Writing Reproducible Code

Literate, linear programming

Daniela Palleschi

2024-05-14

Table of contents

Learning objectives

- learn what literate programming is
- create and render a dynamic report with Quarto
- load data
- include a table and figure

Reproducible code

- how you write your code is the first step in making it reproducible
- the first principle is that your code must be *linear*

  - this means code must be written in a linear fashion
  - this is because we typically run a script from top-to-bottom

```
read_csv(here("data", "my_data.csv"))

library(readr)
library(here)
```

Writing linear code

- you need to load a package *before* you call a function from it
  - if we're just working in an R session, *before* means temporally prior
  - with linear code, *before* means higher up in the script
- such pre-requisite code must
  a. be present in the script
  b. appear above the first line of code that uses a function from this package
- missing pre-requisite code might not throw an error message
  - but might produce output we aren't expecting
  - e.g., forgetting to filter out certain observations
  - or forgetting *that* some observations have been filtered out

Literate programming

- introduced in 1992 by Donald Knuth (Knuth, 1984)
- refers to writing and documenting our code so that humans can understand it
  - important for us: we are (generally) not professional programmers, nor are our peers
- we need to not only know what our code is doing when we look back at it in the future/share it
- the easiest way: informative comments
  - the length and frequency of these comments is your choice

Example R script

```
# Analysis script for phoneme paper
# author: Joe DiMaggio
# date: Feb. 29, 2024
# purpose: analyse cleaned dataset

# Set-up ###

# load required packages
library(dplyr)
library(readr)
library(ggplot2)
library(lme4)
library(broom.mixed) # tidy model summaries
library(ggeffects) # model predictions
library(here) # project-relative file path

# load-in data
df_phon <- read_csv(here("data", "phoneme_tidy_data.csv"))

# Explore data ###
```

- begins with some meta-information about the document, including its purpose
  - aids in knowing which scripts to run in which sequence
- there are three hashtags after some headings (`###`)
  - this is helpful because it structures the outline of the document in RStudio
- the purpose of chunks of code are written above
  - description of specific lines of code are also given

Dynamic reports

- R scripts are useful, but don't show the code output
  - and commenting can get clunky
- dynamic reports combine prose, code, and code output
  - R markdown (`.Rmd` file extension) and Quarto (`.qmd`) are extensions of markdown

    - can embed R code 'chunks' in a script, thus producing 'dynamic' reports
  - produce a variety of output files which contain text, R code chunks, and the code chunk outputs all in one

Structure your reports

- describe the function/purpose at the beginning
- document your train of thought and findings throughout the script
  - e.g., why are you producing this plot, what does it tell you?
- give an overview of the findings/end result at the end
- it's wise to avoid very long, multi-purpose scripts
  - rule of thumb: one script per product or purpose
  - e.g., data cleaning, exploration, analysis, publiation figures, etc.

Session Information

- R and R package versions are both open source, and are frequently updated
  - you might've run your code using `dplyr` version `1.1.0` or later, which introduced the `.by` per-operation grouping argument
  - what happens when somebody who has an older version of `dplyr` tries to run your code?
    - They won't be able to!
  - the reverse of this situation is more common:
    - a newer version of a package no longer supports a deprecated function or argument

Printing session info

- so, print your session info at the end of every script!

```
sessionInfo()
```

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRlapack.dylib;  L

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Berlin
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

loaded via a namespace (and not attached):
 [1] compiler_4.4.0    fastmap_1.2.0     cli_3.6.2         htmltools_0.5.8.1
 [5] tools_4.4.0       rstudioapi_0.16.0 yaml_2.3.8        rmarkdown_2.27
 [9] knitr_1.47        jsonlite_1.8.8    xfun_0.44         digest_0.6.35
[13] rlang_1.1.4       renv_1.0.7        evaluate_0.23
```

Knuth, D. (1984). Literate programming. *The Computer Journal*, *27*(2), 97–111.