La Recherche Reproductible : C'est quoi ? Pourquoi en faire ? Comment ?

Christophe Pouzat

IRMA, Université de Strasboug et CNRS

christophe.pouzat@mah.unistra.fr

Mini colloque 3Rs, Celphedia, Bordeaux, 3 novembre 2021

Qu'est-ce que la « recherche reproductible » ?

- Pour faire « simple », c'est une approche qui cherche à diminuer l'écart entre un idéal : les résultats devraient être reproductibles

Qu'est-ce que la « recherche reproductible » ?

- Pour faire « simple », c'est une approche qui cherche à diminuer l'écart entre un idéal : les résultats devraient être reproductibles ; et la réalité : il est souvent difficile, même pour leurs auteurs, de reproduire des résultats publiés.

Qu'est-ce que la « recherche reproductible » ?

- Pour faire « simple », c'est une approche qui cherche à diminuer l'écart entre un idéal : les résultats devraient être reproductibles ; et la réalité : il est souvent difficile, même pour leurs auteurs, de reproduire des résultats publiés.
- Concrètement, c'est une démarche qui consiste à fournir aux lecteurs d'articles, d'ouvrages, etc, l'ensemble des données et des programmes – voir l'environnement logiciel complet – accompagnés d'une description algorithmique de la façon dont les programmes ont été appliqués aux données pour obtenir les résultats présentés.

Pourquoi en faire ?

Arrivé là, deux questions sont souvent posées :

- Pourquoi s'embêter à rendre un travail reproductible (au sens précédent) si personne ne le demande ?
- Super, mais comment fait-on ?

Une remarque

- dans la pratique, ce qui est donc entendu ici par « reproduction » est tout ce qui vient *après* la collecte des données – il serait donc plus juste de parler d'analyse reproductible des données – ;
- mais comme l'approche requiert un *accès libre* à celles-ci, elles deviennent critiquables et comparables : un pas important vers une reproductibilité des données elles-mêmes.

Le *Stanford Exploration Project*

En 1992, Jon Claerbout et Martin Karrenbach écrivent :

> *Une révolution dans la formation et dans le transfert technologique résulte du mariage du traitement de texte et des interpréteurs en ligne de commande de type script. Ce mariage permet à un auteur d'associer à chaque légende de figure une étiquette référençant tout ce qui est nécessaire à la régénération de la figure : les données, les paramètres et les programmes. Ceci fournit une exemple concret de reproductibilité en science computationnelle. Notre expérience, au Stanford Exploration Project, montre que la préparation de ce type de document électronique ne demande pas beaucoup plus de travail que celui nécessaire à la préparation d'un rapport classique ; il faut juste tout archiver de façon systématique.*

Communication dont la « substantifique mœlle » sera extraite par Buckheit et Donoho (1995) qui écriront :

> *An article about computational science in a scientific publication is* ***not*** *the scholarship itself, it is merely* ***advertising*** *of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures.*

Les outils du *Stanford Exploration Project*

Les géophysiciens du SEP effectuent l'analyse de gros jeux de données ainsi que des simulations de modèles géophysiques « compliqués » (basés sur des EDPs) ; ainsi :

- ils ont l'habitude des langages compilés comme le `FORTRAN` et le `C` ;
- ils emploient des moteurs de production (*workflows*) comme `Make` ;
- ils écrivent leurs articles en TEX et $LATEX$;

Les outils du *Stanford Exploration Project*

Les géophysiciens du SEP effectuent l'analyse de gros jeux de données ainsi que des simulations de modèles géophysiques « compliqués » (basés sur des EDPs) ; ainsi :

- ils ont l'habitude des langages compilés comme le `FORTRAN` et le `C` ;
- ils emploient des moteurs de production (*workflows*) comme `Make` ;
- ils écrivent leurs articles en TEX et $LATEX$;
- leur idée clé est d'utiliser le moteur de production, non seulement pour générer les « exécutables », mais aussi pour les appliquer aux données – et ainsi générer les figures et les tables de l'article –, avant de compiler le fichier `.tex`.

Points forts et faibles de l'approche

Points forts :

- tout (données, codes sources, scripts, texte) est conservé dans une collection de répertoires imbriqués ce qui rend le travail « facile » à sauvegarder et à distribuer ;
- un accent est mis dès le départ sur l'utilisation de logiciels « libres ».

Points faibles :

- l'emploi de TEX (ou $LaTeX$) se prête mal à la « prise de notes » et est un véritable obstacle hors des maths et de la physique ;
- la gestion d'une arborisation de fichiers, pour ne pas dire l'ensemble de l'approche, est « lourde » dans le cadre d'une analyse exploratoire « au quotidien ».

Les langages de balisage léger

Un point « faible » de l'approche précédente, la nécessité d'écrire en LaTeX ou `HTML`, a maintenant disparu avec le développement de langages de balisage léger comme :

- `Markdown` ;
- `reStructuredText` ;
- `Asciidoc` ;
- `Org mode` (utilisé pour préparer cette présentation).

Exemple (version R Markdown)

Un morceau de fichier R Markdown ressemble à :

````
## Résumé des données
Le _*résumé à 5 nombres*_ du jeu de données `jeu_A`
est :

```{r resume-jeu_A}
summary(jeu_A)
```

On voit qu'__aucune saturation ne semble présente__...
````

Exemple (version R Markdown)

Un morceau de fichier R Markdown ressemble à :

````
## Résumé des données
Le _*résumé à 5 nombres*_ du jeu de données `jeu_A`
est :

```{r resume-jeu_A}
summary(jeu_A)
```

On voit qu'__aucune saturation ne semble présente__...
````

Un point important : nous avons ici affaire à des fichiers « textes » (format UTF-8).

Petit comparatif

Les solutions suivantes permettent toutes à quiconque maîtrise déjà `R`, `Python`, `Julia`, d'être productif rapidement dans le cadre d'un travail « exploratoire » ou interactif :

- Le « carnet de notes » (*notebook*) `jupyter` permet d'utiliser *séparément* les trois langages ci-dessus (et même 37 autres), un défaut important de mon point de vue : il n'y a quasiment pas d'aide d'édition ;
- RMarkdown utilisé avec RStudio permet de mettre en œuvre facilement la recherche reproductible avec `R` et (un peu) avec `Python` (avec une bonne aide d'édition).

Gestion de version

Fidèles à la tradition de « détournement » des outils de développements logiciels pour faire de la recherche reproductible, ses praticiens deviennent souvent « dépendants » des logiciels de gestion de version qui permettent de faire évoluer des documents tout en gardant une trace explicite de leurs différentes versions :

- git devient de fait l'outil standard ;
- avec github et gitlab, même des « non-experts » arrivent à l'utiliser.

Gros jeu de données

Lorsque nous commençons à travailler sur de « vraies » données nous nous trouvons généralement confrontés à deux problèmes :

- les données sont de nature « diverse », avec par exemple des enregistrements de potentiel membranaire et une séquence d'images de fluorescence pour estimer la concentration calcique.
- les données occupent un grand espace mémoire.

Les métadonnées

- Le format texte permet de stocker les données et tout le reste...
- $\Rightarrow$ ajouter des informations sur les données :
  - provenance ;
  - date d'enregistrement ;
  - protocole d'acquisition ;
  - etc.
- Ces informations sur les données sont ce qu'on appelle les métadonnées.
- Elles sont vitales pour la mise en œuvre de la recherche reproductible.

Des formats binaires, pour données composites, permettant la sauvegarde de métadonnées

- Le Flexible Image Transport System (`FITS`), créé en 1981 est toujours régulièrement mis à jour.
- Le Hierarchical Data Format (`HDF`), développé au National Center for Supercomputing Applications, en est à sa cinquième version, `HDF5`.

Les dépôts de données

Le chercheur qui travaille sur des données expérimentales (par opposition à des simulations) risque tôt ou tard d'avoir un problème lié à la recherche reproductible : comment rendre de gros jeux de données accessibles / téléchargeables par quiconque ? Heureusement, de nombreux dépôts publiques (et gratuits) sont apparus ces dernières années :

- RunMyCode ;
- Zenodo ;
- L'Open Science Framework ;
- Figshare ;
- DRYAD ;
- Exec&Share.

Évolutions récentes

- Depuis une vingtaine d'années des langages de « haut niveau », interprétés, comme `Python`, `R` ou `Julia` se sont généralisés.
- Certains comme `Python` évoluent très (trop) vite, ce qui pose de gros problèmes de stabilité dans la durée car les développeurs ne se préoccupent pas vraiment de la rétro-compatibilité.
- Les chaînes de traitement de données se sont complexifiées, de même que l'environnement matériel d'exécution : on développe les codes sur un portable et on les utilise « en grand » sur un cluster $\Rightarrow$ les *workflows* et autres *pipelines* comme `snakemake` sont en développement rapide.

Reproductibilité dans la durée

- Une expérience souvent désagréable attend le chercheur qui se lance dans la recherche reproductible.
- Malgré un document dynamique préparé avec le plus grand soin et permettant effectivement de régénérer une étude complète *au moment de la création du document*, cette régénération échoue six mois ou deux ans après.
- Cet échec résulte de la non prise en compte de la dépendance des résultats (numériques) de l'étude vis à vis de l'environnement logiciel dans lequel celle-ci a été effectuée.

Solutions ?

- C'est encore un domaine où la recherche est active ce qui implique que le paysage change vite !
- On peut continuer à travailler avec un langage de « haut niveau » comme `Python` et figer sa pile logicielle au moyen d'un conteneur comme Docker ou Singularity.
- On peut décider de travailler au maximum avec des languages compilés et normalisés comme le `C`, le `C++` ou le `Fortran` : ils évoluent moins vite et les nouvelles versions sont rétro-compatibles.

Conclusions

- Mettre en œuvre une recherche reproductible « au quotidien » ne présente plus aujourd'hui de gros problèmes.
- La recherche reproductible est une approche jeune qui doit faire face à des problèmes pas toujours pleinement anticipés, comme la *reproductibilité dans la durée*.
- Comme toute discipline nouvelle et dynamique elle voit se présenter de nombreuses propositions de solutions, pas toujours compatibles, au problème rencontrés. Nous avons ainsi aujourd'hui de nombreux moteurs de *workflow* à disposition, plusieurs systèmes de conteneurs, etc.
- Comme l'un des enjeux majeurs est la fiabilité dans le temps il va nous falloir nécessairement faire preuve de patience et rester ouverts.

Remerciements

- Les organisateurs du mini colloque « Comment améliorer le respect de la règle des 3Rs ? » pour m'avoir invité ;
- mon employeur, le CNRS, qui me permet de m'embêter à rendre mon travail reproductible même si personne ne me le demande ;
- les développeurs de tous les logiciels (libres) mentionnés dans cet exposé ainsi que ceux des logiciels que j'ai injustement oubliés ;
- vous pour m'avoir écouté.

Quelques références

- Le CLOM/MOOC Recherche reproductible : principes méthodologiques pour une science transparente, évidemment !
- *Implementing Reproducible Research*, un livre édité par V Stodden, F Leisch et R Peng, entièrement (et légalement) disponible sur le web.
- *Top 10 Reasons to Not Share Your Code (and why you should anyway)* une présentation de Randy LeVeque, à la fois très drôle et profonde.
- Ricardo Wurmus a donné une présentation courte et lumineuse des enjeux et problèmes des *workflows* dans le cadre de la recherche reproductible au FOSDEM 2021