Sept ans de recherche reproductible au quotidien : de R / Sweave à Org

Christophe Pouzat

Mathématiques Appliquées à Paris 5 (MAP5)

Université Paris-Descartes et CNRS UMR 8145

christophe.pouzat@parisdescartes.fr

Rencontre de réflexion autour de la recherche reproductible

Orléans, jeudi 5 avril 2012

Note au lecteur

J’ai essayé de mettre des liens vers des sites internet et des articles pour la plupart des termes un peu techniques qui apparaissent dans ce document. Pour y accéder, il suffit de pointer vers un mot qui vous semble bizarre et de cliquer dessus. Avec un peu de chance, des explications apparaîtront dans votre navigateur oueb préféré. J’ai aussi ajouter des notes avec des références et des liens à la fin de ce document, après les remerciements.

Le contexte

Je travaille principalement sur des données neurophysiologiques :

- enregistrements extracellulaires multiples avec « matrices d'électrodes » ;
- enregistrements de la fluorescence d'une sonde sensible au calcium.

De plus en plus, je « donne des coups de main » pour l'analyse de données en biologie comme pour l'étude de l'effet du régime alimentaire sur l'espérance de vie de lémuriens.

L’analyse de mes données

Trois cas de figure se présentent quand j’analyse mes données :

1. refaire une « analyse standard » sur un nouveau jeu de données ;
2. développer des « codes courts » – 10 à 100 lignes dans un langage « riche », comme `R` – pour un nouveau type d’analyse ;
3. mettre en œuvre une nouvelle méthodologie avec de codes de plus de 1000 lignes en langage compilé (`C` et, maintenant, `Common Lisp`).

Reproductibilité

Pour rendre l'analyse reproductible et *critiquable* il faut :

- décrire l'environnement matériel : processeurs (`cat /proc/cpuinfo`), mémoire (`cat /proc/meminfo`), etc. ;
- décrire l'environnement logiciel : noyau, version des compilateurs, etc. ; travailler avec des logiciels « libres » facilite clairement les choses ;
- *documenter les codes* ;
- documenter tous les choix non automatiques effectués lors de l'analyse.

Une remarque

La reproductibilité d'un résultat dans une science expérimentale suppose que :

1. les données sont reproductibles ;
2. l'analyse l'est aussi.

Je ne vais discuter ici, *stricto sensu*, que le second point. Mais, l'analyse reproductible implique l'accès aux données brutes. Les données brutes accessibles deviennent plus facilement critiquables et comparables, ce qui devrait faire progresser le premier point.

Passage de la théorie à la pratique : les outils (1)

Convaincus du bien-fondé de l'approche *recherche reproductible* (RR) comme le sont en général les étudiants et les jeunes chercheurs – moins les chercheurs établis, dans mon domaine – il nous faut trouver les outils pour passer de « la théorie » à la pratique :

- les outils sont cruciaux pour une mise en œuvre *systématique* de la RR ;
- les « premières approches », comme celles du Stanford Exploration Project (Claerbout et Karrenbach, 1992), basées sur LaTeX, des codes en C et fortran et une compilation contrôlée par make, sont trop lourdes pour *mon* travail quotidien qui implique beaucoup d'analyse exploratoire.

Passage de la théorie à la pratique : les outils (2)

- j'ai besoin de travailler principalement avec des logiciels / langages de programmation de « haut niveau » (`Matlab`, GNU Octave, R, python) ;
- je génère beaucoup de figures ;
- avoir un minimum de fichiers à gérer pour mettre en œuvre la RR aide beaucoup ;
- avoir des fichiers ouvrables et dans un langage compréhensible par mes collaborateurs biologistes (au moins pour la partie texte) est un gros plus.

Je vais maintenant brièvement décrire deux outils qui satisfont à tous ou partis de ces critères.

R

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 avec la portée lexicale (lexical scoping) mais à une syntaxe type C : on écrit `2+2` et 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 ;
- Sweave dérive de l'approche de programmation lettrée (literate programming) proposée par Don Knuth ;
- 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
$\alpha$ est :
<<resume-jeu-alpha>>=
summary(α)
@
On voit qu'\textbf{aucune saturation ne semble
présente}...
```

Extensions

- il est possible de contrôler finement l'inclusion de l'évaluation d'un morceau de code (c.-à-d. garder le résultat, le résultat et le code, etc.) ;
- il est possible de « cacher » (écrire sur le disque) le résultat de l'évaluation d'un morceau de code pour ne pas le recalculer à chaque fois ;
- les données brutes peuvent être distribuées avec le fichier `Sweave`, ainsi que des codes `C` appelés par `R`, sous forme d'un paquet `R` ; c'est le concept de *compendium* de Gentleman et Temple-Lang (2007).

Inconvénients

- il faut connaître R et LaTeX (ou HTML) ;
- pour moi, LaTeX est très bien pour écrire des articles scientifiques et des tutoriels, 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.

Org

- Org est un mode de emacs, le « couteau suisse » des éditeurs ;
- `Org` génère du `PDF` via LaTeX, de l'`HTML`, du `DocBook`, du `ODT` (*OpenDocument Text*, le format de `OpenOffice` / `LibreOffice`) avec une syntaxe simplifiée type Markdown ;
- « `* Ma section` » marque le début d'une section intitulée « Ma section » ;
- « `** Ma sous-section` » marque le début d'une sous-section, etc. ;
- `*gras*` apparaît **gras**, `/italique/` apparaît *italique*, `_souligné_` apparaît souligné.

Je ne fais ici qu'érafler la surface de Org…

babel

Babel est une extension de `Org` dont le nom vient de :

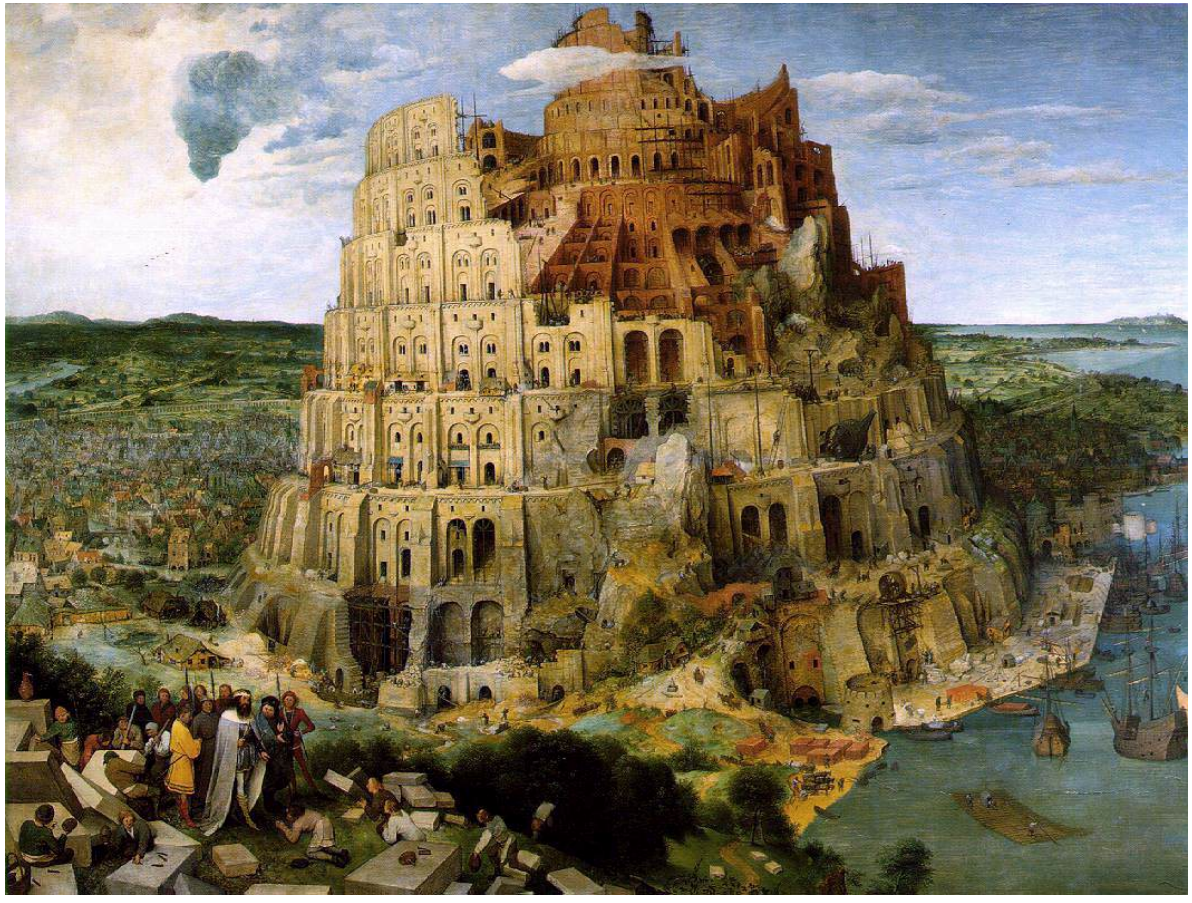

Figure : Les langages dont il est question ici sont les langages de programmation et de « scripts ».

babel

- babel permet d'inclure des « blocs de code » dans un fichier Org ;
- à ce jour 39 langages différents dont : R, Python, Perl, C, C++, Matlab, Maxima, SQL, Java, Gnuplot, peuvent être utilisés en même temps ;
- babel permet de faire de la métaprogrammation, c.-à-d. que le résultat d'un bloc de code Maxima peut être utilisé comme argument d'entrée d'un bloc de code Matlab ;
- babel permet de mettre en œuvre aussi bien la RR que la programmation lettrée ;
- on peut tout avoir dans un seul fichier.

Exemple

L’exemple précédent donne :

```
** Résumé des données
Le /résumé à 5 nombres/ du jeu de données
α est :
#+NAME: resumé-α
#+BEGIN_SRC R :exports results
summary(α)
#+END_SRC
On voit qu’*aucune saturation ne semble
présente*…
```

Remarques en guise de conclusions

- `Org` est très simple à apprendre ;
- ceux qui utilisent déjà `emacs` pour coder en `Matlab`, `Python` `R`, etc., peuvent immédiatement passer à `Org` + `babel` ;
- un seul fichier source donne différentes sorties ;
- avec sa syntaxe simplifiée et l'utilisation de l'`UTF-8`, même des gens qui ne connaissent rien à `emacs` et ne l'utilisent pas peuvent modifier le fichier source sans faire de bêtises, cela m'aide énormément dans mes collaborations avec les biologistes ;
- en fait, `Org` évolue d'un « simple » `mode emacs` vers un format et, à terme, `emacs` ne sera plus indispensable pour traiter les fichiers `Org`.

Remerciements

- les organisateurs de cette « Rencontre de réflexion autour de la recherche reproductible » ;
- les développeurs de `R`, LaTeX, `emacs`, ... (la liste est trop longue) et surtout `Org` et `babel` ;
- mes étudiants et postdocs (M. Delecluse, T. Lieury, S. Joucla, R. Franconville) que j'ai « fortement incités » à se mettre à la `RR` ;
- Bastien Guerry pour ses suggestions ;
- mon employeur, le `CNRS`, qui me permet de faire tout cela ;
- vous pour m'avoir écouté.

Notes (1)

J'ajoute dans les quelques pages qui suivent des notes et des liens pour prolonger le contenu de cette présentation :

- `R` est, contrairement à `Matlab`, un langage dont la définition est disponible http ://cran.r-project.org/doc/manuals/R-lang.html ;
- lorsque `R` a été développé par Ross Ihaka et Robert Gentleman des choix ont été délibérément effectués par ces derniers et sont expliqués dans un article téléchargeable à l'adresse : http ://www.stat.auckland.ac.nz/%7Eihaka/downloads/R-paper.pdf ;
- d'autres choix auraient été possibles et, à posteriori, préférables, comme discuté dans l'article : http ://www.stat.auckland.ac.nz/%7Eihaka/downloads/Compstat-2008.pdf.

Notes (2)

- un des papiers fondateurs de Richard Stallman sur `emacs` peut être téléchargé à l'adresse : ftp ://publications.ai.mit.edu/ai-publications/pdf/AIM-519A.pdf ;
- après le tutoriel et la visite guidée (http ://www.gnu.org/software/emacs/tour/), les cours de Phil Sung : http ://web.psung.name/emacs/ constituent un bon accès à `emacs` ;
- pour apprendre à utiliser `R` avec `emacs`, voyez les tutoriels de Stephen Eglen : http ://www.damtp.cam.ac.uk/user/sje30/ess11/index.html.

Notes (3)

- le site de `Org` : http ://orgmode.org/ ;
- pour apprendre les différentes (et très nombreuses !) fonctionnalités, voyez `Worg` : http ://orgmode.org/worg/ et ses tutoriels ;
- le site de `babel` : http ://orgmode.org/worg/org-contrib/babel/ ;
- pour commencer avec `babel`, voyez le papier de Eric Schulte, Dan Davison, Thomas Dye et Carsten Dominik (2012) A Multi-Language Computing Environment for Literate Programming and Reproducible Research *The Journal of Statistical Software* **46** (3) : http ://www.jstatsoft.org/v46/i03.

Notes (4)

Je termine par un peu d'auto-publicité :

- un article avec M. Delescluse, S. Joucla, R. Franconville et T. Lieury sur la recherche reproductible : http ://hal.archives-ouvertes.fr/hal-00591455 ;
- un page oueb : http ://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/ReproducibleDataAnalysis/ReproducibleDa

Ce document a été préparé avec `emacs`, `Org` et XeTeX.