Harvard Data Science Review • Issue 5.3, Summer 2023

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

Richard Ball[1,2]

[1]Department of Economics, Haverford College, Haverford, Pennsylvania, United States of America,
[2]Project TIER, United States of America

Published on: Jul 27, 2023

DOI: https://doi.org/10.1162/99608f92.9e002f7b

Introduction

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 (2023) guidelines and the World Bank Development Impact Evaluation (DIME) manual (Bjarkefur et al., 2021) may appear too technical and complex to introduce to undergraduates. Even the TIER Protocol (Project TIER, 2023a), which was designed to be accessible to students at all levels, is elaborated with a degree of specificity and detail that could give instructors the impression that incorporating reproducibility into undergraduate classes and research supervision would be a costly and disruptive undertaking.[1]

This essay argues that, on the contrary, integrating reproducibility into the undergraduate curriculum is eminently feasible. To support this claim, we present a simple exercise of the kind that might be assigned in an introductory quantitative methods class, and then develop 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. The key features of each version of the exercise are summarized in Table 1.

Table 1. Properties of the four versions of the income comparison exercise.

Version	Elements of Reproducibility Introduced	Work Submitted by Students	How the Report Is Written
Version 1: Interactive and nonreproducible	None	A report (a *.pdf* document)	Text composed with a word processor or markup language; table and figure inserted by copying and pasting
Version 2: Scripts, the project folder, and the working directory	• 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 or markup language; 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: The reproducibility trifecta	• Establishing a well-defined folder hierarchy within the project folder • Designating the project folder as the working directory • In scripts, using 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

The discussion in the text of this article is software-neutral. The methods and principles introduced can be applied by users of any scriptable statistical software, including open source programs like R and Python, as well as commercial packages like MATLAB, SAS, SPSS, and Stata.[2] Concrete illustrations are provided by a suite of examples available in an online supplement. Each example includes the instructions that would be given to students, as well as models of the work students would submit, for all four versions of the exercise. A number of variations of the examples illustrate how the exercise manifests with different types of software (R and Stata) and with different choices about workflow (in particular, different choices about how the working directory is managed). Although each variation is based on particular choices about software and workflow, the purpose of showing a range of examples embodying different choices is to demonstrate the general applicability of the methods being presented.

The Exercise

In all versions of the exercise, students are given an extract of data from the 2018 American Community Survey (Ruggles et al., 2023; U.S. Census Bureau, 2019), 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) display 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 reports 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 nonreproducible.

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 graphical user interface (GUI) or by typing commands interactively. Students may write the report using the word processor or markup language of their choice; they insert the table and graph into the report by copying and pasting output displayed on their monitor. The only work students turn in is a single document—the report.

Version 2: Scripts, the project folder, and the working directory.

Version 2 is identical to the nonreproducible 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.

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. Unfortunately, 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. 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 learning fundamental principles of reproducibility.

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. In version 2, these issues are resolved very simply: the instructions tell students to store all the files for the exercise—including the data file—in a single project folder, and the project folder is designated as the working directory.

Several different methods of managing the working directory are possible, and instructors can advise students to adopt whatever approach is best suited to their particular circumstances. The exercises in the online supplement illustrate two such methods: manually setting the working directory, and creating a project (e.g., an RStudio project file with a *.Rproj* extension, or a Stata project icon with a *.stpr* extension).[3] Regardless of the method chosen to manage the working directory, we recommend adopting the convention of designating the project folder as the working directory at all times.

The instructions for version 2 also provide guidance on several recommended 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, 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 their choice of word processor or markup language, and copy and paste the results from their monitor into the report. In version 2, however, the work they submit consists not just of 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, ensuring (by whatever method has been adopted for the exercise) that the project folder is designated as the working directory, 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. Because the project folder is designated as the working directory, that is where the two output files are saved when they are generated.

If students have learned how to compose documents in a markup language, saving the output files makes it possible to automate the process of inserting the table and bar graph by embedding links to the output files at appropriate points in the source file. For students who use word processors, copying and pasting is of course still feasible.

The work students submit for version 3 again consists of the entire project folder, which in this case contains five files: the data file, the script, the report, and the two output files.

Version 4: The reproducibility trifecta: Folder hierarchy, the working directory, 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 within the project folder and distributing the various files among them. The organizational scheme prescribed 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 often convenient to build a more developed folder hierarchy, often including several levels of subfolders within the **Data**, **Scripts**, and **Output** folders. But the simple scheme used in version 4 is sufficient to introduce three practices that are the key to achieving reproducibility given any folder structure adopted in a particular application. We refer to these three practices 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 acquired or generated as work on the project progresses.

2. **Be explicit about the working directory.**
   - For every script you write, choose one of your folders to be designated as the working directory when the script is executed.
     - We recommend making the main project folder the working directory, but you could also choose any subfolder within the project folder.
   - Each time you, or someone else interested in your project, begins a session, they must ensure the folder you have chosen is in fact designated as the working directory.
     - Use whatever method you prefer to set the working directory (e.g., interactively setting the working directory, creating a project, or writing a directory path macro).
   - After the working directory has initially been set to the chosen folder, there should be no need to change it. Do not change the working directory interactively or write change directory commands in your scripts.
   - In the header for each script, include a note that informs the user of the conventions you have decided upon for managing the working directory and reminds them to be sure that the correct folder is in fact designated as the working directory before the script is executed.
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).[4]

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 a simple setting, version 4 of the exercise makes it easy for them to grasp how the pieces fit together and prepares them to deal with more complex projects.

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 making new folders, moving files from one folder to another, or editing any scripts).

Synchronizing the folder hierarchy, working directory, and relative directory paths according to 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, ensuring that the working directory is set to whatever folder has been designated by the author. Depending on the method chosen for managing the working directory, this initial task may be to manually set the working directory to the chosen folder, launch the session by double-clicking on a project file, or editing a macro defining a path to the chosen folder. The qualifier ‘almost’ before the term ‘automated reproducibility’ reflects the fact that the user must take some such preliminary action before executing the scripts that reproduce the computations.

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 their hardware is adequate and 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 additional features, but is still accessible to students in introductory courses, might consider the Project TIER exercise titled “Animal House in Alcohol-Free Dorms?” (Project TIER, 2023b). A notable feature of that exercise is that students are not provided with clean data that is ready to be analyzed; instead, they are instructed to download a raw data set from the website of a research data archive. They then write both a processing script that cleans and organizes the raw data as necessary to prepare it for analysis and a script that executes the analysis and saves the results.

When students move beyond structured exercises and begin research projects of their own, they may benefit from the TIER Protocol (Project TIER, 2023a), 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 (Project TIER, 2023c).

Acknowledgments

This article is based on a talk given for a panel titled “Should Teaching Reproducibility Be a Part of Undergraduate Education or Curriculum?” at the 2022 Annual Meeting of the Southern Economic Association, Ft. Lauderdale, FL, November 20, 2022. This panel was one of a series of events comprising the Conference on Reproducibility and Replicability in Economics and the Social Sciences, organized by Lars Vilhuber and Aleksandr Michuda.

I thank Lars Vilhuber and Aleksandr Michuda for the opportunity to participate in the conference. I acknowledge my debt to Barack Obama and Robert T. Builder for the inspirational slogan I have borrowed to use in the title of this article. The ideas about teaching reproducibility in this article have been developed over more than two decades of collaboration with Norm Medeiros.

Disclosure Statement

This article is an expanded version of remarks given in the Conference on Reproducibility and Replicability in Economics and the Social Sciences (CRRESS) webinar series, which is funded by National Science Foundation Grant #2217493.

References

AEA Data Editor. (2023, May11). *The AEA Data Editor defines and monitors the AEA journals approach to data and reproducibility.* https://aeadataeditor.github.io/

Bjarkefur, K., Cardoso de Andrade, L., Daniels, B., & Jones, M. R. (2021). *Development research in practice: The DIME analytics data handbook*. World Bank. http://hdl.handle.net/10986/35594

National Academies of Sciences, Engineering, and Medicine. (2019). *Reproducibility and replicability in science*. National Academies Press. https://doi.org/10.17226/25303

Project TIER. (2023a, May 11). *TIER Protocol 4.0*. https://www.projecttier.org/tier-protocol/protocol-4-0/

Project TIER. (2023b, May 11). *Soup-to-nuts exercises*. https://www.projecttier.org/tier-classroom/soup-nuts-exercises/

Project TIER. (2023c, May 11). *Demo project*. https://www.projecttier.org/tier-protocol/demo-project/

Ruggles, S., Flood, S., Sobek, M., Brockman, D., Cooper, G., Richards, & Schouweiler, M. (2023). *IPUMS USA: Version 13.0* [data set]. IPUMS. https://doi.org/10.18128/D010.V13.0

U.S. Census Bureau. (2019). *American Community Survey (ACS): Public Use Microdata Sample (PUMS), 2018* [data set]. U.S. Department of Commerce.

Footnotes

1. Throughout this article, the term ‘reproducibility’ refers to the concept of computational reproducibility, as elaborated in the National Academies (2019) Consensus Study Report *Reproducibility and Replicability in Science*. ↩

2. But the modifier ‘scriptable’ is essential: the methods presented here cannot be applied to a workflow that relies on point-and-click manipulation of data in a spreadsheet. ↩

3. Another method that is common in some research communities is to manage the working directory by defining a macro for the absolute path to the working directory that users can edit to match the directory structure on their computer. ↩

4. Unless you create an editable macro for the absolute path to the working directory. ↩