LAB 2: Aligning reads to reference

ASSIGNMENTS

First, take a look at the **LAB2_Tips** file and practice with each section described. We strongly believe that they could be helpful with the following assignments.

Assignment 1: Global alignment

Under the assumption that both input sequences a and b stem from the same origin, a global alignment tries to identify matching parts and the changes needed to transfer one sequence into the other. The changes are scored and an optimal set of changes is identified, which defines an alignment. The dynamic programming approach tabularizes optimal subsolutions in matrix D, where an entry $D_{i,j}$ represents the best score for aligning the prefixes $a_{1..i}$ with $b_{1..j}$.

To better clarify how global alignment works, take a look here: http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Needleman-Wunsch

Then, write a Python program that given two sequences (passed as first and second argument from command line) and match, mismatch and gap costs (passed as third, fourth, fifth argument from command line):

1. Compute matrix D and output it on the terminal, along with the final alignment score
2. Output the final alignment (if two sequences have more than one alignment with the same score, provide one of them e.g. check website for 'AACCG' and 'AACG')
3. Check your alignment on Freiburg website

Usage should be something like this:

```
python global_alignment.py AACCG AACG 1 -1 -2
```

Output:

```
Global alignment score: 2.0
[[  0.  -2.  -4.  -6.  -8.]
 [ -2.   1.  -1.  -3.  -5.]
 [ -4.  -1.   2.   0.  -2.]
 [ -6.  -3.   0.   3.   1.]
 [ -8.  -5.  -2.   1.   2.]
 [-10.  -7.  -4.  -1.   2.]]

Final alignment:
AACCG
||| |
AAC-G
```

Assignment 2: Local alignment

A local alignment approach tries to identify the most similar subsequences that maximize the scoring of their matching parts and the changes needed to transfer one subsequence into the other. The dynamic programming approach tabularizes optimal subsolutions in matrix S, where an entry $S_{i,j}$ represents the maximal similarity score for any local alignment of the (sub)prefixes $a_{x..i}$ with $b_{y..j}$, where x,y>0 are so far unknown and have to be identified via traceback.

To better clarify how local alignment works, take a look here: http://rna.informatik.uni-freiburg.de/Teaching/index.jsp?toolName=Smith-Waterman

Then, write a Python program that given two sequences (passed as first and second argument from command line) and match, mismatch and gap cost (passed as third, fourth, fifth argument from command line).

1. Compute matrix D and output it on the terminal, along with the final alignment score (**remember that the maximum value you can have in the matrix is 0!!**)
2. Output the final alignment (if two sequences have more than one alignment with the same score, provide one of them)
3. Check your alignment on Freiburg website
4. **Local alignment has a lot of concepts similar to global alignment, so you can reuse a lot of code you have previously written for global alignment!**

Usage should be something like this:

```
python local_alignment.py AATCG AACG 1 -1 -2
```

Output:

```
Local alignment score: 2.0
[[0. 0. 0. 0. 0.]
 [0. 1. 1. 0. 0.]
 [0. 1. 2. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 2.]]
Final alignment:
AA
||
AA
```

Assignment 3: Run BWA

After implementing a simplified form of global and local alignment, let's now try to use BWA, a widespread alignment tool and practice with its output formats.

Before proceeding with this assignment, follow the instructions in the LAB2_Tips file to install the tool and download reference files for chromosome 10 and chromosome 18.

Remember to download **mate_1.fq** and **mate_2.fq** files from the Teaching Portal.

BWA

1. Create a bwa index for the reference sequence (chr10 and chr18). This process requires some minutes to be performed. Remember that indexing must be performed just once, when you have a new sample from the same species to process you can start directly from step 2.

```
mkdir path_to_folder_where_BWA_index_will_be_created
```

```
bwa index -p path_to_folder_where_BWA_index_will_be_created
path_to_reference_sequence/reference_chr10_chr18.fa
```

2. Perform the alignment of mate_1.fq and mate_2.fq

```
bwa mem path_to_folder_where_BWA_index_has_been_created
path_to_reads_file/mate_1.fq path_to_reads_file/mate_2.fq >
path_to_place_results/results.sam
```

Assignment 4: Create a bash script file

Now, create a bash scripts that allow a user to run BWA taking all the arguments necessary to run the script from the command line. In particular, make sure that the user is informed at each step about what is happening (use the echo command - e.g. echo "BWA aligner is currently running. This step could require time, be patient!").

Example:

```
sh run_bwa.sh genomeref mate_1 ate_2 outputdata
```

```
Output:
```

```
BWA initializing...
```

```
BWA aligner is currently running. This step could require time, be
patient!
```

```
Alignment is finished! You can now check your results.
```