Practical and pleasurable reproducibility

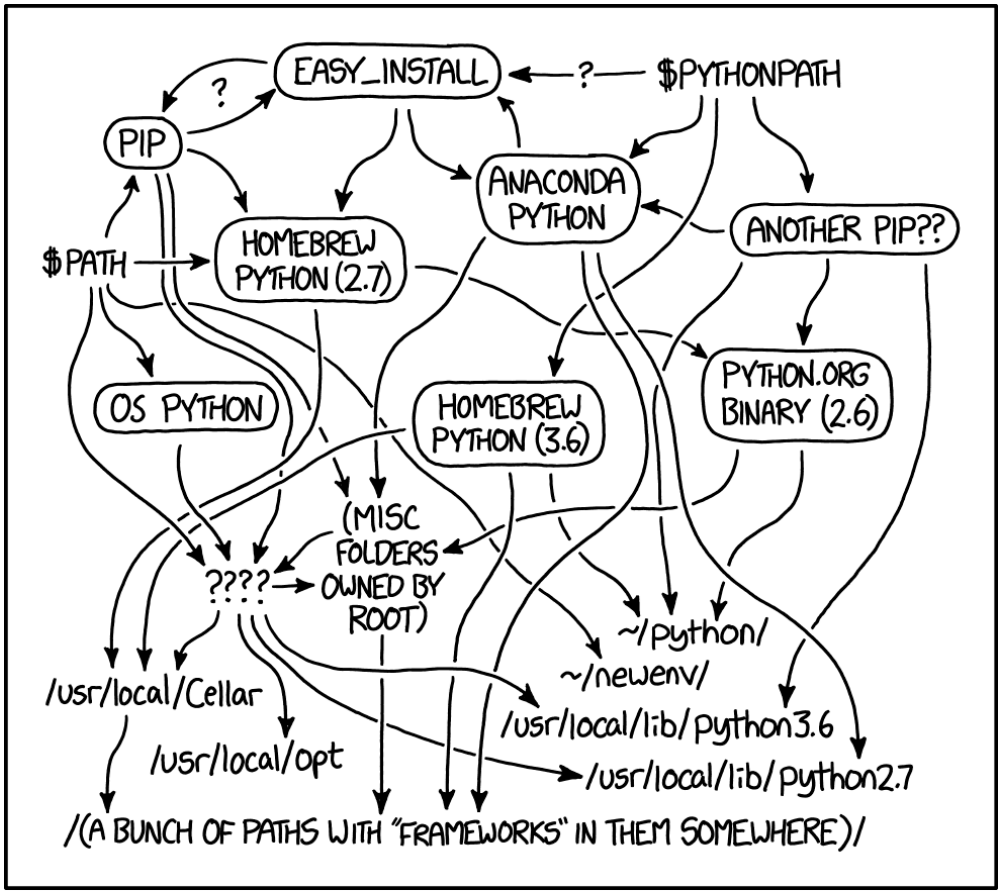
Using *Guix*, the *Common Workflow Language* and *ravanan*

Arun Isaac (Mott lab, University College London)

April 21, 2026

The Problem

xkcd 1987: Python Environment



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

A reproducibility problem

- Computers are the most **deterministic machines** ever, yet ...
- Why can't we reproduce our own computational results?
- Daily frustrations
  - The *it worked on my machine* problem
  - The *it worked last month* problem
  - The *parameter tweak* problem
  - The *new collaborator* problem
  - compiling a package from source *never works*!

Why current methods fall short

- **Software installed by cluster admins** Unknowable provenance, can be updated under your feet, not portable to collaborators
- **conda/pip** Easy installation and portability, no reproducibility
- **Docker/Singularity** Reproducible (if you're sharing the built image) but an uninspectable black box
- **Lab notebook with commands** Very accessible, but hard to keep consistent

Software installed by cluster support

- Unknown and unknowable **provenance**
- **Where** did the software come from?
- **Which version** is it?
- **Which libraries** were it linked against?

conda, pip and cran packages

- `install.packages(…)`, `pip install …`, `conda install …`
- Package managers that are used to install packages easily on different machines
- Make installation easy (**portability**), but don't help with **reproducibility**

- Application bundles (commonly called containers)
- Software that is fully compiled and **distributed along with all required dependencies**
- Essentially **shrink wrapping** your computer and giving it to someone else
- Uninspectable, and unknowable provenance

Lab notebook with commands

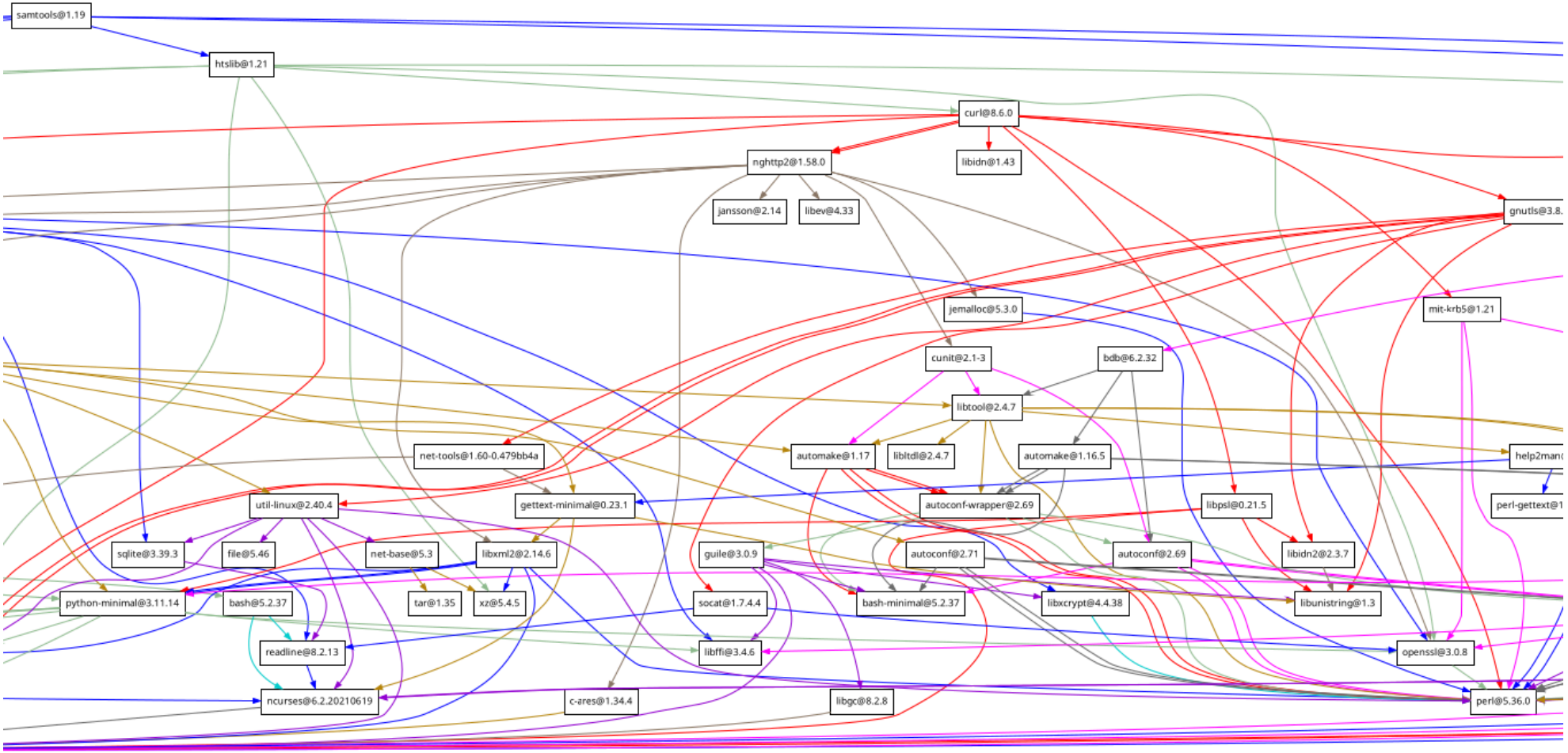
- The good old README file
- **Simple**—no need to learn any new tools
- Very hard to keep consistent—**fragile**

Guix, Common Workflow Language (CWL) and ravanan

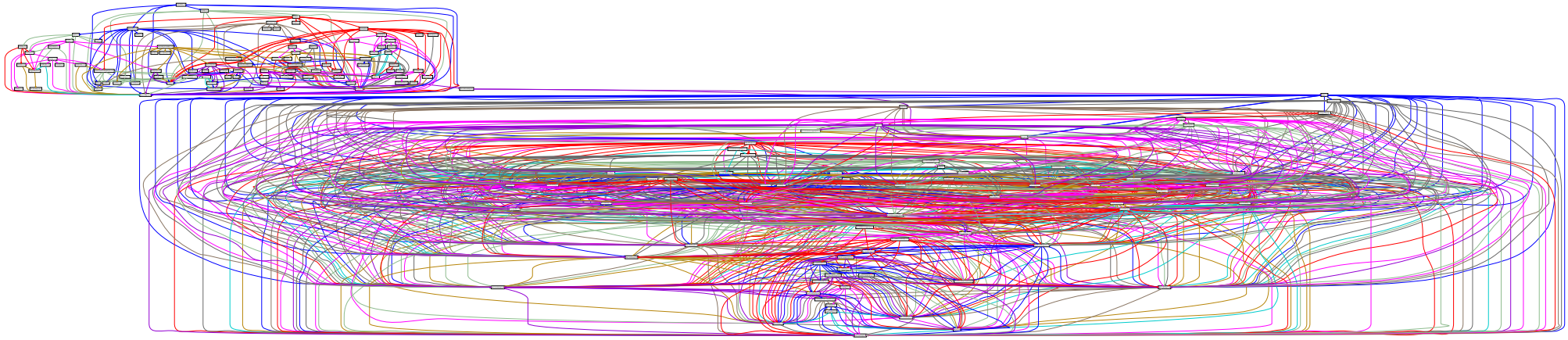
- **Guix** (like *conda*, *cran*, etc.) the package manager—install **reproducible packages**

- **Common Workflow Language (CWL)** (like a *shell script*) a format to write **portable pipelines**
- **ravanan** (like *bash*) combine Guix and CWL to **run pipelines** portably and reproducibly

Software is complex

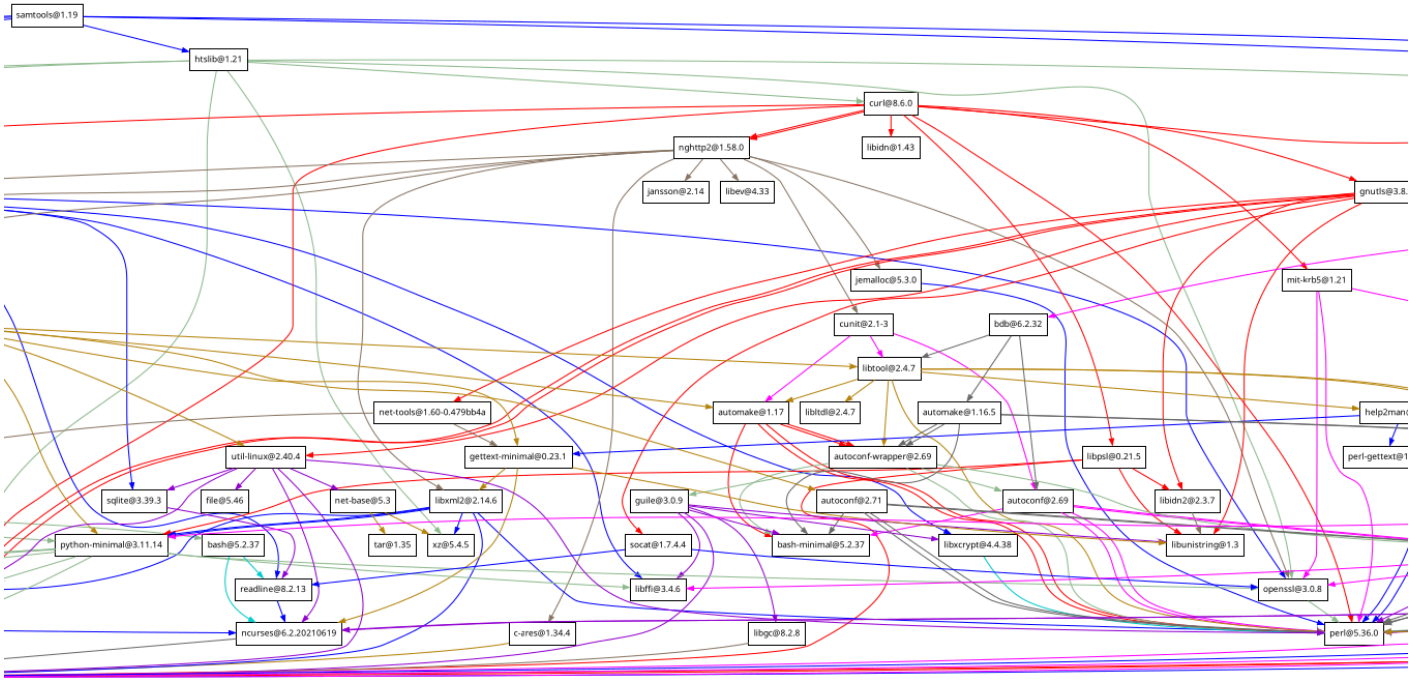


Software is very complex



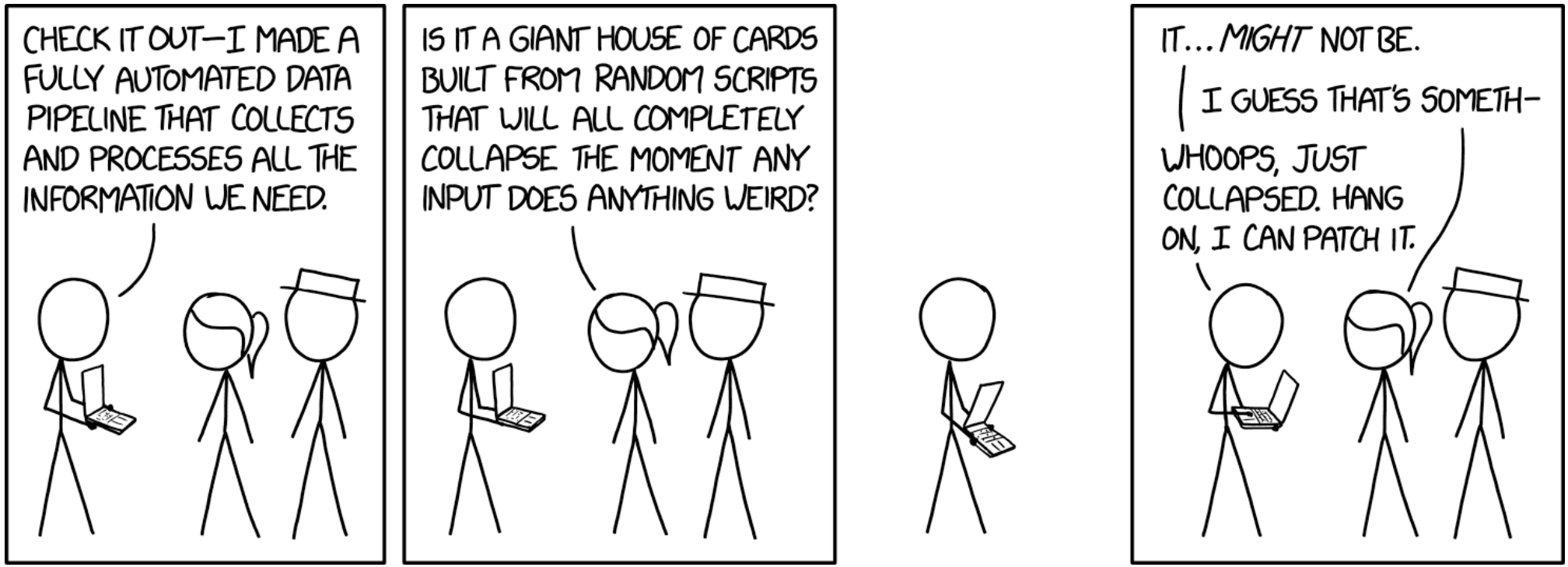
- Package manager that provides both **portability** and **reproducibility**
- Guix packages are defined by their **complete recipe**—source code, compilation steps, and their entire tree of dependencies
- Built packages are accompanied by a **fingerprint** (a hash):
  `4yh36ouvqvo6g7muov7m6rllsjotvyes-samtools`

Guix for reproducible package management



- **bit-identical** reproducibility guarantee: any change to any dependency or build step is observable in the fingerprint: `4yh36ouvqvo6g7muov7m6rllsjotvyes-samtools`
- Pin your **entire** software environment for all time

xkcd 2054: Data Pipeline

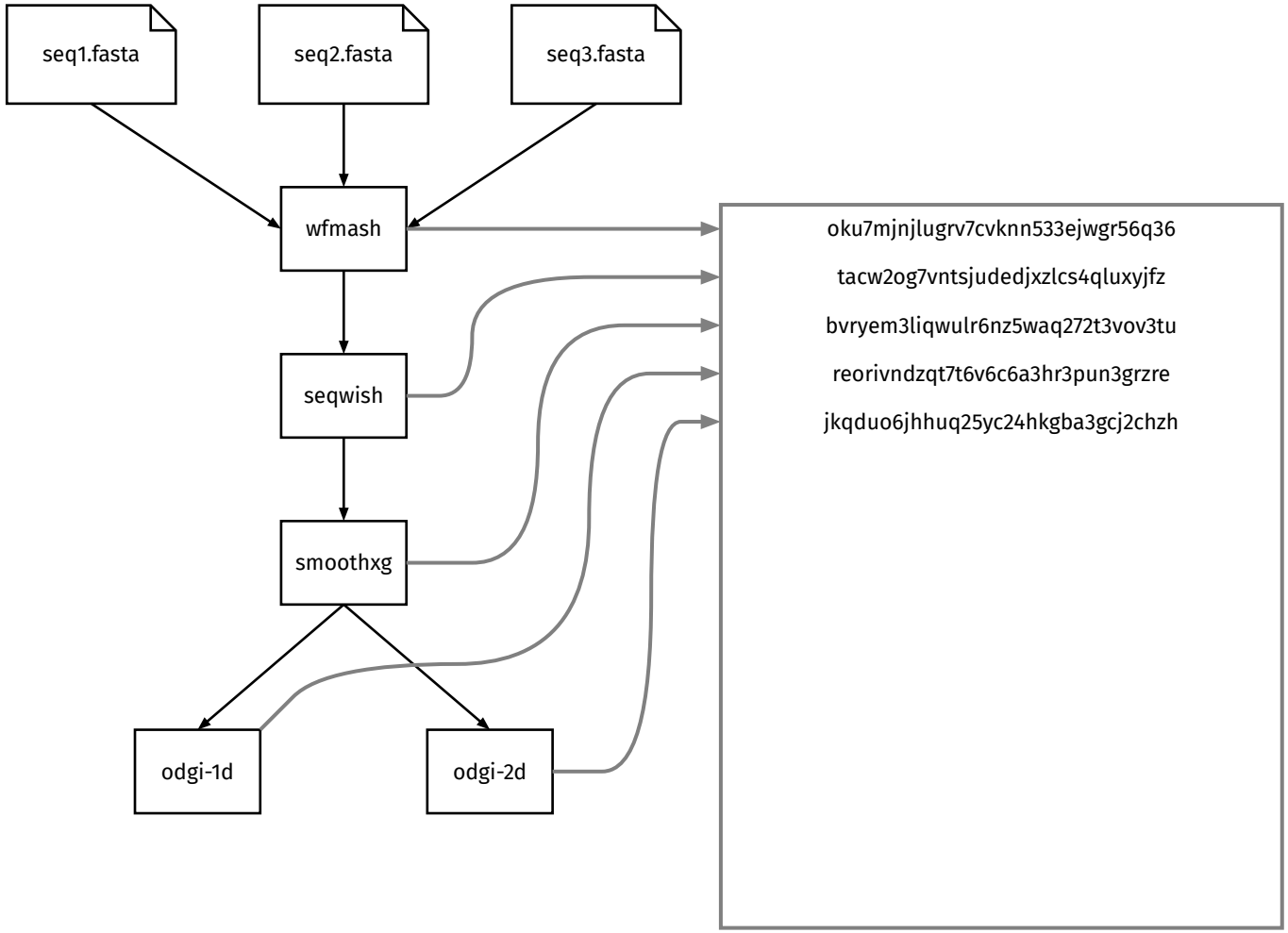


Common Workflow Language: describe pipelines, don't script them

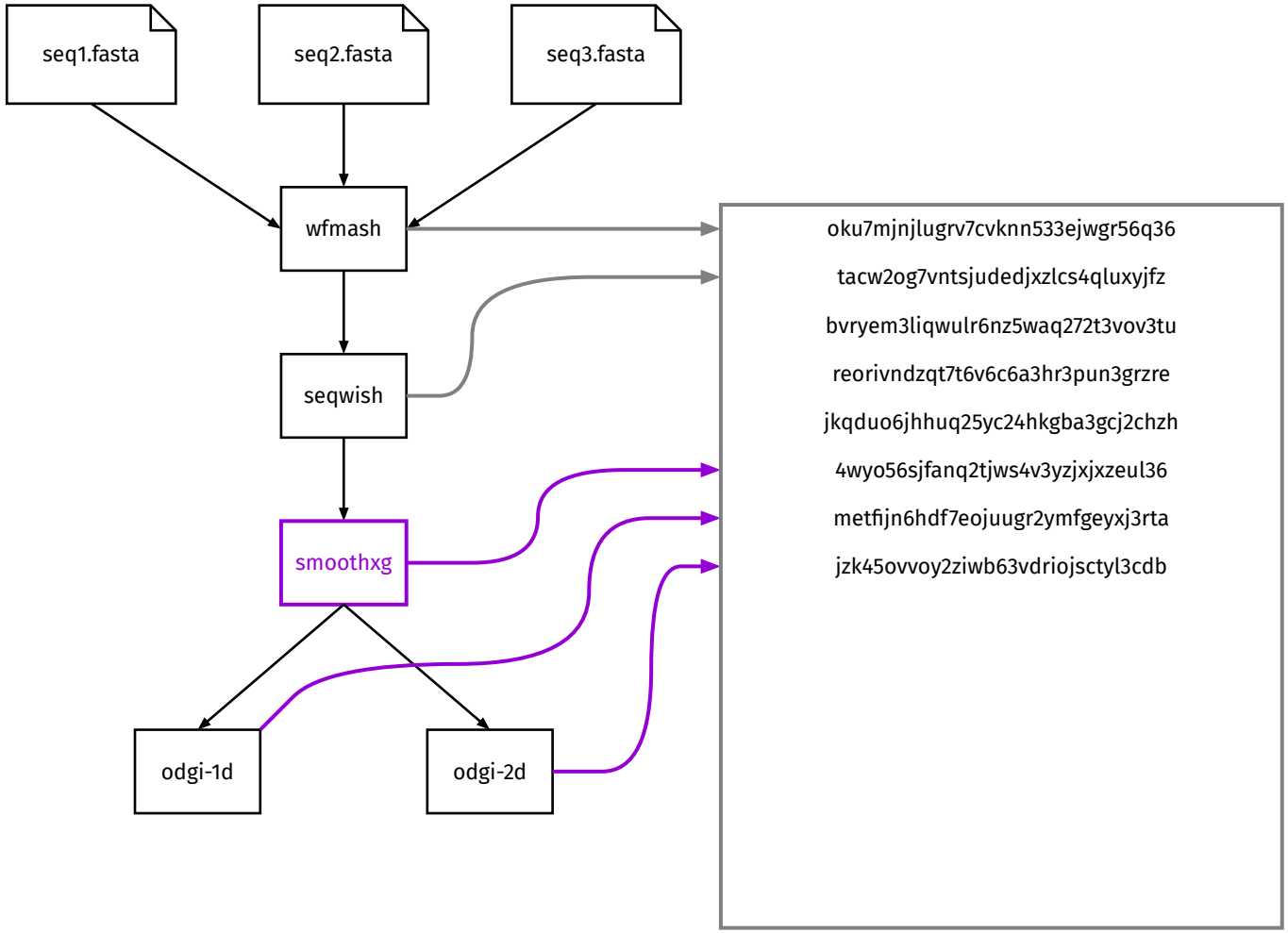
- **Portability** Pipelines run equally well on your laptop, on HPC and on collaborators' machines
- **Parallelism** Independent steps are run in parallel automatically without you have to manage parallel job submissions yourself
- **Longevity** Community standard, not the product of a single company; your pipeline will last for 200 years if need be!

- Automatic **caching**
  - Result of every step is stored in a cache keyed to the cryptographic fingerprint of all inputs, parameters and software that went into that step
    - Changing an input re-runs that step
    - Changing a parameter re-runs that step
    - Even changing the software environment (even a little change somewhere down the dependency tree) re-runs that step
- Like a sophisticated **Make** that tracks changes to software, pipeline steps, etc.

ravanan caching



ravanan caching



- Freedom to **iterate freely**:
  - Change a parameter
  - re-run the workflow
  - only what needs to change, changes.
- No anxiety. No manual bookkeeping. No cognitive load.
- **No "was this output from the old run or the new one?"**

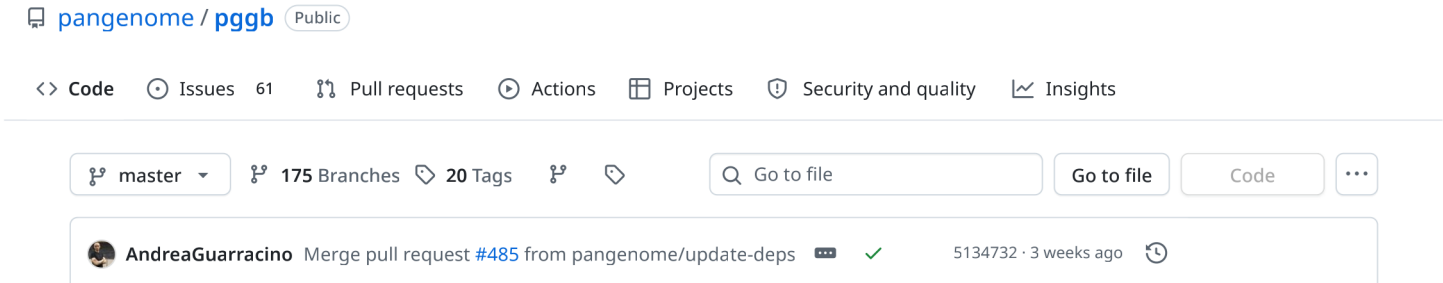
PGGB—A CONCRETE EXAMPLE

Brief Communication | Published: 21 October 2024

Building pangenome graphs

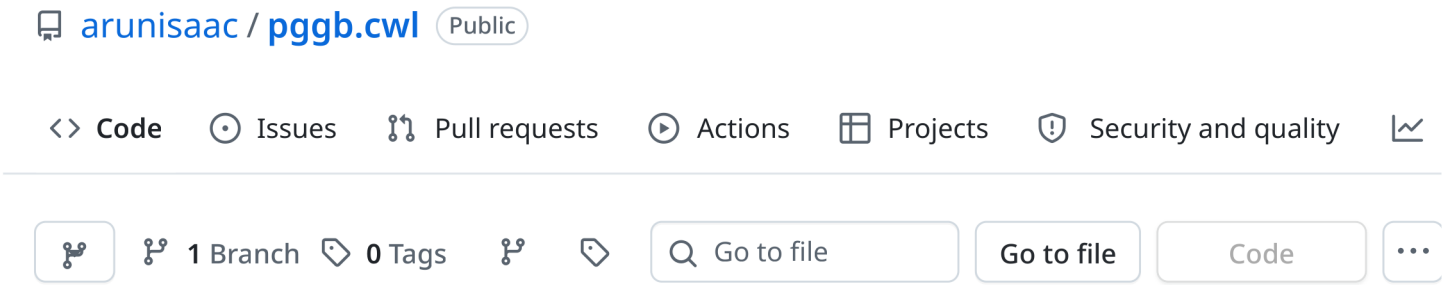
Erik Garrison ✉, Andrea Guarracino, Simon Heumos, Flavia Villani, Zhigui Bao, Lorenzo

- turn a set of sequences into a variation graph without privileging a reference
- a published pipeline in Nature Methods
- main steps are
  - **wfmash** all-to-all alignment
  - **seqwish, smoothxg and gfaffix** create variation graph from alignment, normalize and cleanup
  - **odgi** visualization



https://github.com/pangenome/pggb

- pggb is a bash script
  - lots of manual file management
  - hand-written `--resume` flag
  - manual tracking of parameters in filenames
  - strictly sequential
  - hard to inspect
  - biologically interesting bits tangled up with lots of bookkeeping



https://github.com/arunisaac/pggb.cwl

- pipeline is a dataflow graph
- wfmash all-to-all alignment is automatically parallelized
- odgi visualization steps are automatically parallelized
- caching that tracks inputs, parameters and software, not just output file presence: easy to tweak and experiment with parameters
- software environment pinned precisely and forever

Conclusion

Future you is a collaborator

- your most important collaborator is **yourself**, 6 months from now
- that you has forgotten
  - what parameters you used
  - which version of the script produced which result
  - and is probably under deadline pressure!
- leave future you with an **unreproducible mess** or the **gift of a reproducible pipeline**
- reproducibility is not a box to tick to placate reviewers
- and it is for future you's **sanity** and maybe even reading pleasure
- reproducible science is **just good science**, nothing more

For postdocs and students: UGI ECR Computational Biology “Group Therapy”

Next one on May 7

There will be free pizza 🍕 and coffee ☕ for all!