Guidelines for Code and Data Submission

Specific Guidance on Reproducible Research (RR)

Benjamin Hofner, Fabian Scheipl (RR Editors, Biometrical Journal)
E-mail: fabian.scheipl@stat.uni-muenchen.de

Document Version: 1.7 (2016/10/28)

Contents

Why do we support reproducible research (RR)?

- Code (and data) makes it easier (or even possible) to reproduce the results and thus facilitates understanding of the methods and results.
- Publicly available code helps to promote the results and methods of the article and increases scientific impact.
- Good code increases the precision of the description of methods and results and thus improves the article.
- Code facilitates communication with other researches who intend to apply the proposed methods or develop alternatives to them.

For more information on reproducible research, a review of the RR policy of the Biometrical Journal and its impact, and on the RR guideline with background information we refer to Hofner *et al.* [1].

1 Submission material

Please prepare your (code) submission as follows:

1.1 Code and data

- The code must be well documented. Each user-defined function (Stata: program) must be documented including all function arguments (Stata: options).
- Unnecessary comments (such as alternative code lines) should be avoided.
- Please use a consistent style throughout your code.
  - R: For a good example see, e.g., http://adv-r.had.co.nz/Style.html. Proper spacing (spaces after commas and around operators etc.) and consistent, reader-friendly line lengths (max. 80 characters per line) can be achieved very easily by running the `tidy_source()` function in the `formatR` package on your source files.
- Please indent your code properly. Usually 2 or 4 spaces per indentation level are a good choice. Maintain a consistent indentation style throughout your document.
  - R: If you use RStudio (Download), proper indentation can be easily achieved by selecting everything (`CTRL-A`) and then typing `CTRL-I` (re-indent lines).
  - Stata: use tab-indentation (which defaults to 4 spaces in the Stata do-file editor).
- Try to put the code in as few code files as possible but use separate files where useful and sensible, e.g., one for the function definitions, one for the simulation study and another for the data analysis.
  - R, Stata, etc.: Do not use separate files for each function, but do structure the code files properly using comments for each "section".
  - MATLAB: use one file per function.
- Read other files, such as the function definitions, into the main file performing the analysis, instead of mixing code that performs the analysis with code that defines the functions used in the analysis:
  - R: use `source()`.
  - Stata: use `do`.
  - SAS: use `%include`.
- Make sure that the results are easy to compare with the manuscript:
  - Add comments that state what figure / table in the manuscript the code produces.

  - Structure your output such that it corresponds to the structure of the tables given in the manuscript.

- Avoid absolute paths (e.g., `C:/My Code`) whenever possible. Use paths relative to the current file instead. If this is impossible state in your README where/how to change the paths.

  - In R and Stata absolute paths are not necessary (unless in some very rare occasions). If you really want to set or change your working directory do this only once, at the very beginning of your master code file. For an example that uses relative paths in R see Appendix A.
  - In SAS one can either define a macro variable that contains the relevant path and use this wherever the path is needed or one needs to clearly state in the README which path needs to be replaced in which files.

- Use descriptive file names and avoid spaces in file names. Use underscores (_) or hyphens (-) instead.

- Each submission should contain:

  - all the code to reproduce all the results, images and tables. Make sure that the code submitted is complete and covers all calculations used in the finally published paper (and only these). Make also sure the code is well documented. This especially includes (but is not restricted to) the documentation of user written functions and arguments therein.
  - all the data to reproduce all the results, images and tables.

- If the code takes a long time to run on a modern PC (longer than 2h, say), consider saving intermediate results in a separate folder. Keep the main folder clean so that the code is executed without using the intermediate results by default. For an example of the structure, see Section 1.3. Please state in the README how to use the intermediate results (e.g. "Copy the intermediate results to the folder ..."). Alternatively, give advice how to change certain parameters (such as the number of repetitions) to speed up the computation drastically but still obtains similar results.

- If the data is confidential, authors should consider using simulated data that mimics (the main features of) the real data and produces similar results.

- Simulations and everything else that relies on random numbers / random samples must set a random generator seed to make the results reproducible:

  - R: use `set.seed(#)`
  - Stata: use `set seed #`
  - SAS: use for example `call streaminit(#)`, or `call randseed(#)` in `proc iml`

(replace `#` by a number)

- Make sure that the code can be *executed without the need of alterations by the user*. This avoids user errors and streamlines the reproduction of results for both the reviewers and readers. If user input is required, the authors must state and explain this in the README.

- Before submitting the code carefully check that it reproduces the published results. Especially make sure that it is executable in a clean workspace. In R, remove all objects from the environment via `rm(list = ls())` before running the code or use `R CMD BATCH` to start the code.

1.2 README

- Each code submission must be accompanied by a `README.txt` or a `README.pdf` file that contains:
  - the title of the manuscript
  - the authors of the manuscript
  - please indicate which authors are mainly responsible for writing the code in the README. Please name any other persons that made substantial contributions to the development of the code here and also acknowledge them in the paper.
  - the e-mail address of the author or contributor that readers should address with questions, comments and remarks on the code (and to report bugs).
  - a list of configurations on which you tested your code [software, software version (incl. package versions), platform].
    - In R you can use the output from `sessionInfo()` at the end of your code, i.e., after all packages have been loaded.
    - In Stata report the version number and if appropriate the flavor you are using (see `Help > About Stata`).
  - a specification of special hardware or software requirements (if applicable).

- The README
  - explains how to execute the code (e.g., Which program was used? In which order should the files be executed? What are the results obtained by the file? Add flowcharts when they appear helpful.)
  - (if applicable) explains what the data represents, i.e., the type and meaning of all data which are used in the code. You may refer to tables/sections in the paper.
  - state the source of the data.

1.3 Structure of code submission (ZIP-folder)

Code, data and the README must be contained in a single zip container, possibly with sub-folders in the container for different purposes. E.g. one could have the following structure

```
README.txt
/case_study
    ./data
    case_study.R
/simulation
    ./intermediate_results
    simulation.R
```

A special folder for the code might not be necessary. The README should be placed in the top level of the zip file (see example above). Make sure that the code can be directly executed after unpacking the file without any need to alter the code.

Please name the zip container as follows:

- If it contains only data: `Data.zip`
- If it contains only Code: `Code.zip`
- If it contains data and code: `Code_and_Data.zip`

2 Submission process

Please use the appropriate mode of submission:

- **When you submit your paper:**

  If your paper fits the RR option of Biometrical Journal you are welcome to submit code and data along with the first submission of the manuscript. Please upload the zip file, structured as described in Section 1.3, on Manuscript Central via "Data and Software".

- **During the review process:**

  You may be asked during the review process to submit code and data or to change them if they were already supplied at the time of the initial submission. Then please upload the zip file, structured as described in Section 1.3, on Manuscript Central via "Data and Software".

- **After (conditional) acceptance:**

  Please send the code (structured as described in Section 1.3) to the RR Editor (fabian.scheipl@stat.uni-muenchen.de). If the manuscript was changed please send also the manuscript (incl. sources) by email. If the files are too large to be emailed please inform the RR Editor who will give further instructions on how to proceed. All files will be uploaded to Manuscript Central at the end of the RR check.

If any questions on the submission process arise please contact the RR Editor and keep the Editor in cc.

3 Reference the code

Please refer to the code in the submitted manuscript where appropriate, for example in the section of the data analysis and/or in the section where the simulation study or a numerical study are described. You can do this for example as follows:

> "Source code to reproduce the results is available as Supporting Information on the journal's web page (`http://onlinelibrary.wiley.com/doi/xxx/suppinfo`)."

If the Supporting Information contains several files you may list them explicitly such that the reader (of the electronic version) can navigate through the Supporting Information. The links will be set by the publisher.

4 Example

A good example is given by W. Sauerbrei, A. Buchholz, A.-L. Boulesteix & H. Binder (see http://onlinelibrary.wiley.com/enhanced/doi/10.1002/bimj.201300222/).

5 Possible improvements

To further simplify and enhance reproducibility of results, submissions using literate programming approaches to combine LaTeXand R code such as Sweave [2, 3]/ knitr [4, 5]/... are encouraged. Please also supply an R code file in that case. For SAS and Stata there exists a similar package for dynamic documents called StatWeave.

Authors of papers that use multiple user-specified functions coded in R are encouraged to provide an R package (which can be, but does not necessarily has to be, hosted on CRAN or Bioconductor).

References

[1] B. Hofner, M. Schmid, and L. Edler. Reproducible research in statistics: A review and guidelines for the Biometrical Journal. *Biometrical Journal*, 2015. Online First, DOI 10.1002/bimj.201500156.

[2] Friedrich Leisch. Sweave, part I: Mixing R and LaTeX. *R News*, 2(3):28–31, December 2002. http://CRAN.R-project.org/doc/Rnews/.

[3] Friedrich Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, October 2003. http://CRAN.R-project.org/doc/Rnews/.

[4] Yihui Xie. *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2013. ISBN 978-1482203530, http://yihui.name/knitr/.

[5] Yihui Xie. *knitr: A general-purpose package for dynamic report generation in R*, 2014. R package version 1.6, http://yihui.name/knitr/.

A Using relative paths in R

Normally, you can assume that the working directory is the folder that contains the master code file. Now, one can navigate relative to this directory. Assume the following (unnecessarily complex) structure:

```
/data/dataset.Rda
/code/case_study.R
/code/functions/fun_dev.R
/results/
```

We assume that the working directory is the folder of the master source file, i.e., `code`. Now, we want to load the function definitions `fun_def.R` into the master file `case_study.R`, load the data set and save the results in the `results` folder. The code in `case_study.R` looks as follows:

```
########################################################################
## This an example with pseudo code to show the use of relative paths
########################################################################

## make sure the current working directory is the folder code/

## now source the function definitions:
source("functions/fun_dev.R")

## load the data
load("../data/dataset.Rda")
## ../ means to go up one directory level,
## ../../ would mean to go up two levels etc.

## do something with the data

## save the results to results/...
pdf("../results/figure1.pdf")
plot(...)
dev.off()

write.csv(some_results, file = "../results/table1.csv")
```