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: HongYe Wang

1 Specific feedback

1.1 The Good (what you did well!)

1. Found all the expected weekly directories in your parent directory.
2. Quite clean project organization and code.
3. Your Git repo size when I last checked was about 15 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. You had a .gitignore throughout, with meaningful exclusions specific to certain weeks. Good; glad you used `https://www.gitignore.io`. However, I suggest looking and removing useless exclusions (e.g., `develop-eggs`) and adding ones you might stil be missing.
5. You had an overall Readme file including a description of what the overall project structure is. You also had one within each week. The weekly Readmes were clear and informative, but maybe with too much info about test directories like `sandbox`.
6. Good job with the coding overall. Great progress over the weeks (with some caveats - see below)!
7. You did most extra credit Qs.
8. The temperature Autocorrelation practical report: good job. However, you did not report the observed correlation coefficient! Also, you could have plotted the observed correlation coefficient on the histogram of permuted ones, and included some more interpretation of the results. Compare with the solution code.

1.2 The Bad (errors, missing files, etc)

1. There were a few syntax and path specification errors. Most of these were easy to fix.
2. There were also, on occasion, some extra files.
3. the Food web network did not print to file (week 7)

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

1. The description of code in the weekly readmes were a bit vague. Also, in the readmes, you should 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.
2. Scripts from Weeks 4 and 5 were not part of the assessment, but you could have kept these organized as well.
3. 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.
4. The code was inconsistently well-commented (sometimes too many comments, sometimes none/too few in important portions of the code). You will learn to find the right balance of clean vs commented code that works for you.
5. Some docstrings were missing.
6. 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!
7. 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.
8. Please do compare as many of your solutions with the ones I have given (e.g., UnixPrac1.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.

2 Overall Assessment

You did a fine 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 believe 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: 72

Signed: Samraat Pawar

March 9, 2020