Software Papers in Computers & Geosciences

Edzer Pebesma*

Version 0.1, September 19, 2014

1 Introduction

Today, many scientific papers are based upon scientific computing, which in turn is based on using scientific software. Increasingly, developing original scientific software is an integral part of the process that leads to obtaining new scientific insights. The *research software impact manifesto*[1] for instance argues that *software has become the third pillar of research, supporting theory and experiment.*

We describe the *software paper* as a specific type of scientific paper that describes an original scientific software contribution which has the goals to (i) help other scientists understand and reproduce previously published research, (ii) increase the transparency of the scientific research where the software was used, (iii) allow reuse of the software to carry out new research, and (iv) allow proper citing of the software contribution, e.g. when it is used in consecutive papers. Following goal (ii), we will only consider papers describing open source software. We describe the *software paper*, its purpose, its requirements, and recommendations. We aim at providing a blueprint for future software papers published in *Computers & Geosciences*, or elsewhere. It has consequences to the review process, as reviewers should address the software published too.

2 Purpose

A *software paper*, as described here, needs to be an original scientific contribution. Where usual scientific paper describe novel theories or findings that follow from analysing data, the software paper describes original scientific software. As with all scientific papers, the paper should describe the software in the context of previous work. It should argue why there was a need for the software, describe similar or competing solutions, and demonstrate the use of the software. It should properly reference the relevant related literature, be it other

*edzer.pebesma@uni-muenster.de

[1] http://software.ac.uk/blog/2011-05-02-publish-or-be-damned-alternative-impact-manifesto-research-software

software papers or papers where the software described was put into practice. It should also discuss the virtues of the software, and describe its use, and the user experiences.

3 Requirements

The following minimum requirements need to be maintained in order to obtain comprehensive description of the software publishe.

originality The paper should point out why the software contribution is original, and how it improves over existing solutions. It should answer the question why there was a need to develop the software.

context The paper should point how the software developed relates or compares to existing solutions, and how existing solutions were integrated or re-used.

source code A pointer should be given to the source code, which can either be a URL pointing to a tar ball, more typicall a source code repository (git, subversion).

versioning The paper should clearly mention which version of the software is described, either encoded in the file name, or as the tag, revision, or commit number identifier in a repository.

availability The software should be available to the reviewers at the time of submitting the paper, in source code form, meaning that it is part of the paper at time of first submission, and subject to review.

readability The software should should be readable, have comments in place where relevant, and be written having in mind that others would want to understand it. The software should be organized in a way that makes it useful for prospective users.

portability The paper should describe the platform(s) targeted by the software, and describe *requirements* for installing, compiling (if needed) and running the software. Complete *instructions* for installing the software are not needed in the paper, but should be easily identifiable (e.g. in README files) in the source code itself.

examples The paper should demonstrate show how the software works with real examples, describing input, interaction (if any), and output. The examples should be easily reproducible with the source code available.

citing The paper should cite all relevant software mentioned, or used to build the software described. This does not include generic tools (like `make`, or `libc`) used for most software, but specific tools on which the described software critically depends.

copyright, license The paper and software should both clearly state who owns the copyrights, and under which license it is distributed.

3.1 Requirements to reviewers

Reviewers should primarily review the paper, but also address the software described. Minimum checks include:

- can the software be downloaded?
- can the software be installed, and run?
- does it run the examples shown in the paper?

The reviewers should address the question whether the contribution of the software paper is original, substantial, and sufficiently embedded and/or discussed in the context of existing solutions and/or embedding environments.

4 Discussion

Research software is rarely written with the main purpose of writing a paper about it, and publishing this. Doing this nevertheless makes only sense if the author believes that a certain readership is helped with this, and will be inclined to read and hopefully cite the paper. Citations may be seen as a reward that users are typically inclined (if not morally obliged) to give when they use the software described for their scientific work. Although direct pointers to the software itself, e.g. by URLs, tags like CRAN:packagename[2], or DOIs pointing to the software[3] in the paper, from a software paper the readership may expect that the software has been properly described and discussed for a scientific audience.

In general, it is recommended that authors write a software paper after a certain amount of user feed-back has been collected, and after the software has reached a certain stability. This allows to report on usage, user feedback, and reflect on its usability[4]. Needless to say, publishing software requires properly paying attention to software testing, naming conventions, organisation, complexity, scalability, intelligable comments and user documentation, and so on.

It is not unusual that research software is being further developed after a paper on it has been published; the paper is usually does not allow updating. It is therefore useful that a fixed version that matches the published paper is identifiable; this can be done by a source code git clone made by the journal editor (Computers & Geosciences), or by a full copy of the software (Journal of Statistical Software). Alternatively, an updated copy of the paper can be provided by the software author, along with the software, as is often done with vignettes accompanying R packages[**?**].

[2]TODO
[3]DOI-from-github-TODO
[4]HyndManBlogJSS

Persistence of web sites, and of links
long-term curation and github.
[pointers to other descriptions of / requirements to software papers]

5 Links

- http://software.ac.uk/so-exactly-what-software-did-you-use
- http://robjhyndman.com/hyndsight/jss-rpackages/