Practical 3: Genetics Practical

GD5302: Digital Health Practice

Due date: XXXX XXXXX XXXXX 2021 (Week X) at XX:00
30% of the coursework grade.
Please note that MMS is definitive for weighting and deadlines, which, occasionally have to be changed.

Aims

The main aim of this practical is to identify an unknown sample using DNA sequencing. You will assemble this DNA sequencing data to generate a genome assembly (a representation of the genetic code of this organism) and identify what organism it was.

Task

The hospital has some patients with unknown diseases and they need our help diagnosing what the patient is suffering from.

We cannot diagnose this infection based on symptoms alone. Therefore, we have isolated and sent the unknown (clean culture) sample for DNA sequencing (Illumina `https://en.wikipedia.org/wiki/Illumina_dye_sequencing`). The output from the DNA sequencing matching are fastq files. The format can be found here: `https://en.wikipedia.org/wiki/FASTQ_format`. Based on the sequencing output we will identify the causative disease.

We will use these fastq files (if the files end in .gz, this means they are compressed compression is always a good idea to save space), which are millions of fragments of the original genetic sequence. These millions of fragments will go into a "genome assembler" software package and result in an estimation of the original genetic code, more commonly called the genome sequence. Genome assemblies are represented as fasta format: `https://en.wikipedia.org/wiki/FASTA_format` and graph format `https://en.wikipedia.org/wiki/Sequence_graph`.

From the sequencing data originally obtained from a single patient we will generate a genome sequence, predict the genes from this genome and sequence similarity search against known databases in order to identify what it was. In this task you will need to do this for four patients (patients A, B, C and D) each of which may have different disease.

The solution is expected to consist of several steps:

1. Quality control the fastq data. This involves the removal of low quality bases (where there is a low probability of a specific base being called correctly), see the fastq format above. You might want to use fastqc and trimmomatic software packages for this.

2. Assemble the fastq data into a genome assembly. Use different kmer values and different assembler software packages (e.g. Velvet, Spades, and Unicycler) to yield the best assembly (in terms of contiguity). Note later you have to find which assember is the best make sure you justify this later.

3. Predict the genes from the genome assembly. This is a theoretical prediction of the coding sequences encoded within this genome sequence. Genes directly relate to protein sequences and thus are the functional unit within the cell. You may want to use the software package prokka for this.

4. Write a python script to parse the genome assembly and calculate the GC content (as a percentage e.g. 12%) of the assembled genome (use the fasta file). A genome assembly should be entirely made up of the DNA bases A,T,C and G. Sometimes you will see an N. This is an unknown base. Just to clarify this is the total percentage of C and G letters, out of all the letters in the assembly. Use the GC value to identify if any of the (four) patients have the potential to have the same disease.

5. Write a python script to parse the genome assembly (use the fasta file) and work out the percentage of each of the letters A,T,C and G. Plot the percentage of the total number of each of these letters in the assembly.

6. Write a python script to calculate the length of all the fragments (use the fasta files), nearly all genome assembly attempts result in many fragments. Plot these fragments as a histogram. Try this for all your assembly attempts and use this to prioritise and justify which of the assemblers (Velvet, Spades, and Unicycler) and assembly (fasta file) is the best for each patient/disease.

7. Use sequence similarity search to identify what the organism was by comparing it to public databases (Genbank nucleotide), you might want to use the blastn software package, alternatively you can use the online web version: `https://blast.ncbi.nlm.nih.gov/Blast.cgi?PAGE_TYPE=BlastSearch`. Explain why and how this organism has been identified.

Each of these steps should be clearly explained in the report. In all cases, you should explain what you did, why you did it that way, what the results show, what the results mean and put these into context. Evidence and justify your arguments both with relevant citations and with the output of your analysis.

Try to keep the report informative and focussed on the important details and insights – the report also demonstrates an understanding of what is important. There is a maximum page limit of 10 pages, including figures but excluding references, note that this is a limit not a target. All figures must be referenced in the main body of the text, and have captions and must have legible axis labels.

Deliverables

Hand in via MMS, by the deadline of XXXXXXXXXXX:

- The bash scripts and the Python code that you write. But do not include any outputted files or linked software.
- A report in PDF format which contains details of each step of the process, justification for any decisions you take, and an evaluation of the final analysis. This should also contain evidence of functionality (via your results in the report) and any notable figures you have produced with relevant citations throughout.

Please create a `.zip` file containing both and submit this to MMS. Please note that MMS is definitive for weighting and deadlines, which, occasionally have to be changed.

Marking and Extensions

This practical will be marked according to the graduate school mark descriptors. All documents relating to the mark descriptors (and their conversion to the 20 point scale) can be found on the graduate school webpages: `https://www.st-andrews.ac.uk/graduate-school/students/msc-and-mlitt-documents/`.

For the mark descriptors see: `https://www.st-andrews.ac.uk/assets/university/graduate-school/documents/marks-descriptors.pdf`. You should also be aware of the following marking guidelines for this practical:

- To get a mark **in the 0-22% band** is a submission which shows little evidence of any attempt to complete the work.
- To get mark **in the 23-39% band** is a submission which shows little evidence of any acceptable attempt to complete the work, with no substantial relevant material submitted.
- A *partial working implementation* **in the 40-49% grade band** is a submission which shows evidence of a reasonable attempt addressing some of the requirements, or is accompanied by a very weak report which does not evidence good understanding.
- A *basic implementation* **in the 50-59% band** is a submission which achieves a solution in a straight-forward way and contains some evaluation, but is lacking in quality and detail, or is accompanied by a weaker report which does not evidence good understanding.
- An implementation **in the 60-68% range** should complete all parts of the specification, consist of clean and understandable code, and be accompanied by a good report which clearly describes the process and reasoning behind each step and contains a good discussion of the achieved results including graphs and evaluation measures.
- To achieve a grade of **69% and higher**, should have excellent justification and experimentation into the methods used with many relevant citations linking with the literature. Using more than one algorithm/software package does not constitute an extension unless accompanied by additional insight and analysis gained from this.

Note that the goal is *solid methodology and understanding* rather than a collection of extensions – a good scientific approach and analysis are difficult, whereas running many different algorithms on the same data is easy. Be thorough in your solution and strengthen your basic argument and methodology.
Also note that:

- We will not focus on software engineering practice and advanced Python techniques when marking, but your code should be sensibly organised, commented, and easy to follow. The result of your code must be in the pdf report to evidence that your code worked.
- Overlength penalty: Scheme A, 1 mark for work that is 10% over-length, then a further 1 mark per additional 10% over. See `https://www.st-andrews.ac.uk/policy/academic-policies-assessment-examination-and-award-coursework-penalties/coursework-penalties.pdf`
- Lateness penalty B. 5% of the maximum available mark per 8-hour period, or part thereof. See `https://www.st-andrews.ac.uk/policy/academic-policies-assessment-examinat` `coursework-penalties.pdf`

- Details for good academic practice are outlined on the University webpages here: `https://www.st-andrews.ac.uk/students/rules/academicpractice/`
- For more details on Graduate School penalties, extension request etc, please refer to `https://www.standrews.ac.uk/graduate-school/students/rules/`