How Safe & Persistent Is Your Research?

Challenges in Managing Trustworthy Large-scale Digital Science

Ben Evans

AGU2017 Union U41A-03  Ben.Evans@anu.edu.au
How Safe and Persistent is your Digital Science?

- If someone gave you their digital science output, what is needed to convince you that it was correct?
- If this required a complex system that you couldn’t run yourself, what evidence do you need to trust it?
- What standards should apply?
- What would make that a trustworthy process?
- Do you know what the assumptions and dependencies are?
- Do you even really know the details of the underlying system platform or service?
- How much could you rely on their results persisting over a longer time?
- Would you use them as automated inputs to your own workflows that you want others to trust?
Increasingly research has moved from experimental “hand-crafted” steps into codified Digital Science that requires:

- A precise way for “documenting” all the steps
- Adopts self-describing data, common vocabularies and units
- Adopts self-describing programs (e.g., Python notebooks)
- Increasingly uses linked data and other semantic informatics systems
- Workflow systems that track processes

*So, are we done?*
What do we mean?

Definitions of the Association for Computing Machinery (ACM):
https://www.acm.org/publications/policies/artifact-review-badging (based on International Vocab of Metrology - BIPM)

**Repeatability (Same team, same experimental setup)**
The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials. For computational experiments, this means that a researcher can reliably repeat her own computation.

**Replicability (Different team, same experimental setup)**
The measurement can be obtained with stated precision by a different team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using the author’s own artifacts.

**Reproducibility (Different team, different experimental setup)**
The measurement can be obtained with stated precision by a different team, a different measuring system, in a different location on multiple trials. For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.

*We are not done with needing to define terms – Available, Reusable, Contestable/Visible*
Repeatability (Same team, same experimental setup)
Timescale: < 1 month

Scenario 1: Code crashes / doesn’t complete

• Impact:
  • If operationally sensitive (e.g., NWP, data services) then clearly not “fit for purpose”
  • If trying to process data, then output will be substandard and the results could be misleading
  • Scientific discovery - low robustness means hard to manage, and means Time-to-Science is very high
  • Loss/misuse of scarce computational resources / grant / funds / precious time
• General Considerations:
  • Scientific “corner cases” need to be addressed
  • Is there potentially silent data corruption?
• Large-scale Considerations:
  • Large simulations are more likely for the model to encounter a system fault (system resilience, MTBF)
  • Complex systems have dependencies on fragile components
  • Checkpoint sensitivities (exact layout and preconditions)
  • System heuristics (for pre-exascale, debugging by jitter to better understand random crashes)
  • How many times should you run a computation to trust the results?
Repeatability (Same team, same experimental setup) continued
Timescale: < 1 month

Scenario 2: Performance variability

• Impact
  • Major consideration if part of your work is to compare against alternative software

• General Considerations
  • Time share systems (i.e., queueing times): scheduler processing time/optimising for best packing
  • Competing for shared resources (shared CPU, shared memory, poor systems management)
  • Lack of systems resources means internal program contention (e.g., insufficient memory bandwidth)
  • Caching affects – has both positive and negative, depending on whether they can be managed

• Large-scale Considerations
  • Topology (e.g., from local resource placement through to general uniformity across the system)
  • Operating system jitter
  • Size of algorithmic messages, efficient parallel algorithm, asynchronous processes
  • I/O does not scale with compute - particularly metadata
Repeatability (Same team, same experimental setup) continued
Timescale: < 1 month

Scenario 3: Numerical Variability in Results

- Impact
  - Basic assurance of the computational method

- General Issues
  - What is the precision required?
  - Software improvements
  - Bitwise comparison -> helps be assured that software is debugged
  - Variability in platform due to OS, compiler, libraries upgrades

- Large-scale issues
  - Underlying computer arithmetic is not reproducible (e.g., Gustafson recently proposed new changes to IEEE)
  - Pre-exascale systems looking at cost of “bitwise reproducible(?)” algorithms/software/math libraries
    - Sensitivity to domain decomposition chosen (bitwise helpful in debugging)
    - Sensitivity to the number of cores
    - Special math libraries do exist - ~10x computational cost. Significant investment to check your own parallel code
  - Multiple execution - rerun the same calculation and confirm that the results are “equivalent”
  - Ensemble calculations – testing numerical stability of the underlying model
Persistence: Repeatability over a medium timeframe?

What can help?
• Version control of user level software is maintained with the modules system.
• Well managed, professional software systems with Continuous Integration
• Orchestration/Containers: e.g., Puppet, Docker

Scientific codes
• Full version control on the software
• Large software systems – also integrated version control on the whole systems, with confidence tests

Platform and workflow dependencies
• Platform management, and robust processes for deployment, management and assurance
• Access to reference data, but now via services (traded persistency of URL with way of tracking version updates)
• Complex workflow systems now use micro-services

Issues
• How many systems maintain a log of the "version" of the system (or any changes) that a user can gain access?
• If you had full access, could you work out what the state was of the underlying system?
• Can you gain access to the confidence measures of the platform?
• Is this beyond what we can reasonably do, or are there levels of confidence in our digital science that we accept?
### 3-5 years
- Current research will focus on refreshed code base with new algorithms and higher resolution
- Data Changes/refresh
- Complete refresh for all dependencies: e.g.,
  - New physical technology refresh: e.g., lower arithmetic precision b/w or performance consideration ???
    - e.g., Many core, GPU, TPU/DPU, FPGA?, Quantum
    - On-premise vs cloud vs ???
  - Software paradigm changes for the hardware or platform

### 10+ years
- Whole software codebase will be refreshed
- Dependencies change again -> new paradigm, new efficiencies, new costing model

**So, what is a reasonable expectation?**
- *Would you really redo old work?*
- *Can we just preserve some assets,*
- *Expect the latest work to be backwardly-compatible model?*
If it's not feasible/practical to repeat or completely verify the work at the large scale, then...

What do we need to gain trust?

For Large computational digital science models:

- Unit tests, pretesting against scenarios and follow procedures to ensure ok?
- Share major benchmark cases? (e.g., CMIP)
- How do we share this with others to build confidence/trustworthiness?
- How do we share our level of confidence in the digital science results?
- How do we record and share the level of variability/precision?
- How do we put in place auditable systems to stop cherry picking results?

In production systems its best practice to separate Dev and Ops. We don’t really do that in digital science, but many talk about need a separate person to verify (Replicate)

- HPC systems maintainers track underlying underpinning uniformity across the system and performance
- Underlying systems dependencies (does anyone get details of the “versions” for their big and unique systems?)
**FAIR data principles**

- Access to data is increasingly via intelligent open data services, so to what extent are we able to extend the information infrastructure to address data services or protocols?
  - One issue is that we expect to understand and select versions of datasets.
  - Do we also need the version of a server or service?
  - What about the software or major models?

**CoreTrustSeal**

- What information infrastructure should we put in place around these?
  - Core Trustworthy Data Repositories, Extended Guidance (v1.0, Oct 2017)
  - This is a good, broad set of goals but also need to be extended to other parts of the digital science ecosystem.
  - Are they useful to address the questions we have around models?
  - Are these sufficient?

**Quality Assurance Measures**

- To what extent do these need to be shared and citable for the service to be trusted?
Emerging issues

• To what extent should be providing information about uncertainty? And in what form?

• How do we address next generation of workflow systems (the micro-services)
  We can’t put PIDs on the chain. Does a certified Blockchain play a role?

• To what extent do we expect Provenance records to be available?
  Is PROV ready to handle the range of questions and return in a usable form?
  Can these be available to provide information about the underlying platforms?

• What needs to be put in place to Trust Deep Learning/ Machine Learning and AI?
• **Repeatability** is difficult – short term and long term ....

• And yet we aim to be Open and Transparent to help *Replicability and Reproducibility*

• We need trustworthy processes and trustworthy dependencies that support confidence and can handle scrutiny

• We need to continue to build the information infrastructure and informatics systems to support Transparency

• Transparency is supported by FAIR principles, and can be adopted more broadly than “data”
  
  But this requires more discipline, higher costs and other ‘hidden’ overheads

• VREs, software, services developers and compute platforms managers still need improvement

• We need a good scientific rationale of what is trying to be done and the level of precision required

```
“You must not fool yourself — and you are the easiest person to fool”
Richard Feynman.
```