Best Practices – Reproducitibility in Scientific Computing

Fernando Seiti Furusato - 228011

June 21, 2019

This is a guide for best practices on reproducibility in scientific computing research, which I have compiled from my experience while working on a project, designed to be reproducible.

The convention, when working to guarantee the reproducibility, is that there are 5 main subjects of concentration, which can be essential for making a research fully reproducible:

- **Code**: how the project's code can be distributed, programming language used.
- **Data**: much like the code, the data must also be distributed, so how to distribute the project's data. Keeping in mind that the data can be very large.
- **Documentation**: documenting the work is important when the goal is reproducibility. It makes it easier to reproduce when one has a documentation to refer to when trying to execute someone else's work.
- **Environment**: what is needed in order to get the same execution results of someone else's code, in terms of software dependencies, libraries, operating system etc.
- **Workflow**: guaranteeing the workflow of a research is not always trivial, due to intermediate data generation, seeding data used and plotting data maintenance. Having a workflow plan and a tool to help its development makes this task smoother and more natural (less painful).

Note, from the list above, that it is bulleted with symbols, representing each of the subjects described. In this guide, you will a table, named ”DOs and DON'Ts”, with some tips collected from experience and marked with the bullets from the list, according to which subject or subjects it covers.

DOs and DON'Ts

Tip	**Item**
Be as clear as possible regarding the location, how to obtain the data and where to save them. Do not assume your data is easily usable and locatable.	🛢
Specially if your work involves big data: collect your data at the earliest of your project. Not only that, but also learn how to work with it as soon as you have them. That is because if you leave it to the end, you might end up with no data at all.	🛢
Use an environment distribution solution, such as virtual machines, containers and such. Docker [2] is a good starting point. It is relatively easy to implement and light, compared to virtual machines.	🎚
Implement the workflow controlling from the beginning of the project, or as soon as possible. That is because doing so too late might make it nearly impossible.	🖧
Make the data processing – either input, intermediate or output – automated. This will make it easier for you and others to have a ready-to-go dataset.	>_ 🛢
If your paper involves generating a model for making predictions (*e.g.* machine learning), it is interesting to make it possible for whoever is reproducing it to run only the prediction if they wish so.	>_ 📄

DOs and DON'Ts

Tip	**Item**
Be as up-to-date as possible when using third-party libraries. That way, deprecation of snippets of the dependencies might take longer to get to your code.	>_ 🎚
Keep all the library or module dependencies documented. Test your code in a clean environment to detect undocumented dependencies. Do not keep unnecessary dependencies within the code. This might affect your project portability.	>_ 🎚
Understand why you need your dependencies and be as specific as possible in regards of version and how to install them. Get rid of unnecessary dependencies as soon as possible. That way, your environment can be as compact as possible and you will not be stuck with unnecessary dependencies.	>_ 🎚
Plan beforehand (or as soon as possible) what you are going to include in the final version of your paper, such as which data will be used and what part of the code you are going to present.	>_ 🛢 🗎
Include the pre-processing of the data in your paper, in case whoever is reproducing the paper can also generate the analytical data. In other words, share the raw data and how to process it.	>_ 🗎 🛢 🖧
If your work generates models for predictions, provide means to download the model so one can use it for testing the predictions.	>_ 🗎 🎚 🖧
Keep the documentation synchronized with coding. Do not leave the writing for when the work is complete. This includes conversion from jupyter notebook to pdf for the ones using it. That way, it will be possible to keep track of each part of the documentation.	🗎 🖧

Successes and Failures

In this section I present some cases in which I had success or failure on the reproducibility tips listed previously. These are basically the points where I have acquired the knowledge for this document.

- The entire code can be downloaded, visualized and even executed if one wishes so. In this case I have used github [4]. ✓
- Not all the parts of the work is reproducible. *E.g.* part of the data preparation, plotting and training are not straightforward to reproduce and might demand some grinding. ✗
- The data is entirely available in a figshare [3] project, even the ones that are not referenced in the final version of the paper. ✓
- The data preprocessing is not available in the final version of the paper, so users are only able to use the final analytical data. ✗
- I did not miss any dependency. That is because each time I tested my paper, I started a clean environment in a docker container. ✓
- I used docker to make the environment of reproduction available. That way, whoever needs to run my project, can do it as simply as executing a 'docker run' command. ✓
- For the final version of the paper, I ended up leaving one dependency that I could finally get rid of. If I had paid attention to it from the beginning, my final environment distribution could have been much more compact. ✗

Steps

Considering what has been discussed along this guide, what steps could be taken towards the direction of full reproducibility? Keep in mind that reproducibility is a cyclic and continuous effort. Thus, depending on the kind of work being done, the steps presented below might be taken in a different order.

1. Know your data: download, transfer, copy – whatever needs to be done – as soon as possible and start working with it! Understanding the data makes everything easier.

2. Set your workflow: workflow is planning. And planning is essential. Learn how to use a workflow tool and use it from the beginning. Make it part of your routine.

3. Code fluently: I am not telling you to master the programming language you are going to use, but either know what each line of code does. That will make you the master of **your** code.

4. Start writing right away: as soon as you start coding, start documenting. That might be annoying, but it is necessary. After all, the documentation is what you are going to show to your peers. And synchronizing it with the code will make it easier to manage when it gets big.

5. Make it reproducible: environmental tools are the best. They save you from having to create installation scripts that break all the time and are not always compatible with your peers' environment. *I.e.* containers and virtual machines might save you from exchanging lots of messages.

Conclusion

In conclusion, and fairly obviously: planning is key. That might be hard to fully achieve, but one should be as forward as possible when the intention is to develop a reproducible research. Luckily, there are various solutions for reproducible research when the subject is computational work. The challenge lies on choosing which one to use and, more importantly, how to use them, since some of them are not really straightforward.

Working on this project will certainly drive my future works towards reproducibility and as a plus, gave me some nice tools to guarantee my own environment configurations – either for when I need to run a project that has not been touched for a month, or to execute the same tests a thousand times within a week.

References

[1] X. Danaux and D. Gandy. The fontawesome package. http://ctan.dcc.uchile.cl/fonts/fontawesome/doc/fontawesome.pdf, May 2016.

[2] Docker. Docker documentation. https://docs.docker.com/.

[3] figshare. Figshare. https://figshare.com/.

[4] F. S. Furusato. Fernando furusato's reproducible work. https://github.com/ferseiti/reproducibility.