CMEE Masters: Computing Coursework Assessment

Assignment Objectives: To work on a series of computing/programming exercises and problems in a coherent, modular, reproducible workflow under version control.

Note that:

- *All script/code files, errors and other info mentioned below are in the weekly log/feedback files.*
- *The overall assessment will typically have significantly lesser marks than a simple weighted average of each week's points because the overall assessment is based on not just the "Computing Coursework Assessment Criteria", but also the the "Marking Criteria for Exams, Essays and Coursework". Both sets of marking criteria are in the Assessment Appendix of the online TheMulQuaBio notes and git repository.*
- *In your 1:1 post-assessment feedback session, we will discuss where you gained or lost marks, and what you could have improved further. To the extent possible, please come with questions about specific scripts based upon the overall and weekly feedback you have received. This may require you to compare your code with the solution code in many cases.*

Specific feedback

The Good (what you did well!)

1. Found all the expected weekly directories in your parent directory.
2. Overall clean project organization.
3. Your Git repo size at the end of computing weeks was reasonable — suggesting you did not keep unnecessary binary files under version control, and that you did not commit excessively. It could also mean that you did not commit enough, and/or somehow along the the way lost parts of your git history — but I haven't checked these possibilities!
4. You had an .gitignore, good! You made exclusions specific to certain weeks (including/excluding subdirectories/files/patterns). You might find this useful in the future if you haven't seen it already: `https://www.gitignore.io`.
5. You had a good overall Readme which gave a sufficient overview of the Repo as a whole, and gave general details of things like languages used. You also had weekly Readme files which went into more details relevant to each week. You included versions of languages and dependencies/packages used. Also check out this resource: `https://github.com/jehna/readme-best-practices`.
6. You made your bash scripts robust to handle missing inputs.
7. Your Unixprac solutions were correct. Do compare with my solutions.
8. You did keep Weeks 4-6 under version control - I did not ask for this, so well done.
9. You made good use of functions to tidy up some (particularly R) scripts; basically making some of those scripts modular.

10. Your Autocorrelation practical was OK – the code was compact and ran quickly. The report as good in that it provided reasonable statistical and biological/ecological interpretations.

11. Your Groupwork practicals were all in order, and your group did well in collaborating on it based on the commit/merge/pull history. More feedback on this in the 1:1 sessions.

The Bad (errors, missing files, etc)

1. Some scripts threw errors (`basic_io1.py`, `basic_io2.py`, `basic_io3.py`, `DrawFW.py`, `Numpy_pra.py`).

2. `Lv3.py` missing.

The Ugly (niggling issues like commenting, cosmetics, complexity of code, etc)

1. You could have formatted the output of certain scripts to be a little neater/organised/informative – for example lc1.py is perfectly functional, but the output could have been improved (compare with my solution). I did not explicitly ask for this but was expecting students would come to this conclusion on their own as they became more experienced / skilled over the first term.

2. Although your file organisation is generally neat, some of your script outputs (e.g. pdf report / figs) were saved to your `code` subdirectory, rather than placed into the relevant results subdirectory. Not a huge issue, but keep an eye on this as such things make it hard for you and/or users of your code to find things within your project structure, particularly when it comes to more complex projects.

3. Commenting could be improved – you are currently erring on the side of overly verbose comments at times (including in your readmes), which is nonetheless better than not commenting at all, or too little! This will improve with experience, as you will begin to get a feel of what is "common-knowledge" among programmers, and what stylistic idioms are your own and require explanation. In general though, comments should be written to help explain a coding or syntactical decision to a user (or to your future self re-reading the code!) rather than to describe the meaning of a symbol, argument or function (that should be in the function docstring!).

4. In many places you could have broken the description of certain complex commands or code lines into key components using a comment. Whilst docstrings in a Python module serve some of this purpose it is good to not totally rely on those to be sufficient, particularly as you move into the 'main' module. As you write more complex code (like in week 7) more comments are needed to aid those who read your code.

5. Please do compare as many of your solutions with the ones I have given as possible. There are simpler ways to solve some of them, and in general it will be insightful to see how the same code/solution can be written/found.

Overall Assessment

Overall, a very good job. Although some of your scripts did retain some fatal errors, most of them ran without issue. Try to be a little more vigilant in chasing down errors in future.

Your commenting is very thorough, perhaps a little too much so, but this is also likely to be a tendency that fixes itself with experience. A solid job overall, well done.

Provisional Mark: 77

Signed: Samraat Pawar

March 27, 2023