Notes on Reproducible Workflows

Mental Health and the Justice System in Durham County

Maria Tackett

6/11/24

Table of contents

Reproducibility **1**

Data **2**

Code **4**
- R scripts . . . 4
- Quarto documents . . . 4
- Version control . . . 6

Organization **6**

Environment **7**
- References . . . 7

> **Note**
>
> The source code for this document is available at github.com/matackett/mhjs-workflow. Please open an issue or submit a pull request if you have any edits or additions.

Reproducibility

In *Telling Stories with Data* (Alexander 2023), Rohan Alexander as this about reproducibility (emphasis mine):

> "Alexander (2019) defines reproducible research as that which can be exactly redone, given all the materials used. This underscores the importance of providing

the **code, data, and environment**. The minimum expectation is that another person is independently able to use your code, data, and environment to get your results, including figures and tables."

Using this definition as a guide, we can think about how to make our research reproducible by establishing good practices in each of the following areas:

1. Data
2. Code (and analysis)
3. Organization
4. Environment

We'll address each of these areas separately to help organize our thinking, but in reality they all intertwine, so we'll often need to discuss one while discussing the other. For example, we can't establish best practices for coding without thinking about the data.

- Why is it important to do the work on this project in a reproducible way?
- What are practical considerations to keep in mind as we develop reproducible workflows for this project?

Data

In the 2021 paper Datasheets for Datasets Gebru et al. (2021) name two stakeholders for a data set: *data creators* and *data consumers*. For many data sets the creator and consumer are one in the same. Whether they are the same or different parties, clear and detailed data documentation is critical for both parties to understand what's in a data set, how the data can be analyzed, and the context and limitations of any results gleaned from the data.

At a minimum, every data set should have a *codebook* (aka data dictionary) that documents what each row represents, and the following information for every column:

- Column name
- Data type
- Precise definition
- Possible values / categories
- Units (if applicable)

Here are some examples from our data sets. Some of this information was pulled from the data dictionary provided by the ACE team who created the deidentified data set.

- **Column name**: `RACE`
- **Data type**: Categorical

- **Precise definition**: Race record from DCDF
- **Possible values / categories**:
  - A
  - AMERICAN INDIAN OR ALASKAN NATIVE
  - ASIAN OR PACIFIC ISLANDER
  - B
  - BLACK
  - H
  - I
  - U
  - UKNOWN
  - W
  - WHITE
  - `NA`
- **Units (if applicable)**: N/A

> **!** Let's write the codebook entry for a variable in the encounters data set and for a variable in the cost data set.

Another important piece of documentation is the datasheet that provides more information regarding the motivation for the data set, data collection, contents of the data, and ethical considerations. Gebru et al. (2021) gives a list of questions we can use to create "a datasheet that documents its motivation, composition, collection process, recommended uses, and so on." Datasheets serve a different purposes for data creators and data consumers.

- **Data creators:** "to encourage careful reflection on the process of creating, distributing, and maintaining a dataset, including any underlying assumptions, potential risks or harms, and implications of use"
- **Data consumers:** "to ensure they have the information they need to make informed decisions about using a dataset"

> **!**
> - Why might datasheets be useful and important for our project?
> - Use the questions from Gebru et al. (2021) to build a template datasheet that could be used as documentation for data sets in our project. The datasheet doesn't need to include all the questions from the paper, but be sure it includes at least one question from each section: Motivation, Composition, Collection Process, Preprocessing/cleaning/labeling, Uses, Distribution, Maintenance.

Code

The R code and analyses will be written in one of two file types: R scripts (.R) and Quarto documents (.qmd). In general, use a script for processes that produce output but not analyses (e.g., creating data sets, writing functions, and building Shiny apps). In contrast, use Quarto documents for analysis (e.g., exploratory data analysis, model building, etc.) that require narrative and interpretation in addition to the code and output.

R scripts

Making the code in scripts reproducible not only means ensuring the same output is produced every time the code is run but also that the code is readable to future you and others. One way to make code more readable is to use comments periodically throughout the script to briefly describe what the code is doing. Additionally, use comments to explain why you're doing something and what you're trying to achieve, especially for things in the code that are unusual or not immediately obvious (Alexander 2023).

Use `#` to start each line of a comment. Comments can be written on new lines or at the end of a line of code.

In addition to writing comments, you can make code more readable by writing code in sections with a header to mark each section and writing a preamble at the top of the script that provides some metadata about the script. See Section 3.5.4 Code comments and style in *Telling Stories with Data for* more information and examples.

> **!** Suppose we want to create a data set to build a longitudinal model to understand factors associated with rebookings over time. To get started, a team member copies the *bookings* data set into the Summer 2023 folder. They glance at the data in Excel and notice some of the charges are misspelled. They use `CTRL+F` to find and correct all the instances of the misspelling. Then they load the corrected data set into R and use it to make the desired model building data set.
>
> Is this process reproducible? Why or why not? If not, what are potential issues from handling the data this way? How can the team member make the process more reproducible?

Quarto documents

Write all analyses using a Quarto document (similar to R Markdown but more powerful!).

> Quarto provides a unified authoring framework for data science, combining your code, its results, and your prose. Quarto documents are fully reproducible and support dozens of output formats, like PDFs, Word files, presentations, and more. (Hadley Wickham 2023)

You can read Chapter 29 Quarto in *R for Data Science (2e)*to learn more about using Quarto and the visual editor in RStudio (Section 29.3).

Similar to the comments in an R script, write narrative throughout the Quarto document to describe your analysis process, motivation for taking particular steps (e.g., why did you choose a specific type of model), and the interpretation from the code output. You can think of this like a research notebook; your narrative does not need to be polished at first (bullet points are fine), but it should be detailed enough that you and others understand the process and interpretations. As you continue working on the analysis, you can render the Quarto document to see the updated results.

One of the best features of Quarto documents is that they can easily be rendered in different formats. Therefore the same document used to analyze the results can be used to generate reports and presentations. As you get closer to sharing the results, you can polish the narrative and use code chunk options, such as `echo: false` and `eval: false` to hide code chunks and skip chunks of code that may no longer be relevant.

Quarto.org is an excellent resource with up-to-date code and tips for producing documents in Quarto.

Tip

Tips for writing reproducible R scripts and Quarto documents:

- Load all packages at the beginning of the script or Quarto document. This is especially important for Quarto, as the code renders in order from beginning to end.
- Use short and meaningful variable names. See Section 5.1 Names in *R for Data Science (2e)*)
- Write code according to the Tidyverse style guide to make it more readable. See Chapter 5 Workflow: code style in *R for Data Science (2e)* for an abbreviated guide)
- Use the visual editor to write Quarto documents. The visual editor makes writing in RStudio similar to writing in Google dos or MS Word (but with code!). See Section 29.3 Visual editor in *R for Data Science (2e)* to learn more.

Version control

Version control is the process of systematically tracking changes to a file. Instead of saving new versions of a file each time we update it (i.e., `my-analysis.qmd`, `my-analysis-v2.qmd`, `my-analysis-final.qmd`, `my-analysis-final-updated.qmd`), we can practice version control by documenting the changes made to a single file. Practicing version control makes it easier for future you and others to see the history of changes made to a file. It is also useful for debugging code, as version control allows you go more easily go back to a previous version of a file before issues occurred.

Git is a software used for version control. You can install git on your computer (or PACE machine) and use basic git functions in the Git pane in RStudio. As you work on an analysis, periodically write a short and informative message describing the changes you've made (e.g., `Add logistic regression model with interaction terms`) and then *commit* those changes. The *History* button in the Git pane displays a history of all the commits made to the current R project.

We often think of GitHub and GitLab when talking about version control. These are both platforms made to store git-based projects. You can think of them as ways to backup your git-based work. Given the security restrictions in PACE, we are unable to store work on these platforms; however, we are still able to implement version control using git within PACE.

See Section 3.4 Version control in *Telling Stories with Data* for more information about version control and using git in RStudio.

Organization

Projects are a nice way to organize your work. They are "widely used in software development and exist to keep all the files (data, analysis, report, etc) associated with a particular project together and related to each other." (Alexander 2023). Projects also allow you to reference files in "self contained" using relative paths (e.g., `data/mydata.csv` instead of `User/mt324/my-project/data/my-data.csv`). Relative paths are important for reproducibility, as they still work when the R project is opened on different computers.

A project is essentially a folder on your computer, so you can organize the files in a way that will be easy for you and others to navigate. Section 3.3 R projects and file structure in *Telling Stories with Data* gives an example organization scheme for an R project. Section 7.2.3 RStudio Projects in *R for Data Science (2e)* gives more detail about how to create and use projects.

Within the project, use file names that "machine readable", "human readable", and are in a default order. Section 6.1.3 Saving and naming in *R for Data Science (2e)* provides the the guidelines below for good file name practices:

1. File names should be **machine** readable: avoid spaces, symbols, and special characters. Don't rely on case sensitivity to distinguish files.
2. File names should be **human** readable: use file names to describe what's in the file.
3. File names should play well with default ordering: start file names with numbers so that alphabetical sorting puts them in the order they get used.

> **Tip**
>
> See Section 6.1.3 in *R for Data Science (2ed)* for examples of file names that are machine readable and human readable.

> **!** How might we organize the work on this team into projects, e.g., a project for each analysis? each team member? etc.?

Environment

In the context of reproducibility in R and RStudio, the *environment* essentially refers to the version of all R packages used in a project. Because R packages are regularly updated, being able to reproduce the project environment is important for long-term reproducibility of projects. As time goes on, function names, arguments, or other features of a function may change. This could potentially cause errors if the code is run in the future with the updated packages. By saving the environment, you are saving the state of the R packages at the time in which the code was written, thus more fully reproducing the code and coding environment.

A rigorous approach is using the renv package (Ushey 2023) which stores the R packages in the project directory (Hadley Wickham 2023). Another option is to run the function `sessionInfo()` and save the results in the project directory. This function will print give information about the R version, details about the computer, packages, and their version.

References

Alexander, Rohan. 2023. *Telling Stories with Data*. Chapman; Hall/CRC. https://tellingstorieswithdata.com.

Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III au2, and Kate Crawford. 2021. "Datasheets for Datasets." https://arxiv.org/abs/1803.09010.

Hadley Wickham, Garrett Grolemund, Mine Çetinkaya-Rundel. 2023. *R for Data Science (2e)*. O'Reilly. https://r4ds.hadley.nz/.

Ushey, Kevin. 2023. "Renv: Project Environments." https://CRAN.R-project.org/package=renv.