Reproducibility, Git, and Research Artifacts

Jared Edgerton

Why Reproducibility Matters

Reproducibility asks a simple question:

- Can someone else obtain the same results
- Using the same data and code
- Without informal guidance

This is a minimum standard for credible research.

The Replication Crisis

Across the social and behavioral sciences:

- Many published findings fail to replicate
- Effect sizes shrink or disappear
- Results depend on analytic choices

This is not about fraud; it is about systems.

Sources of Replication Failure

Common contributors include:

- Researcher degrees of freedom
- Small samples and noisy estimates
- Selective reporting
- Multiple hypothesis testing

Failures often emerge at scale.

Multiple Comparisons in Practice

Many studies implicitly test:

- Multiple outcomes
- Multiple specifications
- Multiple subgroups

Even with a single natural experiment, false positives accumulate without correction.

Natural Experiments and Multiplicity

Natural experiments often involve:

- Many outcomes per treatment
- Event-study coefficients over time
- Sensitivity across windows and controls

Multiplicity is structural, not accidental.

Specification Search

Common but risky practices:

- Trying alternative controls
- Varying bandwidths or cutoffs
- Exploring subgroups post hoc

Without discipline, robustness becomes overfitting.

What Replication Really Tests

Replication evaluates:

- Data access
- Code executability
- Analytic transparency
- Sensitivity to assumptions

Exact numerical equality is rarely the point.

Computational Reproducibility

Even with shared code and data:

- Results may differ across machines
- Library versions change behavior
- Randomness is handled differently

Computation is part of the research design.

Operating System Differences

Results can vary across:

- Linux vs macOS vs Windows
- File system behavior
- Floating-point implementations

Assuming identical environments is unsafe.

Version Drift

Over time:

- R and Python packages update
- Default behaviors change
- Deprecated functions break pipelines

Code that ran once may not run again.

Environment Capture

Best practice includes:

- Explicit package versions
- Lockfiles (e.g., renv, requirements.txt)
- Minimal system dependencies

Environments must be specified, not inferred.

Git as a Scientific Tool

Git supports:

- Versioned code history
- Branching and experimentation
- Transparent collaboration

Git records *how* results evolved.

Commits as Research Units

Each commit should:

- Represent a logical change
- Be reversible
- Be interpretable

Commit history is a research log.

Containers

Containers:

- Bundle code and dependencies
- Isolate execution environments
- Improve portability

They reduce, but do not eliminate, variability.

Containers Are Not Magic

Containers do not fix:

- Data errors
- Conceptual mistakes
- Poor documentation

They address execution, not inference.

Research Artifacts

A reproducible project produces:

- Data
- Code
- Documentation
- Execution instructions

The artifact is the contribution.

Documentation Is Essential

Reproducibility requires:

- Clear READMEs
- Data provenance
- Known limitations
- Explicit assumptions

Undocumented pipelines are irreproducible.

What We Emphasize in Practice

- Treat reproducibility as design
- Anticipate multiplicity
- Capture environments early
- Make artifacts usable by others

Discussion

- What kinds of results are hardest to reproduce?
- Where does multiplicity enter your work?
- What is the minimal reproducible artifact?