La reproductibilité des calculs en Python

Konrad Hinsen

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

11 avril 2023

Plan

1. Principes de la reproductibilité computationnelle
2. Application à l'univers Python
3. Conseils pratiques

Schéma du calcul scientifique

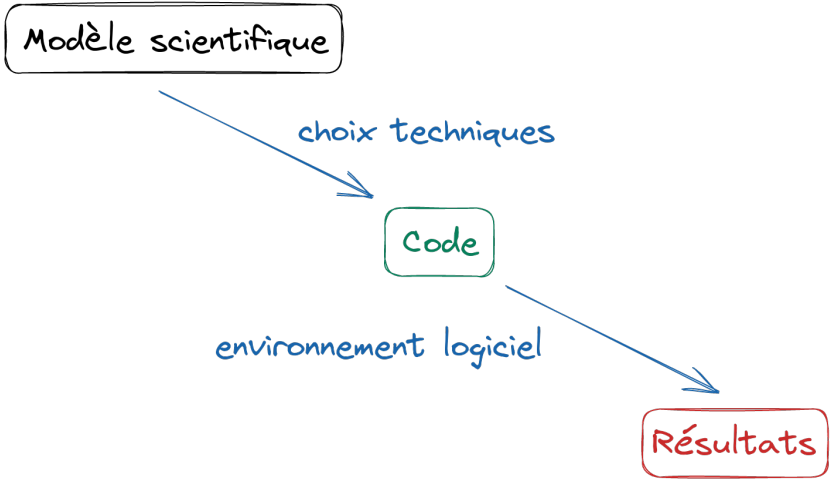
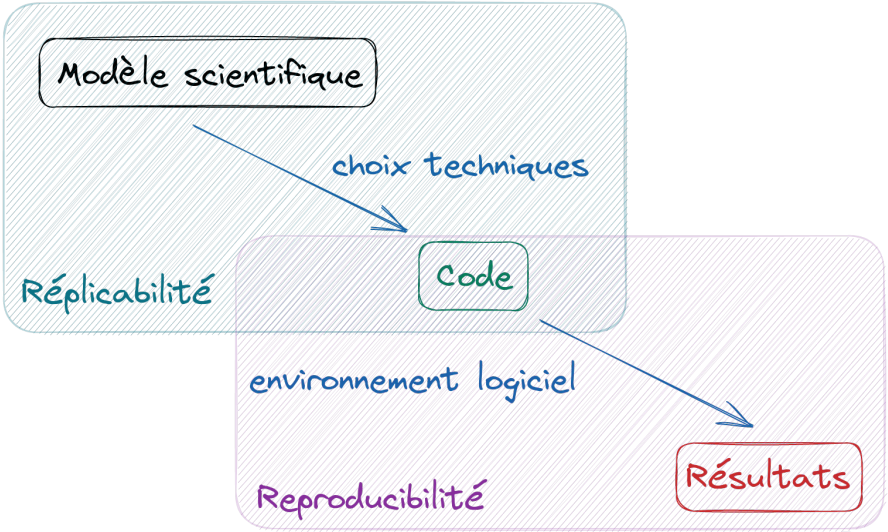


Schéma du calcul scientifique



Re. . .bilité(s)

Reproductibilité

Obtenir des résultats **identiques** (octet par octet)
sur une **autre** machine,
avec le **même** code et les **mêmes** données

Documentation du calcul : code, données, environnement logiciel

Re. . .bilité(s)

Reproductibilité

Obtenir des résultats **identiques** (octet par octet)
sur une **autre** machine,
avec le **même** code et les **mêmes** données

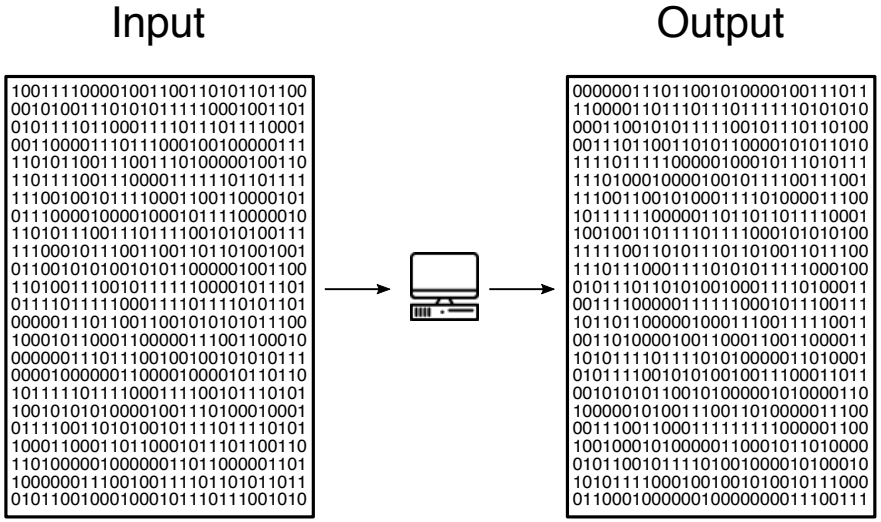
Documentation du calcul : code, données, environnement logiciel

Réplicabilité

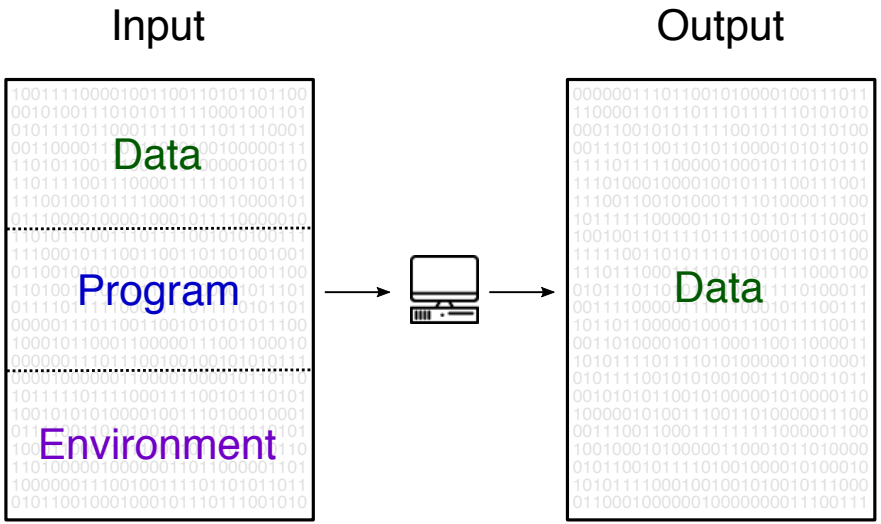
Obtenir des résultats **équivalents**
avec une **autre** implémentation
du **même** modèle scientifique

Documentation du modèle scientifique
Validation des choix techniques

What's a computation ?

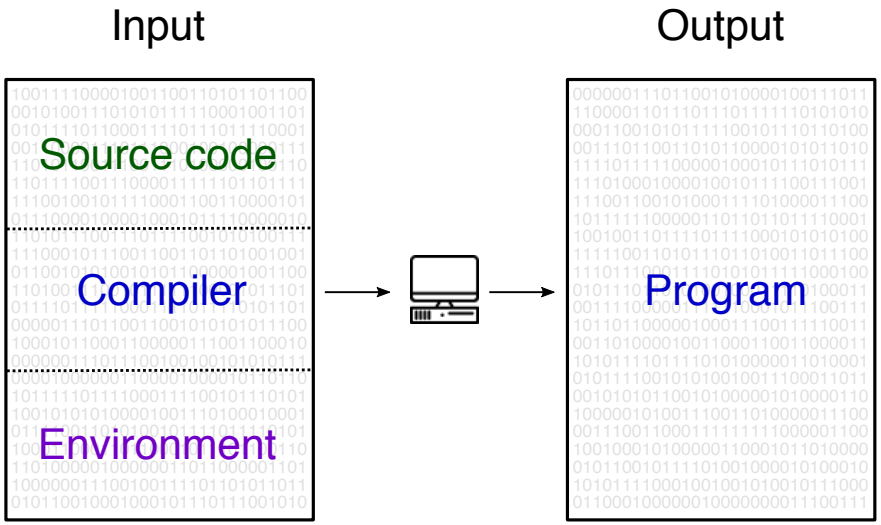


What's a computation ?



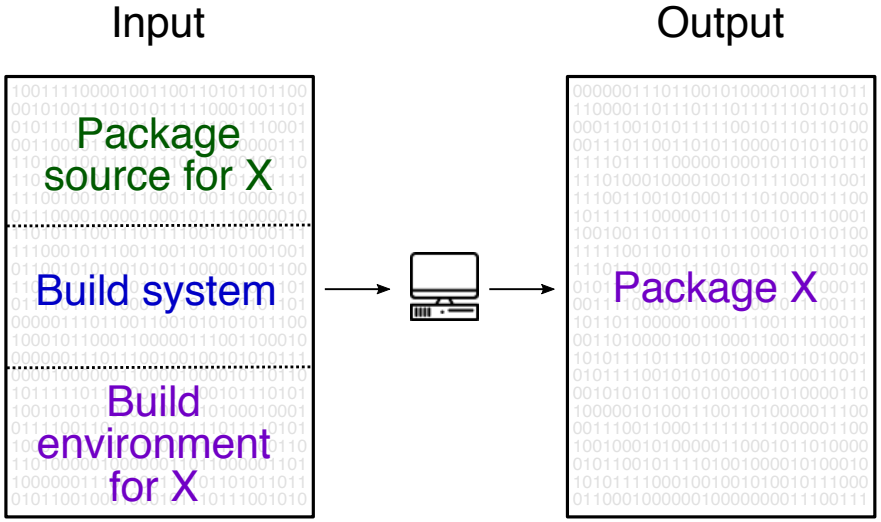
Computer by Creative Stall from the Noun Project

Compilation is computation



Computer by Creative Stall from the Noun Project

Package building is computation



Computer by Creative Stall from the Noun Project

Reproducibility

Computation is deterministic !

- Same inputs $\longrightarrow$ same output

Reproducibility

Computation is deterministic !

- Same inputs $\longrightarrow$ same output
- Same inputs = same data, same code, same environment

Reproducibility

Computation is deterministic !

- Same inputs $\longrightarrow$ same output
- Same inputs = same data, same code, same environment
- **Irreproducible output $\longrightarrow$ something has changed !**

Reproducibility

Computation is deterministic !

- Same inputs $\longrightarrow$ same output
- Same inputs = same data, same code, same environment
- **Irreproducible output $\longrightarrow$ something has changed !**
- Most often it's the environment.

Deployment vs. archiving

Deployment

- **Goal** : facilitate the use of software tools
- Prepare for distribution
- Install
- Update
- *Newer is better*

Deployment vs. archiving

Deployment

- **Goal** : facilitate the use of software tools
- Prepare for distribution
- Install
- Update
- *Newer is better*

Archiving

- **Goal** : preserve a precise software stack for reproducibility
- Make a complete snapshot
- Re-deploy a snapshot
- *Newer is different*

Package managers

Designed for deployment, not archiving

`apt`, `rpm`, `homebrew`, `pip`, ...

all but two : **Guix** and **Nix**

and a half : **conda**

Package managers

Designed for deployment, not archiving

`apt`, `rpm`, `homebrew`, `pip`, ...

all but two : **Guix** and **Nix**

and a half : **conda**

- Compatibility checks by version number and $\geq$ relations

Package managers

Designed for deployment, not archiving

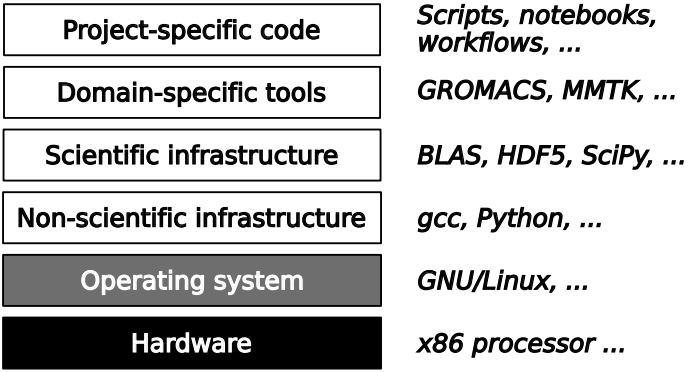
`apt`, `rpm`, `homebrew`, `pip`, ...

all but two : **Guix** and **Nix**

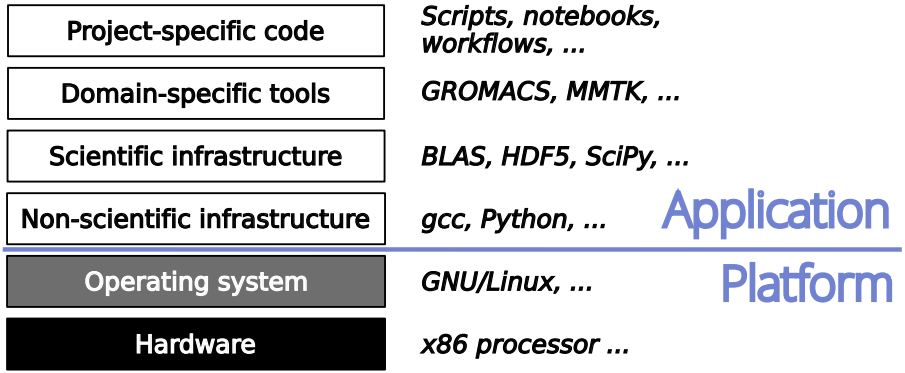
and a half : **conda**

- Compatibility checks by version number and $\geq$ relations
- The build environment is not recorded...
- ... and is thus not reproducible ...
- ... meaning packages are not reproducible either !

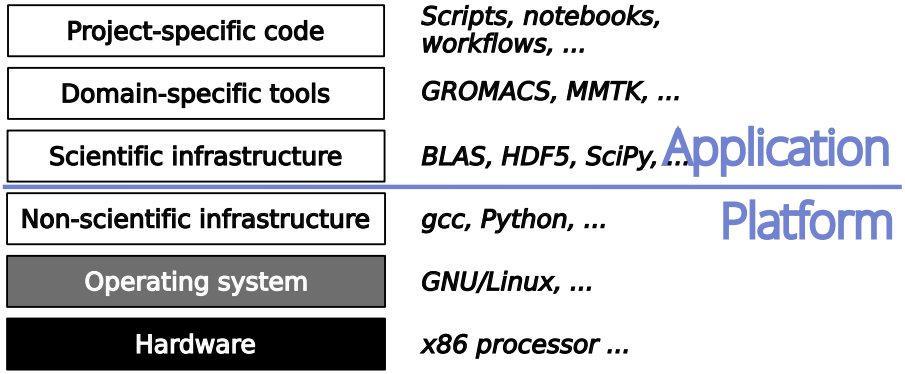
A typical scientific software stack



Foundation : the platform



Foundation : the platform



Stable platforms are rare

Python-specific package managers
cannot ensure reproducibility !

Stable platforms are rare

Python-specific package managers
cannot ensure reproducibility !

The only OS that supports
reproducibility is **Linux** !

The gold standard for reproducibility

Platform :

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

The gold standard for reproducibility

Platform :

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

Software environment :

- Composed of Guix/Nix packages
- Defined by
  - A list of channels, with a commit ID for each channel
  - A list of package specifications

The gold standard for reproducibility

Platform :

- Hardware : x86 or ARM processor
- Operating system : Linux kernel plus POSIX-compatible file system
- Package manager : Guix or Nix

Software environment :

- Composed of Guix/Nix packages
- Defined by
  - A list of channels, with a commit ID for each channel
  - A list of package specifications

Scripts, notebooks, workflows :

- Run in Guix/Nix containers
- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

Good enough

Platform :

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

Good enough

Platform :

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

Software environment :

- Composed of Debian packages
- Defined by
  - The timestamp of the snapshot
  - A list of package names

Good enough

Platform :

- Hardware : x86 or ARM processor in a virtual machine
- Operating system and package manager : Debian snapshot

Software environment :

- Composed of Debian packages
- Defined by
  - The timestamp of the snapshot
  - A list of package names

Scripts, notebooks, workflows :

- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

Acceptable for short-term reproducibility

Platform :

- Conda on top of a supported OS

Acceptable for short-term reproducibility

Platform :

- Conda on top of a supported OS

Software environment :

- Composed of Conda packages
- Defined by
  - The list of channels, ordered by priority
  - A list of package with versions and hashes

Acceptable for short-term reproducibility

Platform :

- Conda on top of a supported OS

Software environment :

- Composed of Conda packages
- Defined by
  - The list of channels, ordered by priority
  - A list of package with versions and hashes

Scripts, notebooks, workflows :

- Stored in a version-controlled repository
- Repository can also contain pure Python dependencies as source code

Conda vs. Guix/Nix

Conda

- Linux, Windows, macOS

Guix

- Linux only

Conda vs. Guix/Nix

Conda

- Linux, Windows, macOS
- one environment level

Guix

- Linux only
- recursive environments

A Conda environment *with hashes* is either reproducible or fails.

Conda vs. Guix/Nix

Conda

- Linux, Windows, macOS
- one environment level
- packages *not* reproducible

Guix

- Linux only
- recursive environments
- packages reproducible

A Conda environment *with hashes* is either reproducible or fails.

A Conda environment *without hashes* is cross-platform but not reproducible (and may fail).

Containers

Designed for deployment, not archiving

- Large binary files (“images”)
- In general not reproducible
- Not inspectable
- Not modifiable

Useful complement for an environment definition, but not a substitute !

Don’ts

- **Don’t confuse reproducibility with deployment for reuse**
- Don’t use Python-only package managers
- Don’t use packages that you cannot build yourself from source
- Don’t refer to “current” or “latest” versions
- Don’t use container images without a build recipe (`Dockerfile`)