

Comment rater la reproductibilité de ses expériences ?

Compas 2023

Quentin GUILLOTEAU , Adrien FAURE , Olivier RICHARD ,
Millian POQUET*

Univ. Grenoble Alpes, INRIA, CNRS, LIG

`firstname.lastname@inria.fr`

*IRIT, Université de Toulouse

`firstname.lastname@irit.fr`

2023-07-06

Reproductibilité ?

Définitions (ACM)

- **Répétabilité** \leadsto Mêmes personnes, mêmes conditions
- **Reproductibilité** \leadsto Personnes différentes, même conditions
- **Réplicable** \leadsto Personnes différentes, autres conditions

Évaluation des artefacts et Badges



**High-Performance and Scalable
Agent-Based Simulation with BioDynaMo**

Réutilisable / Disponible / Résultats reproduits

Pourquoi cette présentation ?

Pourquoi s'intéresser à la reproductibilité ?

- Se reposer sur des travaux **solides**
- Utilise/développe/test/distribue software
 - installer manuellement l'env. ?
 - Env partagé à toute l'équipe et machines de test
 - ~~"Ça marche sur ma machine"~~
- Recherche reproductible
 - expériences dans le même env. exactement
 - introduction précise de variation

Mais...

- Problématiques pas vraiment comprises
- "Reproductibilité" \neq stocker des images et rejouer un Jupyter !
- Souvent trop rigides et non entendables... Besoin de **variation**.

Comment partager un environnement logiciel ?

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` 😊

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹
 - Conteneur \leadsto quid de l'OS/Kernel/Drivers ?

Comment partager un environnement logiciel ?

- `requirements.txt` → quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) → oublis ?
- Module → modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹
 - Conteneur → quid de l'OS/Kernel/Drivers ?
 - → la recette plutôt ! (Dockerfile, Kameleon, etc.)

Comment partager un environnement logiciel ?

- `requirements.txt` → quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) → oublis ?
- Module → modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹
 - Conteneur → quid de l'OS/Kernel/Drivers ?
 - → la recette plutôt ! (Dockerfile, Kameleon, etc.)
- Spack ? → mieux, mais pas (totalement) reproductible

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹
 - Conteneur \leadsto quid de l'OS/Kernel/Drivers ?
 - \leadsto la recette plutôt ! (Dockerfile, Kameleon, etc.)
- Spack ? \leadsto mieux, mais pas (totalement) reproductible
- Pérennité du partage ?

Comment partager un environnement logiciel ?

- `requirements.txt` \leadsto quid des autres dépendances ?
- Liste de paquets (`apt-get install ...`) \leadsto oublis ?
- Module \leadsto modification ? pérennité ? partage ?
- Image (conteneur, VM, système)
 - `tgz-g5k`, `cc-snapshot` ☹
 - Format binaire ? ☹
 - Conteneur \leadsto quid de l'OS/Kernel/Drivers ?
 - \leadsto la recette plutôt ! (Dockerfile, Kameleon, etc.)
- Spack ? \leadsto mieux, mais pas (totalement) reproductible
- Pérennité du partage ?
- Comment introduire de la variation **précise** ?

Exemple de Dockerfile rencontré

```
FROM ubuntu
RUN apt-get update -y && \
    apt-get install -y \
        build-essential \
        ... \
        simgrid
RUN git clone https://.../chord.git
WORKDIR chord
RUN curl -L https://tinyurl.com/patchchord \
    -o increase-timeout.patch
RUN git apply increase-timeout.patch
RUN cmake .
RUN make
ENTRYPOINT [". /chord"]
```

Saurez-vous trouver toutes les erreurs ? 😊

Problèmes - Image de base

Quelle version ?!

FROM ubuntu

FROM ubuntu:latest

Traçabilité ? Quid d'une reconstruction future ?

↪ Dépendance à un état extérieur **incontrôlable** !

Mieux ?

FROM ubuntu:23.04

Pérennité/Reconstructibilité de l'image de base ?

Tag actualisé par le mainteneur ?

↪ Dépendance à un état extérieur **incontrôlable** !

Problèmes - Version du miroir

Quelle version ?!

```
RUN apt-get update
```

Traçabilité ? Quid d'une reconstruction future ?
↪ Dépendance à un état extérieur **incontrôlable** !

Mieux ?

```
RUN deb https://snapshot.debian.org/... lenny
```

Pérennité ? Introduction de variation ?
↪ Trop rigide ?

Problèmes - Commit utilisé

Quelle version ?!

```
RUN git clone https://.../mon_repo.git
```

Traçabilité ? Quid d'une reconstruction future ?

↪ Dépendance à un état extérieur **incontrôlable** !

Mieux ?

```
RUN git clone https://.../mon_repo.git?ref=...
```

```
RUN git clone https://.../mon_repo.git?rev=...
```

Pérennité ?

↪ Software-Heritage

Problèmes - Objet téléchargé

Quelle version ?!

```
RUN curl https://tinyurl.com/ma_config
```

Traçabilité ? Quid d'une reconstruction future ? Pérennité ?
↪ Dépendance à un état extérieur **incontrôlable** !

Mieux ?

```
RUN curl https://tinyurl.com/ma_config  
RUN md5sum --check attendu.md5
```

Plus important de ne pas construire une image plutôt que d'en construire une erronée !

Problèmes - En vrac

Jupyter/OrgMode/...

- Attention à l'ordre des cellules
- Attention aux dépendances à votre config ! (e.g., .emacs)

Moteur d'expérience / Moteur de workflow

- EnOSlib, Execo, Snakemake, etc.
- Fait aussi partie de l'environnement logiciel !

Triste réalisation. . . ☹️

- avec les outils et pratiques usuels :
 - faire des env logiciels reproductibles \leadsto pas facile
 - introduire de la variation précise \leadsto pas facile
- les évaluations d'artefacts sont faites juste après la création des envs
 - \leadsto tout à peu près dans le même état
 - mais dans 1 an ? 5 ans ? 20 ans ?

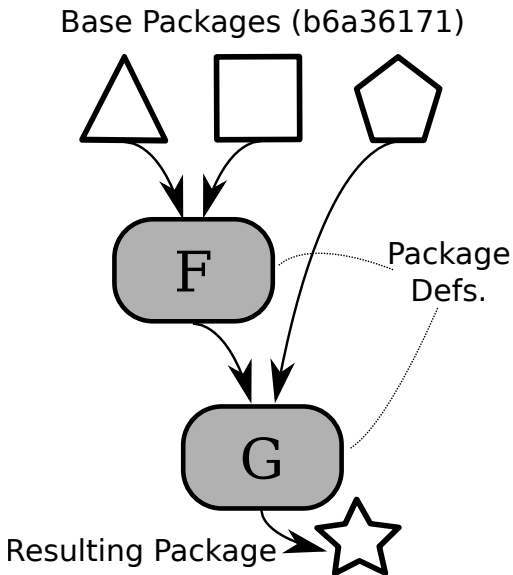
Nécessité d'avoir une autre approche à la gestion d'env logiciel !

Les gestionnaires de paquets fonctionnels à la rescousse !

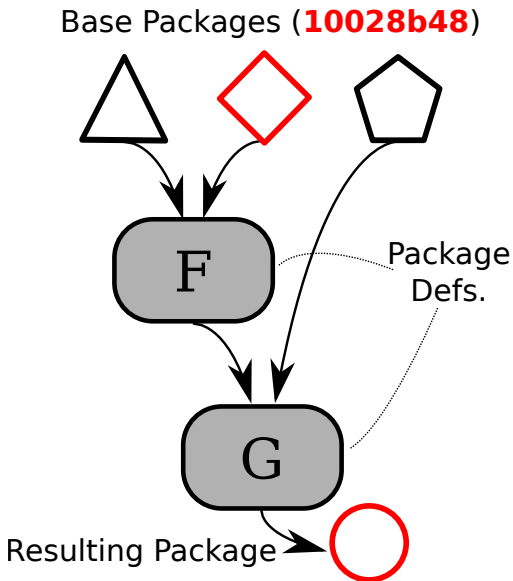
- Nix, Guix (cf. mardi après-midi ☺)
- Reproductible par design
- paquets = fonctions
 - entrées = dépendances
 - corps = commandes pour construire le paquet
 - $\text{chord} = f(\text{simgrid}, \text{boost}, \text{cmake})$
 - $f \simeq \text{cmake} + \text{make} + \dots$
- pas d'effets de bords, sandbox
- peut construire : conteneurs, VMs, images système
- Pas une solution magique : **toujours possible de se tromper !**



Vous avez dit “Fonctionnel” ?!



Vous avez dit “Fonctionnel” ?!



Exemple de Paquet

```
{ stdenv, fetchgit, simgrid, boost, cmake }:
```

Dependances

```
stdenv.mkDerivation rec {
  pname = "chord";
  version = "0.1.0";
```

```
src = fetchgit {
  url = "https://gitlab.inria.fr/me/chord";
  rev = "069d2a5bfa4c40...";
  sha256 = "sha256-ff4f...";
};
```

Sources

```
buildInputs = [ simgrid boost cmake ];
```

Build Info

```
# configurePhase = "cmake .";
# buildPhase = "make";
# installPhase = "mkdir -p $out/bin && mv chord $out/bin";
```

Derivation

Introduire de la variation

```
{ pkgs ? import (fetchTarball {
  url = "https://github.com/NixOS/nixpkgs/[...].tar.gz";
  sha256 = "sha256:[...]";}) {}
}:
```

Pinning

```
let
```

```
  packages = rec {
```

```
    chord = pkgs.callPackage ./chord.nix { };
```

Pkg Def

```
    chord_custom = chord.override {
      simgrid = simgrid-330;
      boost = boost-167;
    };
```

Override

```
    boost-176 = ...;
```

```
    boost-167 = ...;
```

```
    boost = boost-176;
```

```
    simgrid-330 = ...;
```

```
    simgrid-331 = ...;
```

```
    simgrid = simgrid-331;
```

```
  };
```

```
in packages
```

`nix-build -A chord_custom`

Comment stocker les paquets ?

Usual approach: Merge them all

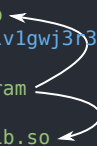
- Conflits
- PATH=/usr/bin

```
/usr
├── bin
│   └── myprogram
└── lib
    ├── libc.so
    └── libmylib.so
```

Store approach: Keep them separated

- + **Variation**
- + Isolés
- + PATH précis
- + Read-only

```
/nix/store
├── y9zg6ryffgc5c9y67fcmfdkyyiivzpj-glibc-2.27
│   └── lib
│       └── libc.so
├── nc5qbagm3wqfg2lvlgwj3r3bn88dpqr8-mypkg-0.1.0
│   ├── bin
│   │   └── myprogram
│   ├── lib
│   │   └── libmylib.so
│   └── ...
```



Critique

Avantages

- + Pas possible d'oublier des deps.
- + Traçabilité (pinned Nixpkgs)
- + `nix-shell/guix shell` = multi-langage `virtualenv`
- + Générer des images (docker, VM, système) minimales \leadsto trivial

Inconvénients

- Contaminant: toutes les deps en Nix/Guix
- Prise en main + changement de pratique
- Quelques comportements implicites
- Stockage externe (github, gitlab,...)

Conclusion

- Du mouvement sur les questions de reproductibilité... 😊
- ... mais problématiques pas tout à fait comprises 😞
- Gestion de l'env logiciel → pas facile
- Très très compliqué d'avoir un env repro avec les solutions usuelles
- Nix/Guix:
 - beaucoup mieux...
 - ... mais toujours possible de se tromper
 - **demande un changement de vision/méthodes** (comme git)
 - (pas qqchse à faire juste pour l'eval des artefacts)

Papier ici: <https://hal.science/hal-04132438>