Literate Programming, Quarto, and Workflows

HES 505 Fall 2023: Session 6

Matt Williamson

For today

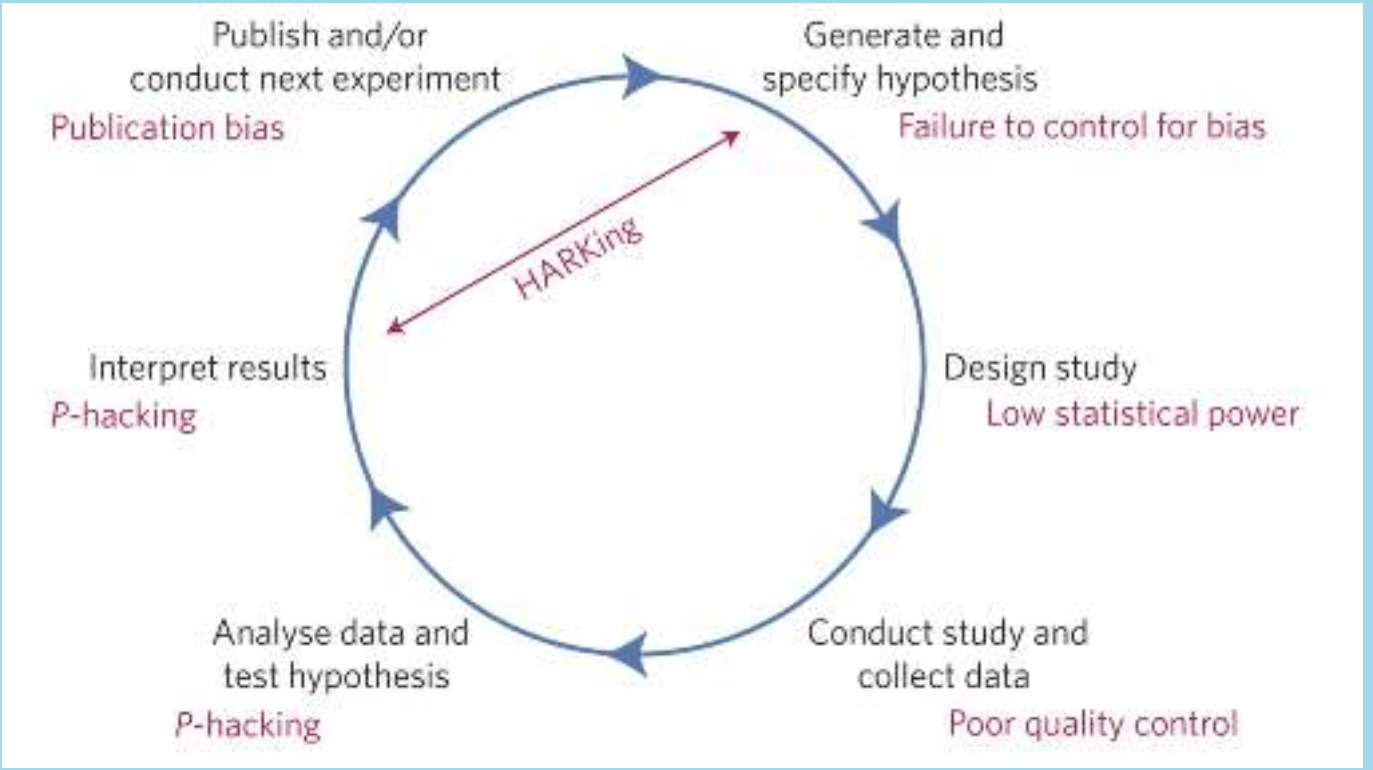
1. Introduce literate programming
2. Describe pseudocode and its utility for designing an analysis
3. Introduce `Quarto` as a means of documenting your work
4. Practice workflow

Reproducibility

Science is a social process!!

Why Do We Need Reproducibility?

- Noise!!
- Confirmation bias
- Hindsight bias



Munafo et al. 2017. Nat Hum Beh.

Reproducibility and your code

- Scripts: may make your code reproducible (but not your analysis)
- Commenting and formatting can help!

```
```{r}
#| eval: false
## load the packages necessary
library(tidyverse)
## read in the data
landmarks.csv <- read_csv("/Users/mattwilliamson/Google Drive/My Drive/TEAC

## How many in each feature class
table(landmarks.csv$MTFCC)
```
```

Reproducible scripts

- Comments explain what the code is doing
- Operations are ordered logically
- Only relevant commands are presented
- Useful object and function names
- Script runs without errors (on your machine and someone else's)

Literate Programming

Toward Efficient Reproducible Analyses

- Scripts can document what you did, but not why you did it!
- Scripts separate your analysis products from your report/manuscript

What is literate programming?

> Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.
> — Donald Knuth, CSLI, 1984

What is literate programming?

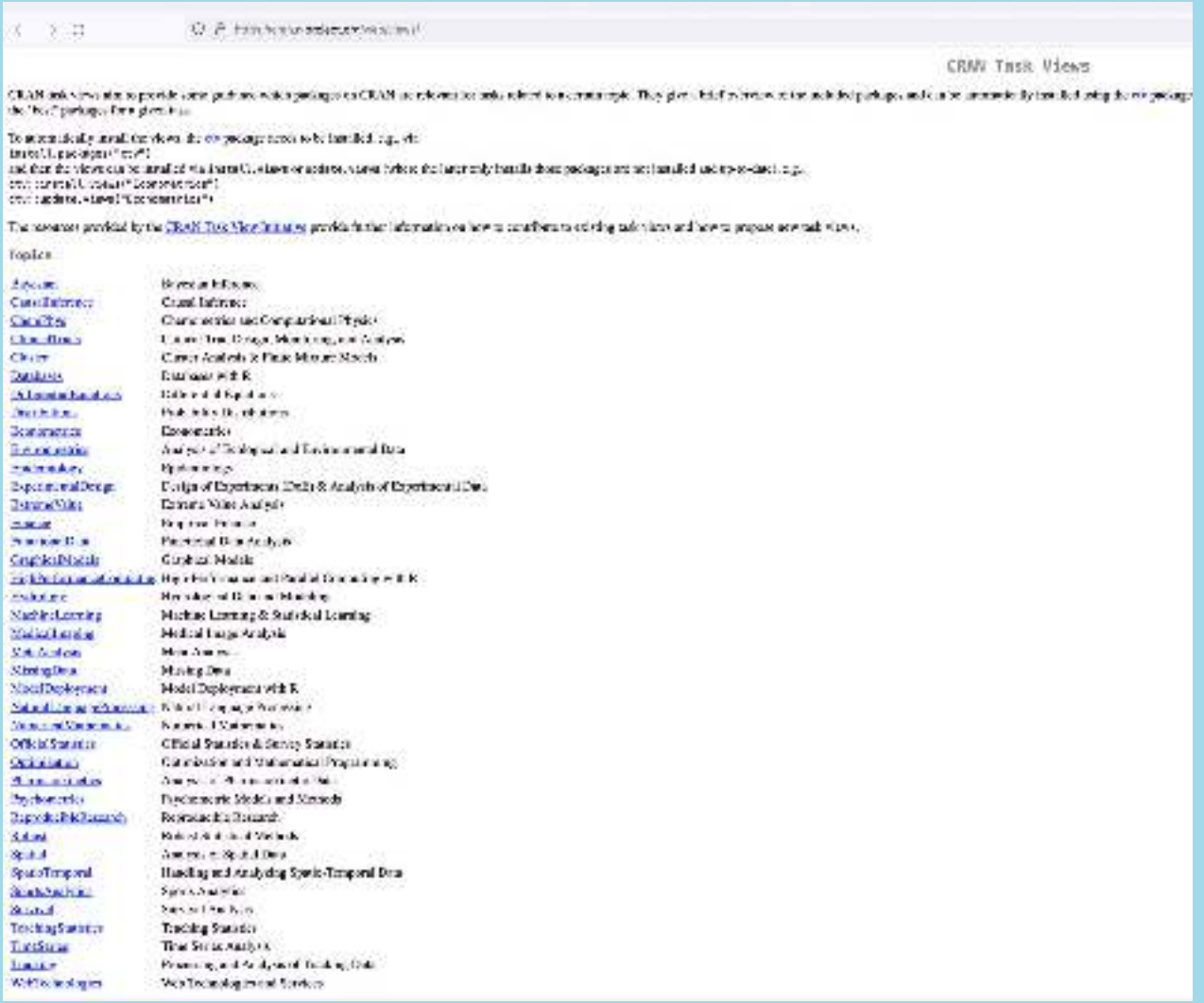
- Documentation containing code (not vice versa!)
- Direct connection between code and explanation
- Convey meaning to humans rather than telling computer what to do!
- Multiple “scales” possible

Why literate programming?

- Your analysis scripts **are** computer software
- Integrate math, figures, code, and narrative in one place
- Explaining something helps you learn it

Planning an analysis

- Outline your project
- Write pseudocode
- Identify potential packages
- Borrow (and attribute) code from others (including yourself!)



Pseudocode

Pseudocode and literate programming

- An informal way of writing the ‘logic’ of your program
- Balance between readability and precision
- Avoid *syntactic drift*

Writing pseudocode

- Focus on statements
- Mathematical operations
- Conditionals
- Iteration
- Exceptions

START: This is the start of your pseudocode.

INPUT: This is data retrieved from the user through typing or through an input device.

READ / GET: This is input used when reading data from a data file.

PRINT, DISPLAY, SHOW: This will show your output to a screen or the relevant output device.

COMPUTE, CALCULATE, DETERMINE: This is used to calculate the result of an expression.

SET, INIT: To initialize values

INCREMENT, BUMP: To increase the value of a variable

DECREMENT: To reduce the value of a variable

Pseudocode

```
1 Start function
2 Input information
3 Logical test: if TRUE
4   (what to do if TRUE)
5 else
6   (what to do if FALSE)
7 End function
```

Introducing Quarto

What is Quarto?

- A multi-language platform for developing reproducible documents
- A ‘lab notebook’ for your analyses
- Allows transparent, reproducible scientific reports and presentations

Key components

1. Metadata and global options: YAML
2. Text, figures, and tables: Markdown and LaTeX
3. Code: `knitr` (or `jupyter` if you're into that sort of thing)

YAML - Yet Another Markup Language

1. Allows you to set (or change) output format
2. Provide options that apply to the entire document
3. Spacing matters!

```yaml
---
title: "Housing Prices"
author: "Mine Çetinkaya-Rundel"
format:
  pdf:
    code-line-numbers: true
---
```

Formatting Text

- Basic formatting via Markdown
- Fancier options using Divs and spans via Pandoc
- Fenced Divs start and end with `:::` (can be any number `>3` but must match)

Adding Code Chunks

- Use 3x ``` on each end
- Include the engine `{r}` (or python or Julia)
- Include options beneath the "fence" using a hashpipe (`#|`)

````
```{r}
#| label: load-packages
#| include: false

library(tidyverse)
library(palmerpenguins)
```
````

Let's Try It!!

Additional considerations

- File locations and Quarto
- Caching for slow operations
- Modularizing code and functional programming