Coding Best Practices

Here are some best practices to follow when writing code.

Start Early!

It's easy to not follow these best practices in the initial phases of coding, as your exploratory code gets more complex, but it's easier to follow these from the start instead of needing to refactor later.

Use Functions

Use functions instead of copying and pasting code blocks. Every time code is copied, it increases the number of lines that need to be edited if that base code is modified, which in turn increases the potential for bugs. If you find yourself repeating code, it's more effective to write a function, which can be more directly debugged and edited.

Write Good Comments

When writing comments, write about "why," not "what." What a particular line of code is doing is pretty straightforward to understand. The logic or goal of a section of code is more important to comment so you or anyone else can put those sections and lines into context

Write Unit Tests

Unit tests are a great debugging and software validation tool. The purpose of a unit test is to check that a particular function or set of functions behaves as expected. They consistent of a few steps:

1. Set the environment's state. Initialize variables to desired values for the test.
2. Execute the desired function(s).
3. Compare the output to the expected output given the specified variables.

It's useful to write a file containing the unit tests and add this to the repository; then others can verify that the code works as it should.

Positive/Negative Controls

Real data is messy, and it can be easy to rationalize and find explanations for results, even if they're erroneous. Make sure that your codes and algorithms work properly by conducting positive and negative control tests. A positive control test shows that code can recover known values, *e.g.* running your calibration code for a data generated by a statistical model with known parameter values. A negative control test shows that your code doesn't falsely detect patterns, *e.g.* running the code on white noise.

Write READMEs

Scientific code repositories aren't always useful for reproduction! This can be because of poorly commented and/or written code or because it isn't clear how to set up a system to run the code. Your `README.md` should explain the purpose of the code, what the dependencies are, including versions (or, if you've shared an `environment.yml` file or something similar, how to set up the environment), and how to reproduce the published results (or extend them if that's desirable).

Use GitHub

A useful Git commit consists of code required to add a new feature or fix a particular bug. This allows easy restoration of a certain code state with a known set of rolled back changes. For larger-scale code revisions, consider creating a new branch which can then be merged with the main branch once that code is fully verified.

Last update: June 26, 2023