Lecture 1

Brooke Anderson

Separate writing and editing

Write code fast, edit code slow.

Editing code

The aims of editing code are to make it easier to maintain and easier to understand.

Editing code

Often, this means making the code **shorter** and **clearer**.

Editing code

Code that is easier to maintain and understand will often have more white space and clearer structure.

Edit out dead ends

Writing code fast results in lots of dead ends.

Murder your darlings

Use good names

If you use good names, it makes your code self-documenting.

Edit your code to have good names for:

- objects
- columns in dataframes
- functions
- files

Good names for objects

- lowercase
- name conveys what's inside
- underscores to make easier to read
- not too short, not too long (although tab completion can help)

Metasyntactic variables

Metasyntactic variables are a fancy way to say placeholders. They're often used by coders when they're editing fast.

- foo, bar, baz (C, C#)
- spam, ham, eggs (Python)
- toto, tata, titi, tutu (French)
- pippo, pluto, paperino (Italian)

These are fine when you're writing code, but be sure to edit them out in favor of better names.

Good names for columns

- Similar rules as for naming objects
- Although it's fine to have the same column name in columns of different dataframes (only one object with a given name per R environment)
- Avoid spaces (you can force them to work, but it's a pain)

Fixing column names

[R function to clean up column names: janitor library, clean_names function]

What this does:

- Converts to lowercase
- Replaces spaces with underscores
- . . .

Breaking up monolithic code

One big aim of editing is to break up monolithic code to divide it into separate blocks and clarify what each block does.

Breaking up monolithic code

In RMarkdown files, separate code into more code chunks.

You may find it helpful to name these chunks (so you can find and fix bugs).

Use whitespace to add structure

- Blank lines are free!
- So is the Return key! (80 character limit)

Add scaffolding and structure

- Indent in appropriate places
- Add code comments (at least one per block of code)
- Use pipelines when appropriate
- Move all library calls (and function definitions) to the top of the code script