

Literate Programming, Quarto, and Workflows

HES 505 Fall 2024: Session 5

Carolyn Koehn

For today

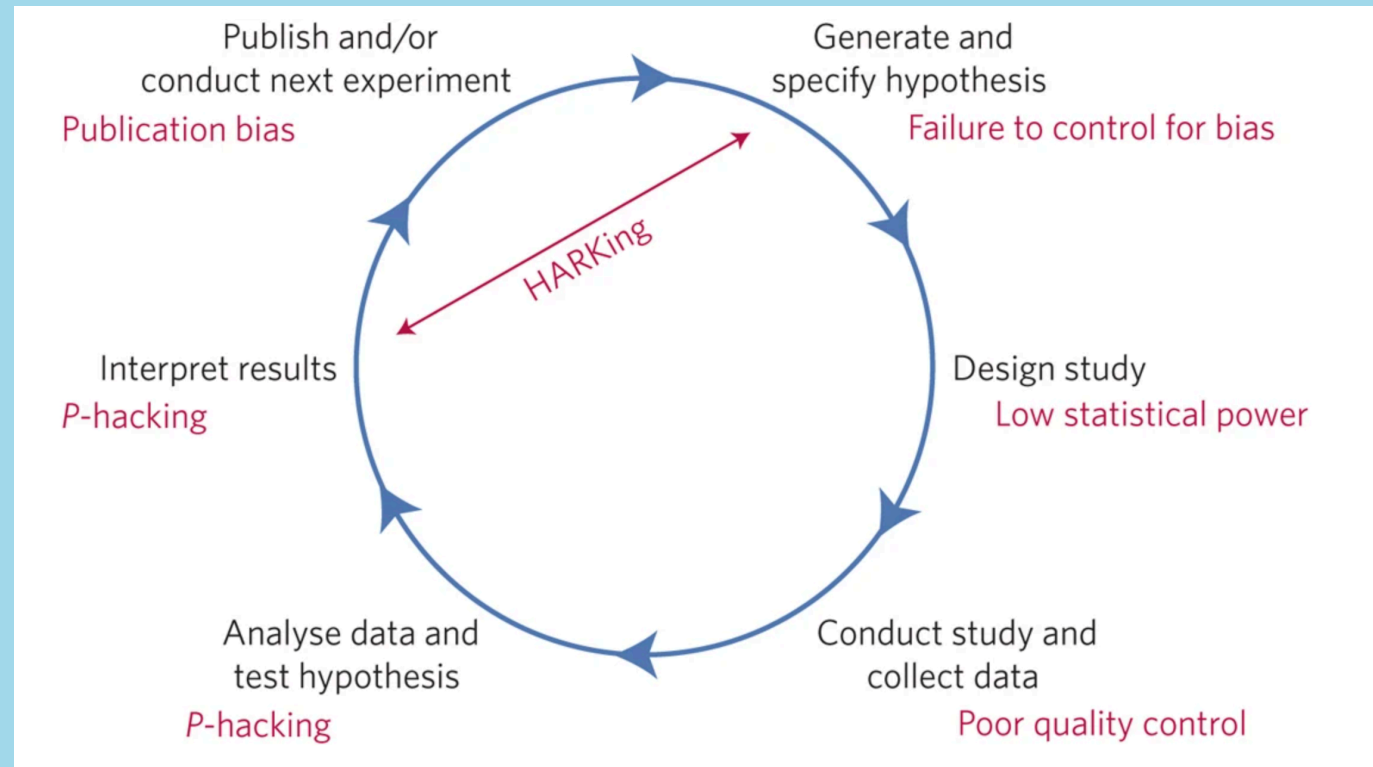
1. Introduce literate programming
2. Describe pseudocode and its utility for designing an analysis
3. Introduce **Quarto** as a means of documenting your work
4. Practice workflow

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!

```
1  ```{r}
2  #| eval: false
3  #|
4  ## load the packages necessary
5  library(tidyverse)
6  ## read in the data
7  landmarks.csv <- read_csv("/Users/mattwilliamson/Google Drive/My Dr
8
9  ## How many in each feature class
10 table(landmarks.csv$MTFCC)
11 ```
```

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?

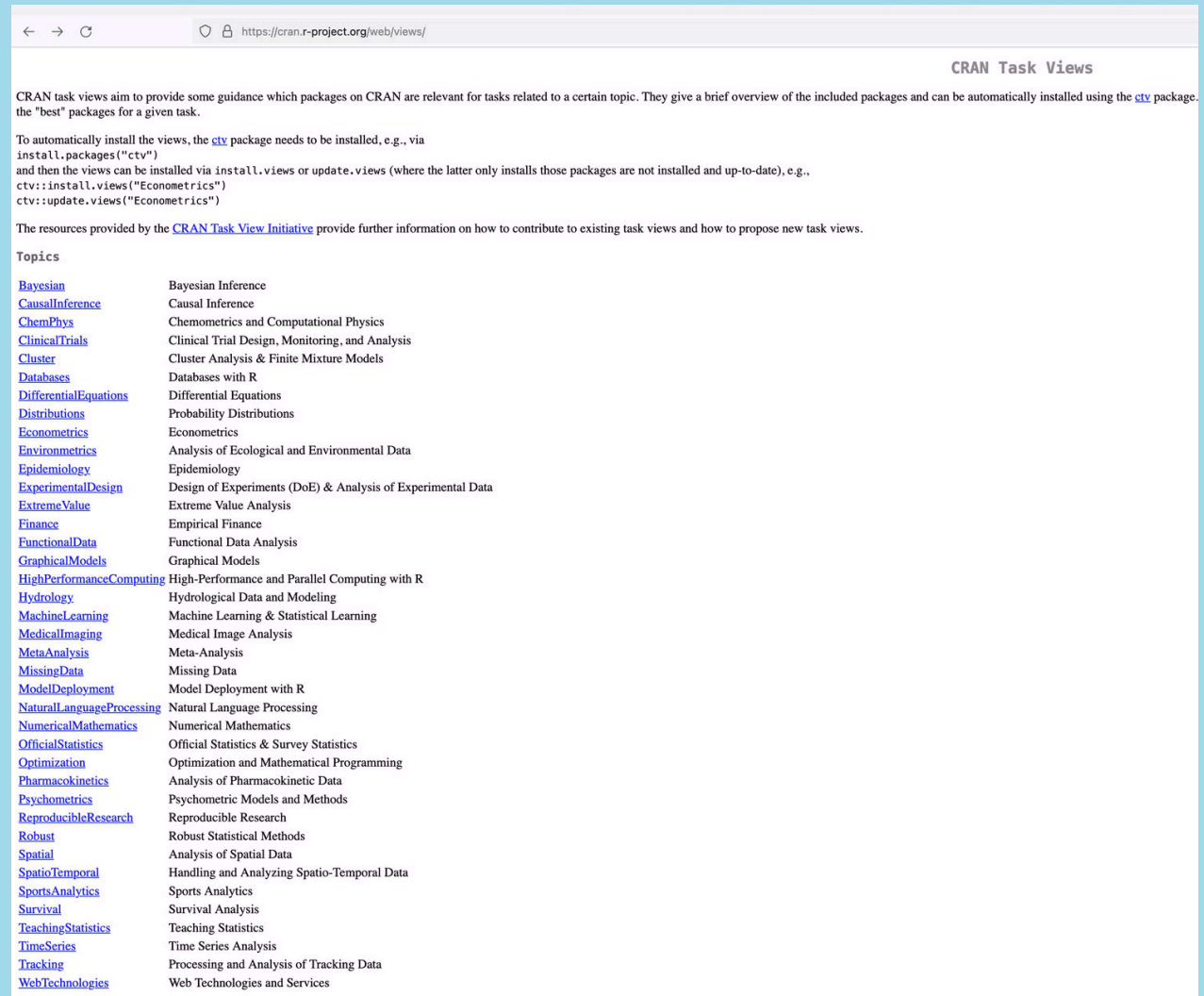
- Documentation containing code (not vice versa!)
- Direct connection between code and explanation
- Convey meaning to humans rather than telling computer what to do!
- Multiple “scales” possible

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!)



The screenshot shows the CRAN Task Views website. The browser address bar displays <https://cran.r-project.org/web/views/>. The page title is "CRAN Task Views".

CRAN task views aim to provide some guidance which packages on CRAN are relevant for tasks related to a certain topic. They give a brief overview of the included packages and can be automatically installed using the [ctv](#) package. the "best" packages for a given task.

To automatically install the views, the [ctv](#) package needs to be installed, e.g., via `install.packages("ctv")` and then the views can be installed via `install.views` or `update.views` (where the latter only installs those packages are not installed and up-to-date), e.g., `ctv::install.views("Econometrics")` `ctv::update.views("Econometrics")`

The resources provided by the [CRAN Task View Initiative](#) provide further information on how to contribute to existing task views and how to propose new task views.

Topics

Bayesian	Bayesian Inference
CausalInference	Causal Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
Databases	Databases with R
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
Epidemiology	Epidemiology
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
ExtremeValue	Extreme Value Analysis
Finance	Empirical Finance
FunctionalData	Functional Data Analysis
GraphicalModels	Graphical Models
HighPerformanceComputing	High-Performance and Parallel Computing with R
Hydrology	Hydrological Data and Modeling
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
MetaAnalysis	Meta-Analysis
MissingData	Missing Data
ModelDeployment	Model Deployment with R
NaturalLanguageProcessing	Natural Language Processing
NumericalMathematics	Numerical Mathematics
OfficialStatistics	Official Statistics & Survey Statistics
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
Spatial	Analysis of Spatial Data
SpatioTemporal	Handling and Analyzing Spatio-Temporal Data
SportsAnalytics	Sports Analytics
Survival	Survival Analysis
TeachingStatistics	Teaching Statistics
TimeSeries	Time Series Analysis
Tracking	Processing and Analysis of Tracking Data
WebTechnologies	Web Technologies and Services

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!

```
---  
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