STAMPED properties for reproducible research objects

Austin Macdonald, Cody C. Baker, Isaac To, Yaroslav O. Halchenko

March 2026

1 Abstract

2 Introduction

Data-driven research depends on specific versions of datasets, software, and computational environments, and a record of how they were used together to produce results. We adopt the term **research object** to denote a collection of data, code, and metadata that together represent the research as a complete unit. It is common for the components of a research object to be organized independently with code in one repository, data on a shared drive, environment setup in a wiki page, provenance in a lab notebook or nowhere at all. This separation undermines rigor, reproducibility, reusability, and efficiency. Even when gathered in one place, research objects are often tightly coupled to specific environments, opaque to outside collaborators, and difficult to reuse or redistribute.

The FAIR principles[1] and FAIR4RS guidelines[2] provide structured guidance for making digital objects "Findable, Accessible, Interoperable, and Reusable". The Workflow Community Initiative (WCI-FW)[3] mapped these principles onto computational workflows but explicitly chose not to define new ones, instead treating workflows as hybrid data/software objects. STAMPED addresses a complementary question: how a research object should be organized and managed so it can actually be re-executed, extended, and redistributed. Both frameworks cover overlapping concerns. For example, persistent identifiers serve both Findability and Tracking, but accent different aspects: FAIR emphasizes discovery and interoperability across repositories, while STAMPED emphasizes improving the operational maturity of scientific workflows in day-to-day research practice.

Many researchers already employ practices that address these operational properties, such as version control, containerized environments, structured project directories, and documented analysis steps. Yet the community lacks a shared vocabulary for referring to these ideas, evaluating how thoroughly they are applied, and communicating that evaluation to collaborators and reviewers. Here we provide that vocabulary: the STAMPED properties (Self-contained, Tracked, Actionable, Modular, Portable, Ephemeral, and Distributable), originating from the YODA Principles[4], which describe the operational maturity of a research object. Rather than a compliance threshold, each STAMPED property is a spectrum from a practical minimum—often what researchers are already doing—to an aspirational ideal. A researcher who stores everything under one project directory, version-controls analysis scripts, and documents how to run them is already meeting the practical minimum for Self-containment, Tracking, and Actionability. As complexity grows, such as for multi-institutional collaborations with heterogeneous workflows spanning platforms enforcing long-term reproducibility, these same properties guide researchers toward more rigorous practices without requiring wholesale adoption of new tools. STAMPED provides a vocabulary for describing and a framework for evaluating and incrementally improving the operational properties of research objects toward enhanced rigor, reproducibility, reusability, and efficiency.

3 Comments

The items containing keywords "MUST", "SHALL", "SHOULD", and "MAY" are to be interpreted as described in RFC 2119[5].

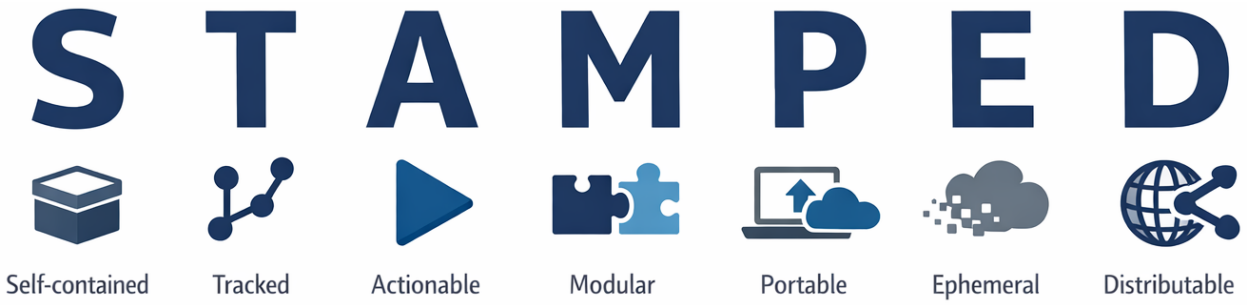

Figure 1: Illustration of the STAMPED properties.

Table 1: The STAMPED Properties. Each property addresses a distinct dimension of research object organization that contributes to reproducibility. Together, they provide a framework for evaluating and improving the reproducibility of computational research.

Letter	**Principle**	**Description**
S	Self-containment	A research object is a complete retrieval unit; it can be obtained and understood in its intended scope without needing to reference external resources.
T	Tracking	The state and provenance of all components is recorded.
A	Actionability	Research object contains machine-actionable information to carry out procedures to obtain or reproduce content.
M	Modularity	All modules are independent and composable.
P	Portability	Research object could be moved to different environments while retaining its STAMPED properties.
E	Ephemerality	Procedural execution is performed within a throwaway environment.
D	Distributability	All modules and procedures are shareable externally in a persistent state.

3.1 Self-containment

- **S.1**: All modules and components essential to replicate computational execution MUST be reachable within a single top-level research object.

Reproducibility fails when a research object depends on resources that are not part of it; an undocumented library, a dataset referenced by a broken URL, a script that assumes tools are installed on the host system. We call this the "don't look up" rule: a research object must never rely on implicit external state. Instead, it must be a complete retrieval unit, where all modules and components essential to reproduce its results are contained within a single top-level boundary (S.1, Table 1).

Components may be included literally (e.g., files committed directly) or by reference (e.g., as subdatasets, registered data URLs). Both approaches satisfy self-containment, provided the references are explicit and part of the research object. At a minimum, everything needed is gathered under one root and any external dependency is explicitly documented. This ensures that nothing is implicitly assumed about the host system, a concern further addressed by Portability. At the ideal end, every reference is precise enough that retrieval is unambiguous; how to achieve that precision is the concern of Tracking (T) and Distributability (D).

Self-containment is the foundational property upon which the remaining STAMPED properties build. Tracking, Modularity, Portability, and Distributability each address a different aspect of how that boundary is organized, versioned, and shared—but none are meaningful without first establishing what is inside it.

3.2 Tracking

- **T.1**: Persistent content identification MUST be recorded for all components.
- **T.2**: All components SHOULD be tracked using the same content-addressed version control system.
- **T.3**: Provenance of all modifications MUST be recorded.
- **T.4**: Code-driven provenance SHOULD be recorded programmatically and MUST include the versions of all components involved.

Research data and code change constantly during a project, and each change compounds the difficulty of reconstructing any previous state: which version of a dataset was used for a particular analysis, what parameters were set, what code produced a given figure. Tracking addresses two concerns: 1) the identity of each component and its exact state at a point in time, and 2) **provenance**, the recorded history of what actions by which identities produced or modified it. These are often managed separately, which makes prior work harder to validate.

The spectrum of Tracking begins with content-addressed version control. Rather than relying on semantic version labels, content-addressed systems identify each state by a checksum of its contents. This distinction matters: two datasets labeled "version 1.0" by different labs are ambiguous; two datasets with identical content hashes are provably identical. Version control commits also capture basic provenance: who made the change, when, and a description of why. This unifies identity and provenance in a single system and also provides a safety net since any change can be reverted and any prior state restored, making it easier to modify and experiment. For tracking to be effective, all components of the research object must be under version control: code, data, environment definitions, and configuration.

Traditional version control is designed for code and other text files, but research objects often include large binary datasets, container images, and other assets that do not fit this model. Compatible tools such as git-annex, DataLad, or DVC extend content-addressed tracking to these components, ensuring that all parts of the research object are managed under the same system.

Toward the ideal end, provenance is captured programmatically by tooling rather than by manual annotation, producing richer records than a human would typically write: each computational step records what command was executed and what inputs it consumed. When modules are independently tracked under compatible systems (M), the version of every module at the time of each commit is also discoverable, providing a complete picture of the state that produced a given result. This enables not just inspection but re-execution. Beyond the research object itself, knowing the state of data and code at each hand-off between experimental, analytical, and engineering stages is what allows a team to coordinate updates without losing the provenance chain—an operational view of Tracking developed by the SciOps framework[6].

3.3 Actionability

- **A.1**: Research object MUST contain sufficient instructions to reproduce all computational results.
- **A.2**: Procedures SHOULD be specified as executable specifications.

A research object may be self-contained and fully tracked, yet having all ingredients and a record of what was done is insufficient without a recipe that can be followed. Requirement A.1 is what makes a research object operationally useful by bridging the gap between having the components and being able to use them.

At a minimum, a research object must contain documentation thorough enough for another researcher to follow every step to reproduce results. Actionability applies to every other STAMPED dimension: at the ideal end, each property is enacted not through documentation alone but through executable specifications (A.2):

- **Self-containment** is more actionable when retrieval of all components is specified and executable (e.g., `git clone`, `datalad install`), not just listed in a manifest.
- **Tracking** is more actionable when provenance records are re-executable (e.g., `datalad rerun`), not just inspectable.
- **Modularity** is more actionable when components can be composed, updated, and replaced via tooling (e.g., `git submodule`), not just organized into directories.
- **Portability** is more actionable when environment specifications are machine-readable and can be resolved on different hosts (e.g., `conda env create`, Nix flakes), not just documented in a README.
- **Ephemerality** is more actionable when computation can be orchestrated in disposable environments (e.g., `docker compose`, Slurm), not just instructed to "run in a clean environment."
- **Distributability** is more actionable when a frozen state can be produced and retrieved by others (e.g., containers, `npm ci`), not just described with pinning instructions.

The principle is tool-agnostic: any system that moves a property from documented to executable moves the research object further along the actionability spectrum. Actionability also extends from tooling to the operational layer of a research group: a SciOps-style workflow makes hand-offs between experimental, analytical, and engineering stages explicit so that artifacts move from one step to the next with unambiguous state[6], an organizational counterpart to the per-component actionability described above.

3.4 Modularity

- **M.1**: Components SHOULD be organized in a modular structure.
- **M.2**: Components MAY be included directly or linked as subdatasets.

Separation of concerns is a foundational engineering practice: organizing a system so that each piece has a clear, distinct role. With clear boundaries between the conceptual components of a research object, it becomes more navigable and maintainable. We define a **module** as a separately distributable unit of a research object. This can include a subdataset, a collection of analysis scripts, or the definition for some computational environment. In practice, module boundaries often align with repository boundaries, but a module may also be a published container, a registered dataset, or any independently versioned unit. Modularity applies this principle to research objects, structuring them as assemblies of modules whose boundaries are explicit. At a minimum, a research object should separate analysis code, input datasets, computational environments, configuration settings, and results into distinct directories. Beyond this, independent versioning of modules enables reuse: a dataset or environment can be updated to a newer version, or swapped for an alternative, without disrupting the rest of the research object. This is possible through manual management but becomes practical when handled by tooling that automates module installation, versioning, and composition. Toward the ideal end, the full research object can be reassembled from its specification through automated, recursive installation of its modules.

3.5 Portability

- **P.1**: Procedures MUST NOT depend on undocumented host environment state.
- **P.2**: Computational environments MUST be explicitly specified.
- **P.3**: Environment definitions MUST be version controlled.

A research object that is self-contained, tracked, and modular may still fail to run on a different machine if it is silently affected by the inherited host state. Common examples of such states include specific developer-level package installations with no released versions, unique system libraries exposed to custom user profiles, hardcoded system paths, or an assumed OS configuration (*e.g.*, path character limits). This coupling is often invisible on the original system and only surfaces when someone else attempts to reproduce the work. Portability requires making all host dependencies explicit, so that the research object can be adapted to a new environment.

Some degree of coupling to the host is unavoidable, such as path naming restrictions ("C:/" vs. "/mnt"), resource limits (minimum CPU/RAM requirements), or platform-specific flags. The spectrum of Portability begins with isolating this host-specific configuration from the rest of the research object, so that adapting to a new environment, however involved that may be, requires only changing configuration. Beyond this, several complementary approaches reduce host coupling: virtual machines provide full system-level isolation; container-based tools (Docker, Singularity/Apptainer) bundle OS-level environments; and declarative package managers (Nix, Guix) specify environments that can be reconstructed from a definition. Toward the ideal end, the computational environment is fully specified and version-controlled within the research object, enabling it to be instantiated on any compatible system.

3.6 Ephemerality

- **E.1**: Computational results SHOULD be produced in ephemeral environments.

A research object may satisfy Self-containment, Tracking, and Portability while its computational environment has only ever been assembled once, incrementally, on the original researcher's machine. During development, researchers routinely modify their environment by installing packages, adjusting configurations, and setting variables. Each change is individually small but collectively difficult to document manually. The result is environment drift, where an environment that works but may not be reconstructable from its specification alone. When this occurs, undocumented steps can go unnoticed when internalized knowledge silently fills the gaps. Ephemerality eliminates this problem by construction: a disposable environment built from tracked specifications cannot carry undocumented state, so any successful execution exercises Self-containment, Actionability, and specification completeness.

The spectrum of Ephemerality ranges from manually setting up a fresh environment from the project's specification (which is usually sufficient to catch environment drift) to fully automated disposable environments created and destroyed per execution. At the ideal end, ephemeral computation also enables testing across different host systems, where same-host execution validates specification completeness and different-host execution validates that specifications are not coupled to a specific system. The FAIRly big workflow[7] and other approaches to using distributed computing platforms such as SLURM or Code Ocean operationalize this ideal, treating each computation as an ephemeral work unit. This approach easily enables horizontal scaling, as independent disposable instances that can be parallelized across subjects, parameters, or datasets. Yet Ephemeral computation does not validate everything. For instance, an ephemeral environment with network access may fail to notice unpinned dependencies fetched at build time, a self-containment gap that only surfaces when those resources move or disappear. Nonetheless, the principle substantially raises confidence that the research object is self-contained, portable, and actionable.

3.7 Distributability

- **D.1**: All referenced modules and components MUST be persistently retrievable by others.
- **D.2**: Environment specifications SHOULD support reproducible builds.

A research object that is self-contained on the author's machine but cannot be obtained by others in the same state is effectively unreproducible beyond its origin. Self-containment (S) establishes that everything needed is within the boundary and Tracking (T) records the state of each component. Distributability ensures that others can obtain the research object in that same state.

The distinction mirrors the concept of a software distribution: a curated, versioned bundle in which all components are resolved to specific versions and packaged for consumption. Simply sharing a research object by uploading scripts to a repository with loose dependencies, or posting files on a website with no version or other persistence guarantees, does not constitute distribution in this sense. A distributable research object is packaged so that it can be retrieved in the same state as intended.

Distributability also has a circular relationship with the other STAMPED properties: someone else's distribution effort often serves as the starting point for a new research object's self-containment. Researchers routinely begin by downloading containers, fetching published datasets, and installing released software—modularly composing the distributions of others to assemble their own self-contained research objects.

The spectrum of Distributability begins where FAIR leaves off: publicly accessible components with documented retrieval instructions. Toward the ideal end, several complementary strategies reduce the risk that external changes will break a research object's reproducibility: bundling dependencies into containers or archives reduces the surface area of components that can independently change or disappear; hosting on archival infrastructure (Zenodo, DANDI, Software Heritage) with content-addressed identifiers allows recipients to verify integrity regardless of source; and mirroring across multiple platforms protects against the failure of any single registry.

3.8 Checklist for compliance to principles

We hope that the preceding sections have provided a clear understanding of the STAMPED properties and their rationale. While this understanding is essential, researchers may also benefit from a practical checklist to assess their research objects against these principles. Similar to the living examples repository, an interactive checklist has been provided at the webpage https://stamped-principles.github.io/stamped-checklist/ to guide assessments of compliance with STAMPED principles, ordered by the strength of the requirement (MUST, SHOULD, MAY). The schema for this checklist is itself a living, version-tracked entity that is likely to grow over time as more common violations of the principles are discovered in practice.

3.9 Enabling tools

No single tool addresses all the dimensions of scientific rigor described above, though many tools contribute to one or several properties simultaneously. For example, container technologies such as Docker or Singularity serve Portability by providing OS-level isolation from explicit environment specifications, enable Ephemerality by disposing per-execution environments, and facilitate Distributability by producing frozen, shareable artifacts. Similarly, version control systems such as Git underpin both Tracking (content-addressed identification and change history) and Self-containment (gathering all components under a single root). In practice, self-containing all data might be undesired or prohibitive due to size or other concerns. Extensions such as git-annex, DataLad, DVC, and Git LFS strike the balance between the two properties by tracking metadata that identifies specific data versions and obtainability information instead, e.g., content checksums and URLs of remote servers that act as a source, and providing actionable interfaces to obtain the data when necessary.

This overlap is not incidental — it reflects the interdependence of the STAMPED properties themselves. Achieving Ephemerality, for instance, inherently exercises Portability and Actionability: a disposable environment must be reconstituted from explicit specifications, encouraging end-to-end automation. Likewise, Distributability builds on Self-containment and Tracking: a research object can only be shared in a consistent, retrievable state if its boundary is well-defined and all components are precisely identified. The tools that serve multiple properties do so precisely because they operationalize these connections.

Table 3 maps some commonly used tools and technologies to each STAMPED property. A comprehensive up-to-date review of existing tools is out of scope for this formalization paper and would quickly become outdated. To fill that niche, we initiated a satellite project with a website to describe tools and examples, and characterize them in terms of STAMPED and FAIR principles [cite https://stamped-principles.github.io/stamped-examples/stamped_principles/].

Table 2: Normative statements about STAMPED properties of research objects.

Principle	Requirements
To be Self-contained:	• all modules and components needed to understand and execute the Research Object are retrievable as a single unit • external dependencies are explicitly documented with retrieval instructions • there are no implicit references to undocumented external resources
To be Tracked:	• every component has version information (commit hash, tag, or identifier) • changes to components are recorded with timestamps and authorship • provenance records capture the computational history, context, and transformations
To be Actionable:	• instructions for executing procedures are present and unambiguous • execution paths can be followed manually or automated programmatically • the Research Object transitions from documentation to operational capability
To be Modular:	• modules can be independently tracked, modified, and distributed • components are organized in logical, separable units • modules can be composed together or used in isolation
To be Portable:	• system requirements and dependencies are explicitly documented • the Research Object is flexible enough to execute on different host environments without modification by the user • environment specifications are machine-readable where possible
To be Ephemeral:	• computation can occur in temporary, disposable environments • results are reproducible without knowledge of previous runs • no reliance on external configurations or host system states (such as OS registry modifications)
To be Distributable:	• all modules and components can be shared in a persistent, retrievable state • dependencies are frozen or pinned to specific versions across systems • the Research Object can be obtained by others in the same state as intended

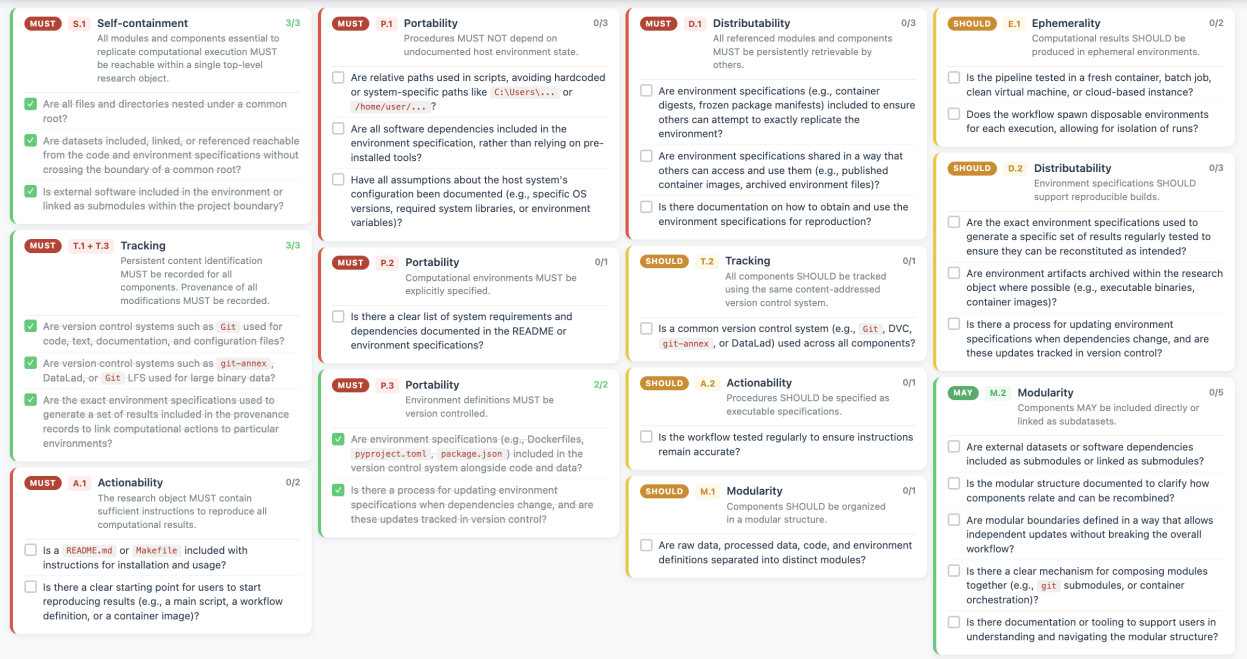

Figure 2: Screenshot of the interactive checklist for helping guide a workflow developer through common cases.

STAMPED properties are tool-agnostic. For example, any system that moves a property from documented to executable moves the research object further along the actionability spectrum, but certain tool categories recur across multiple properties; a well-chosen tool can address several dimensions at once, and a thoughtfully assembled toolchain can cover all of them. Researchers should select tools based on their domain, infrastructure, and team familiarity, recognizing that the tools listed here are illustrative rather than prescriptive: what matters is that the underlying property is satisfied, regardless of which tool achieves it.

Where multiple tools serve the same STAMPED property, the choice between them often involves trade-offs between flexibility and simplicity. A representative example is the choice between git-annex and Git LFS for large-file tracking. Both appear under Self-containment and Tracking in Table 3, but they occupy different points on a complexity-flexibility spectrum. git-annex supports many remote types (e.g. other git repositories over SSH or HTTP, or simple cloud data stores such as S3, local drives, offline USB archives), making it well-suited to multi-institutional collaboration, long-term archival where data sovereignty or offline access is required, as well as local content tracking. Git LFS offers a simpler alternative: it is transparent to users, natively supported by GitHub and GitLab, and broadly adopted. But it is centralized (requiring an LFS server), lacks offline capability, and provides less flexible remote configurations. Lightweight STAMPED implementations using Git LFS are entirely feasible for easier onboarding, trading federation flexibility for reduced operational overhead. The same pattern—flexibility versus simplicity—recurs across other tool choices in Table 3: container-based versus package-based environment management, workflow engines versus shell scripts, and distributed archives versus centralized platforms.

3.10 Examples and Case Studies

Several research projects have already adopted and documented YODA principles—the predecessor to STAMPED—demonstrating practical utility across domains.

BIDS Standard Evolution: A significant validation of the "do not look up" principle occurred within the Brain Imaging Data Structure (BIDS)[8] community. Originally, the BIDS specification nested derived data under `derivatives/` within the original raw dataset, creating upward dependencies that violated portability. Following YODA principles, the BIDS community reversed this relationship: derivative datasets now exist independently and reference raw data as subdatasets, not vice versa.

Table 3: Sample tools to improve scientific practice for each STAMPED property.

Property	Tools / Technologies	Role
S — Self-containment	Git submodules, DataLad, git-annex, DVC, Git LFS	Gather all components (code, data, environments) under a single root via direct inclusion or explicit references (e.g., subdatasets, registered URLs)
T — Tracking	Git, git-annex, DataLad, DVC, Docker and other containers, Kedro, Git LFS, OSF.io, Zenodo and other data archives providing DOIs	Version control for code and data; content-addressed identification; provenance recording (e.g., `datalad run` for programmatic provenance capture); persistent identifiers for content (DOIs)
A — Actionability	Make, Snakemake, Nextflow, CWL, `datalad run`, `git annex addcomputed`	Define executable specifications for workflows; provide clear entry points for reproducing results
M — Modularity	Git submodules, DataLad subdatasets, Kedro	Organize components into independently versioned, composable modules; provide project templates with logical directory separation
P — Portability	Docker, Singularity, Nix, conda, pip (`pyproject.toml`, `requirements.txt`), npm	Explicitly specify and isolate computational environments; container-based (OS-level isolation) or package-based (declarative, reproducible builds)
E — Ephemerality	Docker, Singularity, Docker Compose, Slurm, cloud deployments (e.g., AWS, GCP, Azure, GitHub Actions)	Spawn disposable, temporary environments per execution; validate that specifications are complete by reconstituting from scratch
D — Distributability	Zenodo, PyPI, conda-forge, DockerHub/GHCR, RO-Crate, `npm ci`, `pip freeze`, executable compilers	Archive and persistently host frozen, versioned research objects and their components; enable retrieval by others in the intended state

This architectural shift influenced major neuroimaging tools:

- **fMRIPrep**: Switched default output layout to follow YODA structure
- **OpenNeuroDerivatives**: Entire derivative dataset collection now follows YODA organization, separating processed outputs from raw data dependencies
- **BIDS specification**: Updated to accommodate and recommend YODA-compliant layouts for derivative datasets

Workflow Platforms: Infrastructure projects implementing YODA at scale:

- **BABS**[9]: Platform implementing the "FAIRly big workflow" approach, demonstrating YODA principles for large-scale neuroimaging analyses with containerized pipelines and modular dataset composition
- **CRCNS.org**: Neuroscience data sharing platform providing YODA-structured templates and validation tools

These adoptions demonstrate that STAMPED properties solve practical organizational challenges across scales, from individual studies to community-wide infrastructure.

3.11 Convergent evolution of data-driven scientific practices

The challenges that motivate STAMPED are not unique to any single domain. Over three decades, communities spanning geophysics, statistics, genomics, neuroimaging, and machine learning independently converged on strikingly similar organizational responses to the same recurring problems: fragmented provenance, irreproducible computations, and ad-hoc project layouts. What follows is not a comprehensive review but a tracing of that convergence, organized chronologically, to show that the properties STAMPED formalizes reflect durable responses to recurring challenges.

The earliest calls for reproducibility-as-organization appeared in the 1990s. Claerbout and Karrenbach[10] demonstrated that coupling electronic documents with Makefiles could make geophysical analyses self-contained, tracked, and re-executable—anticipating Self-containment (S), Tracking (T), and Actionability (A). Buckheit and Donoho[11] extended this idea with WaveLab, a compendium packaging code, data, and figures so that "an article ... is not the scholarship itself, ... the actual scholarship is the complete software development environment and the complete set of instructions which generated the figures"—touching Self-containment (S), Actionability (A), Portability (P), and Distributability (D). Gentleman and Temple Lang[12] formalized the research compendium concept, emphasizing modular packaging for reuse and redistribution—Self-containment (S), Modularity (M), Actionability (A), and Distributability (D). Noble[13] provided the first widely adopted project-organization guide for computational biology, offering concrete directory-structure conventions that addressed Self-containment (S) and Modularity (M).

A wave of normative frameworks followed in the 2010s. Stodden[14] articulated legal and social dimensions of sharing computational research, raising concerns central to Distributability (D) and Tracking (T). Sandve et al.[15] distilled ten rules for reproducible computation—track versions, record workflows, archive environments—spanning Tracking (T), Actionability (A), and Self-containment (S). The FAIR principles[1] established findability, accessibility, interoperability, and reusability as community norms for digital objects, later extended to research software by FAIR4RS[2] and to workflows by the Workflow Community Initiative[3]. Wilson et al.[16] codified "Good Enough Practices" that operationalized many of these ideals for working scientists—Actionability (A), Tracking (T), and Self-containment (S). The YODA principles[4] combined these insights into organizational practices for research data management. De Smedt et al.[17] articulated the notion of FAIR Digital Objects as "actionable knowledge units," foregrounding Actionability (A) as a first-class concern complementary to FAIR. Building on YODA and this emphasis on actionability, the VAMP formulation[18] (Versioned, Actionable, Modular, Portable) made Actionability (A), Portability (P), and Modularity (M) explicit organizational requirements, and introduced Ephemerality (E) as validation for Portability (P)—directly prefiguring STAMPED. Notably, the WCI-FW framework[3] extended FAIR to computational workflows but explicitly declined to define new principles, leaving the operational gap—how a research object should be structured and managed—that STAMPED fills.

Independent of these normative efforts, a parallel "Git for data" convergence occurred in tooling. git-annex (2010), Git LFS (2015), DVC (2017), Pachyderm (2014), and Icechunk (2024) all independently arrived at content-addressed identification of large files, logical separation of metadata from bulk storage, and derivation records linking inputs to outputs. Despite differing architectures—some cloud-native, some local-first, some centralized, some distributed—each converged on the same core insight: that version-control semantics proven for code could and should extend to data. This convergence is evidence not of influence but of shared need, and maps most directly onto Tracking (T).

A complementary convergence emerged in platforms and metadata standards. Though not an exhaustive list: Galaxy[19] (first released 2005), Code Ocean (2017), and brainlife.io[20] each provided turnkey reproducibility services, each implementing separation of code, data, and environment as a first-class abstraction—reflecting Self-containment (S), Actionability (A), and Portability (P). On the standards side, the W3C PROV data model (2013), Frictionless Data (2007), RO-Crate[21], and BioComputeObject (IEEE 2791-2020) each addressed interoperable packaging and provenance recording. The facts that FAIR required elaboration to accommodate software[2] and workflows[3;22], and that independent standards converged on provenance and packaging, confirms that these dimensions are recurrently demanded by research practice.

Framework tooling and domain standards reveal the same pattern at the project-layout level. Cookiecutter Data Science (2016), BIDS[8] (2016), Kedro (2019), nipoppy (2023), and others, all imposed opinionated directory structures and modular separation—independently motivated by the costs of ad-hoc layouts—reflecting Modularity (M) and Self-containment (S). Each framework constrains where code and documentation reside, how raw data inputs and derived outputs are handled, which makes such projects navigable by *convention*.

Viewed together, these independent lines of development map onto every STAMPED property. Self-containment (S) appears wherever projects bundle their dependencies into a single rooted structure; Tracking (T) wherever content-addressed storage or version control records the history of changes or acquisition of every component; Actionability (A) wherever Makefiles, workflow managers, or platform APIs make re-execution or acquisition a single command; Modularity (M) wherever directory conventions or subproject composition separate concerns; Portability (P) wherever good coding practice, containers or environment specifications decouple a research object from a particular machine; Ephemerality (E) wherever composition brings together computation on data which is treated as disposable and regenerable; and Distributability (D) wherever licensing, packaging standards, or data logistic mechanisms enable sharing.

The formalization presented here provides a shared vocabulary for what these practices have in common, enabling researchers to evaluate and incrementally improve their workflows regardless of the specific tools they employ. Figure 3 summarizes these parallel timelines and their mapping to STAMPED properties.

3.12 Future Directions

STAMPED principles provide a vocabulary rather than a closed specification, which is rooted in practices that continue to evolve with tooling, policy, and scientific convention. The directions below identify open fronts where the framework's vocabulary will need to stretch, and where complementary work by adjacent communities will shape how STAMPED assists researchers in the coming decade. We begin with directions that apply broadly across computational science — generalizing Ephemerality to workflow specifications, aligning provenance standards, building adoption incentives, and seeding training pipelines — and then devote the remainder of the section to the more specific and fast-moving challenges introduced by AI, where several STAMPED properties need explicit extensions and where the stakes for transparent, reproducible research are most immediate.

3.12.1 Specification-centric research objects as the reproducibility ideal

Ephemerality (E), as formulated here, treats the compute environment as disposable while the code and data it runs on remain durable. A complementary view is emerging in which workflow implementations themselves become regenerable from a durable specification. Declarative workflow languages such as CWL[23], Snakemake[24], and Nextflow[25], together with fully-declarative environment managers such as Nix and Guix, already operate this way: the specification accumulates the domain knowledge — inputs, constraints, expected outputs, validation criteria — while any given implementation can be rebuilt or replaced without

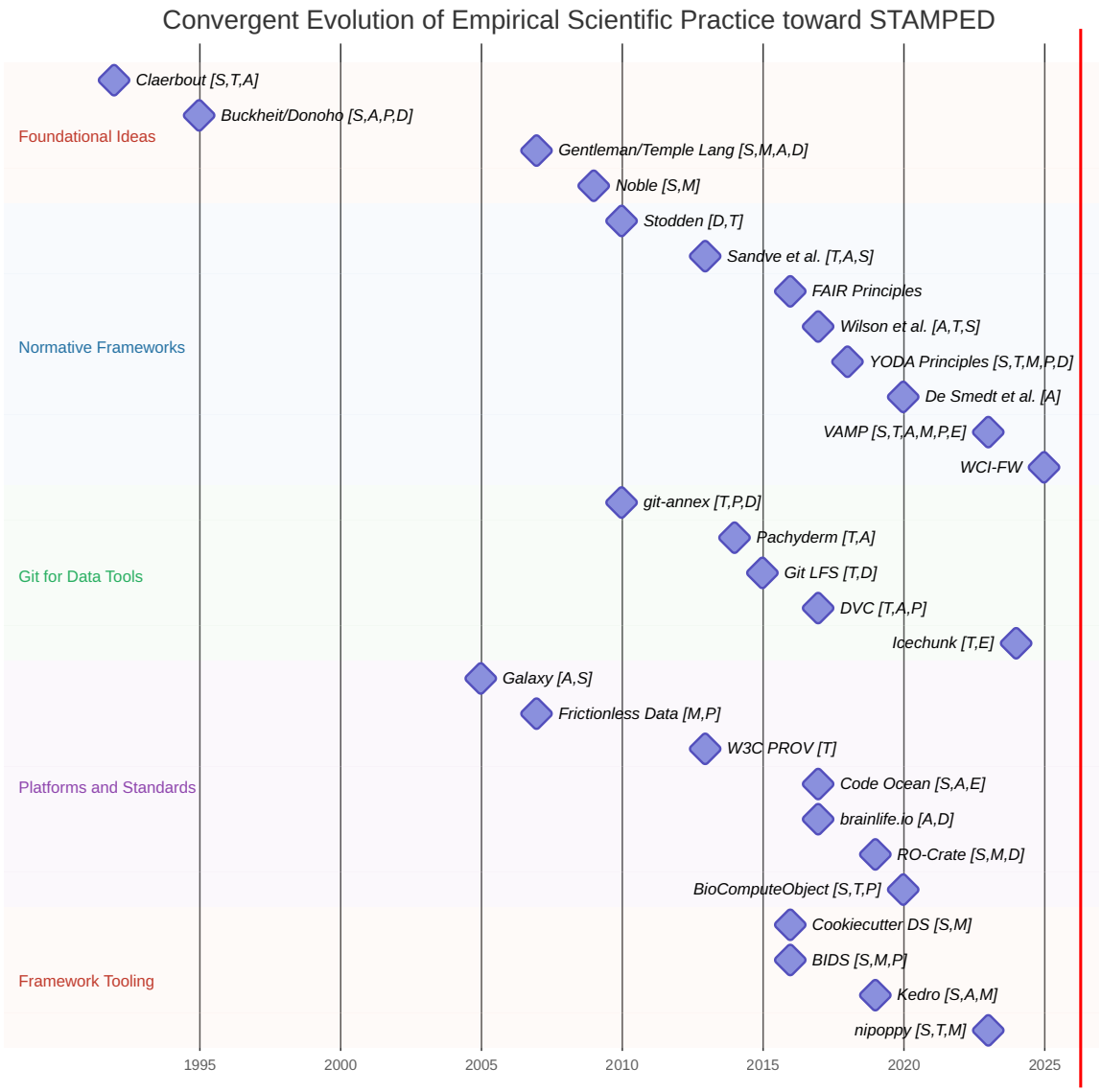

Figure 3: Convergent evolution of empirical scientific practice toward STAMPED. Five independent lines of development—foundational ideas, normative frameworks, data version-control tools, platforms and metadata standards, and framework tooling—are shown on a shared timeline from 1992 to 2025. Each milestone is annotated with the STAMPED properties it anticipates. Despite independent origins, all five tracks converged on the same organizational patterns that STAMPED formalizes. The milestones shown are illustrative rather than exhaustive; many other influential projects and communities contributed to this convergence but are omitted here for brevity.

loss. Under this view, STAMPED properties apply to the specification first and the generated implementation second: the specification is what must be Self-contained, Tracked, Actionable, and Distributable; the running code is ephemeral in the same sense as a container. The connection to pre-registration is direct and generalizes Ephemerality even further[26]: a well-formulated study specification is a commitment that survives any particular implementation, and what is ephemeral expands from compute environment to the full "scientific middleware" — data acquisition, harmonization, preprocessing, analytics — each stage reduced to a specification from which implementations can be derived. That is where concentration on formalization and standardization of overall study specifications becomes the target to apply STAMPED properties, so that the middleware steps could also be automatically carried out in an ephemeral fashion and potentially while exploring various possible implementations. The cost of under-specified middleware is already visible in practice: the NARPS study showed that 70 independent teams analyzing the same fMRI dataset with nominally similar hypotheses produced substantially divergent results[27], precisely because the study specification left most of the middleware up to each team.

3.12.2 Provenance standards convergence

As FAIR leaves specific protocols and standards open, STAMPED likewise does not prescribe a serialization for provenance records, and in practice researchers rely on whatever their tooling emits. Convergence among existing approaches — cross-domain standards such as W3C PROV[28] and RO-Crate[21], tool-level records such as DataLad's[29] `datalad run` provenance, and domain-specific specifications such as BIDS-Prov (BEP028)[30] — would let STAMPED research objects carry portable provenance across toolchains without lock-in. This is less a change to STAMPED itself than a call on these communities: Tracking (T) becomes substantially more useful when provenance records follow shared standards and can be exported, compared, and re-interpreted by tools other than the one that generated them.

3.12.3 Adoption incentives and tool certification

STAMPED compliance today is self-assessed against the checklist provided in this paper and shared at `https://github.com/stamped-principles/stamped-checklist`. Broader adoption will depend on promotion and external incentives: funder mandates (e.g., NIH Data Management and Sharing policies), journal reproducibility badges[31], and venue-level review checklists that converge toward automated scoring — paralleling the trajectory of FAIR, which has moved from principles to machine-assessable indicators with tools such as F-UJI[32]. No tool certifies STAMPED compliance today; a community-maintained benchmark suite, seeded by the satellite examples project associated with this paper (`https://stamped-principles.github.io/stamped-examples`), is a natural precursor.

3.12.4 Teaching and onboarding

Uptake through formal training is slow but compounding: as the "train the trainer" model used by the ReproNim program has demonstrated, students and early-career researchers who encounter these practices during training become some of the most effective propagators of the vocabulary, even if that propagation takes years to surface. Existing resources — EduWrench[33], the Reproducible Research MOOC which teaches `git-annex` and related tools[34], and RDMkit data analysis best practices[35] — already teach most of what STAMPED asks for, without using the term. Curricular integration with open-science and neuroimaging training programs such as Brainhack would close the gap between practitioners who apply these practices by habit and those who can name what they are doing and evaluate it against a shared checklist. YODA principles have already entered the curricula of several such programs, and we expect their references to migrate toward the more formally articulated STAMPED vocabulary; continued community outreach — including a recent introduction to STAMPED[36] delivered to the BRAIN BBQS program — supports that transition.

3.12.5 Research in an AI Era

AI agents are now generating analysis code, selecting methods, and in some systems running entire research cycles end-to-end[37–39]. Without explicit structure, this automation can produce results whose provenance and

reasoning are irrecoverable[40]. STAMPED properties already carry most of what is needed — self-contained workspaces give agents a well-defined boundary, tracked histories make agent contributions auditable and contribute to their "memory", and actionable documentation serves humans and machines alike — but, as with any new methodology or instrumentation entering science, several of these properties need AI-era extensions. Because the agentic-AI ecosystem is evolving rapidly, the directions below are indicative rather than exhaustive.

Self-contained models. Machine learning models are becoming standard components of research workflows, yet they are often fetched at runtime from external repositories with no guarantee of long-term availability or version stability. Users frequently interact with models through platforms without knowing which model version produced their results, and even where models are nominally versioned the degree of control varies widely: Hugging Face provides commit-level pinning, while commercial APIs may silently update the model behind a fixed endpoint. Self-containment (S) extends naturally to this case: models are tracked components of the research object, just as data and code are today. Practical mechanisms exist for open models, such as pinned revisions on Hugging Face or, for durable self-containment, archival copies managed via `git-annex`. Tracking what training data and procedures produced a given model remains an open problem that intersects copyright, benchmarking, and provenance — mostly beyond what STAMPED can prescribe — but community artifacts such as Model Cards[41] and Datasheets for Datasets[42] point toward the shape of an answer.

Provenance for AI-driven changes. When an AI agent modifies code, generates analysis scripts, or makes architectural decisions, those actions become part of the research record. Tracking (T) naturally extends to cover agent actions: wrapping an agent invocation in a provenance-capturing command such as `datalad run` records the model, prompt, and resulting changes alongside the output. Some of this provenance is already captured organically through git commits authored by agents, session transcripts, and co-authorship metadata; emerging tools such as Entire CLI[43] go further, automating capture of the entire AI session alongside the commit. A practical tension follows: agent sessions can produce enormous transcripts, and including them wholesale may dilute the value of more targeted provenance records, so what to capture and at what granularity remain open questions. Bitwise reproducibility is not the target and rarely is the goal even for human-driven steps; what the record must preserve is the hypothesis, the method specification, and enough of the decision trail to support introspection. Self-contained models (above) strengthen this chain: when the model itself is tracked, the provenance record is more complete.

Machine-actionable instructions. Actionability (A) was defined with human operators in mind — entry-point commands documented in READMEs, frequently incomplete or out of sync with the underlying workflow. AI agents are a second consumer of the same prose, one that can parse, follow, and correct instructions on the fly, and plain Markdown has effectively become the interface language: `CLAUDE.md` files, README-driven agent workflows, and spec-kit[44] collections now function as machine-actionable interfaces, making Self-contained and Tracked research objects more Actionable. Where prose instructions fail, the failure is legible — an audit mechanism that did not exist before — even if agent-driven execution remains less reliable than deterministic tooling.

AI-accelerated specification-to-implementation. Specification-centric research objects (above) depend on the cost of regenerating an implementation from a specification being low enough to be routine. AI coding assistants and specification-driven toolkits such as spec-kit move this cost toward zero, and make the spec-centric view practical rather than aspirational — provided the preceding extensions for models, provenance, and machine-actionable instructions are in place. This closes a loop with the end-to-end "AI scientist" systems that opened this section[37–39]: as soon as an agentic pipeline takes a study specification and returns data, code, figures, and a draft manuscript, the specification and every downstream artifact must together satisfy STAMPED — not only to keep such pipelines efficient to re-run at lower cost, but to make their outputs introspectable. Tracked and self-contained research objects let a researcher inspect which model, prompt, and decision led to a particular result, implement "hand over" between humans and agents alike and let subsequent actor re-enter a prior state and continue from it without rediscovering context. At the end, human scientists authoring and iterating over the specification of a STAMPED research object would see

how their edits propagate through the generated implementation. Without these properties, the acceleration might be real but the accountability would be lost. With STAMPED properties spec-driven agentic research becomes a collaborative transparent artifact between scientists and the AI systems that serves them both.

4 Data Availability

5 Code Availability

References

[1] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino Da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott Edmunds, Chris T. Evelo, Richard Finkers, Alejandra Gonzalez-Beltran, Alasdair J.G. Gray, Paul Groth, Carole Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A.C 'T Hoen, Rob Hooft, Tobias Kuhn, Ruben Kok, Joost Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel L. Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, Rene Van Schaik, Susanna-Assunta Sansone, Erik Schultes, Thierry Sengstag, Ted Slater, George Strawn, Morris A. Swertz, Mark Thompson, Johan Van Der Lei, Erik Van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katherine Wolstencroft, Jun Zhao, and Barend Mons. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, March 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.18. URL https://www.nature.com/articles/sdata201618.

[2] Michelle Barker, Neil P. Chue Hong, Daniel S. Katz, Anna-Lena Lamprecht, Carlos Martinez-Ortiz, Fotis Psomopoulos, Jennifer Harrow, Leyla Jael Castro, Morane Gruenpeter, Paula Andrea Martinez, and Tom Honeyman. Introducing the FAIR Principles for research software. *Scientific Data*, 9(1):622, October 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01710-x. URL https://www.nature.com/articles/s41597-022-01710-x.

[3] Sean R. Wilkinson, Meznah Aloqalaa, Khalid Belhajjame, Michael R. Crusoe, Bruno De Paula Kinoshita, Luiz Gadelha, Daniel Garijo, Ove Johan Ragnar Gustafsson, Nick Juty, Sehrish Kanwal, Farah Zaib Khan, Johannes Köster, Karsten Peters-von Gehlen, Line Pouchard, Randy K. Rannow, Stian Soiland-Reyes, Nicola Soranzo, Shoaib Sufi, Ziheng Sun, Baiba Vilne, Merridee A. Wouters, Denis Yuen, and Carole Goble. Applying the FAIR Principles to computational workflows. *Scientific Data*, 12(1):328, February 2025. ISSN 2052-4463. doi: 10.1038/s41597-025-04451-9. URL https://www.nature.com/articles/s41597-025-04451-9.

[4] Michael Hanke, Matteo Visconti di Oleggio Castello, Kyle Meyer, Benjamin Poldrack, and Yaroslav O. Halchenko. YODA: YODA's organigram on data analysis, 2018. URL https://github.com/myyoda/poster/blob/master/ohbm2018.pdf. Published: Poster presented at the annual meeting of the Organization for Human Brain Mapping, Singapore.

[5] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels. Technical Report RFC2119, RFC Editor, March 1997. URL https://www.rfc-editor.org/info/rfc2119.

[6] Erik C. Johnson, Thinh T. Nguyen, Benjamin K. Dichter, Frank Zappulla, Montgomery Kosma, Kabilar Gunalan, Yaroslav O. Halchenko, Shay Q. Neufeld, Kristen Ratan, Nicholas J. Edwards, Susanne Ressl, Sarah R. Heilbronner, Michael Schirner, Petra Ritter, Brock Wester, Satrajit Ghosh, Maryann E. Martone, Franco Pestilli, and Dimitri Yatsenko. SciOps: Achieving Productivity and Reliability in Data-Intensive Research, 2024. URL https://arxiv.org/abs/2401.00077.

[7] Adina S. Wagner, Laura K. Waite, Małgorzata Wierzba, Felix Hoffstaedter, Alexander Q. Waite, Benjamin Poldrack, Simon B. Eickhoff, and Michael Hanke. FAIRly big: A framework for computationally reproducible processing of large-scale data. *Scientific Data*, 9(1):80, March 2022. ISSN 2052-4463. doi: 10.1038/s41597-022-01163-2. URL https://www.nature.com/articles/s41597-022-01163-2.

[8] Krzysztof J. Gorgolewski, Tibor Auer, Vince D. Calhoun, R. Cameron Craddock, Samir Das, Eugene P. Duff, Guillaume Flandin, Satrajit S. Ghosh, Tristan Glatard, Yaroslav O. Halchenko, Daniel A. Handwerker, Michael Hanke, David Keator, Xiangrui Li, Zachary Michael, Camille Maumet, B. Nolan Nichols, Thomas E. Nichols, John Pellman, Jean-Baptiste Poline, Ariel Rokem, Gunnar Schaefer, Vanessa Sochat, William Triplett, Jessica A. Turner, Gaël Varoquaux, and Russell A. Poldrack. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, 3(1):160044, June 2016. ISSN 2052-4463. doi: 10.1038/sdata.2016.44. URL https://www.nature.com/articles/sdata201644.

[9] Chenying Zhao, Dorota Jarecka, Sydney Covitz, Yibei Chen, Simon B. Eickhoff, Damien A. Fair, Alexandre R. Franco, Yaroslav O. Halchenko, Timothy J. Hendrickson, Felix Hoffstaedter, Audrey Houghton, Gregory Kiar, Austin Macdonald, Kahini Mehta, Michael P. Milham, Taylor Salo, Michael Hanke, Satrajit S. Ghosh, Matthew Cieslak, and Theodore D. Satterthwaite. A reproducible and generalizable software workflow for analysis of large-scale neuroimaging data collections using BIDS Apps. *Imaging Neuroscience*, 2:imag–2–00074, January 2024. ISSN 2837-6056. doi: 10.1162/imag_a_00074. URL https://direct.mit.edu/imag/article/doi/10.1162/imag_a_00074/119046/A-reproducible-and-generalizable-software-workflow.

[10] Jon F. Claerbout and Martin Karrenbach. Electronic documents give reproducible research a new meaning. In *SEG Technical Program Expanded Abstracts 1992*, pages 601–604. Society of Exploration Geophysicists, January 1992. doi: 10.1190/1.1822162. URL http://library.seg.org/doi/abs/10.1190/1.1822162.

[11] Jonathan B. Buckheit and David L. Donoho. WaveLab and Reproducible Research. In P. Bickel, P. Diggle, S. Fienberg, K. Krickeberg, I. Olkin, N. Wermuth, S. Zeger, Anestis Antoniadis, and Georges Oppenheim, editors, *Wavelets and Statistics*, volume 103, pages 55–81. Springer New York, New York, NY, 1995. ISBN 978-0-387-94564-4 978-1-4612-2544-7. doi: 10.1007/978-1-4612-2544-7_5. URL http://link.springer.com/10.1007/978-1-4612-2544-7_5. Series Title: Lecture Notes in Statistics.

[12] Robert Gentleman and Duncan Temple Lang. Statistical Analyses and Reproducible Research. *Journal of Computational and Graphical Statistics*, 16(1):1–23, March 2007. ISSN 1061-8600, 1537-2715. doi: 10.1198/106186007X178663. URL http://www.tandfonline.com/doi/abs/10.1198/106186007X178663.

[13] William Stafford Noble. A Quick Guide to Organizing Computational Biology Projects. *PLoS Computational Biology*, 5(7):e1000424, July 2009. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000424. URL https://dx.plos.org/10.1371/journal.pcbi.1000424.

[14] Victoria Stodden. The Scientific Method in Practice: Reproducibility in the Computational Sciences. *SSRN Electronic Journal*, 2010. ISSN 1556-5068. doi: 10.2139/ssrn.1550193. URL http://www.ssrn.com/abstract=1550193.

[15] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten Simple Rules for Reproducible Computational Research. *PLoS Computational Biology*, 9(10):e1003285, October 2013. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003285. URL https://dx.plos.org/10.1371/journal.pcbi.1003285.

[16] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Lex Nederbragt, and Tracy K. Teal. Good enough practices in scientific computing. *PLOS Computational Biology*, 13(6):e1005510, June 2017. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005510. URL https://dx.plos.org/10.1371/journal.pcbi.1005510.

[17] Koenraad De Smedt, Dimitris Koureas, and Peter Wittenburg. FAIR Digital Objects for Science: From Data Pieces to Actionable Knowledge Units. *Publications*, 8(2):21, April 2020. ISSN 2304-6775. doi: 10.3390/publications8020021. URL https://www.mdpi.com/2304-6775/8/2/21.

[18] Michael Hanke. What is DataLad and what can it do for you?, 2023. URL https://files.inm7.de/mih/pres/talks/whatisdatalad_2023.html.

[19] Enis Afgan, Dannon Baker, Bérénice Batut, Marius van den Beek, Dave Bouvier, Martin Čech, John Chilton, Dave Clements, Nate Coraor, Björn A Grüning, Aysam Guerler, Jennifer Hillman-Jackson, Saskia Hiltemann, Vahid Jalili, Helena Rasche, Nicola Soranzo, Jeremy Goecks, James Taylor, Anton Nekrutenko, and Daniel Blankenberg. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, 46(W1):W537–W544, July 2018. ISSN 0305-1048, 1362-4962. doi: 10.1093/nar/gky379. URL https://academic.oup.com/nar/article/46/W1/W537/5001157.

[20] Soichi Hayashi, Bradley A. Caron, Anibal Sólon Heinsfeld, Sophia Vinci-Booher, Brent McPherson, Daniel N. Bullock, Giulia Bertò, Guiomar Niso, Sandra Hanekamp, Daniel Levitas, Kimberly Ray, Anne MacKenzie, Paolo Avesani, Lindsey Kitchell, Josiah K. Leong, Filipi Nascimento-Silva, Serge Koudoro, Hanna Willis, Jasleen K. Jolly, Derek Pisner, Taylor R. Zuidema, Jan W. Kurzawski, Kyriaki Mikellidou, Aurore Bussalb, Maximilien Chaumon, Nathalie George, Christopher Rorden, Conner Victory, Dheeraj Bhatia, Dogu Baran Aydogan, Fang-Cheng F. Yeh, Franco Delogu, Javier Guaje, Jelle Veraart, Jeremy Fischer, Joshua Faskowitz, Ricardo Fabrega, David Hunt, Shawn McKee, Shawn T. Brown, Stephanie Heyman, Vittorio Iacovella, Amanda F. Mejia, Daniele Marinazzo, R. Cameron Craddock, Emanuale Olivetti, Jamie L. Hanson, Eleftherios Garyfallidis, Dan Stanzione, James Carson, Robert Henschel, David Y. Hancock, Craig A. Stewart, David Schnyer, Damian O. Eke, Russell A. Poldrack, Steffen Bollmann, Ashley Stewart, Holly Bridge, Ilaria Sani, Winrich A. Freiwald, Aina Puce, Nicholas L. Port, and Franco Pestilli. brainlife.io: a decentralized and open-source cloud platform to support neuroscience research. *Nature Methods*, 21(5):809–813, May 2024. ISSN 1548-7091, 1548-7105. doi: 10.1038/s41592-024-02237-2. URL https://www.nature.com/articles/s41592-024-02237-2.

[21] Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José M. Fernández, Daniel Garijo, Björn Grüning, Marco La Rosa, Simone Leo, Eoghan Ó Carragáin, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, and Carole Goble. Packaging research artefacts with RO-Crate. *Data Science*, 5(2):97–138, July 2022. ISSN 2451-8484, 2451-8492. doi: 10.3233/DS-210053. URL https://journals.sagepub.com/doi/10.3233/DS-210053.

[22] Carole Goble, Sarah Cohen-Boulakia, Stian Soiland-Reyes, Daniel Garijo, Yolanda Gil, Michael R. Crusoe, Kristian Peters, and Daniel Schober. FAIR Computational Workflows. *Data Intelligence*, 2(1-2): 108–121, January 2020. ISSN 2641-435X. doi: 10.1162/dint_a_00033. URL https://www.sciengine.com/doi/10.1162/dint_a_00033.

[23] Michael R. Crusoe, Sanne Abeln, Alexandru Iosup, Peter Amstutz, John Chilton, Nebojša Tijanić, Hervé Ménager, Stian Soiland-Reyes, Bogdan Gavrilović, Carole Goble, and The Cwl Community. Methods included: standardizing computational reuse and portability with the Common Workflow Language. *Communications of the ACM*, 65(6):54–63, June 2022. ISSN 0001-0782, 1557-7317. doi: 10.1145/3486897. URL https://dl.acm.org/doi/10.1145/3486897.

[24] Felix Mölder, Kim Philipp Jablonski, Brice Letcher, Michael B. Hall, Christopher H. Tomkins-Tinch, Vanessa Sochat, Jan Forster, Soohyun Lee, Sven O. Twardziok, Alexander Kanitz, Andreas Wilm, Manuel Holtgrewe, Sven Rahmann, Sven Nahnsen, and Johannes Köster. Sustainable data analysis with Snakemake. *F1000Research*, 10:33, April 2021. ISSN 2046-1402. doi: 10.12688/f1000research.29032.2. URL https://f1000research.com/articles/10-33/v2.

[25] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4): 316–319, April 2017. ISSN 1087-0156, 1546-1696. doi: 10.1038/nbt.3820. URL https://www.nature.com/articles/nbt.3820.

[26] Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. The preregistration revolution. *Proceedings of the National Academy of Sciences*, 115(11):2600–2606, March 2018. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1708274114. URL https://pnas.org/doi/full/10.1073/pnas.1708274114.

[27] Rotem Botvinik-Nezer, Felix Holzmeister, Colin F. Camerer, Anna Dreber, Juergen Huber, Magnus Johannesson, Michael Kirchler, Roni Iwanir, Jeanette A. Mumford, R. Alison Adcock, Paolo Avesani, Blazej M. Baczkowski, Aahana Bajracharya, Leah Bakst, Sheryl Ball, Marco Barilari, Nadège Bault, Derek Beaton, Julia Beitner, Roland G. Benoit, Ruud M. W. J. Berkers, Jamil P. Bhanji, Bharat B. Biswal, Sebastian Bobadilla-Suarez, Tiago Bortolini, Katherine L. Bottenhorn, Alexander Bowring, Senne Braem, Hayley R. Brooks, Emily G. Brudner, Cristian B. Calderon, Julia A. Camilleri, Jaime J. Castrellon, Luca Cecchetti, Edna C. Cieslik, Zachary J. Cole, Olivier Collignon, Robert W. Cox, William A. Cunningham, Stefan Czoschke, Kamalaker Dadi, Charles P. Davis, Alberto De Luca, Mauricio R. Delgado, Lysia Demetriou, Jeffrey B. Dennison, Xin Di, Erin W. Dickie, Ekaterina Dobryakova, Claire L. Donnat, Juergen Dukart, Niall W. Duncan, Joke Durnez, Amr Eed, Simon B. Eickhoff, Andrew Erhart, Laura Fontanesi, G. Matthew Fricke, Shiguang Fu, Adriana Galván, Remi Gau, Sarah Genon, Tristan Glatard, Enrico Glerean, Jelle J. Goeman, Sergej A. E. Golowin, Carlos González-García, Krzysztof J. Gorgolewski, Cheryl L. Grady, Mikella A. Green, João F. Guassi Moreira, Olivia Guest, Shabnam Hakimi, J. Paul Hamilton, Roeland Hancock, Giacomo Handjaras, Bronson B. Harry, Colin Hawco, Peer Herholz, Gabrielle Herman, Stephan Heunis, Felix Hoffstaedter, Jeremy Hogeveen, Susan Holmes, Chuan-Peng Hu, Scott A. Huettel, Matthew E. Hughes, Vittorio Iacovella, Alexandru D. Iordan, Peder M. Isager, Ayse I. Isik, Andrew Jahn, Matthew R. Johnson, Tom Johnstone, Michael J. E. Joseph, Anthony C. Juliano, Joseph W. Kable, Michalis Kassinopoulos, Cemal Koba, Xiang-Zhen Kong, Timothy R. Koscik, Nuri Erkut Kucukboyaci, Brice A. Kuhl, Sebastian Kupek, Angela R. Laird, Claus Lamm, Robert Langner, Nina Lauharatanahirun, Hongmi Lee, Sangil Lee, Alexander Leemans, Andrea Leo, Elise Lesage, Flora Li, Monica Y. C. Li, Phui Cheng Lim, Evan N. Lintz, Schuyler W. Liphardt, Annabel B. Losecaat Vermeer, Bradley C. Love, Michael L. Mack, Norberto Malpica, Theo Marins, Camille Maumet, Kelsey McDonald, Joseph T. McGuire, Helena Melero, Adriana S. Méndez Leal, Benjamin Meyer, Kristin N. Meyer, Glad Mihai, Georgios D. Mitsis, Jorge Moll, Dylan M. Nielson, Gustav Nilsonne, Michael P. Notter, Emanuele Olivetti, Adrian I. Onicas, Paolo Papale, Kaustubh R. Patil, Jonathan E. Peelle, Alexandre Pérez, Doris Pischedda, Jean-Baptiste Poline, Yanina Prystauka, Shruti Ray, Patricia A. Reuter-Lorenz, Richard C. Reynolds, Emiliano Ricciardi, Jenny R. Rieck, Anais M. Rodriguez-Thompson, Anthony Romyn, Taylor Salo, Gregory R. Samanez-Larkin, Emilio Sanz-Morales, Margaret L. Schlichting, Douglas H. Schultz, Qiang Shen, Margaret A. Sheridan, Jennifer A. Silvers, Kenny Skagerlund, Alec Smith, David V. Smith, Peter Sokol-Hessner, Simon R. Steinkamp, Sarah M. Tashjian, Bertrand Thirion, John N. Thorp, Gustav Tinghög, Loreen Tisdall, Steven H. Tompson, Claudio Toro-Serey, Juan Jesus Torre Tresols, Leonardo Tozzi, Vuong Truong, Luca Turella, Anna E. Van 'T Veer, Tom Verguts, Jean M. Vettel, Sagana Vijayarajah, Khoi Vo, Matthew B. Wall, Wouter D. Weeda, Susanne Weis, David J. White, David Wisniewski, Alba Xifra-Porxas, Emily A. Yearling, Sangsuk Yoon, Rui Yuan, Kenneth S. L. Yuen, Lei Zhang, Xu Zhang, Joshua E. Zosky, Thomas E. Nichols, Russell A. Poldrack, and Tom Schonberg. Variability in the analysis of a single neuroimaging dataset by many teams. *Nature*, 582(7810):84–88, June 2020. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-020-2314-9. URL https://www.nature.com/articles/s41586-020-2314-9.

[28] W3C Provenance Working Group, Yolanda Gil, and Simon Miles. PROV-Overview: An Overview of the PROV Family of Documents, April 2013. URL https://www.w3.org/TR/prov-overview/. Published: W3C Working Group Note.

[29] Yaroslav O. Halchenko, Kyle Meyer, Benjamin Poldrack, Debanjum Singh Solanky, Adina S. Wagner, Jason Gors, Dave MacFarlane, Dorian Pustina, Vanessa Sochat, Satrajit S. Ghosh, Christian Mönch, Christopher J. Markiewicz, Laura Waite, Ilya Shlyakhter, Alejandro de la Vega, Soichi Hayashi, Christian Olaf Häusler, Jean-Baptiste Poline, Tobias Kadelka, Kusti Skytén, Dorota Jarecka, David Kennedy, Ted Strauss, Matt Cieslak, Peter Vavra, Horea-Ioan Ioanas, Robin Schneider, Mika Pflüger, James V. Haxby, Simon B. Eickhoff, and Michael Hanke. DataLad: distributed system for joint management of code, data, and their relationship. *Journal of Open Source Software*, 6(63):3262, July 2021. ISSN 2475-9066. doi: 10.21105/joss.03262. URL https://joss.theoj.org/papers/10.21105/joss.03262.

[30] BIDS Community. BEP028: BIDS Extension Proposal for BIDS Provenance, 2025. URL https://github.com/bids-standard/BEP028_BIDSprov. Published: BIDS Extension Proposal.

[31] Center for Open Science. Open Science Badges, 2023. URL https://www.cos.io/initiatives/badges. Published: Center for Open Science initiative.

[32] Anusuriya Devaraju and Robert Huber. An automated solution for measuring the progress toward FAIR research data. *Patterns*, 2(11):100370, November 2021. ISSN 2666-3899. doi: 10.1016/j.patter.2021.100370. URL https://doi.org/10.1016/j.patter.2021.100370.

[33] Henri Casanova and others. EduWrench: Pedagogic Modules for Parallel and Distributed Computing, 2023. URL https://eduwrench.ics.hawaii.edu/. Published: Online teaching platform, University of Hawai‘ i.

[34] Arnaud Legrand, Martin Quinson, Bruno Bzeznik, Vincent Danjean, Michael Mercier, and Olivier Richard. Reproducible Research II: Practices and Tools for Managing Computations and Data, 2022. URL https://www.fun-mooc.fr/en/courses/reproducible-research-ii-practices-and-tools-for-managing-comput/. Published: FUN-MOOC online course.

[35] ELIXIR RDMkit Community. Data analysis best practices, 2024. URL https://rdmkit.elixir-europe.org/data_analysis. Published: ELIXIR Research Data Management Kit.

[36] Cody Baker. Guidelines for Reproducible Research (STAMPED), March 2026. URL https://www.youtube.com/watch?v=8NTWKHer5Zo. Published: Talk presented at the Virtual BBQS Workshop, Center for Open Neuroscience.

[37] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, December 2023. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0.

[38] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, September 2024. URL http://arxiv.org/abs/2408.06292. arXiv:2408.06292 [cs].

[39] Ludovico Mitchener, Angela Yiu, Benjamin Chang, Mathieu Bourdenx, Tyler Nadolski, Arvis Sulovari, Eric C. Landsness, Daniel L. Barabasi, Siddharth Narayanan, Nicky Evans, Shriya Reddy, Martha Foiani, Aizad Kamal, Leah P. Shriver, Fang Cao, Asmamaw T. Wassie, Jon M. Laurent, Edwin Melville-Green, Mayk Caldas, Albert Bou, Kaleigh F. Roberts, Sladjana Zagorac, Timothy C. Orr, Miranda E. Orr, Kevin J. Zwezdaryk, Ali E. Ghareeb, Laurie McCoy, Bruna Gomes, Euan A. Ashley, Karen E. Duff, Tonio Buonassisi, Tom Rainforth, Randall J. Bateman, Michael Skarlinski, Samuel G. Rodriques, Michaela M. Hinks, and Andrew D. White. Kosmos: An AI Scientist for Autonomous Discovery, November 2025. URL http://arxiv.org/abs/2511.02824. arXiv:2511.02824 [cs].

[40] Lisa Messeri and M. J. Crockett. Artificial intelligence and illusions of understanding in scientific research. *Nature*, 627(8002):49–58, March 2024. ISSN 0028-0836, 1476-4687. doi: 10.1038/s41586-024-07146-0. URL https://www.nature.com/articles/s41586-024-07146-0.

[41] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model Cards for Model Reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 220–229, Atlanta GA USA, January 2019. ACM. ISBN 978-1-4503-6125-5. doi: 10.1145/3287560.3287596. URL https://dl.acm.org/doi/10.1145/3287560.3287596.

[42] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, December 2021. ISSN 0001-0782, 1557-7317. doi: 10.1145/3458723. URL https://dl.acm.org/doi/10.1145/3458723.

[43] Entire. Entire CLI: Git observability layer that captures AI agent sessions as checkpoints alongside commits, 2026. URL https://github.com/entireio/cli. Published: Software, GitHub repository.

[44] GitHub. Spec Kit: Toolkit for Spec-Driven Development, 2025. URL https://github.com/github/spec-kit. Published: Software, GitHub repository.

6 Author Contributions

7 Competing Interests

The authors declare no competing interests.

8 Acknowledgments

This manuscript and companion materials in the https://github.com/stamped-principles/ were prepared with the assistance of agentic AI coding tools and large language models (primarily Claude Opus). All final text, figures, and specifications were edited and/or confirmed by the human authors, who are responsible for the content. Where AI tools notably contributed to a change, we have strived to annotate the corresponding commits with a `Co-Authored-By` trailer, or analogous comments, to identify the tool and potentially the model version, so that the provenance of individual contributions is inspectable in the public git history of the `stamped-*` repositories.

9 Funding

This work was supported by the National Institutes of Health: ReproNim — Center for Reproducible Neuroimaging Computation (NIBIB P41 EB019936), OpenNeuro — An Open Archive for Analysis and Sharing of BRAIN Initiative Data (NIMH R24 MH117179), DANDI — Distributed Archives for Neurophysiology Data Integration (NIMH R24 MH117295), and EMBER — Ecosystem for Multi-modal Brain-behavior Experimentation and Research (NIMH R24 MH136632).