My Final College Paper

A Thesis
Presented to
The Division of Statistical and Data Sciences
Smith College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Audrey M. Bertin

May 2021

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class (for LaTeX) and the R bookdown package, in general.

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Second paragraph of abstract starts here.

Dedication

You can have a dedication here if you wish.

Chapter 1

If you have more two advisors, un-silence line 7

Placeholder

Chapter 2

An Introduction to Reproducibility

2.1 What is reproducibility?

2.2 Why is reproducibility important?

2.3 Current Discussion on Reproducibility in Academia

Chapter 3

Addressing the Challenges of Reproducibility

3.1 Review Of Previous Work

3.1.1 Literature

Publications on reproducibility can be found in all areas of scientific research. However, as Goodman, Fanelli, & Ioannidis (2016) argue, the language and conceptual framework of research reproducibility varies significantly across the sciences, and there are no clear standards on reproducibility agreed upon by the scientific community as a whole. We consider recommendations from a variety of fields and determine the key aspects of reproducibility faced by scientists in different disciplines.

Kitzes, Turek, & Deniz (2017) present a collection of case studies on reproducibility practices from across the data-intensive sciences, illustrating a variety of recommendations and techniques for achieving reproducibility. Although their work does not come to a consensus on the exact standards of reproducibility that should be followed, several common trends and principles emerge from their case studies: 1) use clear separation, labeling, and documentation, 2) automate processes when possible, and 3) design the data analysis workflow as a sequence of small steps glued together, with outputs from one step serving as inputs into the next. This is a common suggestion within the computing community, originating as part of the Unix philosophy (Gancarz (2003)).

Cooper et al. (2017) focus on data analysis in `R` and identify a similar list of important reproducibility components, reinforcing the need for clearly labeled, well-documented, and well-separated files. In addition, they recommend publishing a list of dependencies and using version control. Broman (2019) reiterates the need for clear naming and file separation while sharing several additional suggestions: keep the project contained in one directory, use relative paths, and include a `README`.

The reproducibility recommendations from R OpenSci, a non-profit initiative founded in 2011 to make scientific data retrieval reproducible, share similar principles to those discussed previously. They focus on a need for a well-developed file system, with no extraneous files and clear labeling. They also reiterate the need to note dependencies and use automation when possible, while making clear a suggestion not present in the previously-discussed literature: the need to use seeds, which allow for the saving and restoring of the random number generator state, when running code involving randomness (Martinez et al. (2018)).

When considered in combination, these sources provide a well-rounded picture of the components important to research reproducibility. Using this literature as a guideline, we identify several key features of reproducible work. These recommendations are a matter of opinion—due to the lack of agreement on which components of reproducibility are most important, we select those that are mentioned most often, as well as some that are mentioned less but that we view as important.

1. A well-designed file structure:

- Separate folders for different file types.
- No extraneous files.
- Minimal clutter.

2. Good documentation:

- Files are clearly named, preferably in a way where the order in which they should be run is clear.
- A README is present.
- Dependencies are noted.

3. Reproducible file paths:

- No absolute paths, or paths leading to locations outside of a project's directory, are used in code—only portable (relative) paths.

4. Randomness is accounted for:

- If randomness is used in code, a seed must also be set.

5. Readable, styled code:

- Code should be written in a coherent style. Code that conforms to a style guide or is written in a consistent dialect is easier to read (Hermans & Aldewereld (2017)). We believe that the `tidyverse` provides the most accessible dialect of `R`.

Much of the available literature focuses on file structure, organization, and naming, and `fertile`'s features are consistent with this. Marwick, Boettiger, & Mullen (2018)} provide the framework for file structure that `fertile` is based on: a structure similar to that of an `R` package (R-Core-Team (2020), Wickham (2015)), with an `R` folder, as well as `data`, `data-raw`, `inst`, and `vignettes`.

3.1.2 R Packages and Other Software

Much of this work is highly generalized, written to be applicable to users working with a variety of statistical software programs. Because all statistical software programs operate differently, these recommendations are inherently vague and difficult to implement, particularly to new analysts who are relatively unfamiliar with their software. Focused attempts to address reproducibility in specific certain software programs are more likely to be successful. We focus on `R`, due to its open-source nature, accessibility, and popularity as a tool for statistical analysis.

A small body of `R` packages focuses on research reproducibility. `rrtools` (Marwick (2019)) addresses some of the issues discussed in Marwick et al. (2018) by creating a basic `R` package structure for a data analysis project and implementing a basic `testthat::check` functionality. The `orderly` (FitzJohn et al. (2020)) package also focuses on file structure, requiring the user to declare a desired project structure (typically a step-by-step structure, where outputs from one step are inputs into the next) at the beginning and then creating the files necessary to achieve that structure. `workflowr`'s (Blischak, Carbonetto, & Stephens (2019)) functionality is based around version control and making code easily available online. It works to generate a website containing time-stamped, versioned, and documented results. `checkers` (Ross, DeCicco, & Randhawa (2018)) allows you to create custom checks that examine different aspects of reproducibility. `packrat` (Ushey, McPherson, Cheng, Atkins, & Allaire (2018)) is focused on dependencies, creating a packaged folder containing a project as well as all of its dependencies so that projects dependent on lesser-used packages can be easily shared across computers. `drake` (OpenSci (2020)) analyzes workflows, skips steps where results are up to date, and provides evidence that results match the underlying code and data. Lastly, the `reproducible` (McIntire & Chubaty (2020)) package focuses on the concept of caching: saving information so that projects can be run faster each time they are re-completed from the start.

Many of these packages are narrow, with each effectively addressing a small component of reproducibility: file structure, modularization of code, version control, etc. These packages often succeed in their area of focus, but at the cost of accessibility to a wider audience. Their functions are often quite complex to use, and many steps must be completed to achieve the required reproducibility goal. This cumbersome nature means that most reproducibility packages currently available are not easily accessible to users near the beginning of their `R` journey, nor particularly useful to those looking for quick and easy reproducibility checks. A more effective way of realizing widespread reproducibility is to make the process for doing so simple enough that it takes little to no conscious effort to implement. You want users to "fall into a hole" (we paraphrase Hadley Wickham) of good practice.

`Continuous integration` tools provide more general approaches to automated checking, which can enhance reproducibility with minimal code. For example, `wercker`—a command line tool that leverages Docker—enables users to test whether their projects will successfully compile when run on a variety of operating systems without access to the user's local hard drive (Oracle Corporation (2019)). `GitHub Actions` is integrated into GitHub and can be configured to do similar checks on projects hosted in repositories. `Travis CI` and `Circle CI` are popular continuous integration tools that can also be used to check `R` code.

However, while these tools can be useful, they are generalized so as to be useful to the widest audience. As a result, their checks are not designed to be `R`-specific, which makes them sub-optimal for users looking to address reproducibility issues involving features specific to the `R` programming language, such as package installation and seed setting.

3.2 Identifying Gaps In Existing Solutions

3.3 My Contribution: `fertile`, An R Package Creating Optimal Conditions For Reproducibility

3.4 How `fertile` Works

3.5 `fertile` in Practice: Experimental Results From Smith College Student Use

Chapter 4

Incorporating Reproducibility Tools Into The Greater Data Science Community

4.1 Potential Applications of `fertile`

4.2 Integration Of `fertile` And Other Reproducibility Tools in Data Science Education

Conclusion

If we don't want Conclusion to have a chapter number next to it, we can add the `{-}` attribute.

More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendix A

The First Appendix

This first appendix includes all of the R chunks of code that were hidden throughout the document (using the `include = FALSE` chunk tag) to help with readibility and/or setup.

In the main Rmd file

In Chapter ??:

Appendix B

The Second Appendix, for Fun

References

Placeholder

Blischak, J., Carbonetto, P., & Stephens, M. (2019). Workflowr: A framework for reproducible and collaborative data science. Retrieved from `https://CRAN.R-project.org/package=workflowr`

Broman, K. (2019). Initial steps toward reproducible research: Organize your data and code. *Sitewide ATOM*. Retrieved from `https://kbroman.org/steps2rr/pages/organize.html`

Cooper, N., Hsing, P.-Y., Croucher, M., Graham, L., James, T., Krystalli, A., & Michonneau, F. (2017). A guide to reproducible code in ecology and evolution. *British Ecological Society*. Retrieved from `https://www.britishecologicalsociety.org/wp-content/uploads/2017/12/guide-to-reproducible-code.pdf`

FitzJohn, R., Ashton, R., Hill, A., Eden, M., Hinsley, W., Russell, E., & Thompson, J. (2020). Orderly: Lightweight reproducible reporting. Retrieved from `https://CRAN.R-project.org/package=orderly`

Gancarz, M. (2003). *Linux and the unix philosophy* (2nd ed.). Woburn, MA: Digital Press.

Goodman, S. N., Fanelli, D., & Ioannidis, J. P. A. (2016). What does research reproducibility mean? *Science Translational Medicine*, *8*(341), 1–6. `http://doi.org/10.1126/scitranslmed.aaf5027`

Hermans, F., & Aldewereld, M. (2017). Programming is writing is programming. In *Companion to the first international conference on the art, science and engineering of programming* (pp. 1–8).

Kitzes, J., Turek, D., & Deniz, F. (2017). *The practice of reproducible research: Case studies and lessons from the data-intensive sciences*. Berkeley, CA: University of California Press. Retrieved from `https://www.practicereproducibleresearch.org`

Martinez, C., Hollister, J., Marwick, B., Szöcs, E., Zeitlin, S., Kinoshita, B. P., … Meinke, B. (2018). Reproducibility in Science: A Guide to enhancing reproducibility in scientific results and writing. Retrieved from `http://ropensci.github.io/reproducibility-guide/`

Marwick, B. (2019). Rrtools: Creates a reproducible research compendium. Retrieved from `https://github.com/benmarwick/rrtools`

Marwick, B., Boettiger, C., & Mullen, L. (2018). Packaging data analytical work reproducibly using R (and friends). *The American Statistician*, *72*(1), 80–88. `http:`

//doi.org/doi.org/10.1080/00031305.2017.1375986

McIntire, E. J. B., & Chubaty, A. M. (2020). Reproducible: A set of tools that enhance reproducibility beyond package management. Retrieved from https://CRAN.R-project.org/package=reproducible

OpenSci, R. (2020). Drake: A pipeline toolkit for reproducible computation at scale. Retrieved from https://cran.r-project.org/package=drake

Oracle Corporation. (2019). Wercker. Retrieved from https://github.com/wercker/wercker

R-Core-Team. (2020). Writing r extensions. *R Foundation for Statistical Computing*. Retrieved from http://cran.stat.unipd.it/doc/manuals/r-release/R-exts.pdf

Ross, N., DeCicco, L., & Randhawa, N. (2018). Checkers: Automated checking of best practices for research compendia. Retrieved from https://github.com/ropenscilabs/checkers/blob/master/DESCRIPTIONr

Ushey, K., McPherson, J., Cheng, J., Atkins, A., & Allaire, J. (2018). Packrat: A dependency management system for projects and their r package dependencies. Retrieved from https://CRAN.R-project.org/package=packrat

Wickham, H. (2015). *R packages* (1st ed.). Sebastopol, CA: O'Reilly Media, Inc.