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.*

Student's Name: Liying Huang

1 Specific feedback

1.1 The Good (what you did well!)

1. Found most of the expected weekly directories in your parent directory (no Weeks 4/5).
2. Reasonably clean project organization and code
3. Your Git repo size when I checked week 7 was about 1.5 MB — a good size, suggesting you did not keep unnecessary binary files under VC, 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 won't check these possibilities!
4. Good job with the coding overall. I noted rapid progress over the weeks (with some caveats - see below).
5. You did some extra credit Qs.

1.2 The Bad (errors, missing files, etc)

1. Week 3's project structure was not standard (unlike other weeks). You stored your work in a `MrRCoursework` directory for some reason.
2. I could not find the Autocorrelation practical report
3. There were some syntax and path specification errors that could have been fixed easily.
4. There were also, on occasion, some extra files.

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

1. Scripts from Weeks 4, 5 & 6 were not part of the assessment, but should have kept these organized like otehr weeks.
2. You had a .gitignore throughout, but not enough meaningful exclusions specific to certain weeks. You will likely find this useful: `https://www.gitignore.io`.
3. You had an overall readme file without anything really meaningful in it (e.g., could have included a description of what the overall project structure is).
4. The weekly Readmes could have been more informative. In these you could also have included the language and dependencies requirements. Also check out this resource: `https://github.com/jehna/readme-best-practices`. As you become a seasoned programmer, you will learn to make the readme file descriptions even more informative yet succinct.
5. In many places, especially early weeks (e.g., UNIX), you could have broken the description of certain complex commands or code lines into key components using a comment.
6. The code was inconsistently well-commented, You will learn to find the right balance of clean vs commented code that works for you.
7. Docstrings were frequently missing in Python scripts.
8. For shell scripts, in general, it is a good idea to add some input checks and return a meaningful message with error for utility scripts like these, especially in case somebody else uses it. I had not asked for these explicitly, but was hoping this would be something you would arrive at yourself after gaining some experience with coding and revisiting your code. But most students don't get to this, so don't feel too bad!
9. Also, in as many cases as possible, it is a good idea to write scripts to be self-sufficient (modularize your code). For example, align_seqs.py was nicely done, but it could have been written to be a module also take external inputs optionally (though I did not ask for it specifically). Compare with the solution.
10. Please do compare as many of your solutions with the ones I have given (e.g., Unix-Prac1.txt, using_os.py) as possible. There are simpler ways to solve some of them, especially the last one, and in general it will be insightful to see how the same code/solution can be written/found. In particular:
    (a) using_os.py: the script could have provided some more meaningful output to screen.
    (b) You did a great job with lc1.py, lc2.py, dictionary.py, and tuple.py, but if you compare with the solutions on the repo, you will notice that you could have make them produce better-formatted output.
    (c) Another example is the the temperature Autocorrelation practical report: the solution uses a different approach for the permutations.

2 Overall Assessment

You did an OK job overall.

Overall, as this is the first time you have done programming in a heady mix of UNIX, Python, & R with a sprinkling of LaTeXand git, you did very, very well!

It was a tough set of weeks, but I think your hard work in them has given you a great start towards further training, a quantitative masters dissertation, and ultimately a career in quantitative biology!

Provisional Mark: 66

Signed: Samraat Pawar

March 8, 2020