Organizational notes

- The task should be done as group work (2-3 people). Please specify the names of the group members during submission.
- Don't exchange code between groups. If your solution plagiarizes another solution (highly similar code aside from variable names/comments/docstrings), it will not be counted. Try and write the code yourself.
- Submission deadline is 21.01.2021, 23:59.
- Submit your project as a `.tar.gz` archive, or transfer your private GitHub repository to the user `biodatasciencetulln` (please use only private repositories).
- Please let me know if you find any bugs, or have any questions.

Project description

This mini-project is based on Exercise 12 from Ekmekci et al., 2016: "An Introduction to Programming for Bioscientists: A Python-Based Primer" (doi):

> As a final exercise, a cumulative project is presented below. This project addresses a substantive scientific question, and its successful completion requires one to apply and integrate the skills from the foregoing exercises. Note that a project such as this — and really any project involving more than a few dozen lines of code — will benefit greatly from an initial planning phase. In this initial stage of software design, one should consider the basic functions, classes, algorithms, control flow, and overall code structure.
>
> First, obtain a set of several hundred protein structures from the PDB, as plaintext pdb files (the exact number of entries is immaterial). Then, from this pool of data, determine the relative frequencies of the constituent amino acids for each protein secondary structural class; use only the three descriptors "helix", "sheet", and, for any AA not within a helix or sheet, "irregular". (Hint: In considering file parsing and potential data structures, search online for the PDB's file-format specifications.) Output your statistical data to a human-readable file format (e.g., comma-separated values, .csv) such that the results can be opened in a statistical or graphical software package for further processing and analysis. As a bonus exercise, use Python to visualize the findings of your structural bioinformatics analysis.

The project was implemented and conducted by a third party, and provided as is. Unfortunately, some of the scripts are missing or incomplete. Your task is to add or modify the incomplete scripts as required, to be able to reproduce the results.

Project design

Project goal: Study the propensity of amino acids for protein secondary structures, which can be relevant for secondary structure prediction (Wikipedia: Protein structure prediction, Costantini et al. 2006).

- Milestone 1: Download pdb data → folder with pdb files
  - Wikipedia: PDB, PDB file format
  - wwpdb: FAQ
  - wwpdb.org: pdb file format
- Milestone 2: Parse pdb files → frequency tables for amino acids/sec. structures (csv/tsv files)
  - Biopython Wiki: Struct. bioinformatics FAQ
  - Wikipedia: DSSP
- Milestone 3: Visualize frequency tables → multibar plots (pdf files)
  - Pyplot tutorial
  - Pandas charts tutorial

The user should be able to reproduce all steps and obtain all results by performing these steps:

1. `git clone https://github.com/<username>/<projectname>.git`
2. `cd <projectname>`
3. For each numbered directory, the user should:
   - Enter the directory, e.g. `cd 01_download`
   - Execute the Bash driver scripts, there should be no more than one or two. E.g. `bash ./01_run.sh`
   - After the execution of the scripts, the results should haven been generated and placed in `results/`. The input data required to generate the results should be located in `data/`
   - Leave directory, `cd ..`

Tasks

Your task is to exactly reproduce the results, i.e. the table and corresponding plot, showing the relative frequency of different amino acids in protein secondary structural classes ("helix", "sheet" and "irregular"). Unfortunately, some of the scripts that were used in the project were lost, so you will need to modify or add some scripts.

This also demonstrates the importance of reproducibility. Good project organization allows to easily see what was done and repeat it. Important aspects are a transparent and standardized project structure, reliable driver scripts and using a version control system and code repository for development and sharing with others.

The grading is based on how well the project can be reproduced in the described way. 100 points are maximally achievable. Your code is graded based on these three criteria:

- Correctness: how well the code fulfills the specifications and is free of bugs
- Design: how well the code is written/designed (clearly, efficiently, elegantly, logically)
- Style: how readable the code is (comments, variable names, etc.)

The weighting in % for the three criteria is ca. 60:30:10 (correctness:design:style).

Also, the project is missing a conda environment file `requirements.yml` (carpentries-incubator.github.io). Add an environment file that allows to exactly recreate the environment, with all packages required to reproduce the results. (+5 bonus points)

General notes

- It's recommended to use an IDE like Spyder, VS Code or PyCharm for code development, a linter like Flake8 (code.visualstudio.com) and a code formatter like Black (code.visualstudio.com). Don't forget about comments, meaningful docstrings, naming conventions, programming style, etc.
- It's recommended to first draft the general logic and the functions using pseudocode, than gradually develop the code, immediately testing it using small test cases. A sensible code structure using functions makes this easier, because functions can be tested individually.
- While developing and testing the code, it's beneficial to use a representative subsample of the data, e.g. 1-3 pdb files, rather than the complete dataset.
- Before the final submission (the project with added/repaired scripts), execute all steps in a clean environment to make sure that everything works. A user should be able to reproduce the whole project by unpacking the archive, entering the consecutively numbered directories and executing the run scripts.
- Create your own `README` file. You can use it to add any relevant information (notes, observations, general comments, etc.). You should at least add the names of the group members.

References

- Ekmekci et al., 2016. "An Introduction to Programming for Bioscientists: A Python-Based Primer." PLOS Computational Biology 12 (6): e1004867. (doi)
- Costantini et al., 2006. "Amino Acid Propensities for Secondary Structures Are Influenced by the Protein Structural Class." Biochemical and Biophysical Research Communications 342 (2): 441–51. (doi)
- Noble, 2009. "A Quick Guide to Organizing Computational Biology Projects." PLOS Computational Biology 5 (7): e1000424. (doi)