

Tools of the Trade

HES 505 Fall 2025: Session 2

Matt Williamson

Checking in

1. What can I clarify about the course?
2. Are there any challenges you can already see?

Today's Plan

- Open Science, reproducibility, and R
- What is a (spatial) data workflow?
- Version control for fun and profit

A More Democratic Science?

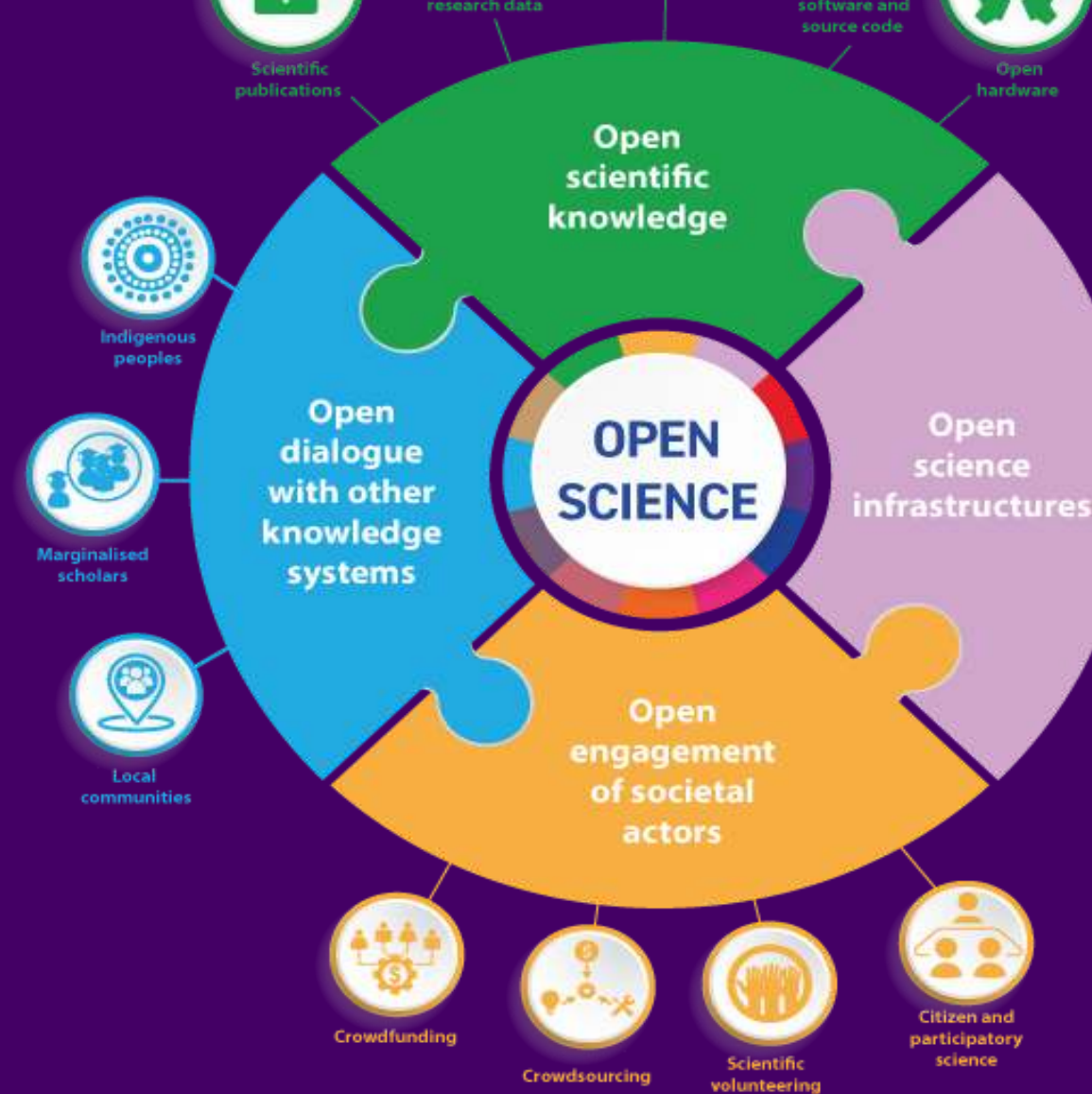
What is open science?



scientific research
and its outcomes
freely accessible to
all

accelerate research
AND improve
public trust

our focus: Open
source software and
code



UNESCO.org, CC BY-SA 4.0, via Wikimedia Commons

Why open source software and code?

- Future-proof: OSS development is fast and ongoing
- Interoperability: Work across hardware types, integrate new software easily
- Free!! (To use and maintain)
- Sharing code and data enables innovation and *reproducibility*

Why (not) R?



Open Source

Large user community

Integrated analysis pipelines

Reproducible workflows



- Coding can be hard...
- Memory challenges
- Speed
- Decision fatigue

Anatomy of an R session

Moving beyond Read-
eval-Print Loops

scripts: contain a
record of the code in
your analysis and the
objects you created

functions: perform
operations on objects

packages: collections of
related functions

Code

Plot

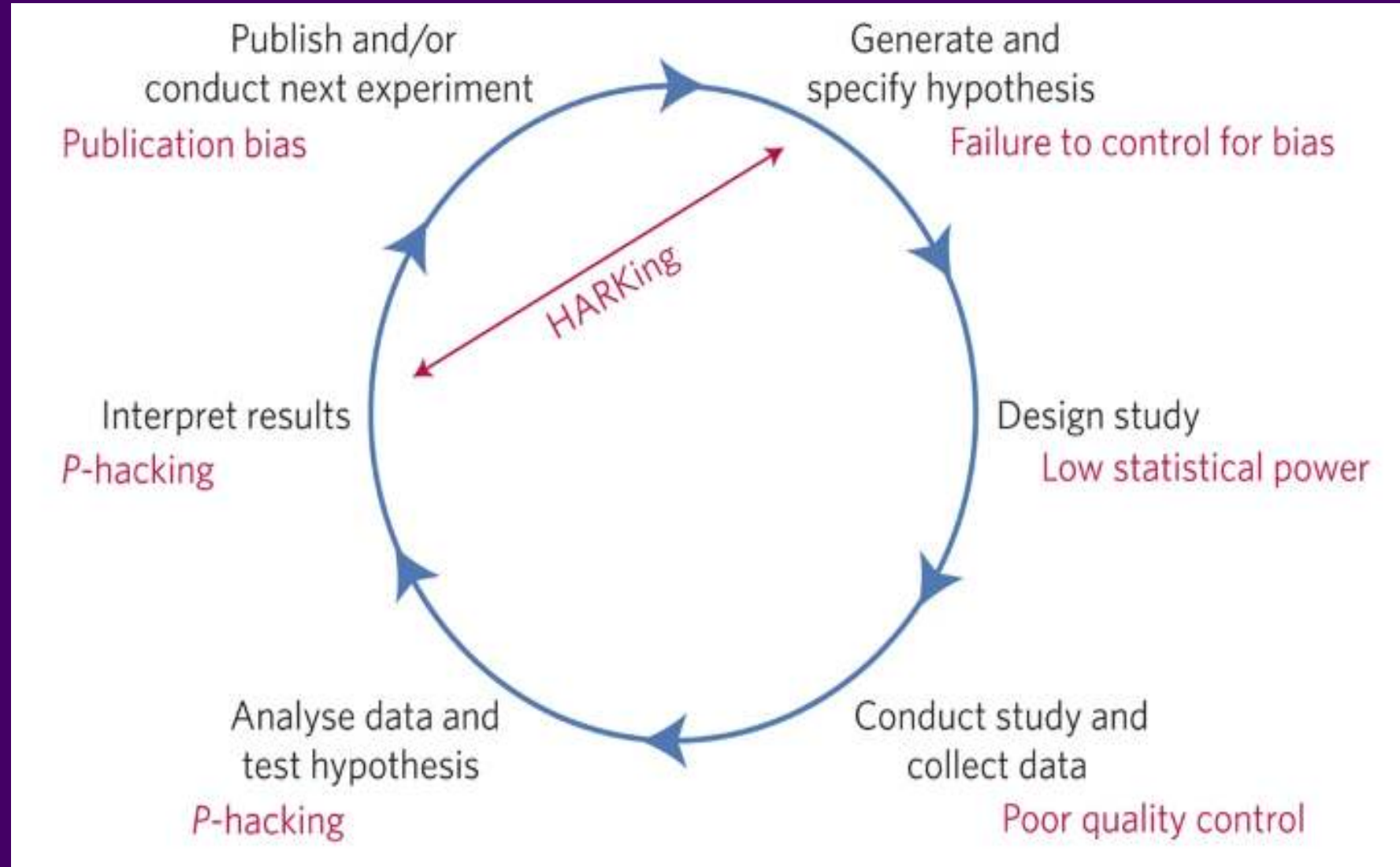
```
1 library
2 library
3 library
4         <- c "#2E74C0" "#CB454A"
5         <- map_data "state"
6         $         <- tolower         $
7         <- left_join
8 <- ggplot data =
9         mapping = aes x =         y =
10                    group =
11                    fill =
12 <-         + geom_polygon color = "gray90"
13                    size = 0.1 +
14         coord_map projection = "albers"
15                    lat0 = 39 lat1 = 45
16 <-         + scale_fill_manual values =         +
17         labs title = "Election Results 2016"
18         fill = NULL
```

Reproducible workflows

Science is a social process!!

Why Do We Need Reproducibility?

noise!!
confirmation
bias
insight bias



What do we mean by reproducible “workflow”?

Reproducibility and your code

- Scripts: may make your code reproducible (but not your analysis)
- Commenting and **formatting** can help!
- Think about future you...

```
1  ```{r}
2  #| eval: false
3  ## load the packages necessary
4  library
5  ## read in the data
6      <- read_csv "/Users/mattwilliamson/Google Drive/My Drive/TEAC
7
8  ## How many in each feature class
9  table          $
10  ```
```

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)

Flipping the script

Toward Efficient Reproducible Workflows

- 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?

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

Why literate programming?

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

Introducing Quarto

What is Quarto?



- End-to-End process between data and report
- Explicit linkage between each step (including iteration)
- Each step involves trials and choices

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)

For this class...

- We'll use headers to outline the analysis
- We'll use code chunks for small, self-contained operations
- We'll create our own functions for repeated operations
- We'll **knit** our documents into a standalone, readable document

Version control, reproducibility, and sanity

Version control in general

- Track changes without version explosion (via `git`)
- Create specific snapshots of a project to facilitate experimentation (via `commit` and `branches`)
- Create centralized backups and ease collaboration (via `GitHub`)

Version control and reproducibility

- Documenting changes to code, manuscripts, figures increases transparency of the scientific process
- Collaboration with other programmers is easier and less risky
- Automates the sharing of code and original data

Version control and sanity

- `commit` early, `commit` often
- use sensible messages to remind yourself where you were
- make sure you always have the most up-to-date version
- It will take some practice to `git` comfortable