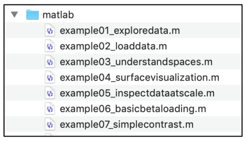
CODING ISSUES

1

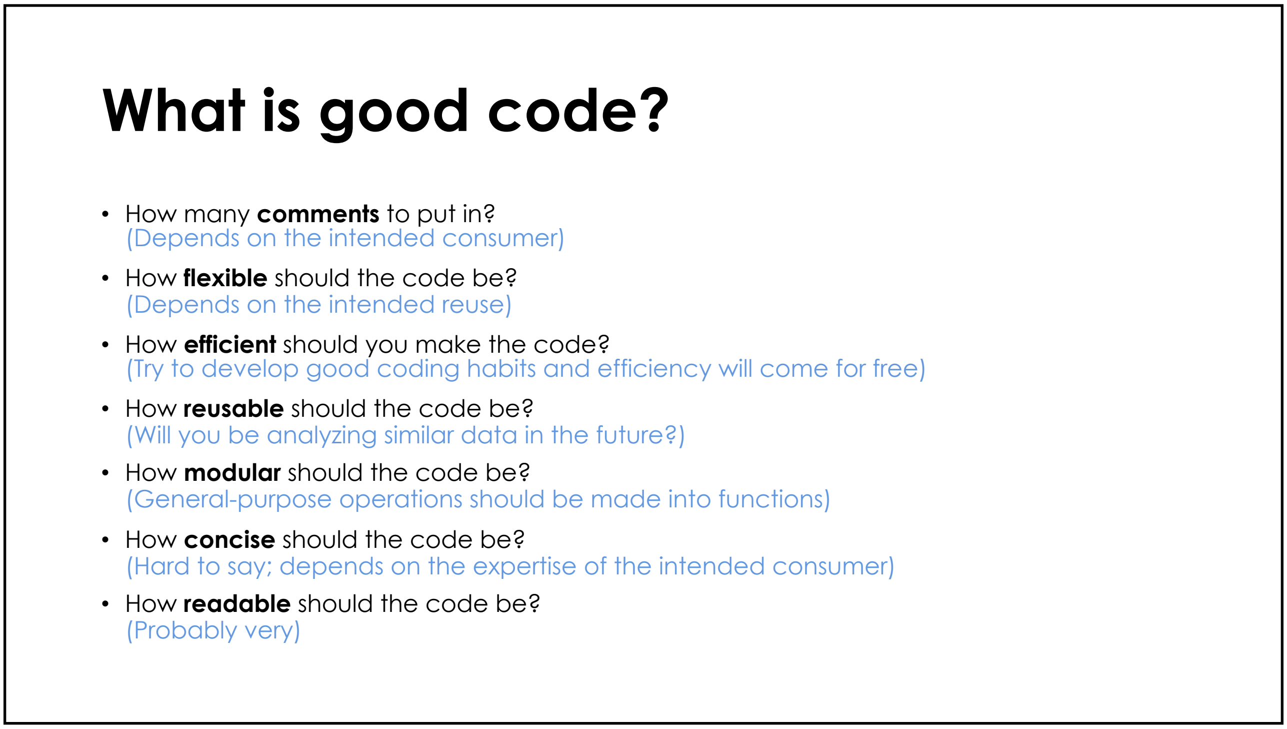
How to manage code



matlab
example01_exploredata.m
example02_loaddata.m
example03_understandspaces.m
example04_surfacevisualization.m
example05_inspectdataatscale.m
example06_basicbetaloading.m
example07_simplecontrast.m

- Important to organize and name your files carefully
- Develop a methodical workflow
  - For each figure you make, do you know the trace of all the code used to generate it?
- Consider creating a formal repository (e.g. github)
- How to organize code for different versions of an analysis?
  - One file (e.g. using an if- or switch-statement)?
    (This can save lines of code but may grow unwieldy)
  - Two parallel independent files?
    (This can streamline each file but may grow hard to manage)

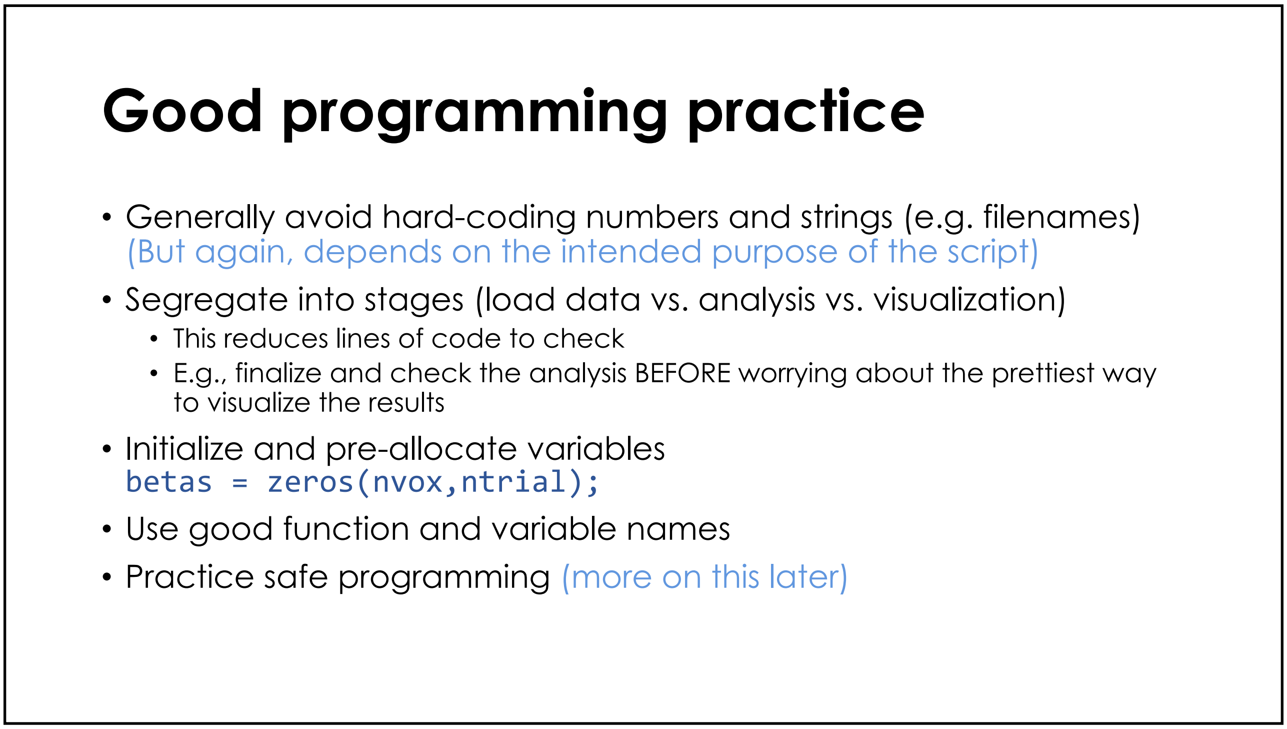
2



What is good code?

- How many **comments** to put in?
  (Depends on the intended consumer)
- How **flexible** should the code be?
  (Depends on the intended reuse)
- How **efficient** should you make the code?
  (Try to develop good coding habits and efficiency will come for free)
- How **reusable** should the code be?
  (Will you be analyzing similar data in the future?)
- How **modular** should the code be?
  (General-purpose operations should be made into functions)
- How **concise** should the code be?
  (Hard to say; depends on the expertise of the intended consumer)
- How **readable** should the code be?
  (Probably very)

3



Good programming practice

- Generally avoid hard-coding numbers and strings (e.g. filenames)
  (But again, depends on the intended purpose of the script)
- Segregate into stages (load data vs. analysis vs. visualization)
  - This reduces lines of code to check
  - E.g., finalize and check the analysis BEFORE worrying about the prettiest way to visualize the results
- Initialize and pre-allocate variables

```
betas = zeros(nvox,ntrial);
```

- Use good function and variable names
- Practice safe programming (more on this later)

4

Hard design choices

- **Choice 1: Function vs. scripts**
  - Functions implies reusability and permanence. Thus, exact correctness, very careful documentation, and (possibly) efficiency are important.
  - Scripts can range from one-offs to production code. For exploration or development, it may be okay to write uncommented messy ugly code. Ugly code can always be polished later...

The importance of functions

- A function is a **promise** to your future self.
- Good function documentation is a skill. One must determine the **proper amount of detail**.
  - No one wants to see computer code in a scientific paper.
  - On the other hand, can you clearly and concisely state exactly what you did to your data?
- **Test** your functions. Bugs are painful. 🕷

Hard design choices

- **Choice 2: Quick-and-dirty vs. production**
  - In the case of quick-and-dirty exploratory analyses (especially when typing directly into the command line), it may be useful to save figures (e.g. take a quick screenshot)

Hard design choices

- **Choice 3: Automated vs. manual**
  - Strong appeal to construct a self-contained and runnable script that does everything: load, compute, save, make figures
    - This way, you can quickly tweak the code and re-run with zero effort. 🏄
  - However, sometimes manual intervention is necessary (e.g. GUIs) or desired (e.g. checking results before proceeding)

7

Safe programming

vs

- When coding, anything and everything can go wrong.
- To help prevent errors:
  - Structure code deliberately and cleanly
  - Perform checks (asserts and queries)

8

Safe programming

vs

- **Asserts**
  - By using assert statements, after code runs successfully without crashing, you KNOW that it is correct (up to the extensiveness of your checks).
  - You can assert all sorts of things:
    - that the number of files matched is correct
    - that a variable is in fact empty
    - that a matrix has specific dimensions
    - that the values in a matrix are all finite values (i.e. not NaN nor Inf nor -Inf)
    - that a variable is of the cell format
    - that the number of elements along a certain dimension matches some specific value

```
assert(isequal(size(data),[10 200]));
assert(length(files)==10);
assert(all(isfinite(data(:))));
assert(size(data,1)==numvoxels);
```

9

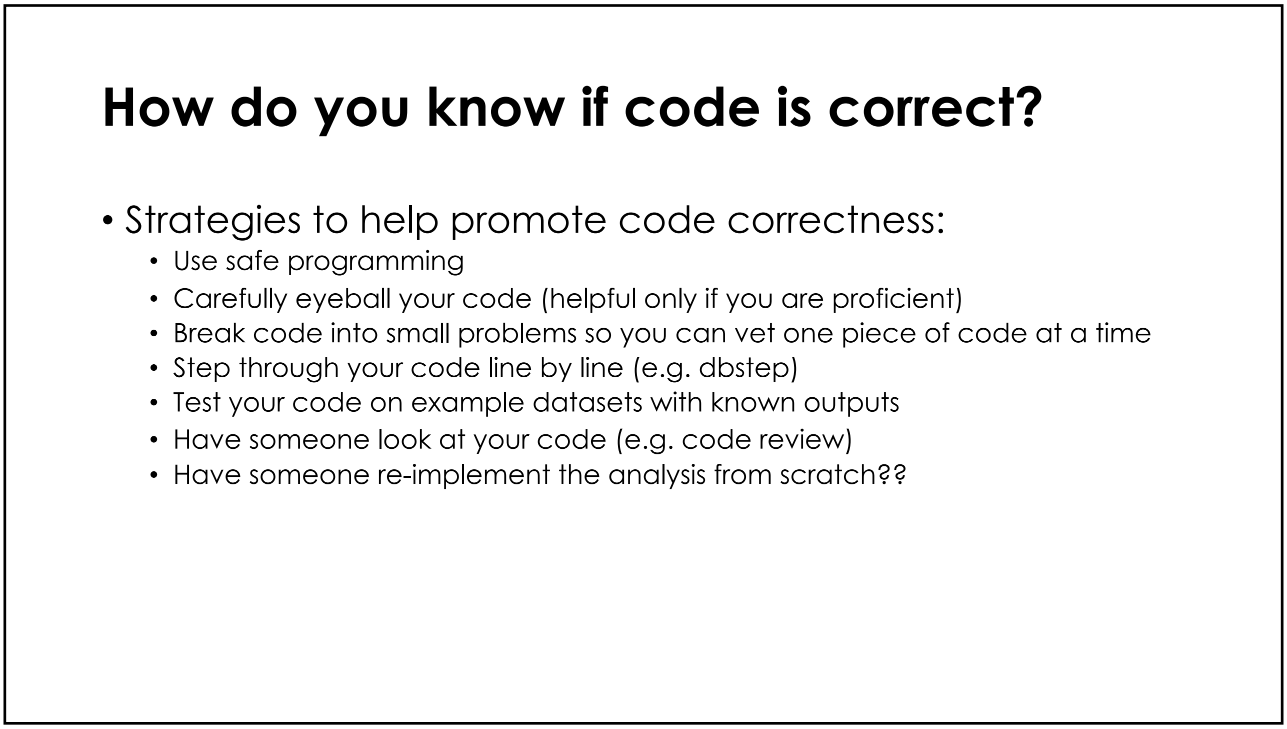
Safe programming

vs

- **Queries**
  - Inspect contents of a variable (while code runs)
  - Inspect dimensionality of a matrix (while code runs)
  - Create a figure and save to disk (e.g. inspect data from every run/subject)
  - Write text to a log file (e.g. which files were processed)

```
data(1)
size(data)
fprintf('%s',filename)
```

10



How do you know if code is correct?

- Strategies to help promote code correctness:
  - Use safe programming
  - Carefully eyeball your code (helpful only if you are proficient)
  - Break code into small problems so you can vet one piece of code at a time
  - Step through your code line by line (e.g. dbstep)
  - Test your code on example datasets with known outputs
  - Have someone look at your code (e.g. code review)
  - Have someone re-implement the analysis from scratch??

11