Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Reproducible development environments with rix

Bruno Rodrigues
Ministry of Research and Higher education, Luxembourg

Philipp Baumann
Plus Affiliation

Abstract

In order to create an analysis that is easily reproducible, it is not enough to write clean code and document it well. One must also make sure to list all the dependencies of the analysis clearly and ideally provide an easy way to install said dependencies. There are several tools that can be used to list dependencies and to make them easily installable by someone that wishes to reproduce a study, such as Docker, a containerization solution. This paper will present the Nix package manager, and an R package called {rix} that lowers Nix's learning curve for users of the R programming language.

Keywords: reproducibility, R, Nix.

1. Introduction: Reproducibility is also about software

The introduction is in principle "as usual". However, it should usually embed both the implemented *methods* and the *software* into the respective relevant literature. For the latter both competing and complementary software should be discussed (within the same software environment and beyond), bringing out relative (dis)advantages. All software mentioned should be properly `@cited`'d. (See also Using BibTeX for more details on BibTeX.)
For writing about software JSS requires authors to use the markup `[]{.proglang}` (programming languages and large programmable systems), `[]{.pkg}` (software packages), back ticks like ‘code‘ for code (functions, commands, arguments, etc.).
If there is such markup in (sub)section titles (as above), a plain text version has to be provided in the LaTeX command as well. Below we also illustrate how abbrevations should be introduced and citation commands can be employed. See the LaTeX code for more details.

Peng (2011) introduced the idea of reproducibility being on a continuum: on one of the ends of this continuum, we only have access to the paper describing the studies, which is not reproducible at all. Then, if to this paper we make the original source code of the analysis that was written to compute the results of the study available, reproducibility is improved, albeit only by a little. Adding the original data to this package improves reproducibility yet again. Finally, if to all this we add what Roger Peng named the *linked and executable code and data*, we reach the gold standard of full replication.

What is this *linked and executable code and data*? Another way to name this crucial piece of the reproducibility puzzle is *computational environment*. The computational environment is all the software required to actually run the analysis. Here too, we can speak of a continuum. One could simply name and list the software used: for example, the R programming language. Sometimes, authors have the courtesy to also state the version of R used. Some authors go further, and also list the packages used, and ideally with their versions as well. Authors rarely state the operating system on which the analysis was done, even though it has been shown that running the same analysis with the same software but on different operating systems could lead to different results, as described in Bhandari Neupane, Neupane, Luo, Yoshida, Sun, and Williams (2019). Authors also only very rarely provide instructions to install the required tools and software in order to reproduce their studies.

There are exceptions of course, a great example of a paper that provides everything needed to reproduce its results is McDermott (2021). The author of this paper set up an accompagnying Github repository to the paper[1] containing all the instructions to install the required software and then run the analysis. If we take a closer look at this repository, we will notice that several tools were used to capture the compatutational environment and make it available to other researchers:

- First, the version of R was stated

2. Models and software

The basic Poisson regression model for count data is a special case of the GLM framework **?**. It describes the dependence of a count response variable y_i $(i = 1, \dots, n)$ by assuming a Poisson distribution $y_i \sim \text{Pois}(\mu_i)$. The dependence of the conditional mean $E[y_i \,|\, x_i] = \mu_i$ on the regressors x_i is then specified via a log link and a linear predictor

$$\log(\mu_i) \quad = \quad x_i^\top \beta, \tag{1}$$

where the regression coefficients β are estimated by maximum likelihood (ML) using the iterative weighted least squares (IWLS) algorithm.

> TODO: Note that around the equation above there should be no spaces (avoided in the LaTeX code by `%` lines) so that "normal" spacing is used and not a new paragraph started.

R provides a very flexible implementation of the general GLM framework in the function `glm()` **?** in the **stats** package. Its most important arguments are

[1] https://github.com/grantmcdermott/skeptic-priors

```
glm(formula, data, subset, na.action, weights, offset,
  family = gaussian, start = NULL, control = glm.control(…),
  model = TRUE, y = TRUE, x = FALSE, …)
```

where `formula` plus `data` is the now standard way of specifying regression relationships in R/S introduced in ?. The remaining arguments in the first line (`subset`, `na.action`, `weights`, and `offset`) are also standard for setting up formula-based regression models in R/S. The arguments in the second line control aspects specific to GLMs while the arguments in the last line specify which components are returned in the fitted model object (of class '`glm`' which inherits from '`lm`'). For further arguments to `glm()` (including alternative specifications of starting values) see `?glm`. For estimating a Poisson model `family = poisson` has to be specified.

> As the synopsis above is a code listing that is not meant to be executed, one can use either the dedicated `{Code}` environment or a simple `{verbatim}` environment for this. Again, spaces before and after should be avoided.
> Finally, there might be a reference to a `{table}` such as Table 1. Usually, these are placed at the top of the page (`[t!]`), centered (`\centering`), with a caption below the table, column headers and captions in sentence style, and if possible avoiding vertical lines.

Type	Distribution	Method	Description
GLM	Poisson	ML	Poisson regression: classical GLM, estimated by maximum likelihood (ML)
		Quasi	"Quasi-Poisson regression": same mean function, estimated by quasi-ML (QML) or equivalently generalized estimating equations (GEE), inference adjustment via estimated dispersion parameter
		Adjusted	"Adjusted Poisson regression": same mean function, estimated by QML/GEE, inference adjustment via sandwich covariances
	NB	ML	NB regression: extended GLM, estimated by ML including additional shape parameter
Zero-augmented	Poisson	ML	Zero-inflated Poisson (ZIP), hurdle Poisson
	NB	ML	Zero-inflated NB (ZINB), hurdle NB

Table 1: Overview of various count regression models. The table is usually placed at the top of the page (`[t!]`), centered (`centering`), has a caption below the table, column headers and captions are in sentence style, and if possible vertical lines should be avoided.

3. Illustrations

For a simple illustration of basic Poisson and NB count regression the `quine` data from the **MASS** package is used. This provides the number of `Days` that children were absent from school in Australia in a particular year, along with several covariates that can be employed as regressors. The data can be loaded by

```
R> data(mtcars)
```

and a basic frequency distribution of the response variable is displayed in **?@fig-quine**.

> For code input and output, the style files provide dedicated environments. Either the "agnostic" `{CodeInput}` and `{CodeOutput}` can be used or, equivalently, the environments `{Sinput}` and `{Soutput}` as produced by `Sweave()` or **knitr** when using the `render_sweave()` hook. Please make sure that all code is properly spaced, e.g., using `y = a + b * x` and *not* `y=a+b*x`. Moreover, code input should use "the usual" command prompt in the respective software system. For R code, the prompt `R>` should be used with `+` as the continuation prompt. Generally, comments within the code chunks should be avoided – and made in the regular LaTeX text instead. Finally, empty lines before and after code input/output should be avoided (see above).

4. Summary and discussion

> As usual…

Computational details

> If necessary or useful, information about certain computational details such as version numbers, operating systems, or compilers could be included in an unnumbered section. Also, auxiliary packages (say, for visualizations, maps, tables, …) that are not cited in the main text can be credited here.

The results in this paper were obtained using R~3.4.1 with the **MASS**~7.3.47 package. R itself and all packages used are available from the Comprehensive R Archive Network (CRAN) at [https://CRAN.R-project.org/].

Acknowledgments

> All acknowledgments (note the AE spelling) should be collected in this unnumbered section before the references. It may contain the usual information about funding and feedback from colleagues/reviewers/etc. Furthermore, information such as relative contributions of the authors may be added here (if any).

References

Bhandari Neupane J, Neupane RP, Luo Y, Yoshida WY, Sun R, Williams PG (2019). "Characterization of Leptazolines A–D, Polar Oxazolines from the Cyanobacterium Leptolyngbya sp., Reveals a Glitch with the "Willoughby–Hoye" Scripts for Calculating NMR Chemical Shifts." *Organic Letters*, **21**(20), 8449–8453.

McDermott GR (2021). "Skeptic priors and climate consensus." *Climatic Change*, **166**(1-2), 7.

Peng RD (2011). "Reproducible research in computational science." *Science*, **334**(6060), 1226–1227.

More technical details

Appendices can be included after the bibliography (with a page break). Each section within the appendix should have a proper section title (rather than just *Appendix*).
For more technical style details, please check out JSS's style FAQ at [https://www.jstatsoft.org/pages/view/style#frequently-asked-questions] which includes the following topics:

- Title vs. sentence case.
- Graphics formatting.
- Naming conventions.
- Turning JSS manuscripts into R package vignettes.
- Trouble shooting.
- Many other potentially helpful details…

Using BibTeX

References need to be provided in a BibTeX file (`.bib`). All references should be made with `@cite` syntax. This commands yield different formats of author-year citations and allow to include additional details (e.g.,pages, chapters, …) in brackets. In case you are not familiar with these commands see the JSS style FAQ for details.
Cleaning up BibTeX files is a somewhat tedious task – especially when acquiring the entries automatically from mixed online sources. However, it is important that informations are complete and presented in a consistent style to avoid confusions. JSS requires the following format.

- item JSS-specific markup (`\proglang`, `\pkg`, `\code`) should be used in the references.
- item Titles should be in title case.
- item Journal titles should not be abbreviated and in title case.
- item DOIs should be included where available.
- item Software should be properly cited as well. For R packages `citation("pkgname")` typically provides a good starting point.

Affiliation:

Bruno Rodrigues
Department of Statistics
18, Montée de la Pétrusse
Luxembourg Luxembourg
E-mail: `bruno@brodrigues.co`
URL: https://www.brodrigues.co

Philipp Baumann

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd