IA Scientific Computing

BRIEFING LECTURE
“A mathematician is a device for turning coffee into theorems” – Erdős / Rényi

requirements → code

thought process

data → insight

code
Scientific computing
* computing as a tool for doing science

Computer science
* the study of computation
Try out an idea ✫ see what happens ✫ refine your idea ✫ try something else ✫ iterate ... ✫ share what you’ve learnt

SCIENTIFIC COMPUTING
CODE AT THE SPEED OF THOUGHT

- Concise one- or two-liners for one-off tasks
- Rich, expressive libraries & glue code
Scientific computing =
Jupyter notebooks + Python + numpy + plotting

First I ran this cell up here
And now this cell is producing strange answers
Then this one, I think.
**What Not to Do**

- Your ML has doubtless been one big file where you threw together all the functions and value declarations.
- Lots of C programs look like this:
  - *We could emulate this in OOP by having one class and throwing everything into it.*
- We can do (much) better.

**OOP Concepts**

- OOP provides the programmer with a number of important concepts:
  - Modularity
  - Code Re-use
  - Encapsulation

**Modularity and Code Re-Use**

- You've long been taught to break down complex problems into more tractable sub-problems.
- Each class represents a sub-unit of code that (if written well) can be developed, tested, and updated independently from the rest of the code.
- Indeed, two classes that achieve the same thing (but perhaps do it in different ways) can be swapped in the code.
- Properly developed classes can be used in other programs without modification.
Look at each line of your code and ask yourself: ‘does this spark joy?’ If not, delete it.

Marie Kondo
while working
- imports
- experiment 1
- debug code
- tweaked experiment 1
- experiment 2
- update to experiment 1
- forgotten import

after you’ve finished
- imports
- utility functions
- run-once setup code
- functions that implement your solutions
- submit solutions to autograder
Impact of universal basic income on inequality

HYPOTHESIS AND METHODOLOGY: I investigated on a system of economic exchange of a flat-rate tax on wealth combined with a universal basic income. For each tax rate in a range of values, I simulate a population of 10,000 individuals, and measured the GINI coefficient. I ensure my simulator has reached steady state by magic.

RESULTS:

CONCLUSION: The graph shows that the larger the tax rate, the smaller the GINI coefficient. The sharpest decrease is around 15%. The limit of 100% tax of course results in a GINI coefficient of 0.
TUTORIALS

0. Programming in Python
   language quirks

1. Numerical computation
   numpy
2. Plotting data
   matplotlib

3. Working with data
   pandas
   A. Data scraping recipes

ASSESSMENT
(8% of maths paper)

No written exam
Four ticks, each marked pass/fail
Ticks 1 and 2: pass the autograder, submit notebook by 23 Jan
Ticks 3 and 4: submit pdf by 6 Feb
Some of you will have in-person ticking
MY CODE PASSED TEST 2. BUT IT WAS BUGGY CODE, AND IT TOOK ME AGES TO DEBUG AND PASS TEST 3.

YOUR GRADER SUCKS.

Scientific computing isn’t about meeting requirements, it’s about discovery.

✍ chart your own path ✍️ write your own tests ✍️ invent a few small test cases ✍️ work them out with pen and paper ✍️ make sure your code agrees
The autograder will run wherever you run Jupyter + Python3
Help and support

- Moodle help forum
- Mini-lectures and help sessions early in Lent term