Bioinformatic Problem – Part B

1 Introduction

This document presents an overview of the bioinformatic problem and the methodological choices employed to solve the problem. Using the programming language ‘Python’ and the command-line interface, I have produced a script highlighting the appropriate solutions for the problem. The bioinformatic problem: “Produce a biologically meaningful summary of sequence variation between two multifasta files relating to BRCA1 variants”. There are two FASTA sequence files associated with the problem they are: ‘BRCA1\_mRNA\_isolate\_D.fasta’ and ‘BRCA1\_partial\_gene\_isolates.fasta’.

2 Background

This section details background information on the BRCA gene, BRCA1 file(s), sequence alignment, alignment techniques & types, alignment programmes (aligners) as well as what is meant by “biologically meaningful”.

2.1 BRCA Gene & BRCA1 File(s)

During translation, mRNA codons are decoded to help build proteins that contain long sequence of amino acids. An example is the mRNA codon ‘AUG’ which specifies the amino acid ‘methionine’ while also acting as a starter codon, signalling the start of protein construction. ‘BRCA’ is the ‘breast cancer gene’. BRCA1 and BRCA2 are genes in the human body which yield tumour suppressor proteins. Therefore, these genes are vital to the cell’s stability and protect against damage. If a mutation occurs in one the BRCA genes, then the ability to create a functional protein and/or repair damage is reduced. The two multifasta files represent different regions and/or snippets of BRCA1 mRNA and a fragment of the BRCA1 gene. By studying genomic BRCA data, it may be possible to identify mutations in the breast cancer susceptibility genes which could help distinguish woman who are at high risk of developing breast cancer.

2.2 Sequence Alignment

The development of sequence comparison algorithms in the 1980s and 1990s revolutionised the computational and molecular biology fields (Zielenzinski et al, 2017). ‘Alignment’ in principle, is the assignment of residue-residue correspondences where nucleotide bases or amino acids, which are in the same order in two or more sequences, are grouped together to form the basis of alignment. Every individual character in an alignment can be categorised in to at least one of three states. The three states are: ‘match’, ‘mismatch’ and ‘gap’. A match is when the two residues in the alignment are the same. Positions that are the same are known as highly conserved or have a high conservation. A mismatch is when the characters in the alignment are not the same, also called a substitution. Positions that are different have a low conservation. An Indel is when the alignment contains a gap character (‘\_’) ‘indel’ stands for insertion/deletion. Most alignment programmes model inserted/deleted states (Zielenzinski et al, 2017). This can help when the sequences are not the same length. Performing an alignment makes it easy to compute similarity between two sequences. Equally, aligning sequences is an important part when we want to study disease causing mutation.

Alignment software’s consist of ‘comparison matrices’ which define scores for the three states described above. These scores are a ‘tally’ of how well the alignment is doing. The software will search for the highest score available (Mullan, 2005). Note: this final score is only relevant to the alignment and cannot be understood outside of the context. Alignment software’s come in three types they are Database Search Only, Pair-wise and Multiple Sequence Alignment.

An alignment technique which compares two biological sequences or DNA/RNA, or protein is the ‘Pairwise’ sequence comparison. A further technique which compares three or more biological sequences is called ‘Multiple Sequence Alignment’ (MSA). MSA are generally coupled with global approaches of alignment (see next paragraph). Examples of MSA alignment tools include MAFFT (Katoh et al, 2002) , MUSCLE (Edgar RC, 2004) and CLUSTALW (Thompson et al, 1994).

We should think about which part of sequences to align. Sometimes we align sequences, end to end, this is known as ‘Global’ alignment. If two sequences have approximately the same length and are similar, they are suitable for global alignment. This alignment is usually done for comparing homologous genes. A general global alignment technique is the Needleman-Wunsch algorithm. Comparatively, if one would like to target regions with the highest level of similarity between two sequences they would use the ‘Local’ alignment technique. Local alignment is suitable for aligning divergent sequences/conserved domains or motifs. A general local alignment method is the Smith-Waterman algorithm. A basic local alignment search tool is ‘BLAST’ (Altschul et al, 1997).

“Biologically meaningful” interpretation is very much a broad statement and will vary in individual cases and/or questions. Examples of the types of things I would consider biologically meaningful in this case would be to study:-

1. Number of nitrogenous base difference(s)
2. Transitions or Transversion occurrences
3. Indels formations (leading to frameshifts or premature stop codon)
4. Single Nucleotide Polymorphisms (SNP’s) occurrences
5. Synonymous or Nonsynonymous variants

Aim: Utilise Python to produce functions which deliver biologically meaningful interpretation, using pairwise comparison, for the two multifasta files: ‘BRCA1\_mRNA\_isolate\_D.fasta’ and ‘BRCA1\_partial\_gene\_isolates.fasta’.

3 Problem Elements

This section details all the aspects of the bioinformatic problem, and the methodical steps taken to solve the problem.

3.1 FASTA Files

I had decided to use the multiple sequence alignment (MSA) programme MAFFT to locally align sequences from each BRCA1 FASTA file. Multiple sequences in each individual FASTA file, associated with the problem, shared a large degree of similarity and using a local technique of aligning specific regions with highest level of syntony would make it easier to identify divergent regions or region with low syntony which could indicate some sort of biological significance between alignments e.g. Indel, substitution, transition, transversion formation. Also aligning highly similar local regions with each other creates better alignments with far less computed gaps (“\_”).

The first thing I had done was open the files and examine the lengths of the all the sequences in the files. In order for an alignment, lengths of sequences must be the equivalent or near. Using the UBUNTU on windows MAFFT (version 1709) I had fed in the FASTA files (separately, as they cannot be aligned to each other – because they reside in different regions) into the MSA programme, MAFFT (Katoh et al, 2002) which was downloaded from https://mafft.cbrc.jp/alignment/software/ubuntu\_on\_windows.html. I used UBUNTU-MAFFT, as a standalone alignment programme on the command line. Once in the correct directory, I ran “MAFFT” and following the input/output style of commands and various other arguments (like strategy [L-INS-i] and output format [2-clustal format] and additional arguments [--ep]), I had produced two separate CLUSTAL format alignment (.aln). Gap characters (‘\_’) were inserted into the alignments to reach similar sequence lengths.

Command-line Script:

@Clustalo.exe -I BRCA1\_mRNA\_isolate\_D.fasta –outfmt=clu -o mRNA\_isolate\_D2.aln

@Clustalo.exe -I BRCA1\_partial\_gene\_isolates.fasta –outfmt=clu -o partial\_gene\_isolates2

Strategy?

@L-INS-i (accurate)

Additional arguments?

@--ep

Command

@[Y]

**Note: I will attach the aln files to this submission. For your reference.**

3.2 Create Pairwise Comparisons

Creating pairwise alignments was an important step in solving this problem. Once the alignments were created in the command line, I had imported them into a Jupiter Notebook using the read in function (‘r’). Here, with help of a dictionary, the ‘Bio: AlignIO’ module and ‘Itertool combinations’ module, every combination of pairwise comparisons was created and accessed using the ‘biologically meaningful functions’ – see below for function details. Firstly, I parsed the aln file using ‘Bio: AlignIO’, this helped me gain access to all the alignment sequences and their sequence headers. Code such as “align.format(‘clustal’)” came in handy to visualise the alignments sequence by sequence in the stdout. After parse, a dictionary was created to hold both key and value, as header and sequence as a list of immutable tuples. The dictionary was formatted as “records.id” for headers and “records.seq” for all sequences from the aln file.

To create a list of all pairwise combinations, I had imported the ‘itertool combination’ module. Using the code “for x, y in (list(itertools.combinations(align, 2))):” this produces all combinations of two (using x & y as placeholder), without repeated elements being produced. The following print statement: “print(‘({0}) vs ({1})’.format(x.id.strip("\_"), y.id.strip("\_")))” was applied so that the comparisons were easily visualised in the stdout. A long list of sequence headers were outputted in the stdout. It was very important to iterate through all the pairwise combinations so that biologically meaningful data could be produced from all the comparisons as computationally possible.

3.3 Biologically Assess the Variations

‘Indel’ is a biological term used to describe an insertion or a deletion of nitrogenous bases in a genome. Indels are structural variants which contribute to genome instability and/or disease e.g. BRCA1/BRCA2 mutations. A transition is a change of ‘purines’ like A-G or ‘pyrimidines’ like C-T. Transversions are exchanges between the purines and pyrimidines.

To biologically assess sequence variation of all pairwise comparisons, a series of Python function have been created to A. detect nitrogenous base differences between pairwise combinations B. detect the number of substitutions and Indels with in a pairwise comparison and C. a function to detect transitions and/or transversions between two sequences.

The first function is called ‘count\_diffs’, which takes two parameters, ‘seq1’ & ‘seq2’. A single count is added to a count function for every time seq1[base] does not equal seq2[base]. A use case example is count\_diffs(x.seq, y.seq) #where x.seq and y.seq are pairs of aligned sequences. The function is being applied to two sequences x and y. What is return is the number of differences as an integer. We could properly format this function using “print("THERE ARE", count\_diffs(x.seq, y.seq), "DIFFERENCES")”.

The second function is called ‘compare\_seqs’. This function takes in two parameters, ‘sequence1’ and ‘sequence2’ and compares pairs of aligned sequences and extracts the base pairs that differ. This function returns a list of lists, each containing two elements sequence1 base and sequence2 base. This function is useful as we can output the pairwise differences as a list of lists.

The third function is called ‘analyse\_diffs’ and takes only a single parameter, ‘diff\_list’. This function takes a list of base pair couples and counts the number of substitutions and indels using a count. This function utilises a list of all possible nitrogenous bases called “bases\_list = [‘a’, ‘t’, ‘c’, ‘g’]” and iterates through base pair couples to see if indeed elements[0] and elements[1] are in “bases\_list”. If the base couples correspond to items in “bases\_list” then a count is added to ‘substitution\_count’, if divergent to the bases\_list, then a single count is added to ‘indel\_count’. The correct format to use this function is “print(analyse\_diffs(compare\_seqs(x.seq, y.seq)))”.

The fourth function is called ‘transition\_transversion’ and takes in two parameters ‘seq1 and seq2’. This function describes base pair ‘sets’ which define a transition only, then analyses pairs of aligned sequences and extracts and produces a count for both transitions and transversions. The format to use this function is “transition\_transversion(x.seq, y.seq))”.

In conclusion sequence alignment is a common experiment performed by biologists. They are hypothesis based, and formally considered an ‘experiment’ because their outcome is normally unknown. A assortment of functions have been created to biologically assess two FASTA sequence files associated with BRCA1 isolates.

References

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D. J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acids research 25 (17), 3389-3402.

Edgar, R. C. MUSCLE: multiple sequence alignment with improved accuracy and speed. 2004. IEEE.

Katoh, Kazutaka, Misawa, Kazuharu, Kuma, Kei, i., Toh, Hiroyuki, Miyata, Takashi, Inst. Chem, R., Kazusa, D. N. A. R. I., Chiba Industry Adv, C., Biohistory Res, H., Waseda, U., Bioinform, C., Sci, E., Kyoto, U. and Create (2004) 2D-08 MAFFT： a multiple sequence alignment program. Genes & Genetic Systems 79 (6), 419-419.

Mullan, L. (2006) Pairwise sequence alignment--it's all about us. Briefings in bioinformatics 7 (1), 113-115.

Thompson, J. D., Higgins, D. G. and Gibson, T. J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic acids research 22 (22), 4673-4680.

Zielezinski, A., Vinga, S., Almeida, J. and Karlowski, W. M. (2017) Alignment-free sequence comparison: benefits, applications, and tools. Genome biology 18 (1), 186-17.

Links

https://mafft.cbrc.jp/alignment/software/ubuntu\_on\_windows.html

GitHub:

https://github.com/mbilal1995/CW1-Bioinformatic-Problem