Humphries Lab Policy on Code Development

V1.1 8/10/2020

Our research is based on code, lots and lots of code. It is thus in our interests to adopt good coding practices, for many reasons including:

- we can have confidence in our results and so also in our published work
- others can use our code, to check our results or extend our approach
- we can re-use our own code, and understand what it does 6 months after it was written

This policy lays out how the lab should strive to develop its code. The lab manual covers our policy on how the lab shares its code.

A key idea in laying out this policy is that research code is not equivalent to software code. Research code is often exploratory, as it is implementing ideas for analysis and models. The author(s) often do not know fully what the code will do in advance. Much research code is written once and abandoned. While there are well-developed ideas for good coding practice for software code, these ideas are not so well developed for research code. Hence this policy is less prescriptive than it would be for software code.

Style

All elements of the code policy are made easier by human-readable code. This means:

- use a consistent, explicit, readable naming convention for variables and functions
- it is common to use either underscore ("add_N_to_X", "plot_an_elephant") or camel-case ("addNtoX", "PlotAnElephant"). Pick one convention, and use it consistently.

Version control

Version control software (e.g. Git) is useful for managing complex code projects; separately, the online cloud hosting of the version control repository (e.g. on GitHub) allows for easy sharing of the code

- any multi-authored code base will use version control by default (see e.g. the "Network Noise Rejection" work), to manage the contributions successfully
- any project whose code is intended to be "product", e.g. a package to compute a metric or algorithm(s), should use version control from the outset.
- Otherwise, single-authored research code is encouraged to use version control at an appropriate point in the project.

Refactoring

Research code often takes the form of long scripts, notebooks etc because of its exploratory nature. If that code becomes central to a paper's results, refactor code into functions for testing and review. Better yet, plan to write functions as the code is developed.

Code for generating figures often contains further processing of data within the plotting code – for example, rescaling the data by subtracting a mean. Figure code should be refactored to include only visualisation code and plotting functions, working on final versions of data.

Testing

Does the code do what you think it does? Code has errors. Code from this lab has had numerous errors, of which we have caught many, but not all.

Testing research code – graphs. Our first test of most of our code is to graph the results of our analysis or models and see if they are what we expected, or if they make sense.

Testing research code – synthetic data*. If doing time-series analysis, a useful check is to test analysis code on artificially generated time-series with known properties.

Unit testing* – for functions, write code that checks the function returns the correct results; re-run unit test after each change to the function.

**note the necessity in these cases of the code to generate the tests also being correct. Testing alone is not expected to catch all errors.*

Code review

Many errors in research code are not bugs, but conceptual errors. For example, the incorrect or missing scaling of a time-series, or the wrong translation of an equation into code.

We will use code review to try to catch these errors. Code review is the process of having code reviewed by someone who is not the author. Again, most code review advice is about software code, not research code.

Our approach will take this form:

- the author will nominate a section of their code to review, typically 100-200 lines long (not including whitespace and comments); they should also point to the location of crucial code those lines call.
- The reviewed code should contain work key to a result or model, such as the processing or transformation of time-series data or other data, the implementation of an algorithm, or the implementation of equations in code. The code for review will likely be an excerpt from a longer piece of code – hence the presentation by the author is needed to give the context of the code.
- The author will send the code or links to it at least 3 working days before the review meeting (inclusive of the meeting day), for reviewers to look at
- The author should send any tutorials, demos, or tests of that code, including graphs, to help the reviewers understand it
- In the review meeting, the author will present the code to the reviewer(s), talking them through the operations of each line or section

- The presentation will ideally be of the live code; but slide-based presentation of code excerpts is allowed
- The reviewers' task is to ask constructive questions to clarify the code's operation, optionally they may make constructive comments on the code's form itself (functions, variable names, initialisation etc)
- Review meetings should strive to be ~30 minutes long

Frequency of review meetings

Code review is separate from the presentation of the scientific insights and directions of our research (data clubs) and the discussion of new science (journal clubs).

We will schedule a regular afternoon tea-break meeting (Monday at 3pm in each journal club week).

- Each tea-break has a named person:
    - if they have code to review, they can use that time to present their code review;
    - if they have no code to review, they can introduce something to the lab – a technique (an algorithm, an analysis approach, a coding tool, etc), or paper(s) of interest, or a line of research they're digging into etc etc. This is not a formal presentation!
- Lab members can ask to present a code review in these tea-breaks at any time, replacing the scheduled person
- Ad hoc: lab members can ask another member of the lab for pair review of code outside these presentation meetings

Who will review

Unless otherwise specified, it is expected that all lab members who are available for a code review meeting will attend the presentation, and are allowed to offer comments. It is recognised that not everyone will necessarily know the coding language being used.