Larissa Helena Mendes

BEST PRACTICES IN REPRODUCIBLE RESEARCH

Scientific research is a remarkably competitive field. The new tools and technologies, the enormous amount and complexity of data increase the pressure on scientists to advance their research. Thus, over the years, fewer great discoveries have been made. On the other hand, a growing amount of irreproducible results critically restrict the capability of the scientific community to build on results and develop the field.

Confronted by these new challenges, the scientific community has been looking for alternatives and good practices in the execution of research to simplify future reproductions. From this need, the term reproducibility in scientific research emerged.

But what is reproducibility?

The terms "reproducibility" and "replicability" are often used interchangeably, and sometimes misplaced. There is always some debate on how these terms are used, but for this paper, the following definitions are going o be followed:

Reproducibility
(different team, same experiment, same result)

In this situation, data and code are reanalyzed by independent scientists to obtain the same result.

Replicability
(different team, different experiment, same result)

With replication, independent scientists address a scientific finding building up evidence for or against it, using different techniques and methods. However, in most field experiments, full replication is often too expensive and laborious as researches need more and more large and complex data. In such cases, the researchers should still try to achieve reproducibility.

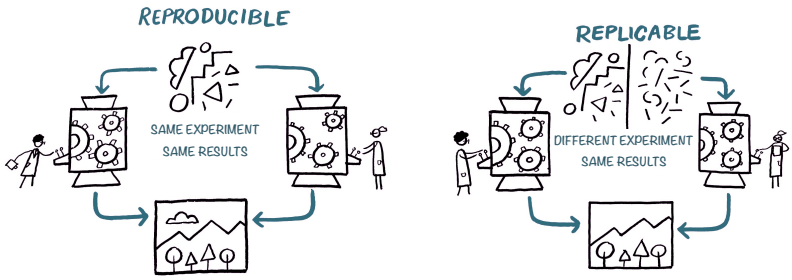

Illustration modified from: The Turing Way Community, & Scriberia. (2019, July 11) **[1]**

Reproducibility is based on the fact that every research must have a detailed log of every action taken. Therefore, to perform reproducible research, one should focus on the following five **key elements**:

Data: Data used in the analysis, normally a subset of the raw data that is clean and well-identified, containing all variables used in the analysis.

Code: Code used to do the analysis or preprocessing of data. Allowing a user to immediately run your code and reproduce the results.

Documentation: All codes and analysis datasets should be easily understandable to researchers attempting to replicate the results. Documentation can include variable dictionaries and outline instruments, data release, ensuring that users can easily understand the data.

Workflow: Adopting a workflow management system means expressing the work in terms of high-level components such as inputs and outputs joined together in a pipeline.

Distribution: Provide the environment used originally because it might be necessary to use programs in the same version.

[1] The Turing Way Community, & Scriberia. (2019, July 11). Illustrations from the Turing Way book dashes. Zenodo. http://doi.org/10.5281/zenodo.3332808

Guidelines for reproducible research

Bellow, a guideline for planning and executing reproducible research. There can be three different stages to produce well-documented and practical research. First, plan your workflow, organize, and document your repository and license your data. Next, provide the environment and its dependencies and write your codes using version control. Finally, when publishing, make available your code along with its data.

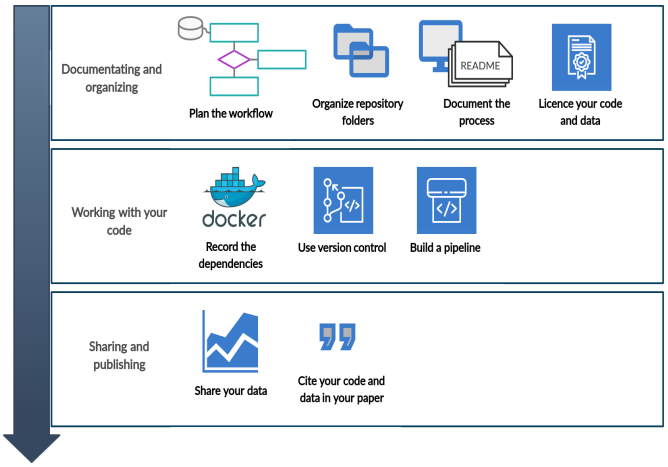

Illustration inspired from: https://doi.org/10.1371/journal.pbio.1001745

DOs and DON'Ts

Distribution:

DO: Use an environment distribution tool, such as virtual machines or containers. There are some interesting options like Docker, VM VirtualBox, and Binder. At best, offering multiple options to the user minimizes the chances of failures in reproduction.

DON'T: Do not forget to report every dependencies and its versions. And do it by the moment you start using them beacuse you can't forget to report any of them. You have to leave the option for the reproducer to work locally.

Data:

DO: License your code and data. People will not be able to share or contribute to your code unless you inform them what rules to play by. If a code has no license, even if it is hosted in GitHub, for instance, this does not imply that you are allowed to use, share or modify the software.

DON'T: Never execute manual data manipulation. Also don't provide your plots that were generated or saved manually. Such manual procedures are disorganized and error-prone, it makes it difficult not just for the user, but also for you to reproduce.

DO: Share the raw data. The raw data should be in the form that was first received, even if it is in binary or some proprietary format. Any code used to clean and tidy the raw data should also be provided. If the data sets you used are not openly available or too difficult to find, if the original dataset license permits, upload the dataset into a repository with a DOI.

Code:

DO: Make your code citable! So far, the citing of code is already supported by platforms like Zenodo. There, each version of the software can be referred by a permanent citable DOI for published repositories on GitHub. Software citations should facilitate access to specific versions of data and code.

DON'T: Do not store old, out-dated code versions and sample codes that are not relevant to your project workflow in the repository. It can make it confusing to understand your data and workflow.

DO: "Commit" to version control! By using project management software with version control you can efficiently track and control changes to your code and documentation. Git is a nice option in which you will find extensive documentation.

Make commits after changes that you want to save, but do not leave to do it after a big set of changes, especially when you are working with a notebook.

DON'T: Don't write a messy code and fix it only at the end. Make the code format and style consistent from the beginning. Variable names should be consistent, distinctive, and meaningful. Comments should be descriptive and indentation should be on point.

Documentation:

DO: Write instructions in your repository. Add README files to describe the project and instructions on reproducing the results. The ReadME file should include:

1. Description of the project and;
2. Contact information, if there is any;
3. Information on the environment and and dependencies;
4. A detailed description of the data, data source(s), and how it will be used;
5. A list of all the scripts and libraries including the input and output parameters (if there are any);
6. A list of all scripts, how to run them, what they are for, and in what order.

DO: Organize your repository. Use a consistent and informative directory structure separating raw from preprocessed data, and code from results. An example of a directory structure could be something like:

- Project/
  - README.md
  - Data (or Inputs)/
    - Raw Data/
    - Processed Data/
  - Code (or Analysis)/
    - Data preprocessing/
    - Data analysis/
  - Output (or Results)/
  - Environment/
  - Publication/

DON'T: Do not leave to do the documentation only for when the work is done. Leaving it to the end can lead to the neglect of important steps that hinder reproducibility. Imagine yourself working on many different projects at the same time, and having to back to a project six months or a year later. That's why documenting what you did to your data and why it is the soul of reproducible research.

DO: Search and read reproducible papers online. A wise person learns from others mistakes, so look for other people's work and learn from them.

Just because a work is labeled reproducible, doesn't mean that it has good instructions or that is stable. GitHub is full of examples, but there are also some reproducible research paper databases available.

Workflow:

DO: Add a workflow diagram to your repository. It will help you to construct your pipeline and explain to a future user about the inputs, outputs, and processes involved. Draw.io and Creately can be very helpful.

DO: If possible, provide an automated workflow. There are very useful tools like Reana, that lets you structure your data analyses through a YAML file that captures sufficient information about the analysis assets, parameters, and processes in which a user can execute the workflow in only one go.

Useful links

- Ten Simple Rules for Reproducible Computational Research
- A Realistic Guide to Making Data Available Alongside Code to Improve Reproducibility.
- AGILE Reproducible Paper Guidelines
- Best Practices for Scientific Computing
- Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks