HOW TO KEEP COMPUTATIONAL EXPERIMENTS ALIVE

Samuel Grayson (Sandia, UIUC), Reed Milewicz (Sandia)

WHAT IS ALIVE?

- (required) Run the experiment
- (would be nice) Results reproducible
  - Within statistical degree

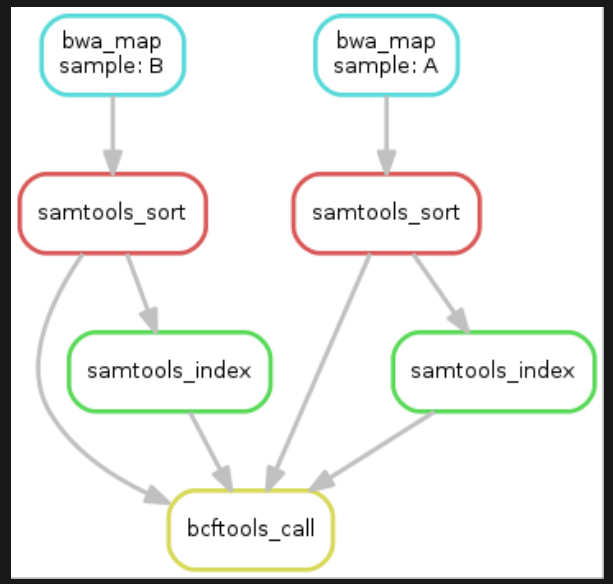
WHY LONG-TERM SURVIVABILITY?

- Parts last for decades
- Be engineers not archeologists
- Provenance: how did we get here
- Rerun: updated knowledge or methods
- Extend: new questions

METHODS

OPEN SOURCE SAMPLES

- Snakemake and Nextflow workflows
- Workflows nodes are container + cmd, edges are files



- One minute to half an hour

RESULTS

HOW MANY WORKFLOWS? HOW MANY STILL ALIVE?

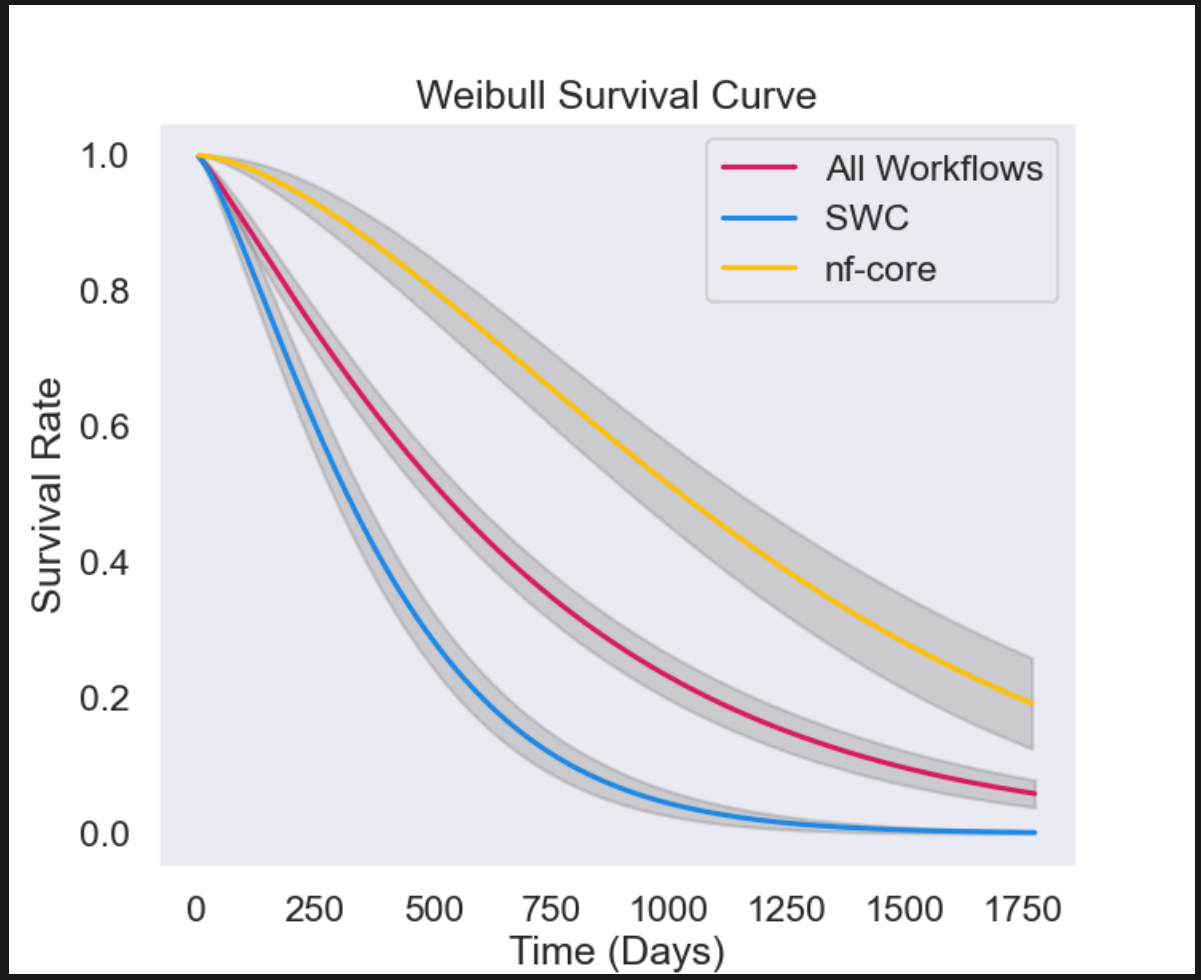
Quantity	All	Snakemake	Nextflow
# workflows	101	53	48
% of workflows with ≥ 1 non-crashing release	53%	23%	88%
# releases	584	333	251
% of releases with no crash	28%	11%	51%

WHAT ARE COMMON ERROR CAUSES

- Timeout
- Network resource changed
- Missing software dependency
- Missing data/config input
- Singularity error
- Conda environment unsolvable
- Unclassified

RESULTS

Kind of crash	All	Snakemake	Nextflow
Missing data/config input	32.2%	43.8%	16.7%
Conda environment unsolvable	10.8%	18.9%	0.0%
Unclassified reason	7.9%	12.0%	2.4%
Timeout reached	7.0%	5.7%	8.8%
Singularity error	6.0%	6.6%	5.2%
Other (workflow script)	5.7%	1.5%	11.2%
Other (workflow task)	1.2%	0.0%	2.8%
Network resource changed	0.7%	0.0%	1.6%
Missing software dependency	0.5%	0.9%	0.0%
No crash	28.1%	10.5%	51.4%
Total	100%	100%	100%


Weibull Survival Curve
All Workflows
SWC
nf-core
Survival Rate
1.0
0.8
0.6
0.4
0.2
0.0
0
250
500
750
1000
1250
1500
1750
Time (Days)

DISCUSSION

MISSING EXAMPLE DATA IS PROMINENT

- Experimenters should include example data (downloaded or generated)
- This would allow these workflows to be tested by idiots
- But, default data can be confusing

CONDA ENVIRONMENT UNSOLVABLE

- Packages can get yoinked
  - Spec Repo
- Conda has no lockfile; only specfile
  - If you have to use Conda, Conda-lock
- Impossible to debug
  - StackOverflow
  - If you have to use Conda, use Mamba frontend

CONTAINER INFRASTRUCTURE IS DIFFICULT (IMAGES)

- Distribute container images or distribute container build files?
- Expensive to store (DockerHub)
  - Is this container safe to delete? Append-only
  - `apt install x y` and `apt install x z` no reuse
  - Flask baseimage, Numpy baseimage, what if I need both?
- Link rot -> registries are ephemeral

CONTAINER INFRASTRUCTURE IS DIFFICULT (BUILD FILES)

- Lose reproducibility benefits if each user has to build
  - `apt update` -> `apt install x` is not reproducible
- Need a reproducible package manager to do the build
- Workable solution here!

SOURCE-LEVEL PACKAGE MANAGERS

- Guix, Nix, Spack
- Source is less expensive to store (delta compress, multiarch)
  - Reuse for `install x==1.0` and `install x==2.0`
  - Guix links to Software Heritage
- Binary buildcache available (more secure)
- Build containers reproducibly
  - Software environment is a DAG not a chain!

CI FOR CONFIGURATION ERRORS

- Too slow to run in CI
- Scale down fidelity and size
- Rerun periodically, not just when code changes
- Store small artifacts
  - File read/write set and hash (ptrace)

CI FOR DETECT SEMANTIC ERRORS

- Parameterize test fidelity and size
- Schedule scaled down test frequently, full fidelity test infrequently
- Write provenance data (DAG of inputs, intermediates, outputs)
  - Collect statistical summary
  - Interoperable with other tools (W3C PROV)
  - Prov diff

FAILURE PREDICTION

- For each experiment, estimate probability of failure
- Probability of failure determines frequency of testing
- Factors:
  - History of that individual workflow
  - History of workflows in that language and ecosystem
  - If A fails and B has many same components, B might fail

SOLUTIONS

- Default example data (for idiots)
- Conda ⇒ Mamba + Conda-lock
- Source-level package managers build containers reproducibly
- Multi-scale CI
- Failure prediction (research)