Improving Reproducibility Through Better Software Practices

David E. Bernholdt
Oak Ridge National Laboratory

Michael A. Heroux
Sandia National Laboratories

Better Scientific Software Tutorial
RF SciDAC 2020 Workshop

License, Citation and Acknowledgements

License and Citation

- **The requested citation the overall tutorial is: David E. Bernholdt, Better Scientific Software tutorial, in RF SciDAC 2020 Workshop, Knoxville, Tennessee. DOI: 10.6084/m9.figshare.11918397**
- Individual modules may be cited as *Speaker, Module Title*, in Better Scientific Software Tutorial…

Acknowledgements

- Additional contributors to this this tutorial include: Anshu Dubey, Mike Heroux, Alicia Klinvex, Jared O'Neal, and Katherine Riley, James M. Willenbring
- This work was supported by the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research (ASCR), and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.
- This work was performed in part at the Argonne National Laboratory, which is managed managed by UChicago Argonne, LLC for the U.S. Department of Energy under Contract No. DE-AC02-06CH11357.
- This work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.
- This work was performed in part at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND NO SAND2017-5474 PE

Terminology

A few of the terms used when talking about this topic…

Reproducibility

Replicability

Reliability

Correctness

Accuracy

Transparency

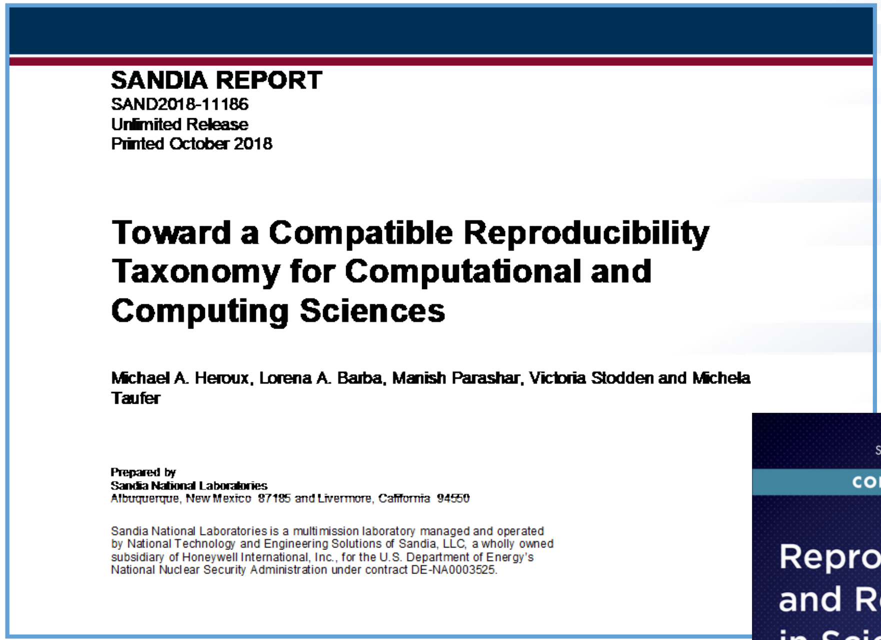
Credibility

They don't mean exactly the same thing…

…but for the purposes of this presentation, the differences don't really matter

Reproducible vs Replicable: A Special Note

- Historically different communities have defined these two differently
- With the increased focus, there has also been an effort to unify the language
- Consensus around the definitions of Claerbout, et al.
  - Others are in the process of switching their terminology to match, i.e., ACM
- **Reproducible**: Another team is able to obtain the same result using the authors' experimental environment
- **Replicable**: Another team is able to obtain consistent results using a different experimental environment



SANDIA REPORT
SAND2018-11186
Unlimited Release
Printed October 2018

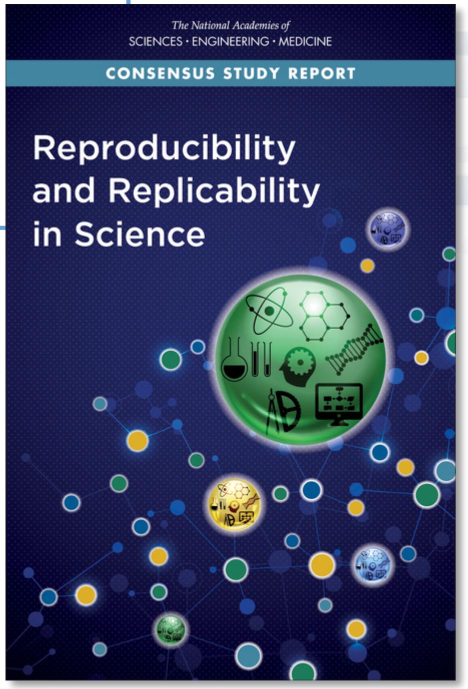
Toward a Compatible Reproducibility Taxonomy for Computational and Computing Sciences

Michael A. Heroux, Lorena A. Barba, Manish Parashar, Victoria Stodden and Michela Taufer

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/SAND2018-11186.pdf


The National Academies of SCIENCES · ENGINEERING · MEDICINE
CONSENSUS STUDY REPORT
Reproducibility and Replicability in Science

https://www.nap.edu/catalog/25303/reproducibility-and-replicability-in-science

IDEAS productivity

ECP EXASCALE COMPUTING PROJECT

Why Reproducibility is Important

SCIENCE

Many Psychology Findings Not as Strong as Claimed

By BENEDICT CAREY AUG. 27, 2015

Staff of the the Reproducibility Project at the Center for Open Science in Charlottesville, Va., from left: Mallory Kidwell, Courtney Soderberg, Johanna Cohoon and Brian Nosek. Dr. Nosek and his team led an attempt to replicate the findings of 100 social science studies. Andrew Shurtleff for The New York Times

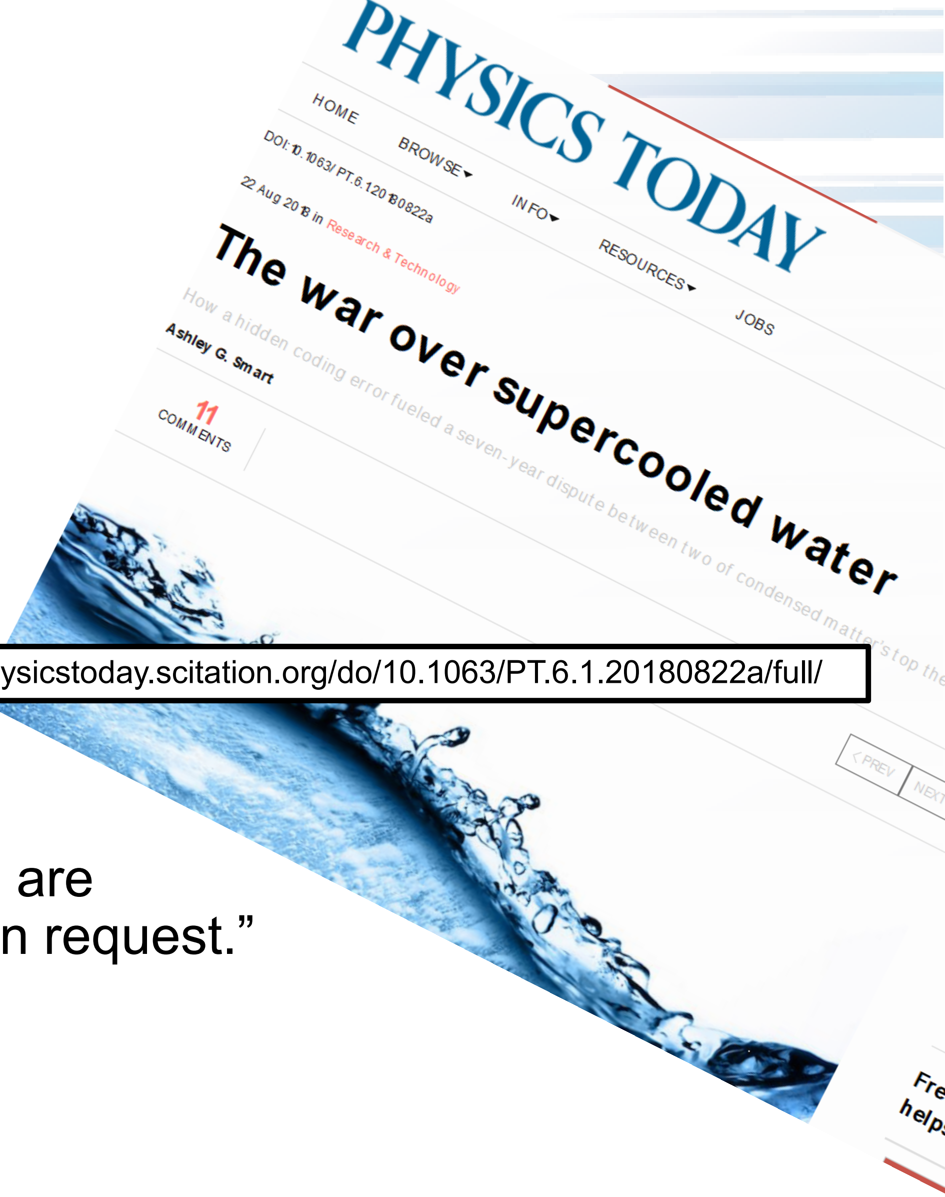
Transparency & Reproducibility

- NY Times highlights “problems”.
- Only one of many cited examples.
- Computational science ***had*** been spared this “spotlight”.

http://www.nytimes.com/2015/08/28/science/many-social-science-findings-not-as-strong-as-claimed-study-says.html

Computational Science Example

- Behavior of pure water just above homogeneous nucleation temperature (~ - 40 C/F).
- Debenedetti/Princeton (2009):
  - 2 possible phases: High or low density.
- Chandler/Berkeley (2011):
  - Only 1 phase: High density.
- No sharing of details across teams until 2016:
  - Chandler in Nature: “LAMMPS codes used in refs 5 and 12 are standard and documented, with scripts freely available upon request.”
  - Debenedetti with colleague Palmer: ”Send us your code.”
  - Received code after requests and appeal to Nature.



PHYSICS TODAY

HOME BROWSE ▾ INFO ▾ RESOURCES ▾ JOBS

DOI: 10.1063/PT.6.1.20180822a

22 Aug 2018 in Research & Technology

The war over supercooled water

How a hidden coding error fueled a seven-year dispute between two of condensed matter's top the

Ashley G. Smart

11 COMMENTS

< PREV NEXT

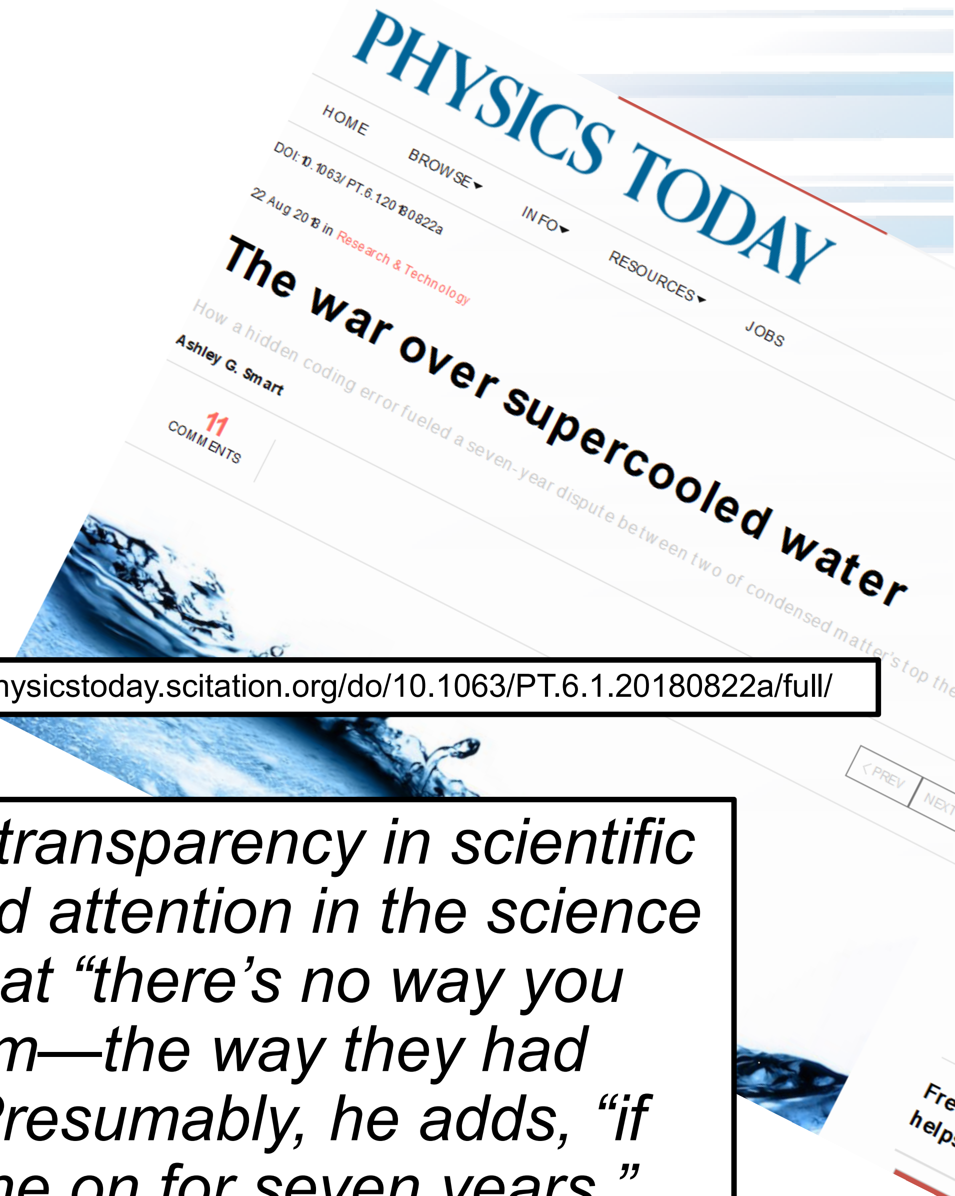
Fre

help

Source: https://physicstoday.scitation.org/do/10.1063/PT.6.1.20180822a/full/

Computational Science Example

- Palmer located bug/feature in Berkeley code.
- Used to speed up LAMMPS execution.
- Replaced with more standard approach.
- Obtained result similar to Debenedetti 2009.
- Resolution took 7 years.


PHYSICS TODAY

HOME BROWSE▾ INFO▾ RESOURCES▾ JOBS

DOI: 10.1063/PT.6.1.20180822a

22 Aug 2018 in Research & Technology

The war over supercooled water

How a hidden coding error fueled a seven-year dispute between two of condensed matter's top the

Ashley G. Smart

11 COMMENTS

< PREV NEXT

Fre
help

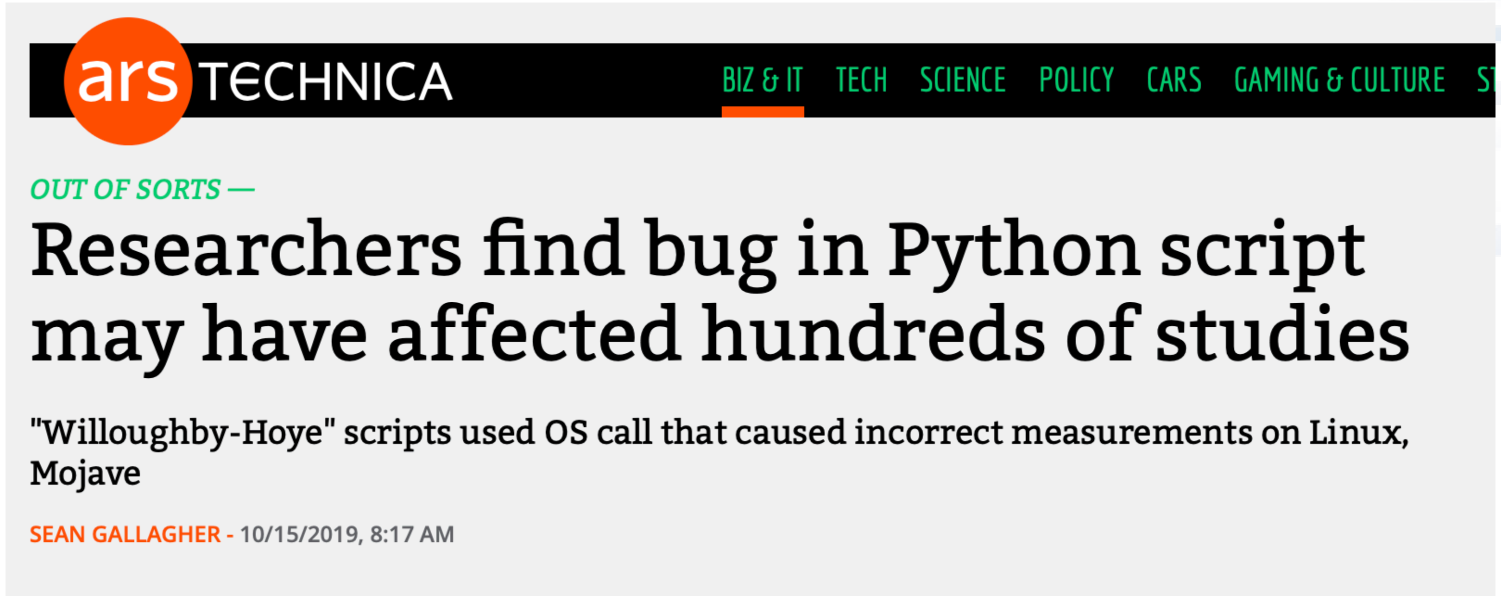
Source: https://physicstoday.scitation.org/do/10.1063/PT.6.1.20180822a/full/

For Palmer, the ordeal exemplifies the importance of transparency in scientific research, an issue that has recently drawn heightened attention in the science community. "One of the real travesties," he says, is that "there's no way you could have reproduced [the Berkeley team's] algorithm—the way they had implemented their code—from reading their paper." Presumably, he adds, "if this had been disclosed, this saga might not have gone on for seven years."

IDEAS productivity ECP EXASCALE COMPUTING PROJECT

Most Recent Example

- Scripts' use of Python's glob module
- Generated different file lists in Linux and Mac Mojave
- Casts doubt on results in 150 papers.
- *Would a unit test have caught this?*



OUT OF SORTS —

Researchers find bug in Python script may have affected hundreds of studies

"Willoughby-Hoye" scripts used OS call that caused incorrect measurements on Linux, Mojave

SEAN GALLAGHER - 10/15/2019, 8:17 AM

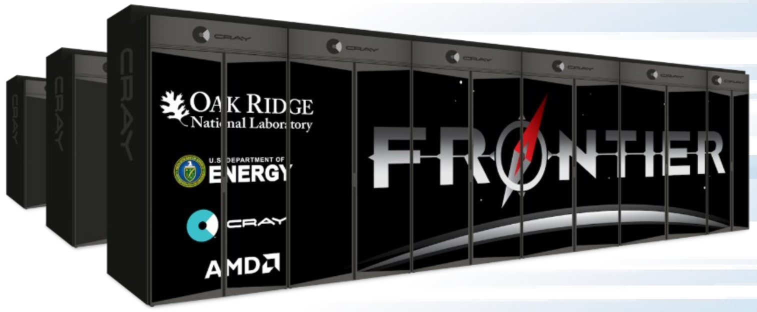
https://arstechnica.com/information-technology/2019/10/chemists-discover-cross-platform-python-scripts-not-so-cross-platform/

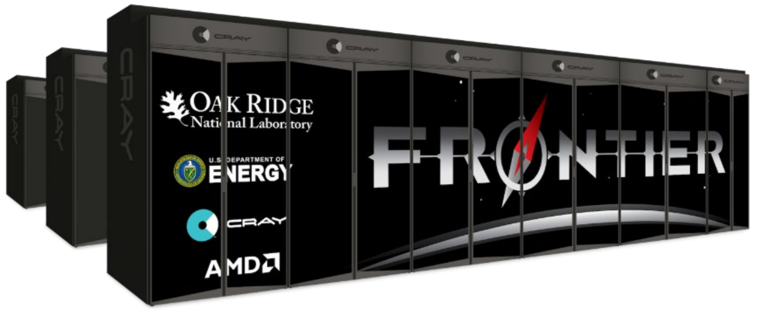
Science through computing is,
at best,
as credible as the software that produces it!

(Additional) Incentives for Paying Attention to Reproducibility

Reproducibility and Supercomputing

- Supercomputer cycles are scarce resources
- No one wants to spend their precious allocation running simulations two or three times to be confident of the results
  - Though this ends up happening more than most people admit
  - And it could still be wrong!
- But lots of people need to have confidence in your results
  - You
  - Your project lead or boss
  - Your sponsor
  - Your reviewers or referees
  - Your readers
- Need to think about how to build credibility *without* repeating runs





IDEAS productivity EXASCALE COMPUTING PROJECT

ACM TOMS Reproducible Computational Results (RCR)

- Submission: Optional RCR option.
- Standard reviewer assignment: Nothing changes.
- RCR reviewer assignment:
  - Concurrent with standard reviews.
  - As early as possible in review process.
  - Known to and works with authors during the RCR process.
- RCR process:
  - Multi-faceted approach, Bottom line: Trust the reviewer.
- Publication:
  - Reproducible Computational Results Designation.
  - The RCR referee acknowledged.
  - Review report appears with published manuscript.

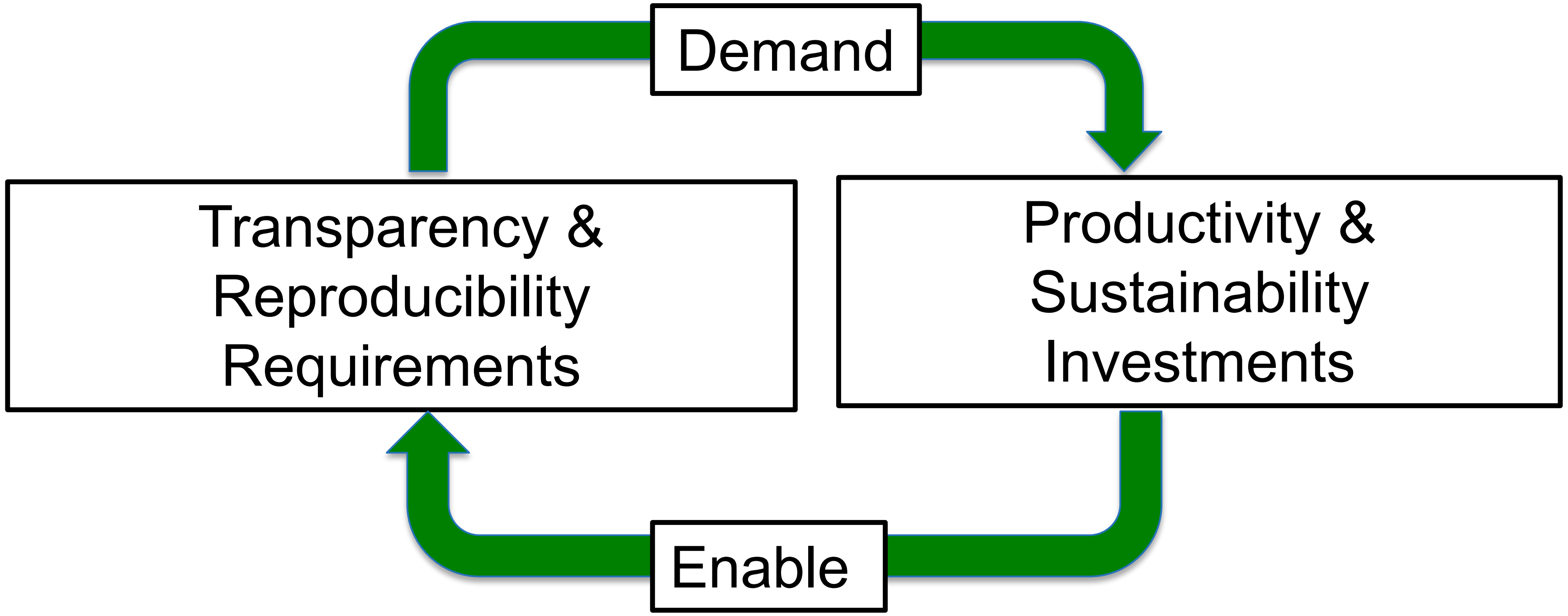
SC20 Transparency and Reproducibility Initiative

- Two appendices:
  - Artifact description (AD).
    - Blue print for setting up your computational experiment.
    - Makes it easier to rerun computations in future.
    - AD appendix is mandatory for paper submissions (since SC19).
    - Largely auto-generated from submission information.
  - Artifact Evaluation (AE).
    - Targets “boutique” environments.
    - Improves trustworthiness when re-running hard, impossible.
    - Remains optional
- Details:
  - https://sc20.supercomputing.org/submit/transparency-reproducibility-initiative/

Coming to Your World Soon: Reproducibility Requirements

- These conferences have artifact evaluation appendices:
  - CGO, PPoPP, PACT, RTSS and SC.
  - http://fursin.net/reproducibility.html
- ACM ~~Replicated~~ Reproducible Computational Results (RCR).
  - ACM TOMS, TOMACS.
  - http://toms.acm.org/replicated-computational-results.cfm
- ACM Badging.
  - https://www.acm.org/publications/policies/artifact-review-badging
- NISO Committee on Reproducibility and Badging.
  - https://www.niso.org/niso-io/2019/01/new-niso-project-badging-scheme-reproducibility-computational-and-computing
  - Publishers: ACM, IEEE, figshare, STM, Reed Elsevier, Springer Nature

Incentives Demand Investments, Enabled by Investments


Demand
Transparency & Reproducibility Requirements
Productivity & Sustainability Investments
Enable

Common statement: “I would love to do a better job on my software, but I need to:

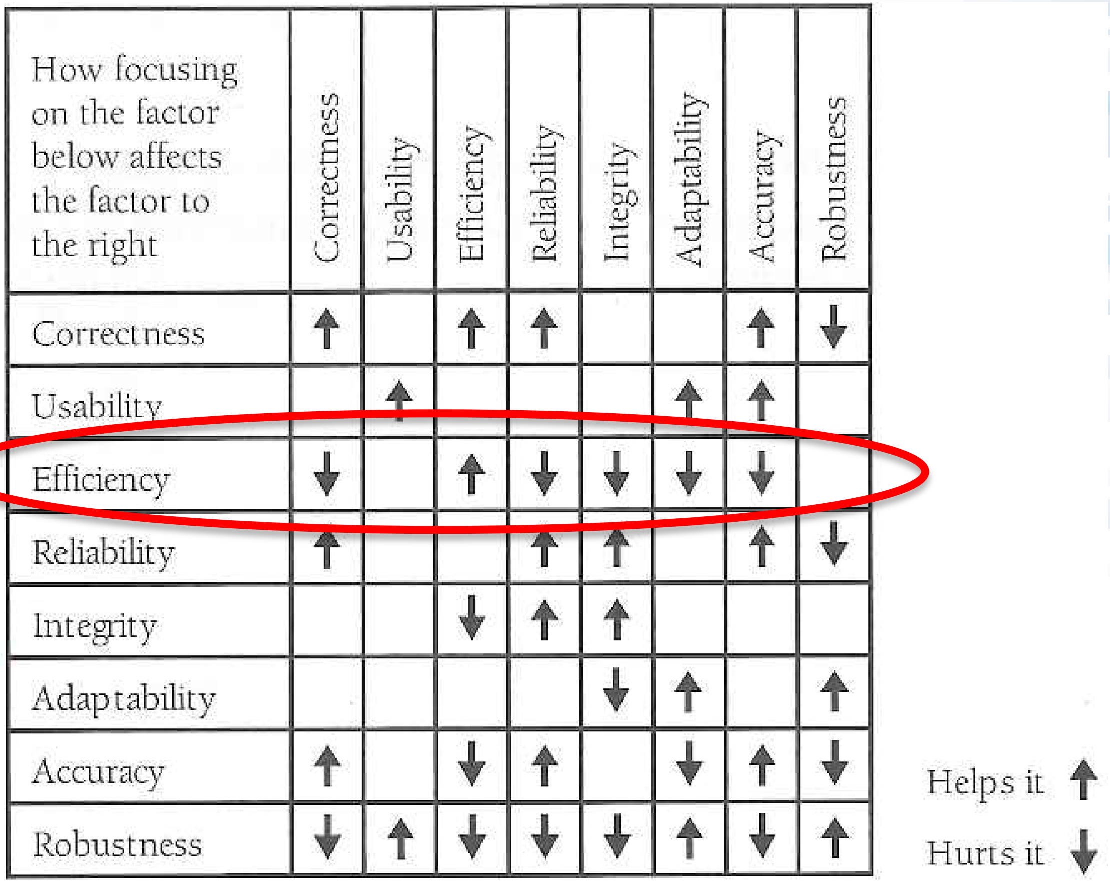
- Get this paper submitted.
- Complete this project task.
- Do something my employer values more.

Goal: Change incentives to include value of better software, better science.

How to Improve Reproducibility

Strategies that Don't Improve Reproducibility

- Trusting the code of inexperienced coders
- Letting deadlines rule
  - I don't have enough time to {test, document, …} the {code, results, …} because I have to finish this {paper, talk, proposal, …}.
- Code performance optimization
- Utilizing dynamic parallelism (standard on modern processors)
- Different code for different hardware
- Ignoring the possibility of silent data corruption



How focusing on the factor below affects the factor to the right	Correctness	Usability	Efficiency	Reliability	Integrity	Adaptability	Accuracy	Robustness
Correctness	↑		↑	↑			↑	↓
Usability		↑				↑	↑	
Efficiency	↓		↑	↓	↓	↓	↓	
Reliability	↑			↑	↑		↑	↓
Integrity			↓	↑	↑			
Adaptability					↓	↑		↑
Accuracy	↑		↓	↑		↓	↑	↓
Robustness	↓	↑	↓	↓	↓	↑	↓	↑

Helps it ↑
Hurts it ↓

from Code Complete by Steve McConnell

Strategies for Improving Reproducibility *<u>During</u>* <u>Development</u> (1/3)

- **Solid versioning practices are fundamental reproducibility**
- Version control of code, documentation, and other artifacts
  - Frequent commits (perhaps to a separate development branch)
- Provide versioning information in key output(s)
  - Version numbers (i.e., semantic versioning) are useful, but when do you increment them?
  - Automatic identifiers (i.e., git commit hash) are less ambiguous, but may not be as meaningful
  - Is the code you're building modified from the version in the repository?
  - *Not often done in practice*
- Maintaining documentation (and other artifacts) in sync with code
  - You'll forget
  - Or you won't have (make) time to go back to it

Strategies for Improving Reproducibility *During* Development (2/3)

- **Build in quality from the start**
- Define and follow coding standards
  - Not just code style
  - Expectations for kinds and extent of documentation, types and rigor of tests
- Develop tests as you code
  - Write tests while the code is fresh in your mind
  - Test Driven Development (TDD) means write tests before code, then code to pass the tests
- Require increasingly rigorous testing as the code becomes more "public"
  - Testing has costs, need to balance level of risk against cost of executing tests
  - Also think about frequency of tests at different levels of cost (c.f. continuous integration)
- Practice peer code review
  - Retrospective if you have a lot of existing unreviewed code
  - Per commit – should meet standards, *and* be understood and judged correct by reviewer
  - Pair experienced reviewers with less experienced coders to help ensure quality

Strategies for Improving Reproducibility *During Development* (3/3)

- **Be judicious in pursuit of performance**
- Start with straightforward, portable implementations
  - They are more easily understood by future developers
  - “Tricky” or more complicated code is more likely to have undetected bugs
- Focus optimization efforts on actual bottlenecks
  - If your algorithm is N^3, accelerating an N^2 operation probably won’t make much difference
  - If you’re creating hardware-specific versions, make sure you’re testing on each platform
- Treat optimization like a refactoring exercise
  - Make sure it is covered by tests
  - Examine both correctness and performance gains
- Include options to eliminate non-determinism
  - Usually has a performance cost, but may improve testability/debugability

Strategies for Improving Reproducibility *After* Development

- **Testing, testing, and more testing!**
- Add more tests
  - Be creative
  - Think about corner cases
  - Think about misuse (unintentional or intentional)
  - Think about synthetic tests with synthetic data
  - Think about low-cost tests that can be "always on" (even if they're not so stringent)
  - Can you detect silent data corruption?
- Test your tests!
  - Make sure tests fail when they're supposed to!
- Thoroughly verify the code
  - Does the code do what you intended it to do?
  - On all relevant platforms

Digression – “Physics” (or Math)-Based Testing Strategies

- **Use what you know (or can construct) about the model you’re studying to test its implementation**
- Synthetic operators with known properties
  - Spectrum (huge diagonals)
  - Rank (by construction)
- Invariance principles
  - Translational, rotational, etc.
  - Physical symmetries
  - Mathematical symmetries
- Conservation rules
  - Fluxes, energy, mass, etc.
- …

Digression – Design by Contract Programming

- **Building testing into your routines**
  - **To complement, *not replace*, other testing**
- The interface to a routine can be thought of as a contract between caller and the routine

– What does the contract expect?	**preconditions**
– What does the contract guarantee?	**postconditions**
– What does the contract maintain?	**invariants**

- Given valid inputs (preconditions satisfied) a routine should guarantee valid outputs (postconditions satisfied, invariants maintained)
  - If the preconditions are not satisfied, the routine should return an error
- Making the contract explicit facilitates correct use of routines
  - Especially when routine is reused in another context
  - Especially by those not intimately familiar with them

Strategies for Improving Reproducibility ***During* Experiments** (1/3)

- **What are your going to do, why, and how?**
- Plan your experiments thoroughly
  - Know what you need (in the code, as inputs, as outputs to capture/analyze, etc.)
  - Know how you're going to process or analyze the results
  - Know what to expect (in results, performance/cost, etc.)
  - How will you convince yourself that your results are trustworthy?
- Perform pilot/test runs to build confidence in correctness, performance, scaling
  - Often useful to pursue an incremental/layered strategy
- Ensure that you have the resources to store and/or analyze the outputs
  - What can you afford to archive?
  - What will you need to process and delete?
  - What will you need to process during execution or stream?

Strategies for Improving Reproducibility *During* Experiments (2/3)

- **Can you reproduce the code used for each and every experiment?**
  - **Three years later?**
- Use only well-defined versions of code (i.e., official "releases", tags, etc.)
  - Master or development branches are often moving targets
  - Capture the exact version of the code used for each experiment
    - Is the code you're building exactly what's in the version control repo?
  - Don't change versions during a related series of experiments (unless you have to)
  - If you have to change versions, know exactly what changed
    - Capture the exact version of the code used for each experiment
- Use only versions of code that have been thoroughly verified
- Consider capturing version information of key libraries, compilers, and other dependencies used to build code
  - *Not often done, in practice*

Strategies for Improving Reproducibility *During* Experiments (3/3)

- **Be thorough in capturing provenance**
  - **Agents (codes), entities (inputs, outputs, etc.), activities (the transformation)**
- Capture code version
- Capture all inputs/configuration information for each experiment
- Use multiple systems to ensure that you can correctly associate inputs, outputs, and code versions
  - Systematic directory and file naming conventions
  - Separate written notes (paper notebook, electronic notebook)
  - Scripts to orchestrate experiments (versioned and captured)
  - Version control (if data is not too large)
- Capture important outputs (as feasible)

Strategies for Improving Reproducibility *After* Experiments

- **Continue provenance capture through data analysis/reduction process**
  - **Agents (codes), entities (inputs, outputs, etc.), activities (the transformation)**
- Script as much of your analysis/reduction as possible
  - Prefer scriptable tools over those requiring human interaction
  - Keep them under version control
- Document your process thoroughly
  - Separately from scripts, etc.
  - E.g., paper or electronic notebook
  - Especially where human interaction is required
- Capture key intermediates in the reduction process
  - The more you capture, the more you have to verify (and find problems) later
- Capture the data (in machine-readable form) used to produce graphs and tables
  - Expected by basic data management plans
  - And an increasing number of publishers

Summary

- The credibility of your science derives from the credibility of your code (and process)
- Science stakeholders are ratcheting up expectations for reproducibility
- There are strategies to improve reproducibility in all phases of the scientific process
  - During development
  - After development
  - During experiments
  - After experiments
- They amount to better software development practices
  - The same kinds of practices advocated for reasons of productivity, sustainability, maintainability, etc.

Other resources

Editorial: ACM TOMS Replicated Computational Results Initiative. Michael A. Heroux. 2015. *ACM Trans. Math. Softw.* 41, 3, Article 13 (June 2015), 5 pages. DOI: http://dx.doi.org/10.1145/2743015

Enhancing Reproducibility for Computational Methods. Victoria Stodden, Marcia McNutt, David H. Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A. Heroux, John P.A. Ioannidis, Michela Taufer Science (09 Dec 2016), pp. 1240-1241

Agenda

Time	Module	Topic	Speaker
1:00pm-1:05pm	00	Introduction	David E. Bernholdt, ORNL
1:05pm-1:30pm	01	Overview of Best Practices in HPC Software Development	David E. Bernholdt, ORNL
1:30pm-2:00pm	02	Agile Methodologies and Useful GitHub Tools	David E. Bernholdt, ORNL
2:00pm-2:30pm	03	Improving Reproducibility through Better Software Practices	David E. Bernholdt, ORNL
2:30pm-2:45pm		Q&A	All
2:45pm-3:30pm		*Break*	
3:30pm-4:15pm	04	Software Design and Testing	David E. Bernholdt, ORNL
4:14pm-4:45pm	05	Continuous Integration	David E. Bernholdt, ORNL
4:45pm-5:00pm		Q&A	All