]

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

Christophe Pouzat

IRMA, Université de Strasbourg et CNRS

Reproductibilité de la recherche, SIF, 10 mai 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 – 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 accompagnés d'une description algorithmique de la façon dont les programmes ont été appliqués aux données pour obtenir les résultats présentés.

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 *Journal of Money, Credit and Banking*

- Au début des années 80, le *Journal of Money, Credit and Banking* a adopté une politique éditoriale demandant aux auteurs les programmes et données utilisés dans leurs articles « empiriques », ainsi que la mise à disposition de ceux-ci sur simple demande (projet financé par la *National Science Foundation*).
- Les auteurs d'une analyse portant sur les 54 articles empiriques publiés par ce journal entre 1982 et 1984 sont parvenus à reproduire les résultats de... 2 d'entre eux (Dewald, Thursby and Anderson, 1986, *The American Economic Review* 76 : 587-603).
- Une autre étude portant sur la période 1996-2003 a trouvé 14 articles reproductibles sur 62 (McCullough, McGeary and Harrison, 2006, *JMCB* 38 : 1093-1107).

Les « points faibles » de l'approche

- Le dépôt des données et des codes dépendaient essentiellement de la bonne volonté des auteurs ;
- aucune spécification de format ou de description des données n'étaient imposée aux auteurs ;
- aucune description des codes n'était demandée – et comme chacun sait, la plupart des codes non documentés sont incompréhensibles même par leurs auteurs après 2 à 6 mois – ;
- aucune description de la façon dont les codes étaient appliqués aux données n'était requise.

Dette publique et taux de croissance

- Plus récemment, les économistes ont occupé le devant de la scène (de la recherche reproductible) avec le controverse sur le lien entre poids de la dette publique et taux de croissance (Reinhart et Rogoff, 2010, *AER* 100: 573–78) ;
- Herndon, Ash et Pollin ont montré que l'article original été problématique : *While using RR's working spreadsheet, we identified coding errors, selective exclusion of available data, and unconventional weighting of summary statistics* (2014, *Cambridge Journal of Economics* 38 : 257–279).
- Il faut mettre au crédit de Reinhart et Rogoff le fait qu'ils ont rendu leurs données accessibles (des tables `Excel` !), ainsi que leurs codes.

Le *Stanford Exploration Project*

En 1992, Jon Claerbout et Martin Karrenbach dans une communication au congrès de la *Society of Exploration Geophysics* écrivent :

> *A revolution in education and technology transfer follows from the marriage of word processing and software command scripts. In this marriage an author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. This provides a concrete definition of reproducibility in computationally oriented research. Experience at the Stanford Exploration Project shows that preparing such electronic documents is little effort beyond our customary report writing ; mainly, we need to file everything in a systematic way.*

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 `ratfor` (une variante du `fortran`) et le `C` ;
- ils emploient des moteurs de production comme `Cake`, une variante de `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`.

Le SEP a depuis développé Madagascar, un outil puissant et « flexible », orienté vers la géophysique et dont le moteur de production est SCons – lui même basé sur `Python`.

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 ».

Bilan (personnel) sur la « RR avec moteur de production »

- si vos projets impliquent le développement d'une « grande » quantité de codes compilés, vous utilisez déjà certainement un moteur de production ;
- inclure la production de l'article (compilation du fichier `.tex`) dans la boucle n'est alors pas un gros problème ;
- c'est la solution que j'utilise – avec des langages standardisés comme le `C` (le `C++` et le `Fortran` le sont aussi) – si je veux quelque chose qui dure ;
- si `Python` ou la `JVM` (avec `Clojure`) vous satisfont, courez découvrir le concept d'ActivePapers vous n'aurez plus à vous « embêter » avec un moteur de production, ni avec une arborisation de fichiers !

Un détour par la programmation lettrée

Lorsqu'en 1976, Donald Knuth reçoit les épreuves de la seconde édition du second volume de son *opus magnum* (*The Art of Computer Programming*), il est horrifié par leur (très) basse qualité typographique ; il décide donc :

1. d'écrire, *TEX*, un logiciel de composition de document ;
2. il en profite pour introduite l'idée de programmation lettrée et développe `WEB`, un logiciel qui permet de la mettre en œuvre.

Avec la programmation lettrée, le code et sa documentation sont « mélangés » – afin de rendre le code facilement compréhensible par un humain, par opposition à un compilateur – dans un même fichier `ASCII` (ou maintenant `UTF-8`). Deux sorties peuvent être produites :

- un fichier `.tex` qui donnera la documentation imprimable → `Weave` ;
- un fichier « source », `.c` (à l'origine, c'était du `Pascal`) qui sera compilé → `Tangle`.

La programmation lettrée expliquée par D Knuth

> *Je crois que le temps est venu pour une amélioration significative de la documentation des programmes, et que le meilleur moyen d'y arriver est de considérer les programmes comme des œuvres littéraires. D'où mon titre, « programmation lettrée ».*
> *Nous devons changer notre attitude traditionnelle envers la construction des programmes : au lieu de considérer que notre tâche principale est de dire à un ordinateur ce qu'il doit faire, appliquons-nous plutôt à expliquer à des êtres humains ce que nous voulons que l'ordinateur fasse.*

> *Le praticien de programmation lettrée peut être vu comme un essayiste, qui s'attache principalement à l'exposition du sujet et à l'excellence du style. Un tel auteur, le dictionnaire à la main, choisit avec soin les noms de ses variables et explique la signification de chacune. Il cherche à obtenir un programme qui est compréhensible parce que les concepts ont été présentés dans le meilleur ordre pour la compréhension humaine, en utilisant un mélange de méthodes formelles et informelles qui se complètent l'une l'autre.*

— Donald Knuth, Literate Programming (source : traduction Wikipédia)

« Détournement » de la programmation lettrée : R et sa fonction Sweave

R est un langage distribué sous licence *GPL*, décrit de la façon suivante sur la page des FAQ :

> *R is a system for statistical computation and graphics. It consists of a language plus a run-time environment with graphics, a debugger, access to certain system functions, and the ability to run programs stored in script files.*

Pour les programmeurs, R s'inspire du scheme mais à une syntaxe type C :

- on écrit 2+2 ;
- pas `(+ 2 2)`.

Sweave

- Sweave est une fonction de R ;
- Sweave traite des fichiers « text » qui mélangent du texte écrit en LaTeX ou HTML et du code R ;
- Sweave copie la partie textuelle telle quelle dans un nouveau fichier, exécute le code R et place le résultat (tableau, figure) dans le nouveau fichier ;
- la syntaxe d'un fichier Sweave est proche de celle d'un fichier noweb.

Avec Sweave, on remplace les figures et les tableaux d'un article par le code qui les génère.

Exemple

Un morceau de fichier Sweave ressemble à :

```
\subsection{Résumé des données}
Le \textit{résumé à 5 nombres} du jeu de données
a est :
<<resume-jeu-a>>=
summary(a)
@
On voit qu'\textbf{aucune saturation ne semble
présente}...
```

Le vrai lien entre Sweave est la programmation lettrée et la syntaxe commune délimitant les blocs de codes dans le fichier source :

- on ouvre le bloc avec «nom-du-bloc»= ;
- on le ferme avec @.

Inconvénients

- il faut connaître `R` et LaTeX (ou `HTML`) ;
- pour moi, LaTeX est très bien pour écrire des articles scientifiques et des didacticiels, mais trop lourd pour mon « cahier de laboratoire » ;
- le passage d'un format de sortie (`PDF`) à un autre (`HTML`) doit se faire avec des outils externes comme TeX4ht ;
- la programmation lettrée au sens classique de Knuth ne peut pas être mise en œuvre.

Développements « récents » : langages de balisage léger

Un point « faible » des approches précédentes, 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).

Avec le logiciel pandoc, développé par le philosophe John MacFarlane, il est possible de passer quasi instantanément de l'un à l'autre et, grâce à l'extension pandoc de Markdown, un débutant avec une heure de pratique peut générer un fichier LaTeX qui ferait envie à un expert.

- Pour les utilisateurs de `R`, la syntaxe `Markdown` est utilisable :
  - grâce au paquet `RMarkdown`,
  - plus simplement, avec RStudio ;
- les utilisateurs de `Python` peuvent employer :
  - `Pweave`,
  - ou le « carnet de notes » (*notebook*) de `jupyter`.

Exemple (version `R Markdown`)

L'exemple précédent avec `R Markdown` devient :

```
## Résumé des données
Le _résumé à 5 nombres_ du jeu de données a est :
```{r resume-jeu-a}
summary(a)
```
On voit qu'__aucune saturation ne semble présente__...
```

Pour comparaison, la version précédente (avec `Sweave`) :

```
\subsection{Résumé des données}
Le \textit{résumé à 5 nombres} du jeu de données
a est :
<<resume-jeu-a>>=
summary(a)
@
On voit qu'\textbf{aucune saturation ne semble
présente}...
```

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, 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) ;
- SageMath, basé sur `Python` 2 mais avec `R`, `Maxima` et une centaine de bibliothèques scientifiques « sous le capot » est certainement la solution la plus complète à ce jour.

Un inconvénient : ces approches sont « orientées script » et pas vraiment destinées à développer du code (même sans aller jusqu'à la programmation lettrée).

Le mode `Org` de l'éditeur `GNU emacs` combine les avantages de `SageMath` et la possibilité de faire de la programmation lettrée ; son seul inconvénient : il faut apprendre `emacs`...

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 :

- 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 ».
- les données occupent un grand espace mémoire.

Ce qu'il faut garder du format texte : 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 ;
  - source ;
  - 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

Rechercher des formats binaires pour :

- travailler avec de grosses données de natures différentes ;
- stocker des métadonnées avec les données ;
- avoir un boutisme fixé une fois pour toute.

FITS et HDF5

- 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.

Un exemple « grandeur nature »

Si le temps le permet, nous pourrons discuter du « matériel supplémentaire » de l'article *A system of interacting neurons with short term plasticity*, disponible à l'adresse :

`https://plmlab.math.cnrs.fr/xtof/interacting_neurons_with_stp`

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 ; présente en plus des approches discutées ici les *workflows* (très populaires chez les biologistes).
- La page *Reproducibility* sur le site de `Madagascar`.
- Le journal *ReScience* dont le but est de publier des réplications d'articles computationnels.
- *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.

Conclusions

- que votre travail nécessite de gros développements logiciels ou du développement de « scripts sur mesure », des solutions maintenant bien rodées permettent de mettre en œuvre « sans douleur ou presque » la recherche reproductible ;
- il va nous falloir maintenant entrer dans une bataille plus « politique » pour que cette approche soit reconnue comme elle le mérite (selon moi), c'est-à-dire pour que disparaisse des commentaires du genre : « Pourquoi s'embêter à rendre un travail reproductible si personne ne le demande ? » ;
- pour cela il faudra :
  - que les politiques éditoriales changent ;
  - que les attributions de financement « réclament » la RR ;
  - que les évaluations des chercheurs la favorise.

Remerciements

- La SIF pour l'invitation ;
- 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.