Reproducible Research Overview

Ben Buzzee, Biometrician I

May 21, 2018

Reproducible Research

For conclusions from a scientific study to be trustworthy, they must be verifiable. All too often, however, the conclusions of scientific studies cannot be verified because the data are not available, the methods are not fully described, or the software researchers used is inaccessible. The goal of the reproducible research movement is to conduct and document research in a manner that allows third parties to fully replicate the results of a study. This allows other researchers to fully understand the methods used and verify the correctness of any conclusions. Using a reproducible methodology also allows for easier collaboration and smoother project hand-offs between team members. This paper will focus on computational and statistical reproducibility.

Components of Reproducible Research

1. The Data

Raw data is the starting point of any statistical analysis and is essential to verifying statistical results from beginning to end. Often, however, the raw data is messy, meaning it is not in a form that can be easily analyzed. As such, the first task in a statistical analysis is often converting the raw dataset into a "tidy" one. A "tidy" dataset is a dataset of a particular form that makes analysis and visualizations relatively easy and straightforward. According to Wickham (2014), a tidy dataset is one where:

- Each column corresponds to one variable
- Each row represents one observational unit
- Data from different types of observational units are stored in different data tables

When converting the raw data to a tidy dataset, it is important not to overwrite the original 'messy' dataset. Both the raw data and the clean data should be saved as separate files. Sometimes mistakes are made when converting the raw data to a useable format, and without the raw data, this cannot be checked for. Other researchers may also choose to process the raw data in a different manner, which requires the original raw data be kept intact.

2. The Code

The code used to conduct an analysis represents a complete description of all calculations and data manipulations. Without the code, a researcher would have to rely on general descriptions of calculations and will rarely know exactly how the analysis was performed. Saving the code and using relative file paths allows a third party to simply download the data and code and immediately run the code for themselves. For example, the script used to convert the raw data to a tidy dataset should be saved so that anyone can download the raw data and cleaning script and then be able to run the script to create the tidy dataset on their own computer.

3. Documentation

A dataset without a description of what the variables represent or how the data were collected is useless to those not involved in its creation. Similarly, code that cannot be human-read or followed is of little help when it comes to understanding or verifying an analysis. Both the data and the code used must be well documented in order for the analysis to be reproducible.

Data should always be accompanied by "metadata." Metadata is a complete description of the data–the who, what, where, when, why, and how behind the data. The metadata file should allow parties not involved in the study or data collection process to fully understand what everything in the raw data file represents and why it is there. The metadata should also explain the sampling or experimental protocols used to collect the data. Creating and saving a metadata file with your dataset also makes it possible for the data to be reused for other purposes beyond the original intentions, even after long periods of time have passed.

In addition to documenting the data, the code itself must be documented. Code should be well commented and written in a manner that is conducive to human reading. Custom functions should also be accompanied by appropriate descriptions and instructions. Writing readable code makes the code less prone to error, easier to understand, easier to debug and improve, and makes collaboration a smoother process. A reader should be able to open a script without any prior knowledge of it and in a reasonably short amount of time have a general understanding of what the code does and how it does it. See http://adv-r.had.co.nz/Style.html for a simple guide to writing readable R code.

For highly technical code, it may be better to write code in an Rmarkdown or Notebook format. Rmarkdown and Notebook documents allow code and output to be interwoven with well-formated sections of text. This can significantly improve the readability of technical documents and allows for (and even encourages) explanations to be more thorough.

4. Organization and Accessibility

Finally, all of the above components need to be organized and accessible. Ideally, any individual wishing to reproduce the results of a particular study should be able to quickly find the data and code used for that study in an archive or repository, and then immediately be able to identify the purpose and location of all the major files and documents.

When dealing with large projects, file clutter can quickly become overwhelming. In addition to slowing down personal workflows, clutter can prevent others from understanding and reviewing code in a time efficient manner. One way to avoid this is to start projects with a consistent, well-organized directory structure. Consistently using a directory structure allows anyone loosely familiar with that structure to immediately know where to look for files based on their purpose. Finding the files in the first place is a critical step in reproducing research.

Finally, to make a project accessible to everyone, it needs to be hosted online. Options range from a formal dedicated archive to simple GitHub repositories. Regardless of where the project is stored, it needs to have an informative name, be tagged with appropriate keywords, and be searchable so that others can find it.

Implementation: GitHub and R Packages

R: Projects and Packages

One way to conveniently bundle components 1 through 3 above is through the use of R Projects and R Packages. An R Project is a convenient way to pair projects up with their own working directory. A working directory is the default folder R will look to when reading in or writing out files. For example, the command write.csv(x, "clean_data.csv") will write convert x to a .csv file called "clean_data" and output it straight to the working directory. By using an R project, all files can be easily managed through the same working directory. This is particularly helpful when sharing code. If all files are kept within the project folder and paths are defined relative to the working directory, it becomes easy for anyone to download the R project and immediately run the code without any file path related errors.

After an R project has been created to aid in organization and shareability, an R package can be created within the project. An R package is a bundle of code, data, and documenation that allows an analysis to be entirely portable. A simple R-package contains an /R folder which contains custom function definitions, a /data folder for data, a /man (short for manual) folder containing documentation for all the functions defined in /R, and one or more folders for analysis using the aforementioned functions and data. After a package has been installed and loaded into the environment, the user can immediately use all functions and data defined in the /R and /data folders. Additionally, a user can type ?name_of_function or ?name_of_data to see documentation that fully describes the object and how to use it with instructions and examples.

Here is an example of what a typical project directory may look like, where / denotes a folder:

- /Project
  - /R
  - /data-raw
  - /data
  - /final_analysis
  - /man
  - /op_plan
  - project.Rproj
  - readme.md

GitHub and GitHub Organizations

Finally, the R Project (containing a package) needs to be put somewhere so that others can access it. Github is a free resource that allows users to create project-specific repositories to store code and other files in. An entire R Project can be uploaded as it's own repository and be made immediately available to anyone with internet acess.

'GitHub Organizations' is a feature that allows multiple users to share the same organizational structure associated with an individual account. Each member of a github organization can create, delete, or modify any of the repositories in the organization. This allows members of the organizations to work together or separately, and contribute their work to a single location that all members have access to. The github organization account also allows repositories to be tagged with keywords and searched for, so as long as appropriate naming schemes and tagging protocols are used, any project can be quickly found by anyone.

Topics worth reading into:

Makefiles precisely describe the relationship between inputs, outputs, and scripts, and allows for easy re-building of the final results.

Docker allows for the isolation of the computational environment. Software will inevitably change over time–sometimes breaking old code. Docker prevents that by also saving the version of software that was used.

Archiving: Github may not be around forever, and outside researchers probably won't search for data they could use there. A formal archive would probably ensure the data stays openly available long-term and make it easier for outside researchers to find and reuse the data for other purposes. We could just supplement our use of github by occasionally submitting projects we think are important or could be used by others to an archive.

Sources

Wickham 2014. https://www.jstatsoft.org/article/view/v059i10/

Similar outline of R packages for reproducible research: https://peerj.com/preprints/3192/

Guide to organizing projects that don't quite need an R pacakge: https://nicercode.github.io/blog/2013-04-05-projects/

GitHub Organizations Intro: https://blog.github.com/2010-06-29-introducing-organizations/