are not abstract ideals, but everyday decisions.

Tools that Empower Transparent Science

Open science is grounded in values and policies, but it is also sustained by tools and technologies that allow researchers to put those principles into practice. Tools shape behavior, lower (or raise) barriers to entry, and implicitly communicate what kinds of work are expected or legitimate. For graduate students in clinical psychology, the tools they learn to use early on will often become the scaffolding for their scientific careers. Ensuring that those tools support reproducibility, transparency, and reusability is therefore an essential part of research training.

Code-Based Workflows

Perhaps the most transformative shift in research practice over the past decade has been the move from costly point-and-click statistical software to scripted, code-based workflows using

open-source programming languages. Whether students use R, Python, or another programming language, learning to write and document their own data processing and analysis scripts fosters a degree of transparency and reproducibility that manual workflows simply cannot match. When analyses are encoded in readable, versioned scripts, others can review them, rerun them, and learn from them. The analysis becomes an object of scientific discourse, not a black box (Sandve et al., 2013).

There is no justifiable scientific reason to withhold analytic code from a published study. While raw data may need to be protected for legal or ethical reasons, the code used to analyze that data is part of the method. Omitting it is equivalent to describing a laboratory procedure vaguely or inaccurately. Yet, despite this, many papers still do not include analysis code, and many students are never taught to think of code as a scientific product in its own right. This gap represents a clear area for curricular reform.

Literate Programming

Literate programming is a paradigm that treats code and narrative text as equally important parts of scientific communication, allowing researchers to explain what their code does and why within a single, coherent document. Tools like Quarto, an open-source publishing system, offer a compelling solution for students and instructors alike (Scheidegger et al., 2025). Quarto enables researchers to seamlessly combine text, code (in R, Python, Julia, or Observable), and output (e.g., figures, tables, diagnostics) in one self-contained document. It supports equations, citations, and cross-references with output formats including reports, slideshows, websites, blogs, books, and articles.[1] This integrated approach facilitates reproducibility and teaches students to communicate their workflows clearly and cohesively, bridging the gap between technical detail and scientific storytelling.

In a graduate course, literate programming encourages students to document their thought process, explain decisions, and produce reports that can be rerun at any time. This format works especially well for method assignments, replication projects, and theses. It also reflects how

[1] This article was created using Quarto. See github.com/jmgirard/cps2025 for the code.

high-quality research is increasingly shared—not just through articles and talks, but through richly documented supplemental materials that promote open science. These same practices prepare students to collaborate on interdisciplinary research teams and open doors to roles beyond academia, including in data science and applied research, where reproducible workflows are essential.

Beyond the Basics

While beyond the scope of most graduate-level methods courses in clinical psychology, three frameworks from computer science can support robust, transparent, and portable workflows. Though less commonly taught in psychology, these approaches align with open science values and are especially useful for students with more programming experience, such as those developing shared codebases, contributing to interdisciplinary projects, or managing evolving analytic pipelines.

Version control systems help students track changes, recover earlier versions, and collaborate without overwriting one another's work. They create a structured record of a project's development, making it easier to understand decisions and revert if needed. Although most commonly used for code, version control also applies to manuscripts, analysis plans, and other evolving documents. This makes it valuable not only for technical work but for any collaborative or iterative writing process. *Git* is the most widely used system, and platforms like GitHub.com support sharing, coordination, and structured discussion of revisions (Chacon & Straub, 2014).

Dependency management tools ensure that code continues to run reliably even as software packages change. Analyses often depend on specific package versions, and updates can cause code to fail or yield different results. Without a way to restore the original setup, reproducing results becomes difficult. For students working in R, *renv* makes it easy to save and later recreate the package environment used in a project (Ushey & Wickham, 2025). This is critical for open science, where reproducibility depends on preserving not just the code, but also the conditions under which it ran.

Containerization goes further by capturing the entire computational environment,

including the operating system, packages, and code. This allows projects to run exactly as they did originally, eliminating the common "it worked on my computer" problem. Tools like *Docker* let students bundle all required components into portable environments that run consistently across systems (Merkel, 2014). Containerization is especially helpful for sharing complex projects, archiving long-term analyses, or supporting reproducibility in public research outputs. Though more technically demanding, it offers a powerful way to future-proof scientific work.

Tools as Values in Action

Importantly, these tools do more than facilitate technical accuracy; they teach a way of thinking. Code-based workflows, literate programming, and version/environment management foster habits of precision, documentation, foresight, and openness. They signal that science is not merely about reaching results but about making one's process intelligible and reviewable by others. These are not just technical skills; they reflect core commitments to rigor and accountability in research.

By teaching these tools explicitly, and modeling their use, graduate programs can help students build workflows that are not only reproducible but also resilient, transferable, and collaborative. In a world where scientific claims are increasingly scrutinized, these capacities are not optional. They are part of what it means to do science well.

Barriers to Implementation

While open science tools have become more powerful and accessible, their integration into graduate training still faces real obstacles. If the benefits are so widely recognized, why haven't practices like preregistration, reproducible code, and registered reports become standard in clinical psychology programs? The issue is not a lack of awareness or opposition to transparency itself, but rather the practical constraints and competing demands that shape the training environment.

The Overload of Early Graduate Training

First-year clinical psychology students are often asked to simultaneously learn statistical theory, coding practices, research design, clinical assessment, ethical frameworks, and therapeutic approaches, while also adjusting to the culture and pace of graduate life. Adding open science to

this already full plate can feel overwhelming, particularly if students are still developing basic competencies in programming or data analysis. Even motivated students may struggle to incorporate reproducible workflows when they are still grappling with the syntax of R or the nuances of applied linear modeling.

This challenge is not a reason to abandon open science education. But it is a reason to approach it with pedagogical sensitivity. Training must be scaffolded, not just in content but in cognitive demand. Students benefit most when tools are introduced incrementally, in the context of meaningful research questions, and with clear rationale. A template-based preregistration assignment in a methods course is very different from asking a second-year student to submit a registered report for their thesis without prior modeling or feedback.

Faculty Constraints and Curricular Inertia

The other half of the equation is the faculty. Many instructors are sympathetic to open science but lack the time, training, or institutional support to revamp existing syllabi. It takes effort to redesign assignments to incorporate code-based reproducibility, especially when one is already stretched thin by teaching, mentoring, clinical supervision, and grant writing. Some faculty may feel underqualified to teach newer tools like R or Quarto, particularly if they themselves were trained in a pre-reproducibility era. Others may support transparency in principle but worry that emphasizing rigid practices will stifle student creativity or delay progress on publishable work.

Addressing these constraints will require departmental leadership and collective commitment. Open science cannot rest solely on individual faculty initiative. Programs should offer faculty development resources, encourage team-teaching or co-mentoring across skill sets, and provide ready-made curricular materials that lower the barrier to entry. Faculty need permission and support to evolve alongside the science.

The Messiness of Real Research

Even when training and support are available, real research often refuses to fit neatly into the boxes provided by preregistration templates (Nosek et al., 2019). This is especially true for students who are still learning statistics and may not yet have the knowledge to specify appropriate

analytic methods in advance. Those working with secondary data, developing new measures, or conducting qualitative research may also find it difficult to anticipate all decisions ahead of time. Many studies evolve in response to unexpected data patterns, reviewer feedback, or shifts in scope. Rigid preregistration requirements can backfire if they encourage box-checking over thoughtful planning, discourage exploration, or promote the false impression that research is a linear process rather than an iterative one.

The solution is not to abandon preregistration, but to teach it as a flexible tool rather than a rigid requirement. Not every project is a good fit for full preregistration. Replication studies or work that builds on well-established methods and literatures are often well suited to prespecifying hypotheses and analyses. In contrast, a student's first exploration of a new topic, dataset, or method may be more appropriately framed as exploratory from the outset. Students should understand that deviations from a preregistration are not failures, but opportunities to document learning and refine their reasoning. They should be taught to distinguish clearly between confirmatory and exploratory analyses, to explain changes transparently, and to see preregistration as a tool for clarifying thought, not constraining it. Similarly, registered reports, while powerful, are not appropriate for every project. Their value lies not in universal applicability, but in setting a gold standard for transparency and rigor in studies where hypotheses and analyses can be clearly prespecified.

The Perfectionism Trap

Finally, one subtle but significant barrier is the culture of perfectionism that can pervade graduate training. Open science, when framed in overly idealistic terms, can unintentionally reinforce the belief that a "good" scientist is one who always writes clean code, anticipates every contingency, and adheres flawlessly to every preregistered plan. For students already struggling with imposter syndrome, this can breed anxiety and paralysis rather than empowerment. Open science is not about perfection, it is about clarity, humility, and continuous improvement. Faculty must actively model this mindset: by sharing imperfect code, troubleshooting errors in front of students, admitting past mistakes, and making their reasoning visible (Strand, 2025). Students

benefit more from seeing a messy, well-documented workflow than from receiving a polished but opaque one. A culture of openness begins not with mastery, but with modeling vulnerability and growth.

Conclusion

Van Til and colleagues have provided a valuable service by empirically documenting what many open science advocates in clinical psychology have long suspected: while interest in transparency and rigor is growing, graduate training has yet to catch up. Their registered report offers both a clear-eyed assessment of the current state and a hopeful reminder that students are eager to engage with these topics when given the opportunity.

But awareness is only the first step. If we want open science to take root in clinical psychology, we must invest in the infrastructure, pedagogy, and culture that make transparency not just possible, but normative. That means treating reproducibility as a core competency, not a niche skill. It means integrating preregistration, code-based analysis, and material sharing into the curriculum early and often. It means supporting faculty with training, templates, and time. And it means creating room for nuance, acknowledging that transparency can look different across projects and that imperfection is not a flaw but a feature of honest science.

Open science is a practice we choose, repeatedly, in the face of complexity, uncertainty, and time pressure. It is a commitment to doing science in a way that is intelligible to others and accountable to the communities we serve. For clinical psychological science, that commitment is essential, and so is the work of teaching it. Open science education is how that commitment becomes sustainable. The path forward is not about mandating any single tool or template. It is about cultivating a scientific environment in which students and faculty alike feel empowered to clearly communicate what they did, why they did it, and how others can evaluate or build upon it. That kind of culture begins in the classroom, is reinforced through mentorship, and is sustained by institutions that value good science by rewarding transparency, rigor, and process.

Acknowledgments

This manuscript was written with the assistance of a large language model (ChatGPT 4o). Specifically, the model was used to refine an initial outline created by the author and to assist with iterative revisions aimed at improving clarity and conciseness. The author has independently verified the accuracy, validity, and appropriateness of all substantive content. As with any tool, LLMs may introduce errors or bias, and the author bears sole responsibility for the final text.

Author Contributions

Conceptualization: J. Girard; Writing – Original Draft Preparation: J. Girard.

Conflicts of Interest

The author declares that there were no conflicts of interest with respect to the authorship or the publication of this article.

References

Boettiger, C., & Eddelbuettel, D. (2017). An Introduction to Rocker: Docker Containers for R. *The R Journal*, *9*(2), 527–536. https://doi.org/10/ghgdtz

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Nosek, B. A., Beck, E. D., Campbell, L., Flake, J. K., Hardwicke, T. E., Mellor, D. T., van 't Veer, A. E., & Vazire, S. (2019). Preregistration is hard, and worthwhile. *Trends in Cognitive Sciences*, *23*(10), 815–818.

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. *PLOS Computational Biology*, *9*(10), e1003285.

Scheidegger, C., Woodhull, G., Dervieux, C., Teague, C., Allaire, J. J., & Xie, Y. (2025). *Quarto*. Posit, PBC.

Strand, J. F. (2025). Error tight: Exercises for lab groups to prevent research mistakes. *Psychological Methods*, *30*(2), 416–424.

Tackett, J. L., Lilienfeld, S. O., Patrick, C. J., Johnson, S. L., Krueger, R. F., Miller, J. D., Oltmanns, T. F., & Shrout, P. E. (2017). It's time to broaden the replicability conversation: Thoughts for and from clinical psychological science. *Perspectives on Psychological Science*, *12*(5), 742–756. https://doi.org/10.1177/1745691617690042

Turrell, A., Monticone, P., Akyol, Z., Holman, J., & Huang, Y. (2025). *Python for Data Science*.

Ushey, K., & Wickham, H. (2025). *Renv: Project Environments*.

Van Til, K., Phillips, N., Du, T., Rose, L., Miller, J., & Lynam, D. (2025). Open science training in APA-accredited clinical psychology programs: A registered report. *Clinical Psychological Science*.

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2nd ed.). O'Reilly Media.

Table 1

Recommended Readings

Tool	Readings
Code-based Workflows	Wickham et al. (2023), Turrell et al. (2025)
Literate Programming	2
Version Control	3
Dependency Management	4
Containerization	Boettiger and Eddelbuettel (2017)