Guix as a tool for reproducible science

Konrad Hinsen

Centre de Biophysique Moléculaire, Orléans, France
and
Synchrotron SOLEIL, Saint Aubin, France

16 September 2022
Guix is ...

- a GNU/Linux distribution
- a package manager
- a manager for reproducible containers
- an account configuration manager (tomorrow!)

a tool for reproducible computation
Research is not done at `$HOME`, it’s done in a `$LAB`. 
Social contexts

- Individual
- Team
- Community
- Public
Lab notebook

From Wikipedia, the free encyclopedia

A laboratory notebook (colloq. lab notebook or lab book) is a primary record of research. Researchers use a lab notebook to document their hypotheses, experiments and initial analysis or interpretation of these experiments. The notebook serves as an organizational tool, a memory aid, and can also have a role in protecting any intellectual property that comes from the research.[2]
PiGx: reproducible genomics analysis pipelines with GNU Guix

Ricardo Wurmus, Bora Uyar, Brendan Osberg, Vedran Franke, Alexander Gosdschan, Katarzyna Wreczycka, Jonathan Ronen, Altuna Akalin


Published: 02 October 2018 Article history ▼

Email alerts
Article activity alert
Advance article alerts
New issue alert
In progress issue alert

Receive exclusive offers and updates from Oxford Academic

Abstract

In bioinformatics, as well as other computationally intensive research fields, there is a need for workflows that can reliably produce consistent output, from known sources, independent of the software environment or configuration settings of the machine on which they are executed. Indeed, this is essential for
Report 9 - Impact of non-pharmaceutical interventions (NP) to reduce COVID-19 mortality and healthcare demand

The global impact of COVID-19 has been profound, and the public health threat it represents is the most serious ever in a respiratory virus since the 1918 influenza pandemic. Here we present the results of epidemiological modeling which has informed policy making in the UK and other countries in recent weeks. In the absence of a COVID-19 vaccine, we assess the potential role of a number of public health measures – so-called non-pharmaceutical interventions (NPIs) – aimed at reducing contact rates in the population and thereby reducing transmission of the virus. In the results presented here, we apply a previously published microsimulation model to two countries: the UK (Great Britain specifically) and the US. We conclude that the effectiveness of any one intervention in isolation is likely to be limited, requiring multiple interventions to be combined to have a substantial impact on transmission.

Two fundamental strategies are possible (i) mitigation, which focuses on slowing but not necessarily stopping epidemic spread – reducing peak healthcare demand while protecting those at most risk of severe disease from infection, and (ii) suppression, which aims to reverse epidemic growth, reducing case numbers to low levels and maintaining that situation indefinitely. Each policy has major challenges. We find that high optimization policies (considering home isolation of suspect cases, home quarantine of those living in the same household as suspect cases, and social distancing of the elderly and others at most risk of severe disease) might reduce peak healthcare demand by 70% and deaths by half, however, the resulting mitigating epidemics would still likely result in hundreds of thousands of deaths and healthcare systems (most notably intensive care units) being overwhelmed many times over. For countries able to achieve it, this leaves suppression as the preferred policy option.

We show that in the UK and US context, suppression will inevitably require a combination of social
Communicating computer-aided research

- Share data
- Share code (the code you care about)
- Share computational environments (the code you don’t care about)
Computational environments

- Tools enabling research and communication
- Formalized and automated research methods
- Must be shared in a team
- **Open Science**: must be shared with community and public
- **Open Science**: must be *verifiable* by community and public
Personal computational environments

- Your $HOME
- Not shared with anyone else
- Do what you want

⇒ Guix, Debian, Ubuntu, Arch, ...
Computational environments for teams

- Precisely documented (think lab notebook!)
- Easy to inspect and modify
- Can evolve rapidly

⇒ Guix
Computational environments for communities

- Archived for years to decades (like a journal article)
- Must work identically for verifiability
- May require *some* effort to deploy
- Forkable

⇒ Guix, Docker + reproducible image, Guix + Docker
Computational environments for the public

- Simple all-in-one user interface
- Must be maintained for as long as the content is relevant
- Must yield equivalent results at all times for verifiability

⇒ Community environment + UI + support + maintenance
Reproducible environments with Guix
The magic incantation

```
guix shell -m manifest.scm

guix shell -m manifest.scm -- command argument ...
```
Guix’ alternative to a Dockerfile

(manifest
  (list "python"
    "python-matplotlib"
    "python-numpy"))
Where are the version numbers?

- Implicitly defined by Guix
- Updating Guix updates all packages in Guix
Reproducible environments

Keep a record of your Guix snapshot

```
guix describe -f channels > channels.scm
```

Use a recorded snapshot

```
guix time-machine -C channels.scm -- guix shell -m manifest.scm ...
```
The channel file

```
(list (channel
  (name 'guix)
  (url "https://git.savannah.gnu.org/git/guix.git")
  (branch "master")
  (commit
    "35b176daf1a466f136f0b77c03de78f482a30702")
  (introduction
    (make-channel-introduction
      "9edb3f66fd807b096b48283debdcddccfea34bad"
      (openpgp-fingerprint
        "BBB0 2DDF 2CEA F6A8 0D1D E643 A2A0 6DF2 A33A 54FA")))
```

Konrad Hinsen (CBM/SOLEIL)  Guix as a tool for reproducible science  16 September 2022  19/26
Two files define your environment

- `manifest.scm` defines the list of packages
- `channels.scm` defines the versions

Store both files in your version-controlled project directory!
Three types of environment

Extended $HOME environment

```
guix shell -m manifest.scm
```
- Your $HOME environment plus the packages from the manifest
- Not transferable, not reproducible
- Good for testing software, using conflicting packages

Clean environment

```
guix shell --pure -m manifest.scm
```
- Resets all environment variables ($PATH etc.)
- Unrestricted access to local files and network
- Good for isolating software but not data

Containerized environment

```
guix shell --container -m manifest.scm
```
- Same isolation/protection as with Docker
- No access to unrelated files, no network
- Best choice for reproducibility
Containers with privileges

**Network access**

```bash
guix shell --container --network -m manifest.scm
```
- Grants unrestricted network access
- No restriction possible

**Local file access**

```bash
guix shell --container --expose=/etc/ssl/certs -m manifest.scm
```
- Grants read-only access to a file/directory

```bash
guix shell --container --share=$HOME/data -m manifest.scm
```
- Grants read and write access to a file/directory
Why Guix rather than Docker?

- Reproducible containers
- Source code for containers
- Source code for all packages in a container
- No huge image files
- Can export to Docker and Singularity
Why Guix rather than Conda?

- Reproducible environments
- Source code for all packages in an environment
The dual nature of software
Summary: reproducible environments

Four commands
- `guix shell`: run in a controlled environment
- `guix describe`: record software versions
- `guix time-machine`: replay software versions
- `guix pull`: update Guix, with all its packages

Two files
- `manifest.scm`: lists your packages
- `channels.scm`: defines versions