“Yes We Can!”: A Practical Approach to Teaching Reproducibility to Undergraduates

Richard Ball

This paper presents a sequence of four exercises illustrating how students can be introduced to reproducible methods of data processing and analysis in a series of small steps. Each step is modest and feasible even in introductory classes, but cumulatively they allow students to achieve state of the art standards of reproducibility. By demonstrating the feasibility of teaching reproducible methods to beginning students, these exercises support the assertion that reproducibility can and should be integrated into quantitative methods training at all levels of the curriculum.

I thank Lars Vilhuber and Aleksandr Michuda for the opportunity to participate in the Conference. I acknowledge my debt to Barak Obama and Robert T. Builder for the inspirational slogan I have borrowed from them to use in the title of this paper. The ideas about teaching reproducibility in this paper have been developed over more than a decade of collaboration with Norm Medeiros.

## Background

Is it feasible to include reproducible research methods in undergraduate training in quantitative data analysis? There are reasons to believe the answer to that question is “no”—that reproducibility is an advanced topic best left to graduate school or early career training. Professional standards such as the [AEA Data Editor’s guidelines](https://aeadataeditor.github.io/aea-de-guidance/) and the [World Bank Development Impact Evaluation (DIME) manual](https://worldbank.github.io/dime-data-handbook/) may appear too technical and complex to introduce to undergraduates. Even the [TIER Protocol](https://www.projecttier.org/tier-protocol/protocol-4-0/), which was designed to be accessible to students at all levels, is elaborated with a degree of specificity and detail that could give the impression that incorporating reproducibility into undergraduate classes and research supervision would be a costly and disruptive undertaking.

This essay argues that, on the contrary, integrating reproducibility into the undergraduate curriculum is eminently feasible. To support this claim, we develop a simple exercise of the kind that might be assigned in an introductory quantitative methods class, and then present four versions of the exercise: a baseline in which the issue of reproducibility is entirely neglected, and three subsequent versions that incrementally introduce essential elements of reproducibility. The additional skills students must acquire for each version of the exercise are modest, but cumulatively they prepare students in computational methods that achieve state of the art standards of reproducibility. These exercises demonstrate the feasibility of teaching reproducibility to undergraduates, and provide instructors with concrete examples of small, practical steps they can take to achieve that goal.

## Main Thoughts

### The Exercise

In all versions of the exercise, students are given an extract of data from the 2018 American Community Survey (citation), and use it to compare average incomes of prime working-age workers by race and sex. The computational tasks are (i) to construct a table showing the means of total income for groups defined by race and sex, and (ii) illustrate those group means in a bar graph. Students then write a report in which they present the table and bar graph, and comment on the patterns they observe.

The report students submit for all four versions are identical. The versions differ in the extent to which students adopt practices that enhance the reproducibility of their results, and in the documentation that is submitted with the report.

**Version 1: Interactive and non-reproducible.** In this baseline version, the issue of reproducibility is entirely ignored. Students open the data file by double-clicking, and then generate the table and bar graph using a menu-driven GUI or by typing commands interactively. They use a word processor to write the report, and insert the table and graph by copying-and-pasting the output displayed on their monitor. The only work students turn in is a single document—the report.

**Version 2: Writing scripts, the project folder, and the working directory.** Scripts are fundamental to reproducible research: it is by executing scripts written and preserved by the author of a study that interested readers are able to reproduce the results.

Instructors and students accustomed to an interactive workflow are often reluctant to adopt reproducible methods because they perceive learning to write code and work with scripts as a hurdle. But version 2 of the exercise shows that the hurdle is not as high as it might appear. Students need not master a programming language to get started: learning the syntax of a few basic commands is sufficient to begin working with scripts and take meaningful steps toward reproducibility.

Version 2 is identical to the non-reproducible version 1, except that instead of interactively typing commands or using menus, students write a script that includes all the commands needed to open the data file and generate the table and bar graph. As in version 1, students write the report with a word processor and copy and paste the results from their monitor into the report.

Because the data file is opened by a command in the script (rather than by double-clicking), it is necessary to be explicit about where the data file is stored and which folder is designated as the working directory. The instructions for version 2 advise students to follow a very simple convention to ensure the software can find the data file:

* All the files for the exercise—the data file, the script, and the report—should be stored in a single folder, which is referred to as the project folder.
* Before executing the script, the user should designate the project folder as the working directory for their software.

The instructions to version 2 also provide guidance on several best practices for writing scripts:

* **Headers.** Every script should begin with a header. Instructors may use their discretion to decide what information they ask students to include in the header for any particular script, but typically headers provide information such as the date, the name of the person writing the script, and a description of the purpose of the script. It is also useful to include a note in the header indicating to the user which folder should be designated as the working directory when the script is executed.
* **Setup.** It is usually convenient to start a script with commands that (i) declare the version of the software being used, (ii) install any other software or add-ons that will be necessary, (iii) clear memory, and (iv) specify any relevant settings for the software.
* **Open the data.** The data file should be opened by a command in the script (not by double-clicking). The command that reads the data must come before any commands that manipulate or analyze the data.
* **Comments.** Throughout the script, it is essential to write detailed and informative comments explaining the purpose of each command. These comments will be helpful to any interested reader who chooses to explore the documentation for a project. Moreover, they are valuable to the students themselves: unless they include good comments in their scripts, they may have trouble deciphering code they wrote only a few days ago.

As in version 1, students write the report with a word processor, and copy and paste the results from their monitor into the report. In version 2, however, the work they submit consists not just the report, but their entire project folder, containing the data file, their script, and the report.

The instructor should then be able to reproduce the table and bar graph simply by launching the software, setting the working directory to the project folder, and executing the script.

**Version 3: Saving output.** In versions 1 and 2, students copy and paste output from their monitor into the report, but their results are not preserved in any other way. In version 3, students write additional code in the script that saves the results in two output files: a text file containing the table, and a graphics file containing the bar graph. As in version 2, students store the data file, their script, and their report in a single project folder, which is again designated as the working directory. Because the project folder is designated as the working directory, that is where the two output files are saved when they are generated.

Saving the output files makes it possible to automate the process of inserting the results into the report. Instead of using a word processor and copying and pasting, students can write the report in a markup language (like Markdown or LaTeX), embedding links to the output files at appropriate points in the text.

The work students submit for version 3 again consists of the entire project folder, but in this case the project folder contains not only the data file, script, and report, but the two output files as well.

**Version 4: The reproducibility trifecta: Folder hierarchy and relative directory paths.** Version 3 involves a number of files of several different types, all of which are stored together in a single project folder. In version 4, students add some structure by creating several subfolders inside the project folder and distributing the various files among them. The organizational scheme in version 4 is very simple:

* The report is stored in the top level of the project folder.
* Three new folders are created in the top level of the project folder: **Data**, **Scripts**, and **Output**.
  + The data file is saved in **Data**.
  + The script is saved in **Scripts**.
  + The output files are saved in **Output**.

For more complex projects, it is usually convenient build a more developed folder hierarchy, often including several levels of subfolders within the project folder. But the simple scheme used in version 4 is sufficient to introduce the key practices for achieving reproducibility given any folder structure adopted in a particular application.

When the files for a project are distributed in a hierarchy of folders within the project folder, the key to reproducibility lies in three practices that we refer to as the ***reproducibility trifecta***.

1. **Establish a well-defined folder hierarchy**.
   * All of the documentation for a project should be stored in a single project folder.
   * The project folder should contain a hierarchy of subfolders in which the various files are organized in some convenient and sensible way.
     + This structure should be established, and the hierarchy of folders (all initially empty) should be built, before work with the data begins.
     + The folders should then be populated with the data, scripts, and other files generated as work on the project progresses.
2. **Be explicit about the working directory.** 
   * For every script you write, choose one of your folders (either the project folder or one of the folders inside the project folder) to be designated as the working directory when the script is executed.
   * We recommend the following convention: Always designate the project folder as the working directory. When you, or an independent investigator interested in your project, launch the software to begin working with your scripts, they will need to check whether the project folder is designated as the working directory; if not, they will need to manually set the working directory to the project folder. After that, there should be no need to change the working directory again.[[1]](#footnote-1)
   * In the header for each script, include a note that (i) indicates which folder you have chosen as the working directory, and (ii) reminds the user to be sure that the chosen folder is in fact designated as the working directory before executing the script.
3. **Use relative directory paths**.
   * A relative directory path is a path through the folders on the computer you are using that begins in whichever folder has been designated as the working directory and leads to a target folder (from which, for example, you wish to open an existing file, or in which you wish to save a newly created file).
   * In your scripts, whenever you write a command in which you need to specify the location of a particular folder, you should do so using a relative directory path. You should not specify a directory path that begins in a particular folder on a particular computer (such as the C: drive on your computer).

The three elements of the reproducibility trifecta are interrelated: when you write a relative directory path, you must know what folder is designated as the working directory (that is where the relative directory path starts), and you must know the structure of the folder hierarchy (since the relative directory path must specify how to navigate through that hierarchy to the target folder). Beginning students need guidance about how to properly synchronize their folder and file structure, the choice of the working directory, and the relative directory paths they write in their scripts. But by introducing these concepts in simple setting, version 4 of the exercise makes it easy for them to grasp how the pieces fit together.

## Conclusion

### Standards of reproducibility

The reproducibility trifecta makes it possible to achieve two important standards of reproducibility, which we refer to as (i) ***(almost) automated reproducibility*** and (ii) ***portable reproducibility.***

Automated reproducibility means that, once a user has copied the project folder onto their own computer, the computations that generate and save the results can be reproduced just by running the scripts, with no need to do anything by hand (such as editing directory paths in scripts or moving files from one folder to another).

Synchronizing the folder hierarchy, working directory, and relative directory paths according the principles of the reproducibility trifecta ensures that automated reproduction is possible—almost. Before the scripts can be executed, there is one task the user needs to complete by hand, namely setting the working directory to whatever folder has been designated by the author. Hence the qualifier "almost" before the term "automated reproducibility".

The standard of portable reproducibility is that any user should be able to perform an (almost) automated reproduction of someone else’s project on their own computer. Provided they have the necessary software installed, they should be able to copy the project folder and all its contents onto their computer, and then (after setting the working directory as necessary) run the scripts that reproduce the results.

The key to achieving portable reproducibility is that all directory paths specified in the scripts must begin and end in folders on the user’s computer. Because the reproducibility trifecta specifies that the working directory should be set to the project folder (or one of its subfolders), and that folder locations should be given by relative directory paths beginning in the working directory, this condition is satisfied the moment a user copies the project folder onto their own computer.

(Almost) automated reproducibility and portability are state of the art standards for professional social science research; they are among the properties that leading conventions such as the AEA Data Editor’s guidelines and the DIME Manual are intended to achieve. The four versions of the exercise we have presented show that these professional standards can be introduced to students in introductory level classes via a sequence of modest, feasible innovations.

### Bells and whistles

To make the fundamental principles and practices as transparent as possible, we have presented a simple exercise that excludes a number of important elements of documentation. But once students have a foundation in the fundamentals, it is easy to introduce additional elements such as a read-me file, more complex directory structures, data citations, a master script, log files, and a data appendix, to name a few.

Instructors looking for a more substantial project that introduces many of these peripherals, but is still accessible to students in introductory courses, might consider the [Project TIER](http://www.projecttier.org) exercise titled  [“Animal House in Alcohol-Free Dorms?”](https://www.projecttier.org/tier-classroom/soup-nuts-exercises/#shorter-exercises-for-teaching-transparency-and-reproducibility:~:text=more%20to%20complete.-,The%20First%20Soup%2Dto%2DNuts%20Exercise%3A%20%22Animal%20House%20in%20Alcohol%2DFree%20Dorms%3F%22,-Download%20the%20first). When students move beyond structured exercises and begin research projects of their own, they may benefit from the [TIER Protocol](https://www.projecttier.org/tier-protocol/protocol-4-0/), which gives detailed guidance about the components of a comprehensive reproduction package. Examples of all the components of the documentation described in the TIER Protocol can be found in an accompanying [demo project](https://www.projecttier.org/tier-protocol/demo-project/).

TABLE 1: PROPERTIES OF THE FOUR VERSIONS OF THE EXERCISE

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Elements of reproducibility introduced** | **Work submitted by students** | **How the report is written** |
| Version 1: Interactive | None | A report (a *.pdf* document) | Text composed with a word processor; table and figure inserted by copying and pasting |
| Version 2: Scripted computations | * Writing all commands in an executable script * Keeping all files in a project folder * Designating the project folder as the working directory | A project folder, containing:   * a report (a *.pdf* document) * the data file * a script | Text composed with a word processor; table and figure inserted by copying and pasting |
| Version 3: Saving output | * All elements of version 2 * Writing additional commands in the script that save output files to the working directory | A project folder, containing:   * a report (a *.pdf* document) * the data file * a script * two output files | Text composed with a word processor; table and figure inserted by copying and pasting  or  Text composed in a markup language; table and figure imported from output files |
| Version 4: Assembling an (almost) automated, portable reproduction package | * The reproducibility trifecta:   + Establishing a well-defined folder hierarchy within the project folder   + Designating the project folder as the working directory   + In scripts, use relative directory paths to specify locations of specific folders | A project folder, containing:   * a report (a *.pdf* document) * a **Data** subfolder, containing the data file * a **Scripts** subfolder, containing a script * an **Output** subfolder, containing two output files | Text composed with a word  processor; table and figure  inserted by copying and pasting  or  Text composed in a markup  language; table and figure  imported from output files |

1. Another common practice is to write a macro in the script that defines an absolute directory path starting in some particular folder on the author’s computer, and then indicate that other users should edit the macro so that it indicates an absolute directory path starting in some folder on the user’s computer. It strikes us as easier and simpler just to agree that every user needs to manually set the working directory to the project folder once at the beginning of each session, and then use relative directory paths starting there to specify folder and file locations. [↑](#footnote-ref-1)