# Quarto, literate programming, and pseudocode

HES 505 Fall 2022: Session 3

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

# Literate Programming

# What is literate progamming?

> 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. `r tufte::quote_footer('--- 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

# 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

# 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!!