Accountable and Reproducible Research

Michael Cysouw

6 December 2014

Introduction

All scientific and scholarly endeavour crucially builds on holding the author accountable. To put it in financial terms: research should have an 'audit trail' of everything that lead to a particular conclusion. This is nothing new, it is basically just plain old being explicit about sources, clear about the method, and precise in citation. In more recent technical discussion, the term *Reproducible Research* has become en-vogue. However, there are at least three different kinds of reproducibility:

- Accountability: did the research accurately report on everything that was done?
- Emulation: is it possible to follow in the footsteps and reproduce everything as claimed in the paper, using exactly the same sources and methods?
- Recreation: is it possible to recreate the result, using new data and new methods that try to follow the earlier work in spirit, but not to the letter?

Recreation is the standard approach to reproducibility in experimental fields of research. In contrast, recreation is not possible in fields of research where data is not (easily) replicable, either because of high costs associated with its collection or because the data are historically bound (e.g., archaeological artifacts, fossils, or manuscripts). The traditional humanities scholar will normally focus on *accountability*. However, as more an more data is becoming available and automatic processing becomes feasible, the focus is slowly shifting towards *emulation*, i.e.

All resources necessary to reproduce the results from an article have to be provided with the article, including explicit documentation on how to reproduce your results from these resources.

Note that the precise prerequisites for Reproducible Research are currently heavily in flux, and it is even unclear what extend of documentation is necessary for it to be counted as "reproducible" (cf. FitzJohn et al. 2014). For some articles it might be feasible to use literature programming (Knuth 1984) by including executable code inside the text of the article (as for example implemented by the Sweave (Leisch 2002) or rMarkDown). For others, a "makefile" approach as initiated by Jon Claerbout (Schwab et al. 2000) might be more suitable. Still other articles might simply depend on explicit textual description of the steps necessary to reproduce the results as presented in the paper. A central criterium for the acceptance of papers is simply whether the reviewers are able to reproduce the results based on the documentation provided.

General recommendations

A few general recommendations that seem sensible for the documentation of your research are the following:

- **Provide all resources as human-readable UTF-8 text-files**.
  - *Human readable* means that a human should be able to make sense of the files and their content, it is of course not intended to mean that the files are for human-eyes-only! Do not use proprietary formats (like Microsoft *doc* or *rtf*, which are very hard to decipher for human eyes) and always include non-compiled code.
  - *UTF-8* means that the files should be in accordance to the Unicode Standard, ideally also using LF line breaks and consistent normalisation (e.g. NFC)
  - *text files* means that the characters in the files are purely UTF-8. The structure of the files is of course not restricted to files ending in *.txt*
- **Use simple, easily understandable, formats**. Formats like CSV, YAML or Markdown are preferred above more complex computer-geared formats, like XML or JSON (although these are of course still fine if necessary). Binary dumps are strongly discouraged.
- **Use open source software**. Open source tools like R, GRASS GIS or Python are strongly preferred above proprietary tools like ArcGIS, SPSS or MATLAB. Using open source tools makes reproducibility easier to adhere to. Note that this is no necessary restriction, but using any commercial tool will strongly restrict the audience for the methods.
- **Provide all resources as a (nested & zipped) directory**, i.e all necessary files are organised in a directory which will be distributed as a single zip-file. Currently, a good option is to collect everything as a repository on GitHub, create a release (i.e. "version 1.0"), and submit a zipped download of the repository with your article. See below for concrete proposals how to structure your repository.
- **Treat sources as read only**, i.e. there will be a set of files that contain the source-data for your research. These files are always kept in their original form and will never be changed. Often it will of course be necessary to recode and subset the source-data, but that process should also be documented explicitly.

- **Treat generated output as disposable**, i.e. it might be good to include any resulting files already finished in your documentation, but these files should be possibly removed without interfering with the reproducibility.
- **Separate function definition from their concrete application** in your code. This is a general guideline for software development, but it can be rather hard to apply to for concrete research using single-function code. However, an attempt should be made to separate the core parts of the code into general functions, and then use these functions in describing the steps you have taken to document your research. Consider preparing the general functions for submission to easily accessible software platforms, like *CRAN* (Comprehensive R Archive Network).

Practical recommendations

A Reproducible Research Resources-directory ("RRR directory" or "RRR package") typically will have something like the structure proposed below (cf. Noble 2009). Note that there is a tendency among computer-literate users to use three-letter abbreviations for such directories (using names like *src*, *doc*, or *bin*), which is understandable from a historical background, but which seems to be superfluous today. I rather prefer longer directory names (and file names) that are more intuitive for anybody trying to reproduce the research. A few general recommendations for the structure and naming of the resource-directory are to include the following kind of files and subdirectories:

- A **readme** file with:
  - Title, author(s) and abstract of the paper
  - An email address where to be contacted
  - An overview of the content of the directory (i.e. a directory listing with some short explanation)
- A **configuration** file with information on about the system on which you tested your code, i.e. hardware and software specifications, everything with detailed version numbers. Ideally, links to dependencies (i.e. necessary software to be downloaded) should also be added.
- A file called **manual** or **make**, listing the concrete steps to reproduce the results, using files from the directories *data* and *code* (see below). A *manual* file will be of a step-by-step listing to reproduce the research, written to be followed by another human being. A *make* file will be a computer-readable set of steps to be taken (using the *GNU make* syntax).
- A subdirectory **sources** which includes all original data from which you started you research. If you include data from a third-party source (e.g. the data from the *World Atlas of Language Structures*), then include the data that you got from there verbatim in this directory, i.e. keep the data unchanged exactly in the form in which you got it from the originator.

- A subdirectory **data** which includes the actual data that you used in your research. Often this is a specific selection or recoding of the data from the directory *sources*. The distinction between *data* and *sources* is important for the reproduction, namely to document exactly which part of the original sources you included in your analyses, and also to show any corrections or recoding you made. In many cases, the transformation of the sources into data is completely automatically, i.e. by using some code from the *code* directory. In that case, the files in *data* are disposable. However, we expect that in many cases some additional tweaking of the data is necessary, which might not be easily documentable in automated form (e.g. manual corrections). In such cases, the explicit addition of the final tweaked data is essential.
- A subdirectory **code** which includes the (documented!) computer-interpretable code to reproduce your research. This should ideally be only a collection of function definitions, to be applied concretely in the *manual* or *make* file. Please add inline commentary and ideally add documentation in the form normally used for the programming language in question. Consider submitting this code to an official code repository like *CRAN*.
- A separate directory **external code** might be included for convenience as well. This will typically be used for already compiled tools that are documented elsewhere (e.g. the *Berkeley Aligner*). Such a folder is traditionally often called *bin* (for *binaries*), but that name seems to be to restrictive to us as the external code might also be non-binary. Necessary external code should always also be documented in the *configuration* file, and any inclusion in the RRR package is for convenience only. In general, this directory is thus disposable, though it might be important to document the precise version of the external software that was used (in case of currently heavily changing external software without detailed versioning).
- A subdirectory **article** (for the source files of the actual article, e.g. the LateX source) can be very helpful. It might include further subdirectories like **figures** or **tables** (for any figures or tables produces from the code). Note that the content of these folders should be disposable, as the figures and tables should be reproducible. Such a directory *article* is traditionally often called *doc* (for *documentation*). However, note that we consider the complete RRR directory as described here to be necessary documentation, so such a name can be confusing.

Other subdirectories can of course be named and added as needed. However, everything should always be documented in the *readme* file, also when the naming as proposed here is followed faithfully.

Note that the files *readme*, *configuration* and *manual/make*, and a directory *sources* seem to be the minimal requirements for a proper RRR package (everything else should be reproducible from there). For theoretical and methodological papers even the directory *sources* might not be necessary (e.g. because it might rely on artificial data). However, adding more resources into an RRR package is generally preferred. There is no reason to try and prepare the absolute minimal RRR package!

References

- FitzJohn, Rich, Matt Pennel, Amy Zanne & Will Cornwell. 2014. Reproducible research is still a challenge. *rOpenSci* http://ropensci.org/blog/2014/06/09/reproducibility/.
- Leisch, Friedrich. 2002. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, eds., *Compstat 2002 - Proceedings in Computational Statistics*. Physica Verlag, Heidelberg, 575-580. https://www.stat.uni-muenchen.de/~leisch/Sweave/
- Knuth, Donald E. 1984. Literate Programming. *The Computer Journal (British Computer Society)* 27(2).97–111. doi:10.1093/comjnl/27.2.97. http://www.literateprogramming.com/knuthweb.pdf
- Noble, William Stafford. 2009. A quick guide to organizing computational biology projects. *PLoS computational biology* 5(7). e1000424. http://dx.doi.org/10.1371/journal.pcbi.1000424
- Schwab, M., N. Karrenbach, J. Claerbout. 2000. Making scientific computations reproducible. *Computing in Science & Engineering*, 2(6).61-67. http://sepwww.stanford.edu/doku.php?id=sep:research:reproducible