L-Università ta' Malta
Centre for Molecular Medicine & Biobanking

Prof. Jean-Paul Ebejer
jean.p.ebejer@um.edu.mt
Dr Panagiotis Alexiou
panagiotis.alexiou@um.edu.mt
Dr Joseph Bonello
joseph.bonello@um.edu.mt

MMB5009: Bioinformatics

Study-Unit Assignment Specification

Version 1.1, 22nd March 2025
(Last Updated: April 4, 2025)

Necessary Preamble: This document describes the assignment for study unit *MMB5009: Bioinformatics* (10 ECTS). Please read it thoroughly. The marks allocated for this assignment are distributed as follows; **90%** are for the implementation and documentation of the specified tasks set out in this document, and **10%** are allocated for a demonstration of your solution. You are expected to allocate approximately 120 hours to complete the assignment. The deadline for this assignment is **Monday, 23rd of June, 2025 at noon**. Late submissions will **not** be accepted. Questions regarding the assignment should **only** be posted in the Assignment VLE forum, and should **not** be emailed directly to the lecturers of this study-unit.

This is an **individual** assignment. Under **no** circumstances are you allowed to share the design and/or code of your task implementations (unless otherwise explicitly stated by the task specification). The Centre for Molecular Medicine and Biobanking takes a strict and serious view on plagiarism. For more details refer to plagiarism section of the University of Malta website[1]. The use of AI tools (e.g. ChatGPT) is governed by the policy outlined at the end of this document. You are to upload all of your developed code, data, and documentation to the study-unit area in the VLE website, **not** via email to the lecturers of this study-unit.

[1] https://www.um.edu.mt/media/um/docs/schools/doctoralschool/PlagiarismandCollusionGuidelines.pdf

The main objective of this assignment is to demonstrate you understood the concepts presented in class. Failure to provide evidence of this will have an adverse effect on your mark! You have to demonstrate an appropriate level of depth and understanding befitting of an M.Sc. level degree.

1 Deliverables

The following deliverables are expected by the specified deadline. Failure to submit any of these artefacts in the required format will result in your assignment **not** being graded. Replace `NAME` and `SURNAME` with your name and surname respectively. The **three** deliverables are:

- **`2024-2025_MMB5009_SURNAME_NAME_assignment_code.tgz`** - A `.tgz` file containing your assignment code and data. This needs to be uploaded to VLE. It is your responsibility to make sure that this archive file has uploaded to VLE correctly (by downloading and testing it). Failure to open the archive file will result in your assignment **not** being graded. **Please include a `README.md` file with additional library requirements and installation and running instructions in the top level directory of your archive file submission**. Please note that if your data (only) is too large to upload to VLE due to upload size limitation (presently 100 MB), you should commit your data (no code) online (in GitHub, Google Drive, or Dropbox only) and specify the (accessible) URL in your documentation. Make sure this URL is **not** publicly available.
- **`2024-2025_MMB5009_SURNAME_NAME_assignment_report.pdf`** - The assignment documentation in `.pdf` format. The documentation has to be uploaded to VLE, together with your code.

1.1 A Note on Presentation

There are no marks assigned to the presentation of your project's documentation. However we expect your work to be of adequate postgraduate-study level all-around (including presentation). Figures and tables should have captions and be numbered, should have axes labels (and units), and should be linked to the text and discussion. Mathematical formulas should be labelled and explained. References should be plentiful and in a correct (i.e. consistent) format. Sectioning should be logical and clearly labelled. Furthermore, we expect you to proof-read your assignment carefully. Please refer to the documentation section for further guidelines on the documentation. Consider each assignment as a training opportunity for writing your M.Sc. dissertation. For further contents suggestions refer to JP's blog post on academic writing[2].

2 Technical Specification

Any code you supply will be run on Linux (distribution Ubuntu 24.04 LTS). It is **your** responsibility to make sure your code runs on this OS platform. You are required to implement the assigned task using `Python` (version 3.12 or later). `Python` libraries that you may use will be installed on the assessment machine. Any other dependencies should be specified in a `README.md` file as specified earlier. Use of an IDE, such as `PyCharm`, is allowed but make sure that your code runs from the command line. In your documentation you should briefly describe the main functionality of each `Python` program you have implemented. Note that you should **not** have any absolute paths hard-coded in any of your programs. The use of conda environments is encouraged.

3 The Five Tasks

In this assignment you are required to apply your skills to bioinformatics problems. You will be given five research questions. You are required to investigate these thoroughly.

1. **Practical Task.** In this task, you will engage in a real-world bioinformatics challenge by working directly with a molecular biologist – Prof. Byron Baron. The goal is to understand a specific biological problem and apply appropriate bioinformatics methods and tools to help address it. This task emphasizes interdisciplinary communication, problem definition, data analysis, and solution development (implementation). Students are expected to document the process, justify their methodological choices, and reflect on the collaborative experience. The focus is on interdisciplinary communication, practical problem-solving, and demonstrating the ability to apply bioinformatics knowledge in a meaningful research context. Assessment will be based not only on the technical soundness of your approach but also on how effectively your solution meets the biologist's needs.

[2]`https://bitsilla.com/blog/2019/03/content-tips-for-your-dissertation-or-project-write-up/`

The deliverables for this task are:

- A report describing both the development process (including problem statement and requirements) and the deliverable itself. The report should also contain a brief section with personal reflections on developing bioinformatics applications.
- A github repository with your implementation.

2. **Generating a dataset of protein data.** Your task is to create a (Python) program that generates a dataset for proteins. You need to read a CSV file with protein accessions (e.g. P05067), provided by the user. You can use the one provided with this assignment as an example. It is suggested that for this question you use the UniProt REST apis[3].

   From UniProt, obtain the following information about each protein in the list:

   - Protein Name (e.g. Amyloid-beta precursor protein)
   - List of accessions, including primary and secondary accessions indicating which type they are (e.g. P05067, B2R5V1)
   - Gene (e.g. APP)
   - Status (e.g. Reviewed)
   - Organism (e.g. Homo Sapiens)
   - Variant IDs (if any)
   - Gene Ontology Annotations
   - Associated Diseases, referencing the UniProt classification of the disease (e.g. Alzheimer's Disease, DI-00085)
   - Families and Domains, clearly indicating the provenance (e.g. 1.20.120.770, Gene3D)
   - Sequence and IsoForms, clearly indicating the differences from the canonical (e.g. 290-305: VCSEQAETGPCRAMIS $\rightarrow$ KWYKEVHSGQARWLML)
   - Regions (e.g. Disordered, Position 194-284)

   In your output, clearly indicate (where available) if the annotation is predicted (and the tool if available) or reviewed and/or the publication from where the evidence was collected. You should format your output in JSON format to ensure it can be shared and manipulated. You should also structure the JSON file in an appropriate format to ensure readability and clarity of the information (i.e. each piece of information should have its own name/value pair).

   In your report, explain the structure of the code (there is no need to print the whole file/s, but you may provide *short* snippets that can help you explain) and the design decisions you took on the structure of the JSON document.

[3] `https://www.ebi.ac.uk/proteins/api/doc/`

3. **Implement the bad character and good suffix rule for exact matching algorithms.** Evaluate and compare your implementation to a naive search (without these optimizations). To your implementation add the pigeon-hole principle to allow for *n* mismatches.
4. **Implement a mini-BLAST.** You are required to implement a simplified version of the BLAST algorithm presented in class. As a starting point you can use the paper, "Having a BLAST with bioinformatics (and avoiding BLASTphemy)" by Pertsemlidis *et al.* (2001) which describes step-by-step how BLAST works. You are to describe each step of the implementation. It is fine to make some assumptions to simplify your implementation, as long as these are listed in the report. You will be given a protein query sequence (in file `query.fa`) and a list of six sequences which make up your target database (in file `target.fa`). You are to use your mini-BLAST implementation to rank these six protein sequences in order of similarity to the query sequence. **Make sure to report your ranking.** What is the function of the query protein sequence?
5. **RNA-Seq Analysis Pipeline.** You are required to run an RNA-Analysis pipeline, in brackets find links to relevant programs which may be of help.
   a) Download publicly available RNA-Seq data[4][5]
   b) Perform dataset quality control using FastQC and identify adapter sequence (this sequence will be overrepresented in the `fastq` file)
   c) Perform sequence adapter trimming (*e.g.* using Trimmomatic[6])
   d) Align reads to the human genome Chr 21 and Chr 22 (STAR Aligner[7], you can produce alignment indexes only for these chromosomes so as to keep file sizes small, you may discard unaligned reads – `samtools`, or your own code *e.g.* using regex)
   e) Download human gene annotation GRCh38 from Ensembl
   f) Estimate gene coverage for each gene on Chr 21 and Chr 22 (number of reads per gene) (*e.g.* `bedtools` coverage[8])
   g) Normalize the gene coverage using two different measures of gene expression (RPM and RPKM – write your own code, take as total reads the sum of reads on all genes you annotated). Answer the following questions:
      i. Why is normalization needed?
      ii. What is the difference between RPM and RPKM?

[4] https://www.encodeproject.org/experiments/ENCSR615EEK/

[5] https://www.encodeproject.org/files/ENCFF493KQW/@@download/ENCFF493KQW.fastq.gz

[6] https://academic.oup.com/bioinformatics/article/30/15/2114/2390096

[7] https://github.com/alexdobin/STAR

[8] https://bedtools.readthedocs.io/en/latest/content/tools/coverage.html

iii. What other normalization methods are there, and how do they compare to simple methods such as RPM and RPKM?

The aim of this assignment is that you show that you are able to use Python and bioinformatics tools to solve the specific challenges posed in the tasks described.

Some required, critical steps (where applicable) in this assignment are:

a) Describe the design approach that you have adopted for your program;

b) Describe the options that a user needs to give to your program;

c) Are there any limitations?

Note that in your documentation you should also discuss strategies you thought were interesting to try but which were not successful (if any). Keep in mind that this is an individual assignment and you should not discuss the strategy you pursue with anyone else. You can find many internet sources which implement solutions to the above questions, but you are required to show a deep understanding of your approach. Sources should be cited. Failure to do so will severely affect your final mark.

4 Documentation

The report should be in PDF format and **not** exceed 25 pages (font Lato, 11pt, 1.15 spacing, default margin sizes), including all figures, tables, equations but excluding the cover page, references, and appendices. Your documentation should **not** include any Python code. A brief general introduction and concluding remarks (stating limitations of your approach and what you have learnt in this assignment) should be included. Appropriate document sectioning which reflects this specification is required. Appendices are **not** assessable.

5 Grading Criteria

The following criteria will be taken into consideration when grading your assignment:

- **Ability to answer the specified research tasks**.
- Expanding on the idea of this assignment to show other trends or interesting findings – how does your solution compare to others in terms of speed and features.
- Thoroughness, completeness, and correctness of solution.
- Ability to critically evaluate your work (shortcomings, assumptions, *etc.*). Note these should **not** be superficial or marginal improvements.

- Readability of code and adherence to coding standards (naming conventions, comments, consistency of style *etc.*).
- Quality of documentation (presentation, proper use of language, writing style, references, figures, tables, captions, *etc.*)
- Consideration of key aspects in bioinformatics software development, such as reproducibility, testing, and usability.

Note that not submitting one (or more) of the tasks, will severely affect your overall mark.

6 AI Usage Policy for MMB5009 Assignment

Students are permitted to use generative AI (such as ChatGPT) or other Artificial Intelligence (AI) models to assist in the development of code or documentation for this assignment, provided they adhere to the following guidelines:

1. **Personal Contribution**
   - The use of AI models for code generation or documentation is **optional**.
   - The assignment must not be solely or predominantly AI-generated. Students are expected to make meaningful personal contributions to the code, demonstrating their individual understanding and effort. AI can serve as a support tool, but the majority of the work should reflect the student's own knowledge and problem-solving approach.
   - If AI models are used, their output should be critically reviewed.
2. **Understanding of Code**
   - Any code generated by AI models must be fully understood by the student submitting it. Students should be prepared to explain the purpose, structure, and functionality of all code included in their submission.
   - In cases where the lecturers suspect a lack of understanding, students may be asked to demonstrate or explain their code in further detail. **Failure to explain workings of any AI generated code will result in a failing grade**.
3. **Disclosure of AI Use**
   - Any use of AI tools (*e.g.* ChatGPT) for code assistance or documentation must be clearly and exhaustively disclosed within the assignment. This can be done in comments within the code or in a separate section of the assignment, specifying how AI was used (e.g. code generation, debugging, etc.).

This policy aims to encourage responsible use of AI while ensuring that students achieve the learning objectives associated with developing coding competency.

THE END