



MathSci.ai

Taming the Chaos of Computational Experiments

Tammy Kolda



SIAM Conference on Dynamical Systems (DS25)

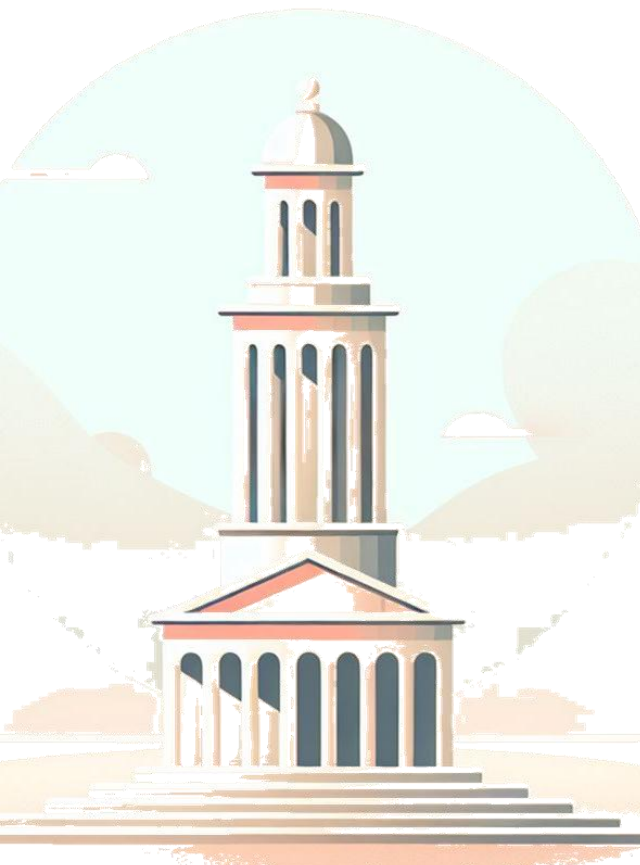
Denver, CO

May 12, 2025

Kolda @ SIAM DS25

5/12/2025



A stylized illustration of a classical building with a dome and columns, set against a light blue and orange background.

“As computational mathematics becomes increasingly important outside the ivory tower because of these large simulation codes, ... *the way we present our work can play a role in the ability of other scientists and engineers to do credible and reliable work that may have life-or-death consequences.* Reproducibility is a cornerstone of the scientific method, and **sharing the computer code** ★ used to reach the conclusions of a paper is often the easiest way to ensure that all the details needed to reproduce the results have been provided.”

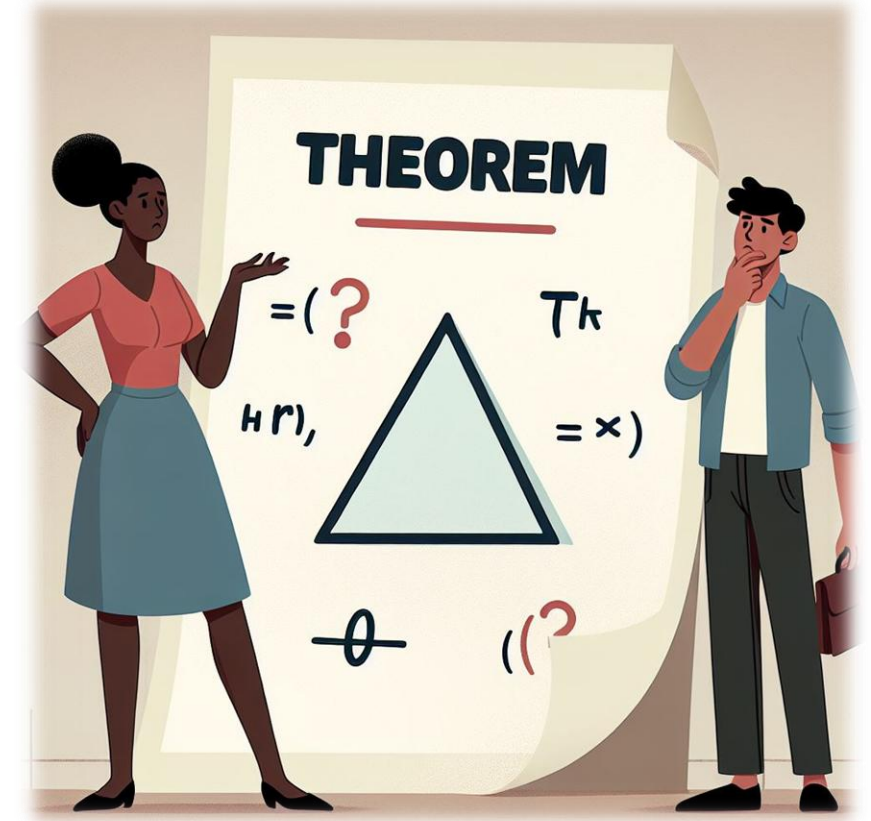
– R. J. LeVeque, “Top Ten Reasons To Not Share Your Code (and why you should anyway),” *SIAM News*, 2013

Publishing Theorems without Proofs?



MathSci.ai

- Thought experiment of LeVeque (2013): there are plenty of reasons not to publish a proof...
 - Too ugly to show anyone else
 - Some details haven't been carefully checked
 - Paper would be too long
 - Referees wouldn't check the proofs anyways
 - And many more!
- Proofs weren't always as formal as today; e.g., calculus initially had tenuous proofs
 - Per Newton (1643-1727) and Leibniz (1646-1716)
 - Formalized by LaGrange (1736-1813), Gauss (1777-1855), Cauchy (1789-1857), and Weierstrauss (1815-1897)
 - See, e.g., Bramlett and Drake, 2013





“Almost everyone who has published a theorem and its proof has found that the process of writing up the proof cleanly enough for publication uncovers subtle issues that must be dealt with, and perhaps even major errors in the original working proof.”

– R. J. LeVeque, “Top Ten Reasons To Not Share Your Code (and why you should anyway),” *SIAM News*, 2013

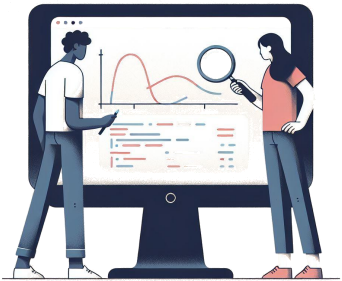
Key Concepts of Experiment Validity



MathSci.ai



- **Replicability** – Experiment can be repeated under identical conditions (compute environment, software, data, parameters)



- **Validation** – Experiment artifacts (code, inputs, outputs) provide suitable evidence for scientific claim
 - Co-authors
 - Referees
 - Readers



- **Reproducibility** – Experiment can be repeated under slightly different conditions (compute environment, software, data, parameters)



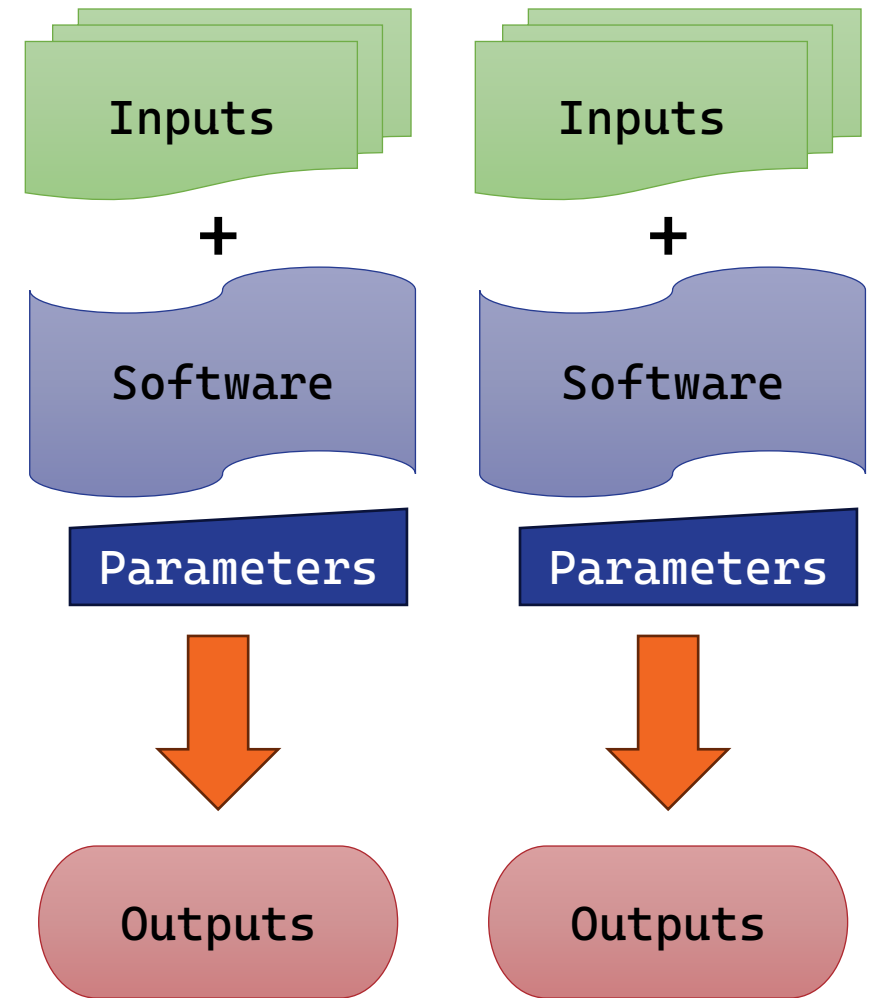
- **Usability** – Proposed methodology (algorithm, software, etc.) can be used for other purposes or extended in various ways

Replication

“To replicate an experiment is to carry out exactly the same task ... with the expectation that the result will be the same. In scientific computing, exact replication would constitute building the same program with the same compiler running on the same hardware and the same operating system.”

– P. Ivie and D. Thain, “Reproducibility in Scientific Computing,” *ACM Computing Surveys*, 2018

*Replication in and of itself
should not be the goal.
May be impossible!*

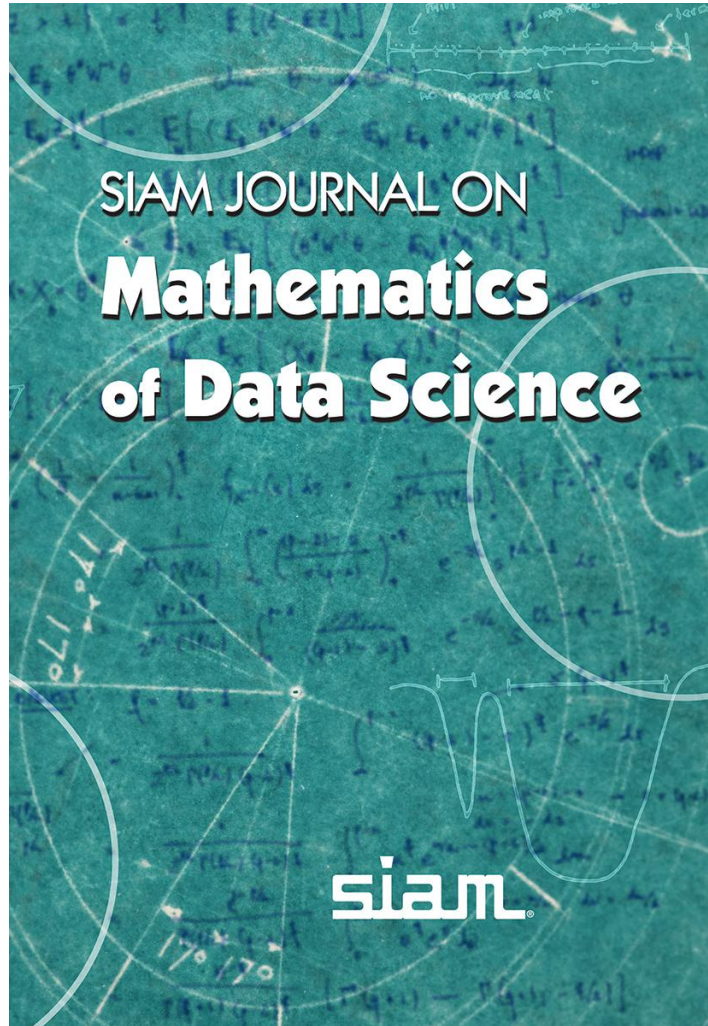


Validation

Evaluating a
computational
experiment to see
if the conclusions
are warranted



Referees Give Partial Validation



“A referee is expected to read the paper with sufficient care to be confident that it is mathematically sound. Nevertheless, it is not necessary to check every detail. The author has final responsibility for the content.”

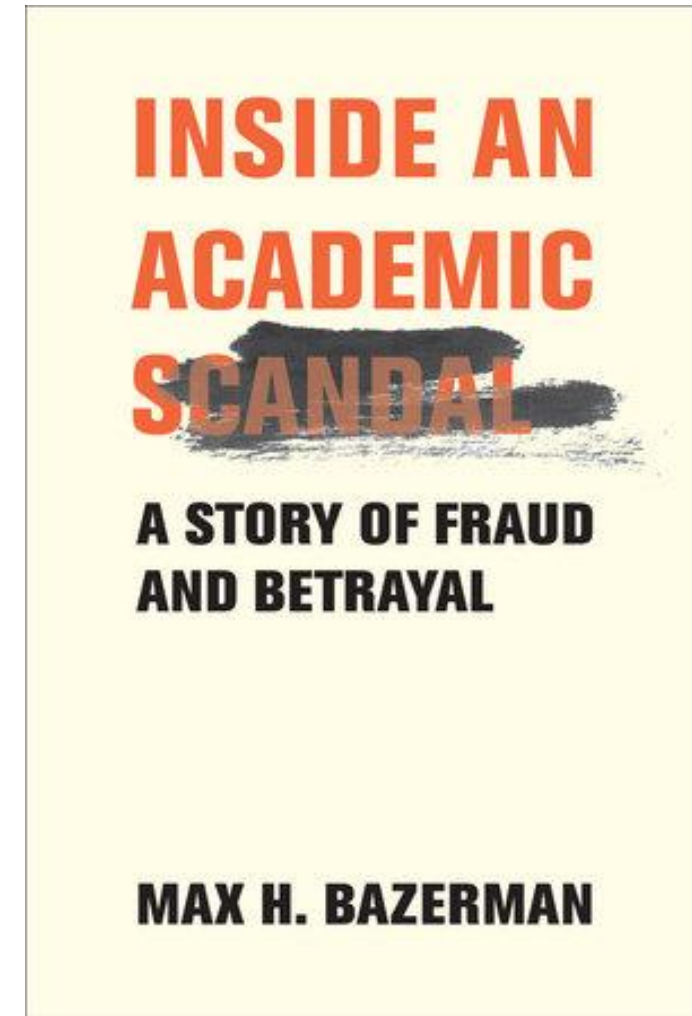
– SIMODS referee instructions

Authors Responsible for Validation

Why Is There So Much Fraud in Academia? - Freakonomics

- Max Bazerman is a Harvard social scientist
- Co-authored papers that have been retracted in a major scientific scandal
- Admits his culpability in the scandal
- Detrimental effect on his field as a whole

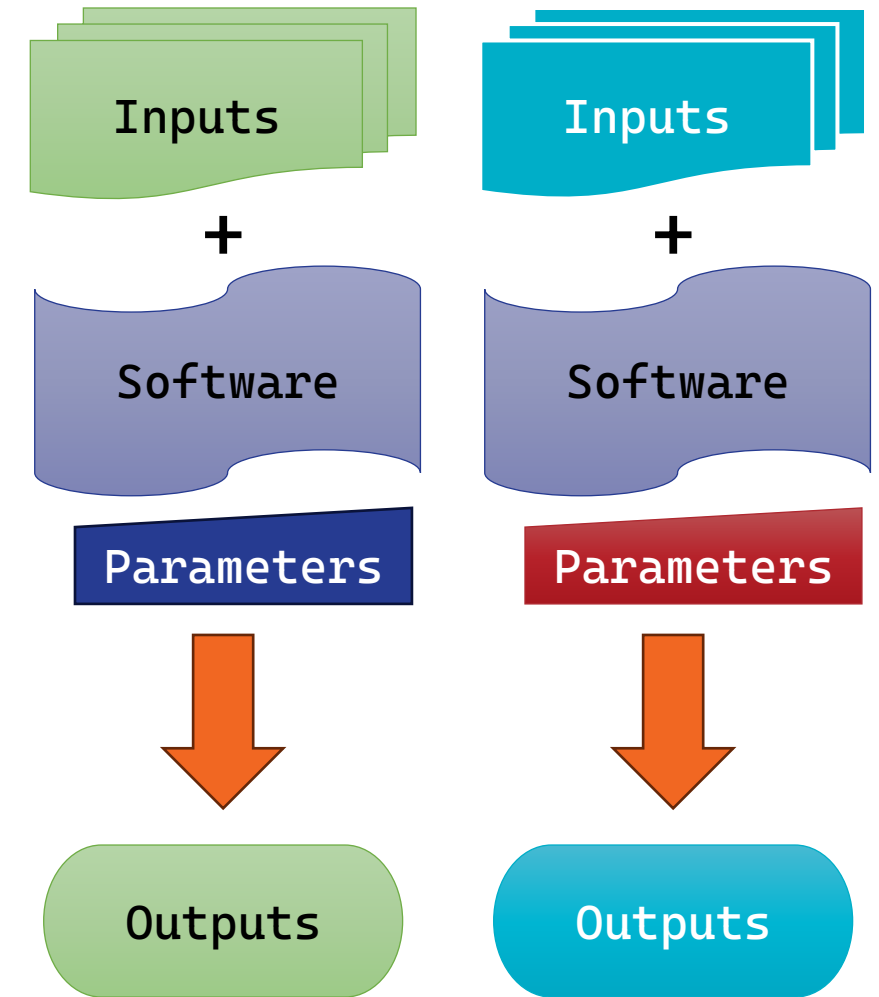
*All authors have a duty to
validate findings in their papers –
this is not just re-running codes!*



Reproduction

“To reproduce an experiment is to carry out tasks that are equivalent in substance to the original, but may differ in ways that are not expected to be significant to the final result. In scientific computing, these differences could range from minor to sweeping. One attempt to reproduce might run the same version of the software on a new version of an operating system, while another attempt to reproduce might involve writing a new piece of software that implements the same algorithm.”

– P. Ivie and D. Thain, “Reproducibility in Scientific Computing,” *ACM Computing Surveys*, 2018



Usability (Impacting Science & Policy)



“The easier it is for a reader to understand the details and implement it themselves, or even borrow code, the more likely it is to be adopted and cited by others.”

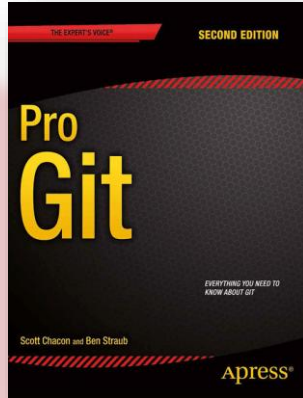
— R. J. LeVeque, “Top Ten Reasons To Not Share Your Code (and why you should anyway),” *SIAM News*, 2013



*But How do We Get to
Replicability,
Validation,
Reproducibility, &
Usability?*



1. Use Version Control (Git)



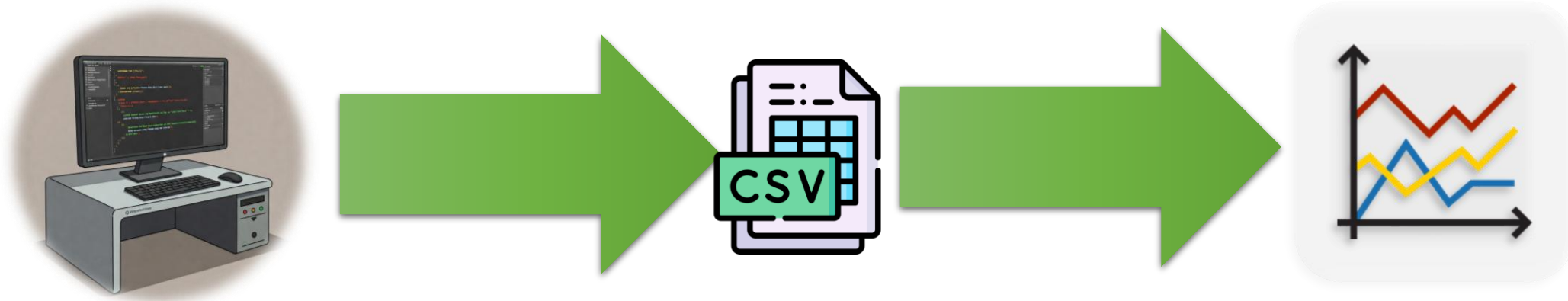
- (Online) Git repositories enable
 - Tagging (hashes)
 - Collaboration
 - Sharing
- Learn the basics using *free* Pro Git book
 - Chapters 1-3
- Recommended
 - GitHub (Overleaf compatible)
 - GitLab
 - BitBucket
 - Not recommended: Google Drive, OneDrive, DropBox

2. Separate Experiments and Figures



- Avoid temptation to turn experiments directly into figures
- Instead, **save data** that can be **translated into figures**
- Keeping results and graphics separate makes it easier to...
 - Re-collate with other metrics (median vs. mean)
 - Make minor changes like axis labels
 - Generate figures with subset of results (e.g., for talk)
- Personally, recommend using PGFPLOTS 😊

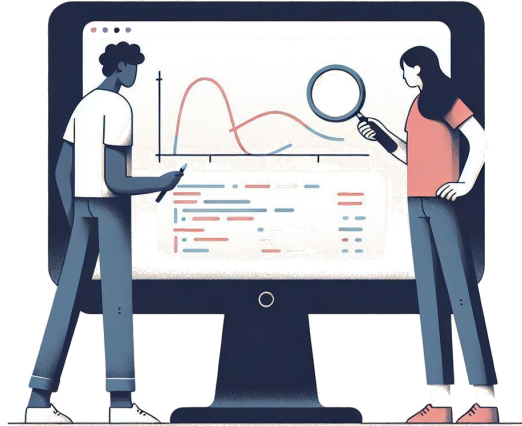
2. Separate Experiments and Figures



- Avoid temptation to turn experiments directly into figures
- Instead, **save data** that can be **translated into figures**
- Keeping results and graphics separate makes it easier to...
 - Re-collate with other metrics (median vs. mean)
 - Make minor changes like axis labels
 - Generate figures with subset of results (e.g., for talk)
- Personally, recommend using PGFPLOTS 😊

3. Create Human-Digestible Logs

*in additional
to data files*



```
---Starting run 3/5---  
Generating random initialization:  
Random Seed in Hex: 41e1119e6ac00000  
Finding CP decomposition:  
Random Seed in Hex: 41da7d077c000000
```

```
CP_ALS (with trace):
```

```
Tensor size: 4821207 x 1774269 x 1805187 (1.544178e+19 total entries)  
Sparse tensor: 1741809018 (1.1e-08%) Nonzeros and 1.544178e+19 (100.00%) Zeros  
Finding CP decomposition with R=25  
Fit change tolerance: 1.000000e-04  
Max iterations: 50
```

- Replication
- Validation
- Additional Info

```
Iter 1: f = 2.797824e-01 f-delta = 2.8e-01 time = 3315.8 seconds  
Iter 2: f = 3.311366e-01 f-delta = 5.1e-02 time = 3316.6 seconds  
Iter 3: f = 3.369195e-01 f-delta = 5.8e-03 time = 3280.3 seconds  
Iter 4: f = 3.384539e-01 f-delta = 1.5e-03 time = 3294.3 seconds  
Iter 5: f = 3.390620e-01 f-delta = 6.1e-04 time = 3273.5 seconds  
Iter 6: f = 3.393420e-01 f-delta = 2.8e-04 time = 3304.0 seconds  
Iter 7: f = 3.394953e-01 f-delta = 1.5e-04 time = 3246.1 seconds  
Iter 8: f = 3.395930e-01 f-delta = 9.8e-05 time = 3264.7 seconds  
Final f = 3.395930e-01  
Total Time (secs): 27287.882
```

~ 8 hours

4. Log Details of Individual Runs

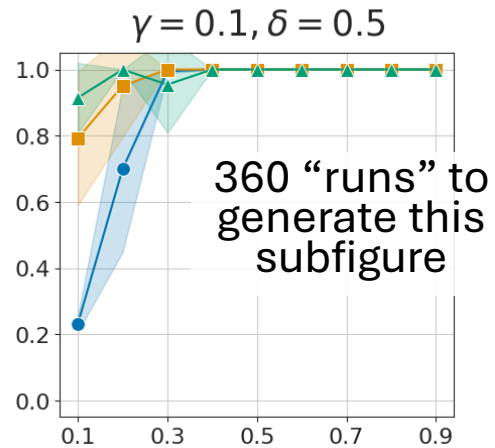


Figure 11B of Li, Luo, & Porter, *SIADS* 2025

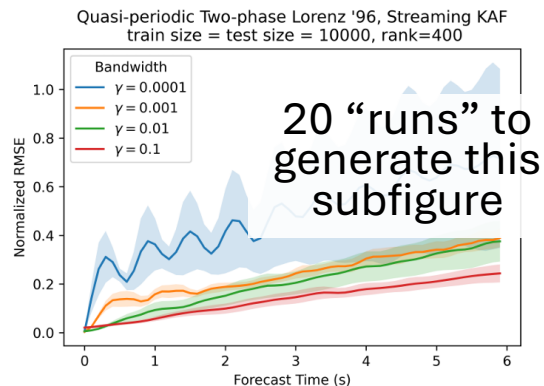


Figure 7b of Giannakis, Henriksen, Tropp, and Ward, *SIADS*, 2023

- Every individual run creates “artifacts”
 - Human-readable files
 - Individual-run data
- Logs include
 - Git hashes on data and codes (and diffs)
 - Date, time, script name, other identifying info
 - **Record of all inputs, including random seed**
- Recommend unique IDs for naming artifacts
 - Script name
 - Date/time
- Enables spot checking
- Ideally, “final” artifacts become part of repository

5. Use Scripts for Automation

- Why automation?
 - Rerun for debugging
 - Rerun after code fixes & optimization
 - Parameter sweeps
 - Incorporating feedback
 - New datasets
 - New models
- Future You Will Thank Current You
 - Easy to regenerate results after changes
 - Sharing with coauthors and others becomes easier
 - Re-use of data, scripts, and codes



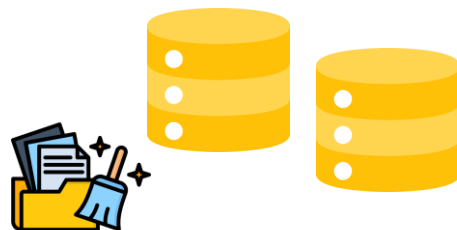
Multitudes of Scripts



- Data preprocessing and generation scripts
- Single run scripts
- Parameter sweep scripts
 - Different methods
 - Different data sets
 - Different input parameters
- Collation scripts
- Figure generation scripts

6. Separate and organize

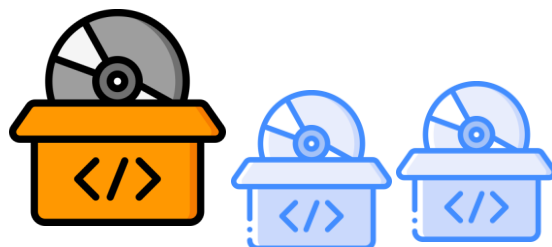
- Data Sets



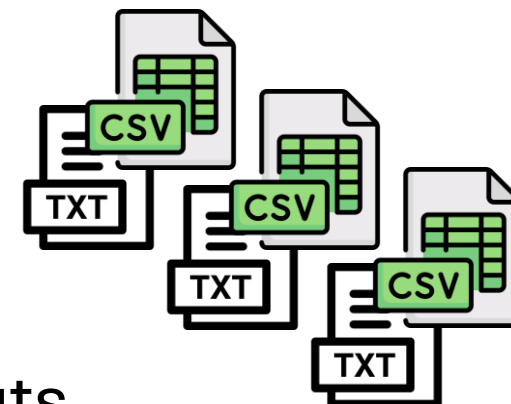
- Scripts



- Codes



- Logs & Outputs



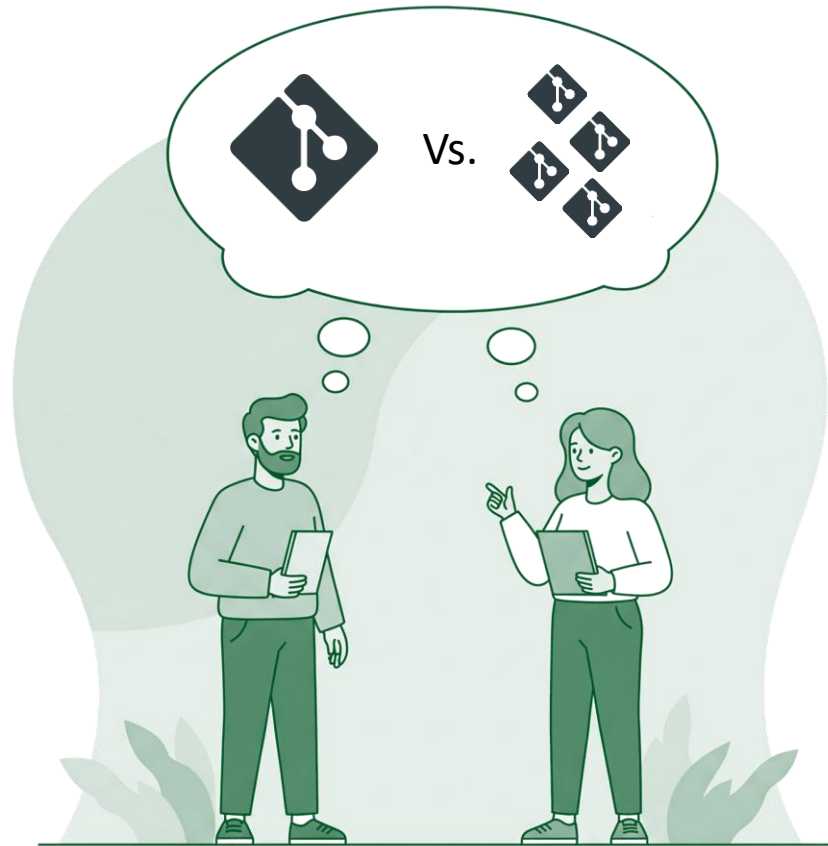
- Paper



- Summary Outputs

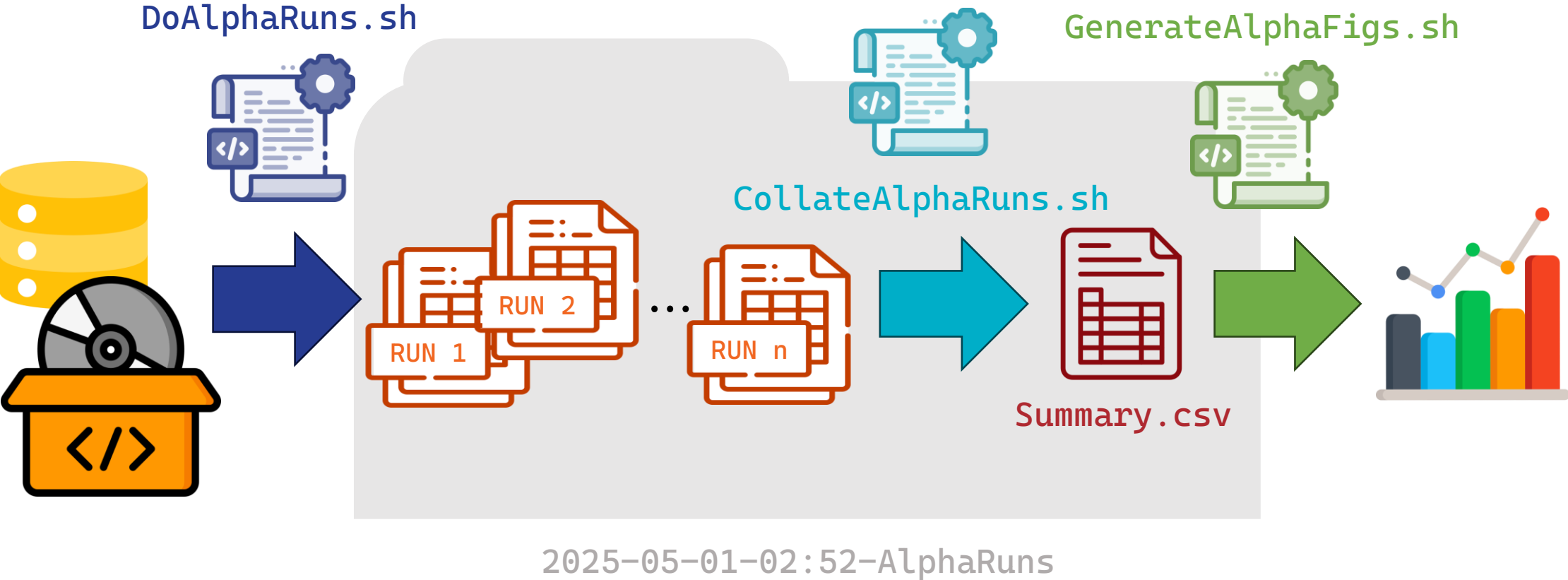


How Many Folders or Git Repositories?



- Are any of the datasets be useful beyond the current paper?
 - Will you use this data again in other papers?
 - Can this data be used by others?
- Can the code be separated?
 - Might it be a useful contribution to an existing software package or library?
- Experiments in their own repo?
 - Organize by folder
 - Maybe the code also resides here?
 - Maybe data also resides here?
- Paper in its own repo
 - Great for linking with Overleaf

Figure Creation Workflow



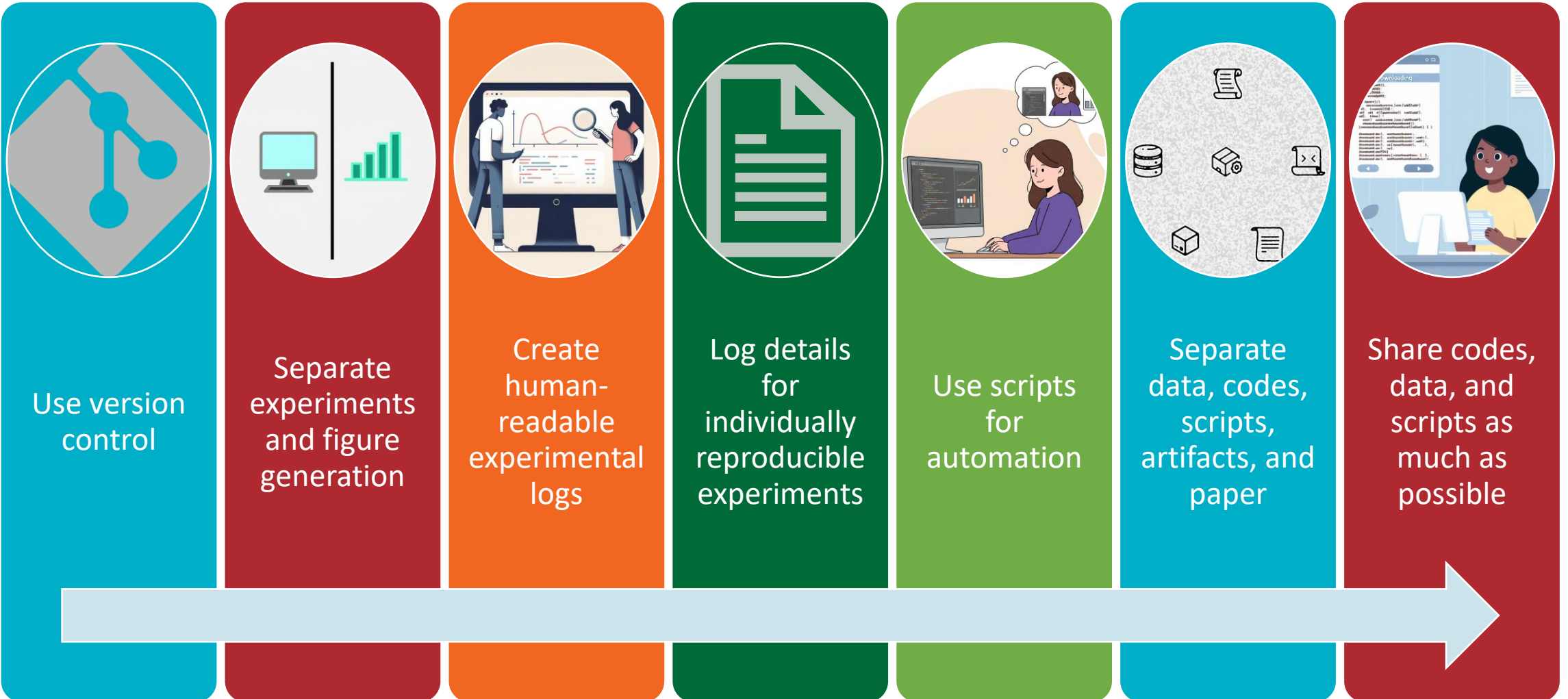
7. Share codes, data, experiments, etc.



“Whatever state it is in, the code is an important part of the scientific record and often contains a wealth of details that do not appear in the paper, no matter how well the authors attempt to describe the method used... The ability to execute the full code and replicate exactly the results in the paper is often of much less interest than the opportunity to examine the most relevant parts of the code.”

– R. J. LeVeque, “Top Ten Reasons To Not Share Your Code (and why you should anyway),” *SIAM News*, 2013

Summary of Recommendations

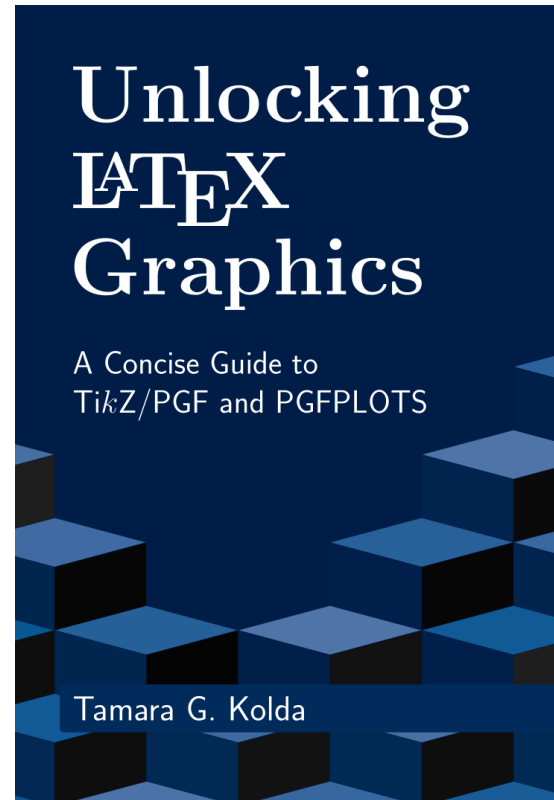


Two Books and a YouTube Channel



mathsci.ai/tensor-textbook

PDF free online and coming soon from Cambridge University Press

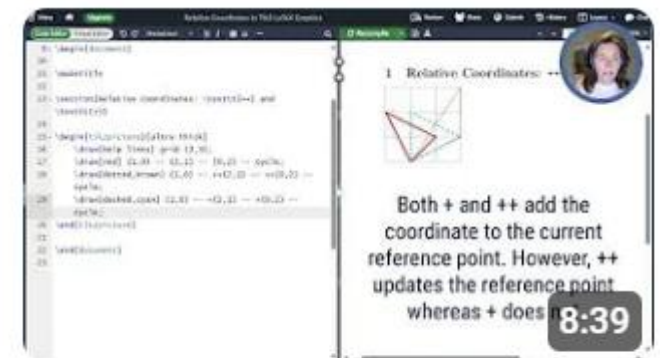


latex-graphics.com

Available at SIAM Booth



<https://www.youtube.com/@UnlockingLaTeXGraphics>



Relative Coordinates (++) and Turns in TikZ LaTeX...

