Computational Thinking 2018/19
Bioinformatics Coursework

Rob Powell

Hand in by 2pm on 3rd May 2019 via DUO.

Coursework Description

This coursework consists of three parts. The first is to develop a Python program for aligning two DNA sequences, the second is to draw by hand a phylogenetic tree from an inter-species distance matrix, and the third is to develop a Python program for drawing phylogenetic tress from inter-species distance matrices. You should submit your two pieces of Python code and a single PDF file (showing your phylogenetic tree and workings) via DUO.

Objective One: Python Programming to align two DNA sequences

Develop a Python program that uses Dynamic Programming (as described in the lectures) to compute the optimal alignment of two sequences. It should meet the following conditions:

- The algorithm should read in two sequences from text files (the sequences may be of different lengths).
- Initialise the scoring matrix and the backtracking matrix.
- Score each alignment using the following scores:
  - +4 for a matching pair of 'A' bases,
  - +3 for a matching pair of 'C' bases,
  - +2 for a matching pair of 'G' bases,
  - +1 for a matching pair of 'T' bases,
  - -3 for a mismatching pair of bases,
  - -2 for a gap.
- Use the backtracking matrix to determine an optimal alignment.
- The program should print out the following:
  - The optimal alignment and its score.
  - Execution time.

As an example consider the following alignment of two sequences of length 4. It would be given score of -1.

A	A	T	G	–
A	C	–	G	T
+4	-3	-2	+2	-2

Note: A template is provided for you that covers the first and last bullet point, and you may use a package such as numpy for matrices if you find it helpful. You should test your program with the test sequences provided on DUO. These will allow you to check your program is working as expected. Your program should be able to handle two input sequences of different length e.g aligning a sequence of length 8 with a sequence of length 10, as well as two sequences of the same length.

Objective Two: Constructing a Phylogenetic Tree by hand

Using the following matrix of inter-species distances, construct a phylogenetic tree using the UPGMA algorithm (as described in the lectures). You should submit a PDF file showing all of your working, and the final phylogenetic tree that links the species **a**,**b**,**c**,**d**,**e**, and **f**. This tree should include edge weights, but I do not expect it to be drawn to scale.

	a	**b**	**c**	**d**	**e**	**f**
a	0	15	24	29	25	37
b	15	0	32	31	23	43
c	24	32	0	30	43	49
d	29	31	30	0	45	57
e	25	23	43	45	0	55
f	37	43	49	57	55	0

Objective Three: Python Programming to calculate and draw phylogenetic trees

Develop a Python program that uses the WPGMA algorithm (as described in the lectures) to draw a phylogenetic tree from an input matrix of inter-species distances. It should meet the following conditions:

- Your code should have a function called WPGMA which takes as its input the name of a text file.
- The algorithm should read in an inter-species distance matrix from this text file, and print this matrix to the screen.
- Follow the WPGMA algorithm, and print out to the screen every reduced distance matrix.
- After fully reducing the matrix, the code should draw the phylogentic tree obtained to a file. This tree does not need to contain edge weights. To achieve this you should use the packages networkx and matplotlib.
- Finally the program should print out the execution time to the screen.

Note: You may use a package such as numpy for matrices if you find it helpful. There are two test matrices stored in text files available on DUO. These will allow you to check your program is working as expected. Your program should be able to handle an input matrix of any size.

Marking Criteria

The breakdown of marks are as follows:

- Dynamic Programming Python code - 35%
- Constructing a Phylogenetic Tree by hand - 30%
- Constructing a Phylogenetic Tree using Python code - 35%

5 of the 35 marks for each programming aspect will be given for how efficiently your code executes. These will be awarded based on a sliding scale against the execution time of a Python program of my own. The code must work on all test cases to achieve these marks.