## P5: Python programming

### Description

The identification of similar DNA or protein sequences has many applications. For example, for a newly determined sequence you want to search for similar sequences (in related species) that can possibly reveal information on the genomic function of the novel sequence. Because of mutations, homologous sequences in related species are rarely identical. To determine the identity of two sequences they have to be aligned. The Needleman-Wunsch algorithm is a method to find the optimal global (=spanning the full sequence lengths) alignment between two sequences.

In this assignment you will study the sequence similarity between several members of a protein family, using the program **`needle`**. As a case study we use a family of Guanine nucleotide binding proteins (G proteins). G proteins and their receptors (GPCRs) form one of the most prevalent signaling systems in mammalian cells, regulating systems such as hormonal regulation. These proteins are composed of three subunits: alpha, beta, and gamma. As a reference, we use the Guanine nucleotide-binding protein alpha-1 subunit of *Arabidopsis thaliana*.

```
Sequence alignment

SEQ1: KSYVPVIHA-
      || ||| ||
SEQ2: KS-LPVVHAN
```

### Input

A FASTA file containing a single record: the GPA1 protein from Arabidopsis (ref.fasta). A FASTA file containing multiple protein sequences: GPA1 proteins from other plant species (related.fasta)
Use the command **`wget`** to download the files from this location:
**`http://www.bioinformatics.nl/courses/BIF-30806/docs/ref.fasta`**
**`http://www.bioinformatics.nl/courses/BIF-30806/docs/related.fasta`**

### Assignment

Write a python script that performs the following tasks:

1. Read filenames specified on the command line (using argv)
2. Parse FASTA files with multiple protein sequences and determine the lengths of the sequences
3. In your python script, run the program **`needle`** to align the protein sequences of related species (related.fasta) to the reference protein from *A. thaliana* (ref.fasta).
   - o Set the gap open penalty to 8 (your program should work for any gap open penalty between 3 and 10)
   - o Set the gap extension penalty to 0.5
   - o Call the output file out.needle
4. Write a function to calculate the hamming distance between two sequences of equal length (defined as: the number of positions at which the corresponding symbols are different.)
5. Write a function to calculate the percent of identity between two aligned sequences (defined as: number_of_identical_positions / alignment_length * 100).
6. Parse the **`needle`** output to extract all pairwise alignments
7. For each alignment, report (tab-delimited):
   - o Sequence labels
   - o Length of the input (=unaligned) sequences (from step 1)
   - o The hamming distance between the aligned sequences, using your own function.
   - o The percent of identity between the aligned sequences, using your own function.

**Output**

The output of your script should look like this: (Note that the labels and numbers in this output are made up. The labels/numbers in your output will be different!)

```
Sequence1    Length   Sequence2    Length   Hamm     Ident
GBB_ARATH    377      GBB_SOLTU    377      71       81.3
GBB_ARATH    377      GBB_MAIZE    380      91       76.4
```

**Environment**

You should work on the server **`altschul.bioinformatics.nl`**
The program **`needle`** is installed there. Try it by typing:

```
     needle –h
```

on the command line. You should see information on the usage and options.
Some more documentation can be found on:
https://www.ebi.ac.uk/Tools/psa/emboss_needle/help/index-protein.html

**Additional notes**

- Put your full name and student number as a comment in your script and put your name in the file name of your script.
- You may use the slides and the code from your exercises from the first course week. You cannot use BioPython or comparable packages, you should write your own fasta parser. You may not directly copy code from the internet, but you may use it as inspiration.
- Running **`needle`** takes only a few seconds or so. But to avoid running it over and over again, make sure your code checks whether the output file exists.
- Think about your code organization. Use subroutines.
- Document your code.
- Make sure you hand in a working script. If it is unfinished, you can leave the unfinished part in comments.

**Turn it in on BlackBoard**

Make sure your **name** and **student number** are in your python script. **Turn in your script on BlackBoard (under P5)**.

***Example submission information for the exam***

***How to hand it in?***

*Monday 28-4-2014 before 17:00, email your **Python script** and **usage string** (e.g. python script_name.py inp1 inp2) to **pythonexam@bioinformatics.nl***

***Assessment***

*We will run your script on the input, check the output, and assess the quality of your code. The grade for this assignment will be 40% of your course grade.*