


Templating Organization of Scientific Research for Reproducibility at Project Inception

Griffin Chure

Department of Biology and Biological Engineering
California Institute of Technology, Pasadena, CA, USA

gchure@caltech.edu

 0000-0002-2216-2057

February 3, 2019

Abstract

The ubiquity of computation in modern scientific research inflicts new challenges for reproducibility. Journals are now requiring that raw (and sometimes processed) experimental data readily retrievable and the computer code used to manipulate said data can be easily accessed, yet this is often done by stating that it is “available upon request” or is haphazardly placed in a public repository on archival and version control service providers such as Zenodo or GitHub. In this writing, I put forward an field-agnostic template for organizing scientific research from project inception with the ultimate goal of reproducibility at the forefront of the research. This template is made publicly available as GitHub repository (DOI XXX) and is accompanied with usage instructions. The organizational structure described here imposes a strict delineation between four major levels of interaction with data: generation, exploration, interpretation, and presentation. The separation between these levels ensures that the processes used to generate or transform the data are clearly identified and are separate from all downstream interaction with the data. A beneficial consequence of this structure is that all code used for a particular experiment or analysis is directly associated with the reported results, ensuring human readability. Finally, I describe my own experience with using this architecture for several research projects and reflect upon how it has changed the way I approach new research questions.

“There is a reproducibility crisis in science” is a sentence guaranteed to illicit a strong response from whomever it is inflicted upon. Regardless of whether such a crisis exists (cite), it is an objective truth that improvements can be made in how science is performed, presented, and reviewed such that other scientists, whether a savant or a neophyte, could reproduce the analysis, generation, and interpretation of your research findings, assuming they have the means to.

Over the past 10 years, numerous articles have been written putting forth opinions on what reproducible research looks like. As of late the concept of a “future paper” in which figures are interactive, code is executable in-browser, and data can be easily downloaded has become popular and advocated for as for what the next generation of science will look like. However, there is almost never a discussion of how one can get to this finish line. I whole-heartedly agree that the

“future paper” is coming, but focusing on these examples without ever chronicling

All fields of science are becoming more quantitative and, as a consequence, relying on computation in the generation, processing, interpretation, and presentation of data. In my experience, the programs written to perform the aforementioned tasks are tailor-made to fit that particular problem. Unfortunately, these bits of code are typically closer to a Frankenstein creature than a tailor-made suit. Hard-coded numbers, ambiguously named variables, and absent (or flat-out wrong) documentation are the most common afflictions found lurking in the code base. While this writing is not meant to educate on how to write scientific software, merely organizing the codebase by separating the *executed code* from the *defined functions* would go a long way towards making the analysis understandable, and therefore reproducible.

As of this writing, I am nearing the end of my PhD and find myself reflecting on what I’ve learned (it’s quite a lot), and more importantly, what I could have done better. Over the course of my graduate research, I’ve had the immense privilege to be involved in projects so-called “scientific socialism” where first authorship of our work was split amongst five graduate students (??). It was during this first collaboration (and my very first scientific paper) that I began to put together what I thought would be the “ideal” layout to keep our data, processing scripts, and statistical analysis. What we developed was far from ideal, but a good first step. Over the next several projects, I adapted and tuned this organizational scheme to something that was much closer to ideal. What follows is a summary of this layout and the rationale behind it.

A Field-Agnostic Project Architecture

Modern scientific research relies on

Data Interaction and Executed Code

Processing

The first folder in Fig. 1 within the project root is the **processing**. This houses all operations which generate the data (such as executing simulations or image processing) or transform the data to a form that can be analyzed (such as “munging” data from an instrument or as a result of web-scraping). Each experiment or class of data (depending on your particular research) is separated into its own folder which houses the computer code executed by the user to perform the data processing. The end result from the processing is stored within that particular experiment folder. This protects against overwriting or accidental deletion of data from another experiment. In addition, the spatial association with the executed code for that particular experiment helps make the structure readable by humans and computers alike.

The code executed for each experiment should be written such that a minimal set of parameters are changed between different experiments.

This is made easier through the generation of processing templates, which will be discussed later.

Analysis

Once your data is processed, analysis through direct manipulation of the data or implementation of statistical inferential models is typically the next step. All computer code or computational notebooks (such as Jupyter Notebooks, Mathematica Notebooks, or MATLAB LiveScripts) used to interpret data and draw conclusions live in the **analysis** folder.

Exploration

Failure is a necessary step of the scientific process. Data analysis routines, troubleshooting, and even question development often require time spent with the data trying new things. The **exploratory** folder serves as a home to all of these processes. In my experience,

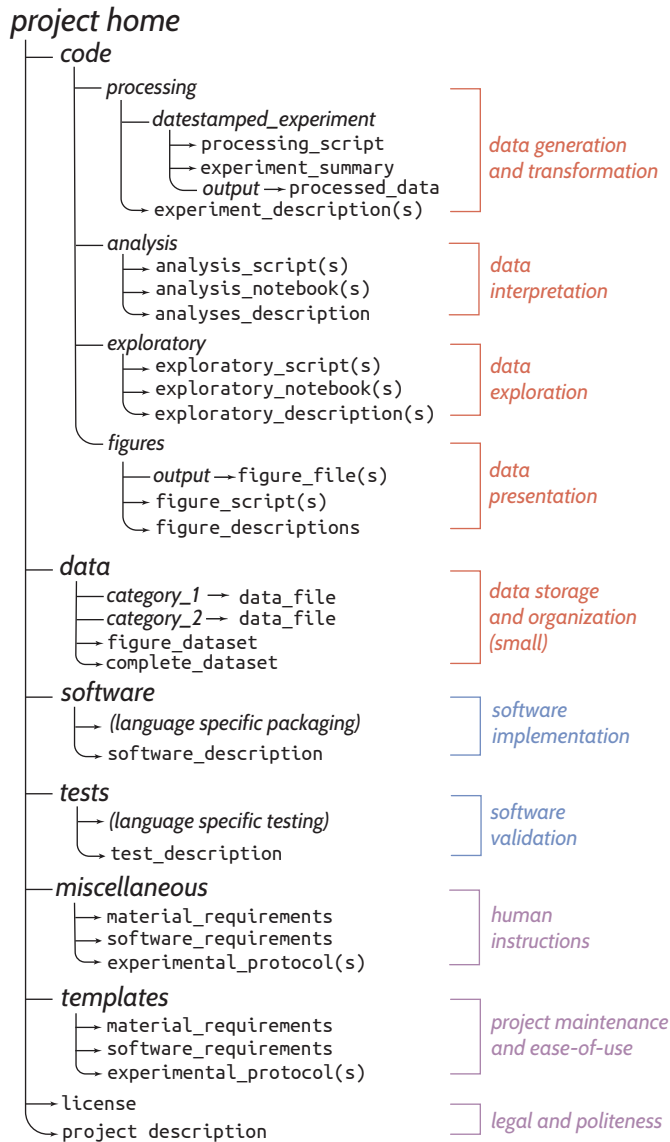


Fig. 1: A template for reproducible scientific research. Interaction with data is separated into four separate classes. All executed computer code in these sections point to data stored in another section of the project root folder. All executed code (run directly by the user) is located within code whereas the custom written code is packaged as a software module. Finally, all instructions for those interested in reproducing the results are defined separately from the data. In the file tree, folders are indicated in italicized text (e.g. *code*). Files are displayed in fixed-width font and are preceded by an arrow (e.g. → `license`). Items in parenthesis depend on the choice of programming language are left unspecified.

Presentation

Storage and organization of (not big) data

Implementation and validation of home-grown software

Making the technical human readable

Making life easy

Protecting your work and staking your claim

Discussion

Separation of human-readable & machine-readable

Separation of execution and definition of code

“It Tolls For Thee”

*“Perchance he for whom this bell tolls may be so ill, as that
he knows not it tolls for him; and perchance I may think my-
self so much better than I am, as that they who are about
me, and see my state, may have caused it to toll for me, and
I know not that.”*

John Donne

Acknowledgements

Justin Bois, Rob Phillips, Manuel Razo-Mejia, Shyam Saladi, Zofii Kaczmarek, Nathan M. Belliveau, Stephanie L. Barnes, (Ask Gail Clement, Tom Morell, Victoria Stodden, Dave VanValen, Quincey Justman for comment).