Review for "Towards Long-term and Archivable Reproducibility"

Dylan Aïssi[1]

[1]Affiliation not available

August 12, 2020

This is a review of manuscript CiSESI-2020-06-0048 submitted to Computing in Science & Engineering: "Towards Long-term and Archivable Reproducibility" (Akhlaghi et al., 2020). Another version of the manuscript was published on arXiv: 2006.03018.

Review

Please summarize what you view as the key point(s) of the manuscript and the importance of the content to the readers of this periodical.

The authors describe briefly the history of solutions proposed by researchers to generate reproducible workflows. Then, they report the problems with the current tools used to tackle the reproducible problem. They propose a set of criteria to develop new reproducible workflows and finally they describe their proof of concept workflow called "Maneage". This manuscript could help researchers to improve their workflow to obtain reproducible results.

What do you see as this manuscript's contribution to the literature in this field?

The authors try to propose a simple answer to the reproducibility problem by defining new criteria. They also propose a proof of concept workflow which can be directly used by researchers for their projects.

What do you see as the strongest aspect of this manuscript?

This manuscript describes a new reproducible workflow which doesn't require another new trendy high-level software. The proposed workflow is only based on low-level tools already widely known. Moreover, the workflow takes into account the version of all software used in the chain of dependencies.

What do you see as the weakest aspect of this manuscript?

Authors don't discuss the problem of results reproducibility when analysis are performed using CPU with different architectures. Some libraries have different behaviors when they ran on different architectures and it could influence final results. Authors are probably talking about x86, but there is no reference at all in the manuscript.

Comments

Operating System

Authors mention that Docker is usually used with an image of Ubuntu without precision about the version used. And Even if users take care about the version, the image is updated monthly thus the image used will have different OS components based on the generation time. This difference in OS components will interfere on the reproducibility. I agree on that, but I would like to add that it is a wrong habit of users. It is possible to generate reproducible Docker images by generating it from an ISO image of the OS. These ISO images are archived, at least for Ubuntu (`http://old-releases.ubuntu.com/releases/`) and for Debian (`https://cdimage.debian.org/mirror/cdimage/archive/`) thus allow users to generate an OS with identical components. Combined with the snapshot.debian.org service, it is even possible to update a Debian release to a specific time point up to 2005 and with a precision of six hours. With combination of both ISO image and snapshot.debian.org service it is possible to obtain an OS for Docker or for a VM with identical components even if users have to use the PM of the OS. Authors should add indication that using good practices it is possible to use Docker or VM to obtain identical OS usable for reproducible research.

CPU architecture

The CPU architecture of the platform used to run the workflow is not discussed in the manuscript. During software integration in Debian, I have seen several software failing their unit tests due to different behavior from itself or from a library dependency. This not expected behavior was only present on non-x86 architectures, mainly because developers use a x86 machine for their developments and tests. Bug or feature? I don't know, but nowadays, it is quite frequent to see computers with a non-x86 CPU. It would be annoying to fail the reproducibility step because of a different in CPU architecture. Authors should probably take into account the architecture used in their workflow or at least report it.

POSIX dependency

I don't understand the "*no dependency beyond POSIX* ". Authors should more explained what they mean by this sentence. I completely agree that the *dependency hell* must be avoided and dependencies should be used with parsimony. Unfortunately, sometime we need proprietary or specialized software to read raw data. For example in genetics, micro-array raw data are stored in binary proprietary formats. To convert this data into a plain text format, we need the proprietary software provided with the measurement tool.

Maneage

I was not able to properly set up a project with Maneage. The configuration step failed during the download of tools used in the workflow. This is probably due to a firewall/antivirus restriction out of my control. How frequent this failure happen to users? Moreover, the time to configure a new project is quite long because everything needs to be compiled. Authors should compare the time required to set up a project Maneage versus time used by other workflows to give an indication to the readers.

Disclaimer

For the sake of transparency, it should be noted that I am involved in the development of Debian, thus my comments are probably oriented.