# Improving Reproducibility Through Better Software Practices

David E. Bernholdt, David Rogers
Oak Ridge National Laboratory

Michael A. Heroux
Sandia National Laboratories

Software Productivity Track, ATPESC 2020

See slide 2 for license details

# License, Citation and Acknowledgements

## License and Citation


- This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

- **The requested citation the overall tutorial is: David E. Bernholdt, Anshu Dubey, Mark C. Miller, Katherine M. Riley, and James M. Willenbring, Software Productivity Track, in Argonne Training Program for Extreme Scale Computing (ATPESC), August 2020, online. DOI: 10.6084/m9.figshare.12719834**
- Individual modules may be cited as *Speaker, Module Title*, in Software Productivity Track…

## Acknowledgements


- Additional contributors include: Patricia Grubel, Rinku Gupta, Mike Heroux, Alicia Klinvex, Jared O'Neal, David Rogers, Deborah Stevens
- 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 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 the Lawrence Livermore National Laboratory, which is managed by Lawrence Livermore National Security, LLC for the U.S. Department of Energy under Contract No. DE-AC52-07NA27344.
- 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.

# Terminology

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

Reproducibility

Replicability

Reliability

Correctness

Accuracy

Transparency

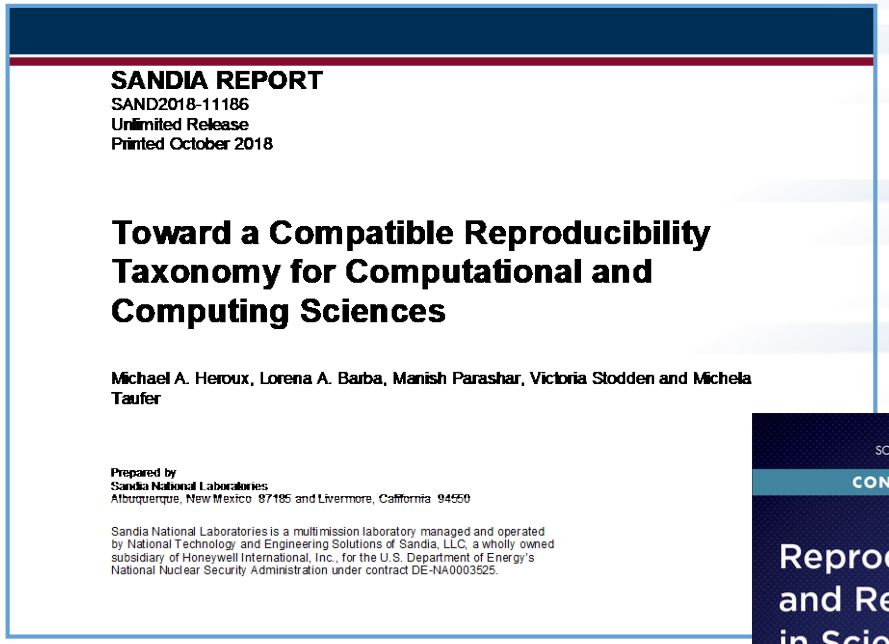
Credibility

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

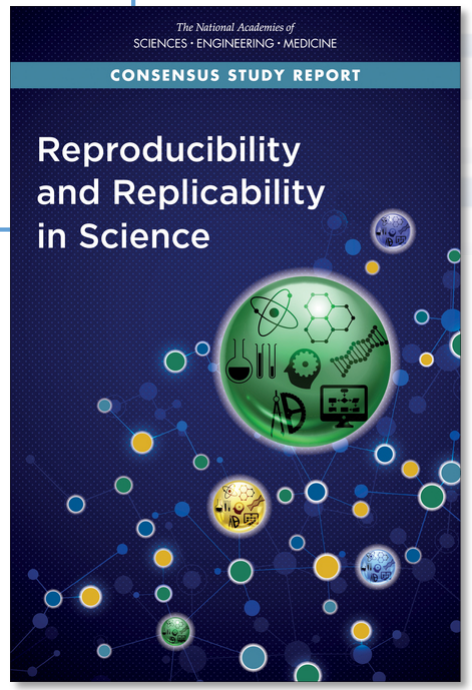
**Toward a Compatible Reproducibility Taxonomy for Computational and Computing Sciences**

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




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

# Why Reproducibility is Important

# Transparency & Reproducibility

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

- 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

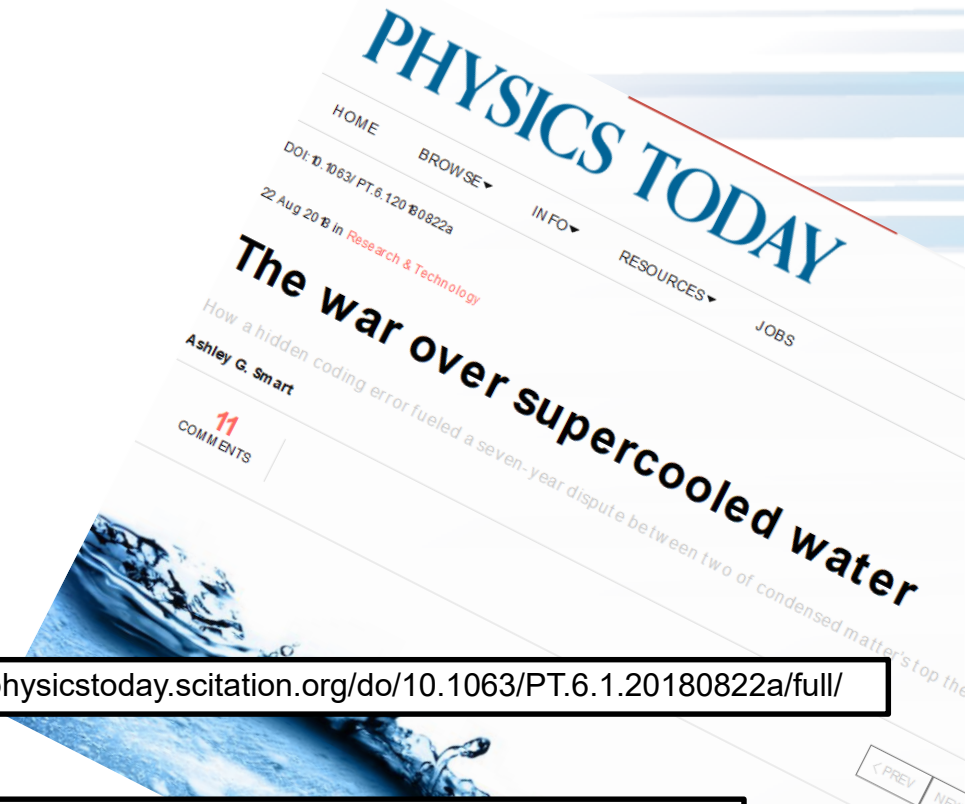
# Computational Science Example

- 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.

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.




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."*

# A Recent Example: Python Behaves Differently on Different Platforms

- Scripts for analyzing experimental nuclear magnetic resonance (NMR) data
- Scripts use Python's glob module (listing filenames matching a pattern)
- Module ordered results differently in Linux and Mac Mojave
- Results depended on the order in which files were processed
- Casts doubt on results in 150 papers.
- *Would a unit test have caught this?*

ars TECHNICA

BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE ST

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

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

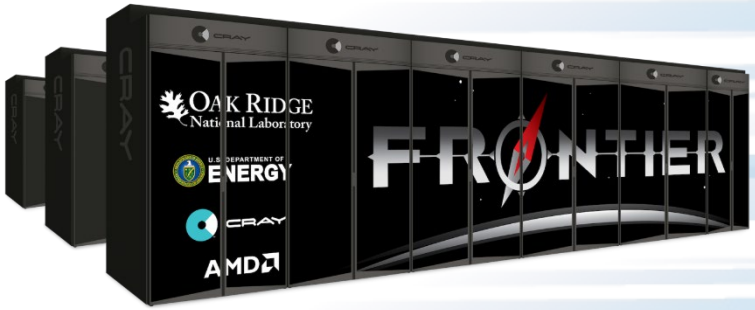
Goal: Change incentives to include valuing of better software, better science

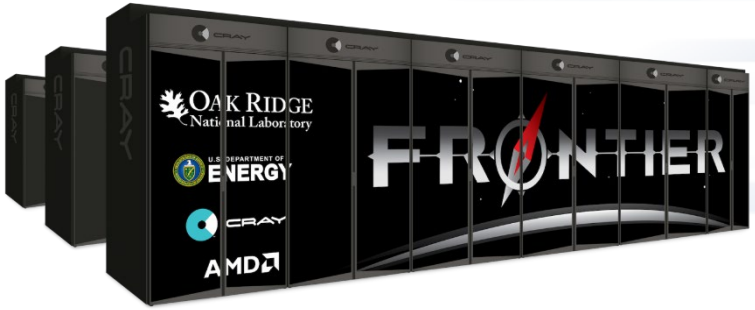


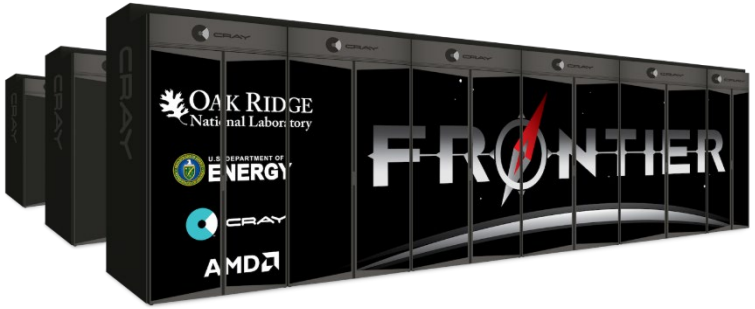
IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

# 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

ECP EXASCALE COMPUTING PROJECT

# Setting Expectations for Your Data

## Data Management Plans

- Most research sponsors require data management plans as part of proposals
- Example: NSF policy on Dissemination and Sharing of Research Results
  - Promptly publish with appropriate authorship
  - Share data, samples, physical collections, and supporting materials with others, within a reasonable time frame
  - Share software and inventions
  - Investigators can keep their legal rights over their intellectual property, but they still have to make their results, data, and collections available to others
  - Policies will be implemented via
    - Proposal review
    - Award negotiations and conditions
    - Support/incentives

## FAIR Data Principles

- Address data producers and publishers to promote maximum use of research data
- Findability
  - Data and supplementary materials have sufficiently rich metadata and a unique and persistent identifier.
- Accessibility
  - Metadata and data are understandable to humans and machines. Data is deposited in a trusted repository.
- Interoperability
  - Metadata use a formal, accessible, shared, and broadly applicable language for knowledge representation.
- Reusability
  - Data and collections have a clear usage licenses and provide accurate information on provenance.

IDEAS productivity

ECP 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
- Incentives:
  - Journal: raises the credibility/quality/rigor of papers it publishes
  - Authors: badging indicates additional credibility/quality/rigor of paper
  - Reviewer: companion publication

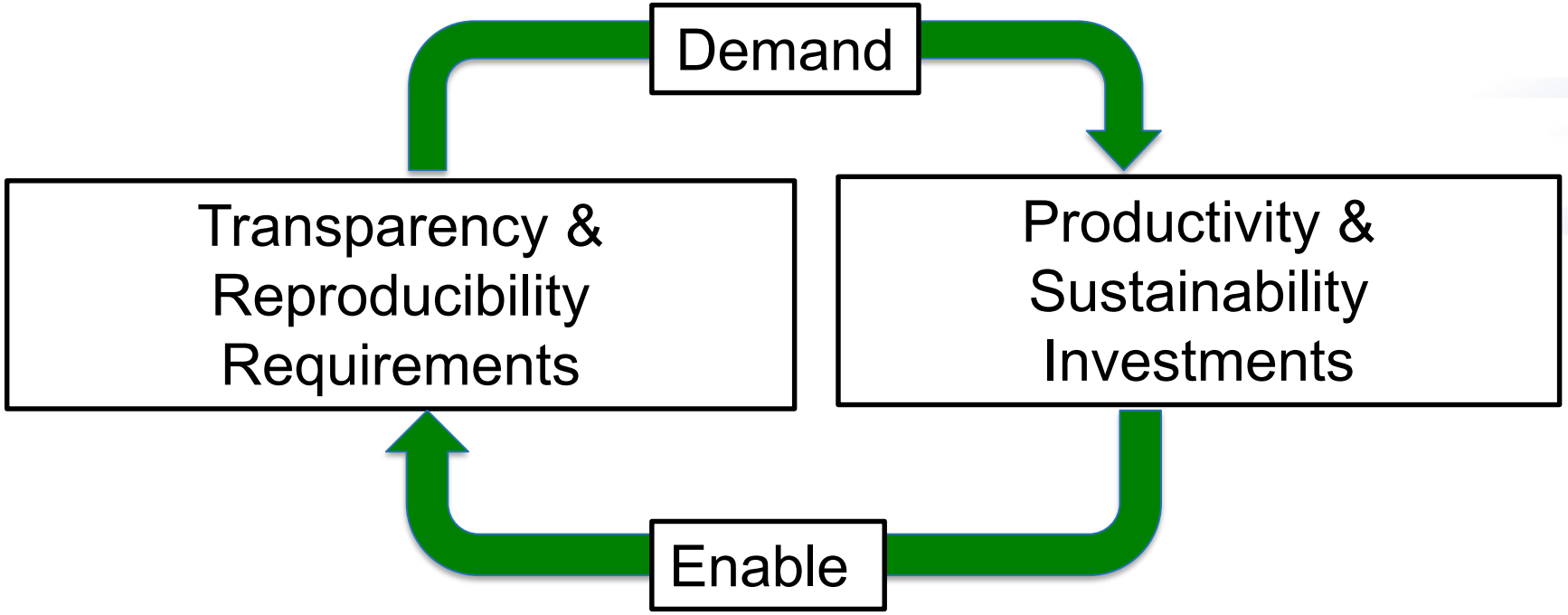
# SC20 Transparency and Reproducibility Initiative

- 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/

# Increasing Attention on Reproducibility

- More publication venues are adding reproducibility **recognition** or **requirements**
- ACM ~~Replicated~~ Reproducible Computational Results (RCR)
  - ACM TOMS, TOMACS
  - http://toms.acm.org/replicated-computational-results.cfm
- ACM Badging
  - Functional, reusable, available, replicated, reproduced
  - https://www.acm.org/publications/policies/artifact-review-badging
- These conferences have artifact evaluation appendices:
  - CGO, PPoPP, PACT, RTSS and SC
  - http://fursin.net/reproducibility.html
- 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

# Creating a Virtuous Cycle




Reproducibility is, ultimately, based on good <u>software development practices</u> and good <u>experimental practices</u>

# How to Improve Reproducibility

# Strategies for Improving Reproducibility *During* Development (1/3)

- **Solid versioning practices are fundamental to 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 creating and executing tests
  - Also think about frequency of tests at different levels of cost (c.f. continuous integration)
- Practice peer code review
  - Per commit – should meet standards, *and* be understood and judged correct by reviewer
  - Pair experienced reviewers with less experienced coders to help ensure quality
  - Retrospective if you have a lot of existing unreviewed code

# Strategies for Improving Reproducibility *After* Development

- **Testing, testing, and more testing!**
- Add "regression tests"
  - If you fix a bug, add a test to make sure that bug doesn't creep back in
- Add more tests
  - Be creative
  - Think about common cases, then 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 (compilers, hardware, etc.)
- Test regularly
  - To identify and document where changes to the underlying platform change code behavior/results



IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

# 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.
- …



IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

# 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 *callee* (the routine)
  - What does the routine expect on input? **preconditions**
  - What does the routine guarantee at completion? **postconditions**
  - What does the routine leave unchanged? **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
  - Emphasize low-costs tests that can be always-on
  - May need to be able to switch enforcement of expensive tests on/off (but try not to!)
- 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 you going to do, why, and how?**
- Plan your experiments thoroughly
  - If you're in a team, designate *one* person to coordinate the experimental campaign
  - 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
- Continue to use regular testing to identify changes due to the underlying platform
  - E.g., compiler release introduces a new optimization that changes numerical results
- 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*



IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

# 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)
    - Lab notebooks aren't just for people who literally work in a laboratory!
  - 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

# Tools May Help with Reproducibility

Just a small sampling…

- Containers to capture the software
- Resources for finding, understanding, and debugging floating point math problems: http://fpanalysistools.org/
- Cloud platforms to publish and reproduce research code and data
  - E.g., https://CodeOcean.com
- DOIs and hosting of data, code, documents, etc.
  - E.g., https://zenodo.org/, https://FigShare.com

**Make sure to test and understand your tools thoroughly *before* using them for something important!**

# 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.



IDEAS productivity
ECP EXASCALE COMPUTING PROJECT

# Other resources

- The FAIR Guiding Principles for Scientific Data Management and Stewardship. Mark D. Wilkinson, et al. https://doi.org/10.1038/sdata.2016.18
- 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
- Simple experiments in reproducibility and technical trust by Mike Heroux and students (work in progress), https://betterscientificsoftware.github.io/Trust-Tools/