Making neurophysiological data analysis reproducible. Why and how?

Matthieu Delescluse, Romain Franconville, Sébastien Joucla, Tiffany Lieury, Christophe Pouzat

Making neurophysiological data analysis reproducible. Why and how?

Matthieu Delescluse[a], Romain Franconville[a,1], Sébastien Joucla[a,2], Tiffany Lieury[a], Christophe Pouzat[a,*]

[a]*Laboratoire de physiologie cérébrale, CNRS UMR 8118, UFR biomédicale, Université Paris-Descartes, 45, rue des Saints-Pères, 75006 Paris, France*

Abstract

Reproducible data analysis is an approach aiming at complementing classical printed scientific articles with everything required to independently reproduce the results they present. "Everything" covers here: the data, the computer codes and a precise description of how the code was applied to the data. A brief history of this approach is presented first, starting with what economists have been calling *replication* since the early eighties to end with what is now called *reproducible research* in computational data analysis oriented fields like statistics and signal processing. Since efficient tools are instrumental for a routine implementation of these approaches, a description of some of the available ones is presented next. A toy example demonstrates then the use of two open source software programs for reproducible data analysis: the "`Sweave` family" and the `org-mode` of `emacs`. The former is bound to `R` while the latter can be used with `R`, `Matlab`, `Python` and many more "generalist" data processing software. Both solutions can be used with Unix-like, Windows and Mac families of operating systems. It is argued that neuroscientists could communicate much more efficiently their results by adopting the reproducible research paradigm from their lab books all the way to their articles, thesis and books.

Keywords: Software, R, emacs, Matlab, Octave, LaTeX, org-mode, Python

1. Introduction

> 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.

Thoughts of Jon Claerbout "distilled" by Buckheit and Donoho (1995).

The preparation of manuscripts and reports in neuroscience often involves a lot of data analysis as well as a careful design and realization of figures and tables, in addition to the time spent on the bench doing experiments. The data analysis part can require the setting of parameters by the analyst and it often leads to the development of dedicated scripts and routines. Before reaching the analysis stage *per se* the data frequently undergo a preprocessing which is rarely extensively documented in the methods section of the paper. When the article includes numerical simulations, key elements of the analysis, like the time step used for conductance based neuronal models, are often omitted in the description. As readers or referees of articles / manuscripts we are therefore often led to ask questions like:

- What would happen to the analysis (or simulation) results if a given parameter had another value?
- What would be the effect of applying my preprocessing to the data instead of the one used by the authors?
- What would a given figure look like with a log scale ordinate instead of the linear scale use by the authors?

*Corresponding author

Email addresses: `matthieu.delescluse@polytechnique.org` (Matthieu Delescluse), `franconviller@janelia.hhmi.org` (Romain Franconville), `sebastien.joucla@parisdescartes.fr` (Sébastien Joucla), `tiffany.lieury@parisdescartes.fr` (Tiffany Lieury), `christophe.pouzat@parisdescartes.fr` (Christophe Pouzat)

[1]Present address: Janelia Farm Research Campus, 19700 Helix Drive, Ashburn, VA 20147, USA

[2]Present address: Institut de Neurosciences Cognitives et Intégratives d'Aquitaine, CNRS UMR 5287, Université de Bordeaux, Bâtiment B2 - Biologie Animale - 4ème étage, Avenue des Facultés, 33405 Talence cedex, France

- What would be the result of applying that same analysis to my own data set ?

We can of course all think of a dozen of similar questions. The problem is to find a way to address them. Clearly the classical journal article format cannot do the job. Editors cannot publish two versions of each figure to satisfy different readers. Many intricate analysis and modeling methods would require too long a description to fit the usual bounds of the printed paper. This is reasonable for we all have a lot of different things to do and we cannot afford to systematically look at every piece of work as thoroughly as suggested above. Many people (Claerbout and Karrenbach, 1992; Buckheit and Donoho, 1995; Rossini and Leisch, 2003; Baggerly, 2010; Diggle and Zeger, 2010; Stein, 2010) feel nevertheless uncomfortable with the present way of diffusing scientific information as a canonical (printed) journal article. We suggest what is needed are more systematic and more explicit ways to describe how the analysis (or modeling) was done.

These issues are not specific to published material. Any scientist after a few years of activity is very likely to have experienced a situation similar to the one we now sketch. A project is ended after an intensive work requiring repeated daily long sequences of sometimes "tricky" analysis. After six months or one year we get to do again very similar analysis for a related project; but the nightmare scenario starts since we forgot:

- The numerical filter settings we used.
- The detection threshold we used.
- The way to get initial guesses for our nonlinear fitting software to converge reliably.

In other words, given enough time, we often struggle to exactly reproduce *our own* work. The same mechanisms lead to know-how being lost from a laboratory when a student or a postdoc leaves: the few parameters having to be carefully set for a successful analysis were not documented as such and there is nowhere to find their typical range. This leads to an important time loss which could ultimately culminate in a project abandonment.

We are afraid that similar considerations sound all too familiar to most of our readers. It turns out that the problems described above are not specific to our scientific domain, and seem instead to be rather common at least in the following domains: economics (Dewald et al., 1986; Anderson and Dewald, 1994; McCullough et al., 2006; McCullough, 2006), geophysics (Claerbout and Karrenbach, 1992; Schwab et al., 2000), signal processing (Vandewalle et al., 2009; Donoho et al., 2009), statistics (Buckheit and Donoho, 1995; Rossini, 2001; Leisch, 2002a), biostatistics (Gentleman and Temple Lang, 2007; Diggle and Zeger, 2010), econometrics (Koenker and Zeileis, 2007), epidemiology (Peng and Dominici, 2008) and climatology (Stein, 2010; McShane and Wyner, 2010) where the debate on analysis reproducibility has reached a particularly acrimonious stage. The good news about this is that researchers have already worked out solutions to our mundane problem. The reader should not conclude from the above short list that our community is immune to the "reproducibility concern" since the computational neuroscience community is also now addressing the problem by proposing standard ways to describe simulations (Nordlie et al., 2009) as well as developing software like *Sumatra*[3] to make simulations / analysis reproducible. The "whole" scientific community is also giving a growing publicity to the problem and its solutions as witnessed by two workshops taking place in 2011: "Verifiable, reproducible research and computational science", a mini-symposium at the SIAM Conference on Computational Science & Engineering in Reno, NV on March 4, 2011[4]; "Reproducible Research: Tools and Strategies for Scientific Computing", a workshop in association with Applied Mathematics Perspectives 2011 University of British Columbia, July 13-16, 2011[5] as well as by "The Executable Paper Grand Challenge" organized by Elsevier[6].

In the next section we review some of the already available tools for reproducible research, which include data sharing and software solutions for mixing code, text and figures.

2. Reproducible research tools

2.1. Sharing research results

Reproducing published analysis clearly depends, in the general case, on the availability of both data and code. It is perhaps worth reminding at this stage *NIH* and *NSF* grantees of the data sharing policies of these two institutions: "Data sharing should be timely and no later than the acceptance of publication of the main findings from the final dataset. Data from large studies can be released in waves as data become available or as they are published" (NIH, 2003) and "Investigators are expected to share with other researchers, at no more than

[3] `http://neuralensemble.org/trac/sumatra/wiki`
[4] `http://jarrodmillman.com/events/siam2011.html`.
[5] `http://www.stodden.net/AMP2011/`.
[6] `http://www.executablepapers.com/index.html`.

incremental cost and within a reasonable time, the primary data, samples, physical collections and other supporting materials created or gathered in the course of work under NSF grants." (The National Science Foundation, 2011, Chap. VI, Sec. D 4 b). Publishers like *Elsevier* (Elsevier, 2011, Sec. Duties of Authors) and *Nature* (ESF, 2007, p. 15) also require that authors make their data available upon request, even if that seems to be mere lip service in some cases (McCullough and McKitrick, 2009, pp. 16-17).

In short, many of us already work and publish in a context where we have to share data and sometimes codes *even if we are not aware of it*. The data sharing issue is presently actively debated but the trend set by the different committees and organizations seems clear: data will have to be shared sooner or later. We can expect or hope that the same will be true for publicly funded code developments.

2.2. Early attempts: database deposits

The economists took on very early the problem of reproducible analysis, that they usually call *replication*, of published results: "In 1982, the *Journal of Money, Credit and Banking*, with financial support from the National Science Foundation, embarked upon the *JMCB Data Storage and Evaluation Project*. As part of the *Project*, the *JMCB* adopted an editorial policy of requesting from authors the programs and data used in their articles and making these programs and data available to other researches on request." Dewald et al. (1986). The results turned out to be scary, with just 2 out of 54 (3.7%) papers having reproducible / replicable results (Dewald et al., 1986). This high level of non reproducible results was largely due to authors not respecting the *JMCB* guidelines and not giving access to both data and codes. A recent study (McCullough et al., 2006) using the 1996-2003 archives of the *JMCB* found a better – but still small – rate of reproducible results, 14/62 (22%). We do not expect the neurobiologists to behave much better in the same context. Policies are obviously not sufficient. This points out the need for dedicated tools making reproducibility as effortless as possible. Besides data sharing platforms, this is a call for reproducible research software solutions. The ideal software should :

- Provide the complete analysis code along with its documentation (anyone working with data and code knows they are useless if the author did not take the time to properly document them) and its theoretic or practical description,
- Allow any user/reader to easily re-run and modify the presented analysis.

We will first present a brief survey of existing tools matching these criteria, before focusing on the two that appear to us as the most promising and generally applicable to neuroscience : `Sweave` and `org-mode`. We will then illustrate their use with a "toy example".

2.3. Implementations of reproducible analysis approaches

A first comprehensive solution: The Stanford Exploration Project. Claerbout and Karrenbach (1992), geophysicists of the Stanford Exploration Project, state in their abstract: "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." Going beyond the database for data and code concept, they decided to link organically some of the figures of their publications and reports with the data and the code which generated them. They achieved this goal by using TEX (Knuth, 1984b) and LATEX (Lamport, 1986) to typeset their documents, a flavor of `fortran` called `ratfor` (for `rat`ional `for`tran) and `C` to perform the computation and `cake` a flavor of `make` to repeat automatically the succession of program calls required to reproduce any given figure automatically. Colleagues were then given a CD ROM with the data, the codes, the final document as well as other required open source software. They then could reproduce the final document – or change it by altering parameter values – provided they had a `UNIX` running computer as well as the appropriate `C` and `fortran` compilers. The approach was comprehensive in the sense that it linked organically the results of a publication with the codes used to generate them. It was nevertheless not very portable since it required `UNIX` running computers. The Stanford Exploration Project has since then done a considerable effort towards portability with their `Madagascar` project[7] (Fomel and Hennenfent, 2007). Madagascar users can work with compiled codes (`C`, `C++`, `fortran`) as well as

[7] `www.reproducibility.org/wiki/Main_Page`

with "generalist" packages like Matlab or Octave or with interpreted scripting languages like Python. The use of the package on Windows still requires "motivation" since users have to install and use cygwin – a Linux like environment for Windows – and this is, in our experience, a real barrier for the users. In addition, although the link is "organic", figures and the code which generated them are not kept in the same file.

WaveLab: A more portable solution. Buckheit and Donoho (1995); Donoho et al. (2009), statisticians from Standford University inspired by their colleague Jon Claerbout, proposed next a "reproducible research" solution based entirely on Matlab in the form of their WaveLab library[8]. Being wavelets experts, and having to talk, work, and write articles with mathematicians and experimentalists, they had to face the problem of keeping a precise track of what different contributors had done. They also had to frequently change the "details" of their analysis based on the asynchronous feedback of the different contributors. A constraint which naturally called for a fully scripted analysis – with adjustable parameters – as well as scripted figures and tables generation. As Matlab users, they naturally solved their problems with this software. The results usable on Windows, Mac OS and Unix, the WaveLab library, accompanies published articles and book chapters and includes Matlab codes, data and the documentation of both. As with Madagascar, codes are distributed separately from the final document. Besides, this approach obviously requires to possess a Matlab license.

Sweave: A comprehensive and portable solution. The first approach we are going to illustrate in this article is portable since it is based on free software, R[9] (R Development Core Team, 2010; Ihaka and Gentleman, 1996) and its user contributed packages[10] for data analysis, and LaTeX or HTML for typesetting. These software are available for every operating system likely to be found in neurobiological laboratories. R is a general data analysis software whose popularity grows every day and is intensively used by statisticians and sometimes by neurophysiologists (Tabelow et al., 2006; Wallstrom et al., 2007; Pouzat and Chaffiol, 2009; Pippow et al., 2009; Joucla et al., 2010). R has a special function called Sweave (Leisch, 2002a,b, 2003; Rossini and Leisch, 2003) to process specific text files where the text of a document is mixed with the code producing the analysis (calculations, figures, tables) presented by the document. The processed file is a new text document (in LaTeX or HTML) in which the text of the original have been copied verbatim and where the code has been replaced by the results of its execution. Sweave stems from the idea of "Literate Programming" (Knuth, 1984a) consisting in mixing a computer code with its description. The goal is to make the code easily readable and *understandable* by a human being rather than by a compiler. The human readable text file with code and description can be preprocessed to generate :

- a file that the compiler can "understand", executing the computations and generating figures
- a file that the TeX processor can "understand"giving a printable documentation as its output.

Sweave's users have to type their texts in TeX, LaTeX or HTML, but a user contributed package odfWeave (Kuhn, 2010) allows users to type the text parts with OpenOffice. Examples of both Sweave and odfWeave will be given in the sequel.

Emacs and org mode: A very versatile solution. We are well aware that the vast majority of our readers is unlikely to give up its usual data analysis software and switch to R just to make its analysis reproducible. We will therefore also detail in this article a solution which appeared recently: the org-mode[11] of Emacs[12]. Emacs (Stallman, 1981) is an extremely powerful text editor. It is open source and runs on nearly every operating system. Emacs has many modes, specific for the edition of text files in different "languages": a C mode to edit C codes, an html mode to edit web pages, several TeX and LaTeX modes to edit files in these languages, etc. Since Emacs is extensible and customizable, its users have extended it in bewildering number of directions over its more than 30 years of existence. We are going to use one of these modes, org-mode, which allows users to type simple texts using ASCII or UTF8 encoding and that can output files in HTML, PDF, DocBook, LaTeX, etc. In other words you can type a very decent LaTeX document with org-mode without having to know the LaTeX syntax. Thanks to the Babel[13] extension of org-mode (Schulte and Davison, 2011), users can also mix text with code, exactly like with Sweave. This becomes therefore a tool for reproducible analysis.

[8] http://www-stat.stanford.edu/~wavelab/
[9] http://www.r-project.org
[10] http://cran.at.r-project.org
[11] http://orgmode.org/.
[12] http://www.gnu.org/software/emacs/
[13] http://orgmode.org/worg/org-contrib/babel/index.html.

Furthermore, users do not have to restrict themselves to codes written in `R`, they can also use `Matlab`, `Octave`, `Python`, etc (33 languages supported as of `org-mode` 7.7). Even more interestingly, they can use different scripting languages in the same document.

3. A toy example

We illustrate the reproducible analysis approach on a simple example of Local Field Potentials (LFPs) detection. The end-product paragraph in a published paper would roughly look like our next sub-section.

3.1. Paper version

Experimental methods. Experiments were performed on mouse embryonic medulla-spinal cord preparations that were isolated at embryonic (E) day E12 and maintained in culture on 4×15 Micro Electrodes Arrays (MEAs). After two days in vitro, neuronal activity was recorded at the junction between the medulla and the spinal cord. Data were acquired at 10 kHz, off-line low-pass-filtered at 200 Hz and 20-times downsampled. The activity was characterized by slow (several tens of μs) LFPs that were simultaneously recorded on a line of 4 electrodes with an inter electrode interval of 250 μm.

Events detection. We want to estimate activity latencies with a simple thresholding method. To this end, we use the information redundancy between the signals recorded on each electrode. To enhance the signal-to-noise ratio, the detection was done on the time derivative of the raw data. An LFP was detected when the signal exceeded 4 times the standard deviation of the noise on at least 3 of the 4 channels. The resulting detection is presented on Fig. 1.

A reader of the paper, or the analyst himself coming back to the study months later, may ask the following questions:

- To what extent taking the derivative of the signal does increase the S/N, as the authors claim? Can we have a look at the raw data, since only the time derivative is shown in the paper?
- How sensitive is the detection to the choice of the threshold?
- How sensitive is the detection to the choice of the number of required channels?
- Can I easily test this method on my own data set?
- How would my own method for LFPs detection perform on that data set?

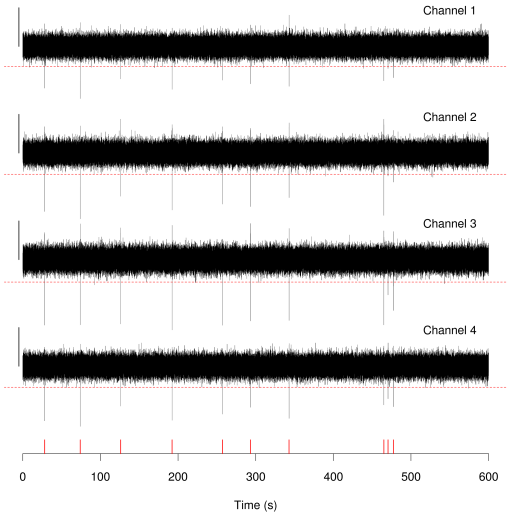

Figure 1: **Detection of LFP activity on the first time-derivative of the data.** The same scale applies to all channels (vertical bar: 200 μV / ms). Channel specific detection threshold shown in dashed red. Detected events are shown as vertical red bar at bottom.

Those kind of issues can be dealt with if, together with the paper, authors provide a document in a "reproducible" format describing precisely the analysis/processing conducted and allowing to reproduce the figures, tables and more generally computations of the paper. As we mentioned earlier, several options are available to produce such a document, and we're going to illustrate the idea with two of them : `Sweave` and `org-mode`.

3.2. Sweave version

Sweave relies on LaTeX or `HTML` for editing the text and processes `R` code.

LaTeX. Like `HTML`, LaTeX relies on markups: the source file does not look exactly like its output; for instance a web browser automatically interprets something like `<i>Warning!</i>` and shows *Warning!*. In the same way the sequence of characters `\textit{Warning!}` in a LaTeX file will appear like *Warning!* in the final `PDF` document. This might seem annoying to people used to word processing software but it has two immediate advantages: you know *exactly* where your formatting commands start and end – how many times did you lose patience using a word processor because you did not want anymore to have the next word you typed to be in italic like the previous one? – and the source file you are working with is a pure text (ASCII or UTF8) file –

meaning it is easy to store, can be directly integrated to your e-mails and is readily exchanged between different operating systems. LaTeX is unsurpassed for writing equations, for its typographic quality[14] and splits the content of a text (its logical structure) from its layout (the fonts used, the margin sizes, etc). The motivation of this last feature is that the user / writer should focus on *content* and *logical structure* and let LaTeX do the formatting work. LaTeX is free software program so you can give it a try without ruining yourself. We recommend the `TeX Live` distribution[15] but many other ones are also available. *The Not So Short Introduction to LaTeX2e* (Oetiker et al., 2011) is an excellent starting point.

R. "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."[16] It is open source software released and distributed by the `R Foundation`. "R is being developed for the Unix-like, Windows and Mac families of operating systems"[17]. `R` can run directly, through editors like `emacs` – the best long term solution in our opinion – or through sophisticated graphical interfaces like `RStudio`[18] (*very* useful for beginners). Readers eager to learn `R` should take a couple of days to go through the lecture notes of Lumley (2006) and through the superb lectures of Ross Ihaka[19]. Many excellent books have been published showing how to use `R` to perform specific tasks. A good general reference is Adler (2009).

Sweave. A Sweave file (conventionally ending with a `.Rnw` or a `.Snw` extension) looks like a regular LaTeX or `HTML` file as shown on Listing 1[20], except that `R` *code chunks* are included. Code chunks start with

```
<<name,option1=...,option2=...,...>>=
```

and end with

```
@
```

both on a single line. Here `name` is a label that can be used to refer to this specific code chunk somewhere else in the document. The first code chunk on Listing 1 (line 10) starts with: `<<load-data>>` (no options are specified). Various options allow to control the way the code chunk is interpreted. For example, the option `eval=FALSE` in the `get-doc-on-readBin` (line 18) chunk tells Sweave to simply print the code without evaluating it. Similarly, the option `fig=TRUE` can be used to display the output of plotting commands as a figure in the final document. It is clear that by giving access to the `.Rnw` file, one allows the reader / user to reproduce or modify an analysis with the corresponding figures. Listing 1 shows the whole `.Rnw` file with the commands required to download the data, compute the time derivatives, detect the events and generate Fig. 1. The first page of the `PDF` file, `LFPdetection.pdf`, obtained after processing `LFPdetection.Rnw` with `R` (see Appendix A.2 for details) and then processing the resulting `LFPdetection.tex` with LaTeX is shown in Fig. 2. Finally, when two "programming" languages like `R` and LaTeX have to be used together it is extremely useful to have an editor providing facilities for both. The two interfaces mentioned in the previous paragraph, `emacs` and `RStudio`, provide such facilities. They do in fact much more since they integrate the two in the sense that code chunks can be evaluated line by line or "en bloc" within a single interface (there is no need to switch back and forth between an `R` running program and a LaTeX editor).

odfWeave. Readers already familiar with `R` but not with LaTeX, or readers simply wanting to learn a single new software at a time can try the user contributed package `odfWeave`[21]. With this package `OpenOffice` is used instead of LaTeX or `HTML` for the text parts. The demarcation of the code chunks and their options is identical to the one used in Sweave as shown on Fig. 3. The processing of these `odfWeave` within `R` is slightly more complicated than the one of the `.Rnw` files as shown in Appendix A.3. These complications should nevertheless offset some of the reticence of readers interested by the concept of reproducible data analysis but not ready to embark into learning two sophisticated languages.

[14] See: `http://nitens.org/taraborelli/latex`.

[15] `http://www.tug.org/texlive/`

[16] `http://cran.r-project.org/doc/FAQ/R-FAQ.html`

[17] `R-FAQ`.

[18] `http://www.rstudio.org/`

[19] Statistical Computing (Undergraduate); Statistical Computing (Graduate); Information Visualisation These courses can be accessed through the following page: http://www.stat.auckland.ac.nz/~ihaka/?Teaching .

[20] This listing has been edited to fit the whole code into a single page. It is therefore harder to read than the original file we worked with (many `R` commands are put on a single line separated by ";"). The `R` code itself is also compact implying it will be difficult to follow for readers unfamiliar with `R`. The part which include the generated `PDF` figure into the final document is missing due to space constraints but all the computations are shown.

[21] http://cran.at.r-project.org/web/packages/odfWeave/index.html

```
\documentclass[a4paper,12pt,english]{article}
\usepackage{fullpage}
\begin{document}
\section{Loading data into \texttt{R}}
The data recorded from 4 electrodes result from a preprocessing briefly described in the main text and are
stored as signed integer coded on 4 Bytes. They must be mutiplied by 0.12715626 on channels 1 and 4 and by
0.01271439 on channels 2 and 3 to get voltage in $\mu$V. They are sampled at 500 Hz and 600 second are stored.
We can then read the data from our web repository and assign them to variable \texttt{Data\_raw} of our
\texttt{work space}:
<<load-data>>=
reposName <- "http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Data_folder/"
dN <- paste("SpinalCordMouseEmbryo_CH",1:4,".dat",sep="");fullN <- paste(reposName,dN,sep="")
nSamples <- 500*600; Data_raw <- sapply(fullN, readBin, n=nSamples,what="integer")
Data_raw <- t(t(Data_raw)*c(0.12715626,0.01271439,0.01271439,0.12715626))
@
For readers unfamiliar with \texttt{R}, the assignment operator ``<-'' can be replaced by the usual symbol
``='' . \texttt{R} users can always get the documentation of native and user contributed functions with:
<<get-doc-on-readBin, eval=FALSE>>=
?readBin
@
The time derivatives of the measurements are simply obtained using a difference equation whose precision is
$o(\delta^2)$:
\begin{displaymath} f'(x) = \frac{f(x+\delta) - f(x-\delta)}{2 \, \delta} \end{displaymath}
<<Data_derivative>>=
Data_derivative <- apply(Data_raw,2, function(x) c(0,diff(x,2)/2,0)*500/1000)
@
Here the unit of \texttt{Data\_derivative} is $\mu$V / ms.
\section{LFP detection}
We are going to detect minima on each channel whose amplitudes are below a \emph{user set} multiple of the
channel standard deviation. We start by computing this quantity for each of the two versions of the data we
might choose to work with, ``raw'' of ``derivative'':
<<SD>>=
SD_raw <- apply(Data_raw, 2, sd); SD_derivative <- apply(Data_derivative, 2, sd)
@
Here \texttt{SD\_raw} and \texttt{SD\_derivative} are \emph{vectors} with as many elements as
\texttt{Data\_raw} and \texttt{Data\_derivative} have columns, that is, as many elements as recording channels.
We are going to use a threshold of 4 times the standard deviation on each channel:
<<threshold-on-derivative>>=
factor <- 4
@
A inquiring reader could easily make another choice like using a threshold of 3.5:
<<threshold-on-raw, eval=FALSE>>=
factor <- 3.5
@
As explained in the main text we \emph{decided} to identify events as minima exceeding (in absolute value) a
threshold on 3 channels simultaneously. To this end we define a variable, \texttt{activeElecNumber}, which
contains our number of required active channels. The value of this variable can easily be changed:
<<activeElecNumber>>=
activeElecNumber <- 3
@
The detection can now proceed:
<<detect-LFPs>>=
Times <- (1:dim(Data_derivative)[1])/500
timeLFP <- Times[apply(t(t(Data_derivative)/SD_derivative) < -factor,1,sum) >= 3]
@
We decide moreover to keep only detected events which are more than 100 ms apart. When ``too close'' events are
found, the second one is discarded. This elimination is done recursively starting with the second event:
<<keep-far-apart-events>>=
timeLFP2 <- timeLFP; nbLFP <- length(timeLFP); last <- 1; new <- 2
while (new <= nbLFP) { tDiff <- timeLFP[new]-timeLFP2[last]
  if (tDiff >= 0.1) {last <- last+1;timeLFP2[last] <- timeLFP[new]};new <- new+1}
timeLFP <- timeLFP2[1:last];rm(timeLFP2)
@
We can now produce our summary Fig.~\ref{fig:detectionOnDerivativeData} with:
<<make-figure,fig=TRUE>>=
Data_derivativeN <- Data_derivative/diff(range(Data_derivative));Data_derivativeN.min <- min(Data_derivativeN)
Data_derivativeN <- Data_derivativeN-Data_derivativeN.min;Data_derivativeN <- t(t(Data_derivativeN)-c(0,1,2,3))
thresh <- -factor*SD_derivative/diff(range(Data_derivative))-Data_derivativeN.min - c(0,1,2,3)
lwr <- 0-Data_derivativeN.min-c(0,1,2,3);upr <- 2/diff(range(Data_derivative))-Data_derivativeN.min-c(0,1,2,3)
plot(0,0,type="n",xlab="Time (s)",ylab="",xlim=c(0,600),ylim=c(-3,1),axes=FALSE)
sapply(1:4, function(i) lines(Times,Data_derivativeN[,i],lwd=1))
sapply(1:4,function(i)text(550,2-i,paste("Channel",i)));abline(h=thresh,col="red",lty=2,lwd=3)
axis(1,at=(0:6)*100,lwd=3);rug(timeLFP,col="red",lwd=5);segments(-5,lwr,-5,upr,lwd=5)
@
\end{document}
```

Listing 1: LFPdetection.Rnw, code chunks have a pale yellow background, documentation chunks have a white one.

1 Loading data into R

The data recorded from 4 electrodes result from a preprocessing briefly described in the main text and are stored as signed integer coded on 4 Bytes. They must be mutiplied by 0.12715626 on channels 1 and 4 and by 0.01271439 on channels 2 and 3 to get voltage in μV. They are sampled at 500 Hz and 600 second are stored. We can then read the data from our web repository and assign them to variable `Data_raw` of our `work space`:

```
> reposName <- "http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Data_folder/"
> dN <- paste("SpinalCordMouseEmbryo_CH", 1:4, ".dat", sep = "")
> fullN <- paste(reposName, dN, sep = "")
> nSamples <- 500 * 600
> Data_raw <- sapply(fullN, readBin, n = nSamples, what = "integer")
> Data_raw <- t(t(Data_raw) * c(0.12715626, 0.01271439, 0.01271439,
+     0.12715626))
```

For readers unfamiliar with R, the assignment operator "<-" can be replaced by the usual symbol "=". R users can always get the documentation of native and user contributed functions with:

```
> `?`(readBin)
```

The time derivatives of the measurements are simply obtained using a difference equation whose precision is $o(\delta^2)$:

$$f'(x) = \frac{f(x+\delta) - f(x-\delta)}{2\,\delta}$$

```
> Data_derivative <- apply(Data_raw, 2, function(x) c(0, diff(x,
+     2)/2, 0) * 500/1000)
```

Here the unit of `Data_derivative` is μV / ms.

2 LFP detection

We are going to detect minima on each channel whose amplitudes are below a *user set* multiple of the channel standard deviation. We start by computing this quantity for each of the two versions of the data we might choose to work with, "raw" of "derivative":

```
> SD_raw <- apply(Data_raw, 2, sd)
> SD_derivative <- apply(Data_derivative, 2, sd)
```

Here `SD_raw` and `SD_derivative` are *vectors* with as many elements as `Data_raw` and `Data_derivative` have columns, that is, as many elements as recording channels.

We are going to use a threshold of 4 times the standard deviation on each channel:

```
> factor <- 4
```

A inquiring reader could easily make another choice like using a threshold of 3.5:

Figure 2: The first page of the PDF file obtained after processing `LFPdetection.Rnw` (Listing 1 – page 1 corresponds to lines 1 to 41) with R and LaTeX.

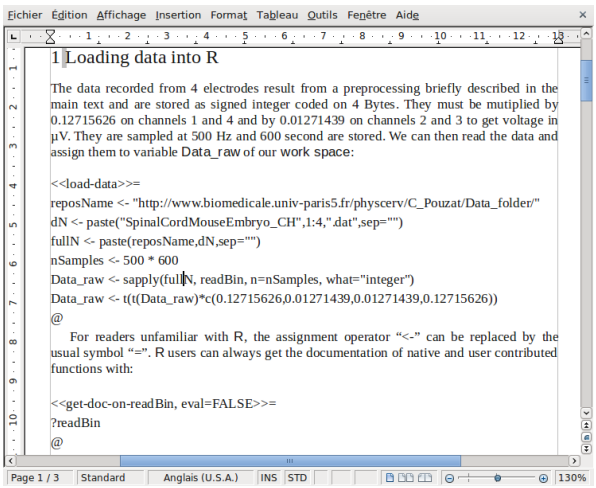

Figure 3: The beginning of the `odfWeave` covering the first two code chunks shown on Listing 1.

cacheSweave. Analysis or simulations performed for a paper or whose results get simply archived in a labbook can be quite long. It becomes therefore interesting, when working with tools like `Sweave`, to be able to store intermediate results –from the code chunks having a long run time. In this way users do not have to recompute everything every time they want to generate a `PDF` document from their `.Rnw` file. Saving intermediate results must clearly be done with care since a code modification in chunk k could change the result of chunk $k+j$ (for j positive) implying that chunk $k+j$ *should also be (re)evaluated even if its result was stored.* The user contributed package `cacheSweave` (Peng, 2011) does precisely that: it saves on disk the results of the successive code chunks *while keeping track of their dependencies* and re-evaluates, following the modification of a single code chunk, as many chunks as necessary to keep the whole analysis/simulation consistent. It is also an obviously useful package when running an analysis in batch mode in a crash prone environment.

3.3. Org-mode version

`org-mode` is a mode of the `emacs` editor. `org-mode` facilitates the implementation of reproducible research in two ways:

- The syntax of the text part is considerably simplified compared to `HTLM` or LaTeX but perfectly valid source files for both of these languages can be generated directly from the same `org-mode` source.
- 28 programming language in addition to `R` are supported including `Matlab`, `Octave` (open source clone of the former) and `Python`.

Emacs. This is the corner stone software of the `Free Software Foundation` meaning that it is open source and that it runs on every operating system except some very exotic ones. This editor is a world in itself but one does not have to know it in depth in order to start using it with `org-mode`. Going once through the `Guided Tour of Emacs`[22] should be enough for the beginner.

org-mode. `Org-mode` files (conventionally ending with an `.org` extension) are quite similar to *Sweave* files since they are made of textual description and *code blocks* (the code chunks of org-mode) as shown on Listing 2. Comparing with Listing 1 we see that the beginning of a new section with its title

```
\section{Loading data into \texttt{R}}
```

in LaTeX (Listing 1, line 4) becomes (Listing 2, line 3)

```
* Downloading data with =Python= and...
```

So section headings in `org-mode` are introduced by "`* `" and get properly converted into LaTeX or `HTML` (or `DocBook`) depending on the user choice. Sub-section heading are introduced by "`** `" and so on. Formatting commands like the "`\texttt{R}`" (typewriter text) of LaTeX become "`=R=`" – using "`/R/`" would produce an italic, while "`*R*`" would produce bold type face. Greek letters are simply entered as "`\mu`" for μ and hyperlink, like the one to the python web site (Listing 2, line 7), are entered as

```
[[http://www.python.org/][python]]
```

but they get immediately reformatted by `emacs` to appear colored and underlined. In short, `org-mode` brings most of the structuring and maths editing capabilities of LaTeX with minimal hassle and it provides output flexibility while preserving the text file nature of the source file. Org-mode files can be opened and edited with any text editor, a feature that can be used with profit for collaborative projects, where some participants do not know how to use `emacs` but still want to be able to modify the text (or the code part).

Code blocks. Source code can be included in `org-mode` files in the same spirit, but with a different syntax, that the code chunks of `Sweave`. The code blocks of `org-mode` admit more optional arguments since they can fundamentally do "more things" than their `Sweave` counterpart. First of all the language used in the code block has to be specified like in

[22] http://www.gnu.org/software/emacs/tour/

```
#+STYLE: <link rel="stylesheet" href="http://orgmode.org/org.css" type="text/css" />
* Downloading data with =Python= and loading them into =octave=
The data recorded from 4 electrodes result from a preprocessing briefly described in the main text and are
stored as signed integer coded on 4 Bytes. They must be mutiplied by 0.12715626 on channels 1 and 4 and by
0.01271439 on channels 2 and 3 to get voltage in \mu V. They are sampled at 500 Hz and 600 second are stored.
We will here then dowload the data from our web repository using [[http://www.python.org/][python]]. To this
end we start by defining a =list= containing the file namse under which we want to store the data on our
hard-drive:
#+srcname: dN
#+begin_src python :session *Python* :exports code :results value pp
  dN=["SpinalCordMouseEmbryo_CH"+str(i)+".dat" for i in range(1,5)]
  dN
#+end_src

#+results: dN
: ['SpinalCordMouseEmbryo_CH1.dat',
:  'SpinalCordMouseEmbryo_CH2.dat',
:  'SpinalCordMouseEmbryo_CH3.dat',
:  'SpinalCordMouseEmbryo_CH4.dat']
After loading the =urllib= library we can proceed and download the data:
#+srcname: download-data
#+begin_src python :session *Python* :exports code :results silent
  import urllib
  reposName = "http://www.biomedicale.univ-paris5.fr/physcerv/C_Pouzat/Data_folder/"
  for n in dN:urllib.urlretrieve(reposName+n,n)
#+end_src
We then load the data in an [[http://www.gnu.org/software/octave/][octave]] session (that's the occasion to
make use of the
[[http://orgmode.org/worg/org-contrib/babel/intro.html#meta-programming-language][meta-programming language]]
capabilities of =org-mode= -- a variable created by =python=, =dN=, is going to be used directly in =octave=):
#+srcname: load-to-octave
#+begin_src octave :session *octave* :exports code :results silent :var fN=dN
  nSamples = 500 * 600;
  Data_raw = zeros(nSamples,4);
  i2v = [0.12715626 0.01271439 0.01271439 0.12715626];
  for i=1:4
    fid=fopen(fN(i,:),'r');
    [C,n]=fread(fid,nSamples,'int32');
    fclose(fid);
    Data_raw(:,i)=C*i2v(i);
  end
#+end_src
The time derivatives of the measurements are simply obtained using a difference equation whose precision is
o(\delta^{2}): \[ f'(x) = (f(x+\delta) - f(x-\delta))/(2 \delta) \]
#+srcname: Data-derivative
#+begin_src octave :session *octave* :exports code :results silent
  Data_derivative = zeros(nSamples,4);
  for i=1:4
    Data_derivative(2:(nSamples-1),i)=(Data_raw(3:nSamples,i)-Data_raw(1:(nSamples-2),i))*500/2/1000;
  end
#+end_src
Here the unit of =Data_derivative= is \mu V / ms.

* LFP detection
We are going to detect minima on each channel whose amplitudes are below a /user set/ multiple of the channel
standard deviation. We start by computing this quantity for each of the two versions of the data we might
choose to work with, ``raw'' of ``derivative'':
#+srcname: SD
#+begin_src octave :session *octave* :exports code :results silent
SD_raw = std(Data_raw);
SD_derivative = std(Data_derivative);
#+end_src
```

Listing 2: First part of LFPdetection.org, code blocks have a pale yellow background.

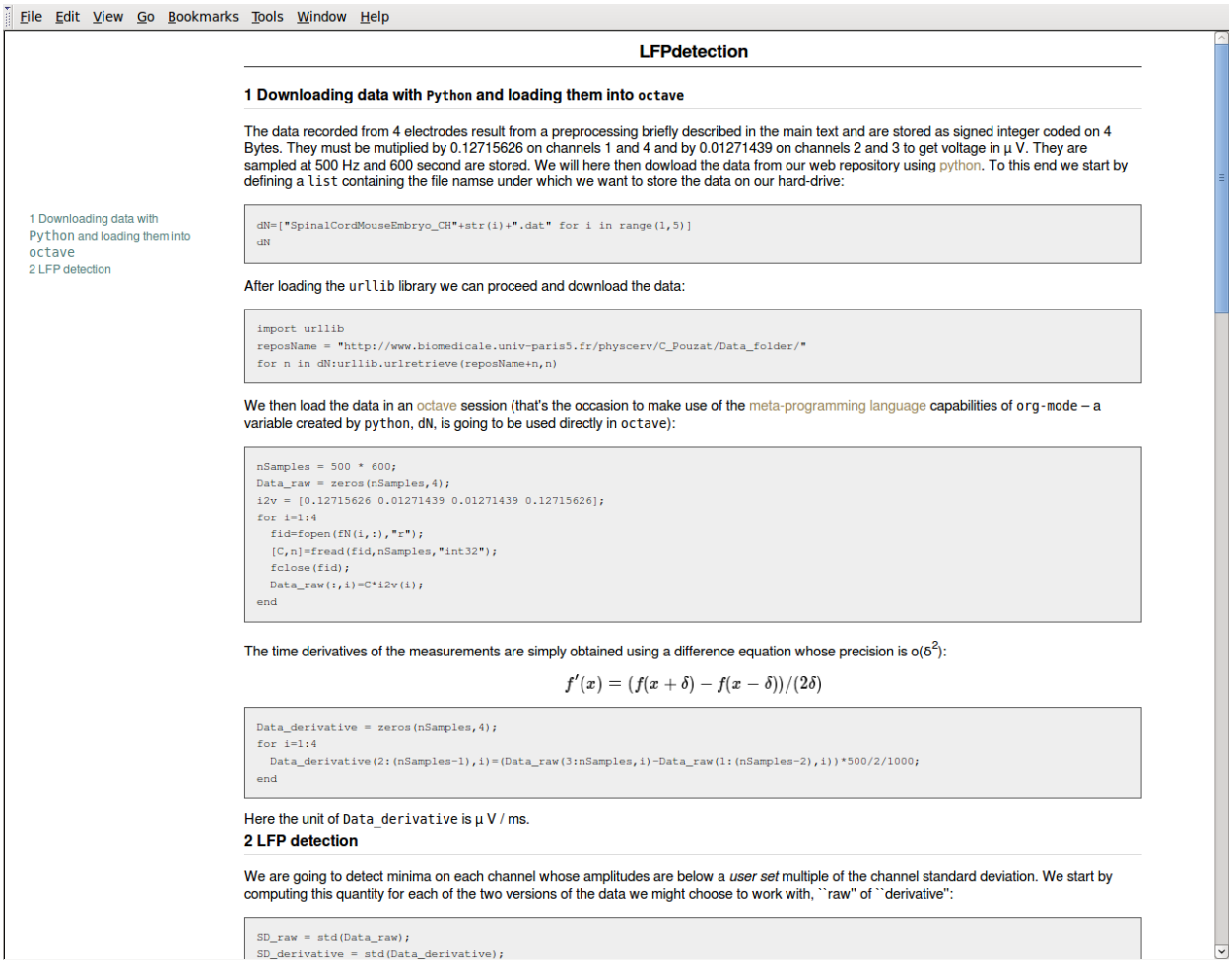

Figure 4: The first third of the HTML output of LFPdetection.org whose listing is shown on Listing 2.

`#+begin_src python`

at the beginning of the first code block in Listing 2 (line 11) where the `Python` language is used. This is the required minimum to open a code block which is closed by (Listing 2, line 14):

`#+end_src`

But optional arguments can also be specified like (Listing 2, line 11):

`:session *Python* :exports code`

followed by:

`:results value pp`

Here "`:session *Python*`" means that the evaluation of the code block will be performed in a `Python` session within `emacs` which will outlive the evaluation itself and that will appear in an `emacs buffer` called "`*Python*`". This session will be accessible independently of the `org-mode` file. Variables created by the evaluation, like `dN` in this code block, can be used and referred to later on in another `Python` code block with the same `:session` name (like the second code block in Listing 2, lines 24 - 28). Optional argument "`:exports code`" controls what is exported when the LaTeX or `HTML` output is generated. Here we just want the code part and not the result of the evaluation to be exported. The output produced can be seen on Fig. 4. Argument "`:results value pp`" controls what happens to the `org-mode` file when the code block is evaluated. We want here the `value` of the last expression evaluated by the code block (`dN` on the second line) and we want it "pretty printed" (`pp`). Listing 2 shows this value below the line starting with "`#+results: dN`" (lines 16 - 20). The reader can see that this value appears only in the `org-mode` file (Listing 2) and not in its `HTML` output (Fig. 4). We could have exported both code and results in our `HTML` file by setting "`:exports both`". Remark that we have set a `name` for this code block with "`#+srcname: dN`". This name is used again in the third code block of Listing 2 starting with (line 35):

`#+begin_src octave :session *octave*`

and continuing with:

`:exports code :results silent :var fN=dN`

This third code block uses a different language, `Octave`[23], something we could not do with `Sweave`. But `org-mode` allows us to do something pretty remarkable here: we are passing to an octave code block, as a variable value with "`:var fN=dN`", the result of *another* code block written *in a different language*: an instance of *meta-programming*. This means that within the `Octave` session the variable `fN` will be given the output value[24] of the code block named `dN` *regardless of the language in which the latter was written*.

Saving intermediate results. `Babel` proposes a feature similar to what the `cacheSweave` package of `R` brings to `Sweave`: the possibility to "store" intermediate results so that the code blocks generating them are not re-evaluated every time a `PDF` or `HTML` output is generated. This can be done by setting the optional argument `:cache` to `yes` (the default is `no`). See Schulte and Davison (2011) as well as the `org-mode` manual for details.

Thanks to `org-mode`, `Matlab` and `Python` users can also easily implement a reproducible data analysis approach, if they are ready to make learn the basics of `emacs`. Being more interactive than `Sweave` (because of the difference between what is exported and what is written in the `org-mode` file upon code block evaluation) `org-mode` files can be used as a true lab book. The obvious benefit is that scripts and notes are stored in a single file.

4. Conclusions

We have advocated here the "reproducible data analysis / reproducible research" approach, illustrating, with a toy-example, several dedicated tools facilitating its implementation. Reproducible research, and more specifically the creation of files mixing text and code blocks should bring four major benefits:

- Analysis reproducibility.
- Analysis transparency.
- Conservation of results and progress on a daily basis.
- Transmission of the accumulated knowledge inside a group *as well as within a scientific community*.

[23] If we had `Matlab` we could simply replace `Octave` by `matlab` here, change nothing else, and get the same result.

[24] One should nevertheless use this functionality being aware that output values, when they exist, are stored as `org-table` objects, that is, floats are converted to ASCII. That can generate severe memory loads and computation slow downs with output vector or matrix containing of the order of 10^5 elements.

Analysis reproducibility. The first point should seem totally trivial to an outsider; after all reproducibility is definitely one of the most basic tenets of the (natural) sciences; so how could we pretend doing science if our work is not reproducible? Sadly empirical evidences show that the gap between reality and principles can be significant (Dewald et al., 1986; McCullough et al., 2006; Vandewalle et al., 2009). It is also perhaps worth clarifying our choice of locution, "reproducible data analysis" in addition to the now more common one, "reproducible research". The latter has emerged in fields like statistics and signal processing where the data are (more or less) taken for granted. But when we talk about reproducibility in neuroscience we cover both the reproducibility of the data *and* the reproducibility of their analysis. We have discussed *only* the latter here. With that in mind it could in fact be better to come back to the original denomination of economists: *replication*. Still, an interesting "side effect" resulting from the approach we have been advocating for –requiring an open access to the raw data– is that experimentalists, having access to others' raw data, can compare with their own. This should allow the community to spot more easily the most obvious problems concerning data reproducibility.

Analysis transparency. The reproducible data analysis approach obviously facilitates the spread and the selection of efficient methods. It should greatly improve the trustworthiness of codes. Indeed as soon as fairly complex codes are developed for data analysis, like for anything else, *bugs are present*. As the open source movement has clearly demonstrated over the years, letting other people see and contribute to code development is a reliable way to "good" software. And a proper documentation, an integral part of transparency in our view, greases the wheels.

Results conservation. Thanks to the tools available on multi-operating systems for several programming language, reproducible data analysis is becoming simpler to practice on a daily basis. Researchers can therefore use it as a tool for writing their lab books. They can not only keep a written trace of what they tried out, but save in the same document their ideas, comments, codes and settings, not to mention the link to the data. A more systematic and comprehensive approach of analysis archiving should reduce mistakes, simplify the routine analysis of many data sets and allow a straightforward investigation of the effect of parameter settings. Moreover, since those files will be containing an exhaustive information, analysis will be easier to reuse, and easier to reuse faster.

Information transmission. Finally, generalizing the reproducible research approach within labs, especially in teams with high turn-over, provides or at least reduces the loss of accumulated knowledge. It is also a perfect medium for transmitting information from one researcher to the other, it facilitates team work and collaborations.

Software version dependence. As any computer user knows, software and hardware are evolving fast resulting in stressful experiences where a document that could be easily opened with the last version of a software cannot be opened anymore with the new version. Even with well designed software one can have bad surprises like a change in the numerical value of a calculation following a library or compiler update (Belding, 2000). This is obviously an issue for the approach advocated here, we could get different results for the "same" analysis after a software update. That implies that reproducible data analysis practitioners should keep rigorous record of the software *version* they have been using[25]. This also encourages these practitioners to use software whose license does not expire and whose "old" versions remain available for a long time; using open source software can help a lot here. Another solution is to build an image of the whole environment one has been using–including the operating system and *all* the dedicated software; image that can be later run directly on a suitable virtual machine. This is the solution recommended by two challengers of Elsevier's "Executable Paper Grand Challenge" (Van Gorp and Mazanek, 2011; Brammer et al., 2011).

Raw data and code servers. Independently of the good will of scientists to share as thoroughly as possible their "production": data, code, etc; the problem of the hardware framework required to implement data sharing on a large scale will have to be addressed. Experimentalists recording for hours from tens of extracellular electrodes at 15 to 20 kHz do not necessarily have the server infrastructure and the technical know-how required to make their data available to all after publication. The situation is luckily evolving fast and initiatives like the "International Neuroinformatics Coordinating Facility"[26] are now supporting data storage services for the neurosciences[27].

[25] R's `sessionInfo` function is very useful in that perspective since it returns the version of R and of all the packages used in a session.

[26] `http://www.incf.org/`.

[27] `http://datasharing.incf.org/ep/Resources`.

Copyright issues. If or, let us be optimistic, when reproducible data analysis practices generalize copyright issues will appear: to what extend could the distribution of the code reproducing a published figure infringe the copyright protecting the figure? The issues on data copyright and use could even be more problematic. We have not touched these questions here since they extensively discussed in Stodden (2009a,b).

Software. We have chosen to illustrate two types of tools to implement the reproducible data analysis approach: the "Sweave family" and the org mode of `emacs`. Hopefully our toy example has convinced our readers that one can reasonably easily go from principle to practice. We also give some "serious" examples on our website[28]. Our experience with both of these tools is that it is really possible to use them systematically when we do data analysis *at every stage of a manuscript preparation, starting with our lab books*. Although we illustrated the use of some of the open-source tools available the reader should not conclude that the ones we did not illustrate are "bad", we just have little or no experience with them and our space is limited. Proprietary software also exists like `Inference for R`[29] – for using `R` with `Microsoft Office` – but we don't have any experience with them. Open-source tools like `Dexy`[30] and `Sumatra`[31] are clearly very promising and have capabilities similar to Sweave and org-mode. `Mathematica`[32] and `Sage`[33] (an open source, `Python` based, environment running "on top" of many open source mathematical software programs like `R` and `Octave`) both include "active notebooks" that can be used for reproducible data analysis. We have moreover focused on "literate programming derived" tools but alternative, "workflow based", solutions exist like `Vis Trails`[34] and `Taverna`[35]. With these approaches, an analysis is constructed and described graphically. The advantage is an easy access for people without programming background, the drawback is, like with any GUI based system, an inefficiency as soon as relatively complex tasks have to be performed[36]. Once the workflow has been specified, it can be shared with other researchers, making the analysis reproducible. In bioinformatics dedicated web sites and servers for workflows sharing are already maintained[37].

To conclude: reproducible data analysis is a challenge but a reachable one. The tools are there and we, like others, have been using them for a few years now (Delescluse, 2005). Give it a try, we will all win at the end!

Appendix A. Reproducing the toy example

Appendix A.1. Getting the source files

The source files of the three versions of our toy example are:

- `LFPdetection.Rnw`: identical to Listing 1 with few extra lines at the end ensuring a proper inclusion of the generated figure in the final `PDF`. This file can be edited with any text editor.
- `LFPdetection_in.odt`: the beginning of this file is shown on Fig. 3, it can be edited with `OpenOffice`.
- `LFPdetection.org`: the beginning of this file is shown on Listing 2, it can be edited with any text editor but is best edited with `emacs`.

These files can be downloaded from: `http://www.biomedicale.univ-paris5.fr/-physcerv/C_Pouzat/ReproducibleDataAnalysis/`.

Appendix A.2. Rnw

To generate `LFPdetection.pdf` from `LFPdetection.Rnw`, start `R` (assuming you have already installed it) from the folder where you have downloaded `LFPdetection.Rnw`. Type

```
> Sweave("LFPdetection.Rnw")
```

Once the command has been evaluated, process the resulting `LFPdetection.tex` in your folder like any LaTeX file (Oetiker et al., 2011) to get the `PDF`.

[28] http://www.biomedicale.univ-paris5.fr/physcerv/-C_Pouzat/ReproducibleDataAnalysis/ReproducibleDataAnalysis.html

[29] http://inferenceforr.com/default.aspx

[30] `http://www.dexy.it/`

[31] `http://neuralensemble.org/trac/sumatra/wiki`

[32] `http://www.wolfram.com/mathematica/`

[33] `www.sagemath.org`

[34] `http://www.vistrails.org/index.php/Main_Page`

[35] `http://www.taverna.org.uk/`

[36] There is only a *very* limited amount of actions or concepts one can unambiguously specify with boxes and arrows–this paper is written with words not with boxes and arrows–so users of workflows or GUIs, end up spending *a lot of time* pointing at- and clicking on- pop-up menus allowing them to specify the required parameters. On a long term, we think that users are better off writing code directly.

[37] `http://www.myexperiment.org/`

Appendix A.3. odfWeave

The package odfWeave allows the processing of an Open Office Writer document mixing text and sweave chunks. This package is not part of R by default which means that after installing R you will have to install the package with:

```
> install.packages("odfWeave")
```

Then start R from the folder where you have downloaded LFPdetection_in.odt. The next step is to load the odfWeave library with:

```
> library(odfWeave)
```

Contrary to Sweave, the document settings in odfWeave, such as page dimensions, font settings, figures or tables margins are all defined in a list of options. It is not recommended to change the settings modifying directly this list since the default settings would be lost for this session. The pre-existing styles can be accessed calling the function getStyleDefs and copied in a new variable, that we call here "myNewStyle".

```
> myNewStyle <- getStyleDefs()
```

Customisations of styles will only be made on "myNewStyle" with:

```
> myNewStyle$ttRed$fontColor = "#000000"
> myNewStyle$RlandscapePage$pageWidth <- "8.3in"
> myNewStyle$RlandscapePage$pageHeight <- "11.7in"
> myNewStyle$RlandscapePage$marginLeft <- "1in"
> myNewStyle$RlandscapePage$marginRight <- "1in"
> myNewStyle$RlandscapePage$marginTop <- "1.6in"
> myNewStyle$RlandscapePage$marginBottom <- "1.6in"
```

where the R commands code for changing the font colour of the displayed code blocks from red to black, and defining new page margins. New styles assignments can be saved and loaded calling the function setStyleDefs.

```
> setStyleDefs(myNewStyle)
```

The image format and sizes are specified using getImageDefs and setImageDefs through a similar process.

```
> imageDefs <- getImageDefs()
> imageDefs$dispWidth <- 4
> imageDefs$dispHeight <- 4
> setImageDefs(imageDefs)
```

Finally, the input file is compiled calling to odfWeave function, with the input file name as first argument, and the output file name as second argument.

```
> odfWeave("LFPdetection_in.odt",
           "LFPdetection_out.odt")
```

Appendix A.4. Org

A note on org-mode. New emacs releases come about once a year while org-mode evolves faster with two or more releases per year. This means that although org-mode is included in emacs it is unlikely that the org-mode of your emacs is the most recent one. So download the last stable version *following the instructions for download and installation* given on the org-mode web site[38].

Required software. This version of the toy example uses Python[39] and Octave[40]. You will therefore have to make these two software available on your computer in order to regenerate this version of the toy example.

Toy example regeneration. After downloading LFPdetection.org and open it in emacs. Press the "key-chord": C-c C-e (where C-c means press the control key and the C key together), in the list that appears select the output format you want: h for HTML, d to generate a PDF and view it immediately. After that emacs will start evaluating the code blocks one after the other, asking you every time to confirm that you want to evaluate them, so answer "yes" every time. After a few "yes" your output file will be ready.

Some tricks. The confirmation asked by emacs upon each code block evaluation can be suppressed by setting variable org-confirm-babel-evaluate to nil. This can be done by typing in the *scratch* buffer of emacs:

```
(setq org-confirm-babel-evaluate nil)
```

before evaluating this expression by placing your cursor just after the closing parenthesis and pressing the key chord: C-x C-e. To learn more about the variables controlling the default working of org-mode, read the manual.

Acknowledgments

We thank Jonathan Bradley, Alain Marty, Avner Bar-Hen and Eric Schulte for comments on the manuscript; Gaute Einevoll and Hans Plesser for comments, discussion and for pointing out Sumatra to us; and two anonymous reviewers for constructive comments and additional references/software suggestions which greatly improved the manuscript's scope.

[38] http://orgmode.org/
[39] http://www.python.org/
[40] http://www.gnu.org/software/octave/

References

Adler, J., 2009. R IN A NUTSHELL. O'REILLY.

Anderson, R.G., Dewald, W.G., 1994. Replication and Scientific Standards in Economics a Decade Later: The Impact of the JMCB Project. Working Paper 1994-007C. Federal Reserve Bank of St. Louis. Available at: http://research.stlouisfed.org/wp/more/1994-007/.

Baggerly, K., 2010. Disclose all data in publications. Nature 467, 401–401.

Belding, T.C., 2000. Numerical replication of computer simulations: Some pitfalls and how to avoid them. Eprint arXiv:nlin/0001057.

Brammer, G.R., Crosby, R.W., Matthews, S.J., Williams, T.L., 2011. Paper mâché: Creating dynamic reproducible science. Procedia Computer Science 4, 658 – 667. Proceedings of the International Conference on Computational Science, ICCS 2011.

Buckheit, J.B., Donoho, D.L., 1995. Wavelets and Statistics. Springer. chapter Wavelab and Reproducible Research. Preprint available at: http://www-stat.stanford.edu/~wavelab/Wavelab_850/wavelab.pdf.

Claerbout, J., Karrenbach, M., 1992. Electronic documents give reproducible research a new meaning, in: Proceedings of the 62nd Annual Meeting of the Society of Exploration Geophysics, pp. 601–604. Available at: http://sepwww.stanford.edu/doku.php?id=sep:research:reproducible:seg92.

Delescluse, M., 2005. Une approche Monte Carlo par Chaînes de Markov pour la classification des potentiels d'action. Application à l'étude des corrélations d'activité des cellules de Purkinje. Ph.D. thesis. Université Pierre et Marie Curie. Available at: http://tel.archives-ouvertes.fr/tel-00011123/fr/.

Dewald, W.G., Thursby, J.G., Anderson, R.G., 1986. Replication in empirical economics: The journal of money, credit, and banking project. American Economic Review 76, 587–603.

Diggle, P.J., Zeger, S.L., 2010. Editorial. Biostatistics 11, 375–375.

Donoho, D.L., Maleki, A., Rahman, I.U., Shahram, M., Stodden, V., 2009. Reproducible research in computational harmonic analysis. Computing in Science and Engineering 11, 8–18. Preprint available at: http://www-stat.stanford.edu/~donoho/Reports/2008/15YrsReproResch-20080426.pdf.

Elsevier, 2011. Ethical guidelines for journal publication. web.

ESF, 2007. Shared responsibilities in sharing research data: Policies and partnerships. reports of an esf-dfg workshop, 21 september 2007. Web. Available at: www.dfg.de/download/pdf/.../sharing_research_data_esf_dfg_0709.pdf.

Fomel, S., Hennenfent, G., 2007. Reproducible computational experiments using scons, in: Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing, p. IV1257–IV1260.

Gentleman, R., Temple Lang, D., 2007. Statistical Analyses and Reproducible Research. Journal of Computational and Graphical Statistics 16, 1–23. http://pubs.amstat.org/doi/pdf/10.1198/106186007X178663.

Ihaka, R., Gentleman, R., 1996. R: A Language for Data Analysis and Graphics. Journal of Graphical and Computational Statistics 5, 299–314.

Joucla, S., Pippow, A., Kloppenburg, P., Pouzat, C., 2010. Quantitative estimation of calcium dynamics from ratiometric measurements: A direct, non-ratioing, method. Journal of Neurophysiology 103, 1130–1144.

Knuth, D.E., 1984a. Literate programming. The Computer Journal 27, 97–111. Reprint available at: http://www.literateprogramming.com/knuthweb.pdf.

Knuth, D.E., 1984b. The TeXbook. Addison-Wesley, Reading, Massachusetts.

Koenker, R., Zeileis, A., 2007. Reproducible Econometric Research. A Critical Review of the State of the Art. Research Report Series / Department of Statistics and Mathematics 60. Department of Statistics and Mathematics, WU Vienna University of Economics and Business, Vienna. Available at: http://epub.wu.ac.at/638/.

Kuhn, M., 2010. odfWeave: Sweave processing of Open Document Format (ODF) files. R package version 0.7.17.

Lamport, L., 1986. LaTeX: A Document Preparation System. Addison-Wesley, Reading, Massachusetts.

Leisch, F., 2002a. Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis, in: Härdle, W., Rönz, B. (Eds.), Compstat 2002 — Proceedings in Computational Statistics, Physica Verlag, Heidelberg. pp. 575–580. Available at: http://www.statistik.uni-muenchen.de/~leisch/Sweave/.

Leisch, F., 2002b. Sweave, Part I: Mixing R and LaTeX. R News 2, 28–31.

Leisch, F., 2003. Sweave, Part II: Package Vignettes. R News 3, 21–24.

Lumley, T., 2006. R Fundamentals and Programming Techniques. Available at: http://faculty.washington.edu/tlumley/Rcourse/.

McCullough, B., McKitrick, R., 2009. Check the Numbers: The Case for Due Diligence in Policy Formation. Research Studies. Fraser Institute. Available at: http://www.fraserinstitute.org/research-news/display.aspx?id=12933.

McCullough, B.D., 2006. Section editor's introduction. Journal of Economic and Social Measurement 31, 103–105. Available at: http://www.pages.drexel.edu/~bdm25/publications.html.

McCullough, B.D., McGeary, K.A., Harrison, T., 2006. Lessons from the jmcb archive. Journal of Money, Credit and Banking 38, 1093–1107. Available at: http://www.pages.drexel.edu/~bdm25/publications.html.

McShane, B.B., Wyner, A.J., 2010. A statistical analysis of multiple temperature proxies: Are reconstructions of surface temperatures over the last 1000 years reliable? To be published in The Annals of Applied Statistics.

NIH, 2003. Nih data sharing brochure. web. Available at: http://grants.nih.gov/grants/policy/data_sharing/.

Nordlie, E., Gewaltig, M.O., Plesser, H.E., 2009. Towards reproducible descriptions of neuronal network models. PLoS Comput Biol 5, e1000456.

Oetiker, T., Partl, H., Hyna, I., Schlegl, E., 2011. The Not So Short Introduction To LaTeX2e. 5.01 edition. Available at: http://www.ctan.org/tex-archive/info/lshort/english.

Peng, R.D., 2011. cacheSweave: Tools for caching Sweave computations. With contributions from Tobias Abenius, R package version 0.6.

Peng, R.D., Dominici, F., 2008. Statistical Methods for Environmental Epidemiology with R. Use R!, Springer.

Pippow, A., Husch, A., Pouzat, C., Kloppenburg, P., 2009. Differences of Ca(2+) handling properties in identified central olfactory neurons of the antennal lobe. Cell Calcium 46, 87–98.

Pouzat, C., Chaffiol, A., 2009. Automatic Spike Train Analysis and Report Generation. An Implementation with R, R2HTML and STAR. J Neurosci Methods 181, 119–144. Pre-print available at: http://sites.google.com/site/spiketrainanalysiswithr/Home/PouzatChaffiol_JNM_2009.pdf?attredirects=0.

R Development Core Team, 2010. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. ISBN 3-900051-07-0.

Rossini, A., Leisch, F., 2003. Literate Statistical Practice. UW Biostatistics Working Paper Series 194. University of Washington. http://www.bepress.com/uwbiostat/paper194/.

Rossini, A.J., 2001. Literate Statistical Analysis, in: Hornik, K.,

Leisch, F. (Eds.), Proceedings of the 2nd International Workshop on Distributed Statistical Computing, Vienna, Austria. ISSN 1609-395X.

Schulte, E., Davison, D., 2011. Active document with org-mode. Computing in Science & Engineering 13, 66–73. Available at: http://www.cs.unm.edu/~eschulte/data/CISE-13-3-SciProg.pdf.

Schwab, M., Karrenbach, N., Claerbout, J., 2000. Making scientific computations reproducible. Computing in Science & Engineering 6, 61– 67. Preprinty available at: http://sep.stanford.edu/lib/exe/fetch.php?media=sep:research:reproducible:cip.ps.

Stallman, R.M., 1981. EMACS: The Extensible, Customizable, Self-Documenting Display Editor. Technical Report AIM-519A. MIT Artificial Intelligence Laboratory. Available at: ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-519A.pdf.

Stein, M.L., 2010. Editorial. Available at: http://www.e-publications.org/ims/submission/index.php/AOAS/user/submissionFile/8887?confirm=6adde642.

Stodden, V., 2009a. Enabling reproducible research: Licensing for scientific innovation. International Journal of Communications Law and Policy 13.

Stodden, V., 2009b. The legal framework for reproducible research in the sciences: Licensing and copyright. IEEE Computing in Science and Engineering 11, 35–40. Available at: http://www.stanford.edu/~vcs/Papers.html.

Tabelow, K., Polzehl, J., Voss, H., Spokoiny., V., 2006. Analyzing fmri experiments with structural adaptive smoothing procedures. NeuroImage 33, 55–62.

The National Science Foundation, 2011. Proposal and award policies and procedure guide. part ii – award & admistration guide. web. Available at: http://www.nsf.gov/pubs/policydocs/pappguide/nsf11001/index.jsp.

Van Gorp, P., Mazanek, S., 2011. Share: a web portal for creating and sharing executable research papers. Procedia Computer Science 4, 589 – 597. Proceedings of the International Conference on Computational Science, ICCS 2011.

Vandewalle, P., Kovacevic, J., Vetterli, M., 2009. Reproducible research in signal processing - what, why, and how. IEEE Signal Processing Magazine 26, 37–47. Available at: http://rr.epfl.ch/17/.

Wallstrom, G., Liebner, J., Kass, R.E., 2007. An Implementation of Bayesian Adaptive Regression Splines (BARS) in C with S and R Wrappers. Journal of Statistical Software 26, 1–21.