CMPSC 300
Introduction to Bioinformatics
Fall 2017

Lab 5: Finding Regions of Potential Disease Initiation
Save this lab assignment to: `labs/lab5`

Objectives

- To generate Python code to manipulate genetic sequences in a variety of ways and make basic comparisons between sequences.
- To automate a pipeline for sequence analysis and mutation detection.
- Understand the structure and orientation of string representations of DNA and protein sequences.
- Gain experience with string manipulation in a chosen programming language and its application to DNA and protein sequence data.

Developing Your Own Tool

A Single Tool for DNA Mutation Detection

Comparisons between genetic material are extremely common in Bioinformatics. As discussed in class, the presence of alternative SNPs between samples of DNA, may help researchers to conclude whether a disorder can be expected from the organisms. Furthermore, mutations between code samples may also also serve to help researchers determine and potentially explain reasons for genetic abnormalities which may appear in protein formation, structure and, therefore, function.

In this part of the lab you will develop a Python program that can manipulate sequences, transcribe and translate them and find mutations or general differences in sequences.

Required:

To keep your code for implementation simple. Your python program should be able to complete the following steps.

1. Please call your Python program: *TandTcompare.py*

2. Transcription and Translation

   (a) There are two FASTA files given with this lab in your repisitory: *mut.fasta* and *wild.fasta*. Both files contains similar sequences of the CFTR gene, however the *mut.fasta* file contains a mutations and alternative SNPs to the version of the CFTR gene code found in the *wild.fasta* file. Your program should be able to open these files to read them. Their

contents should be loaded into string variables for further manipulation. Biopython libraries may be used to help in this file work.

(b) Your program is to find the **RNA** (nucleotide) sequences of both files using Biopython. Record these RNA sequences in a file as prepared by your program (again, Biopython may be used to help with this step). Make sure to label your files so that you can trace them back to their original DNA.

(c) Also using Biopython, convert each of these DNA sequences into **protein** amino acid sequences. Again, output these sequences to files which have been named to describe where their origins from the DNA files.

3. Automatic Sequence Comparison

(a) Although we have not covered this topic in class, you are to implement an automated system of comparing the DNA, the RNA and the protein files containing genetic information.

(b) Your algorithm is to determine where exactly (i.e., at which position) the genetic strings differ and to show the character at each of these locations. Your output should be simple and count the total number of differences encountered.

(c) you are to compare both DNA files to each other, both RNA files to each other and both protein files to each other.

Mutation Detection Algorithm
This section is to give you ideas on how a comparison could be performed. **Input**: Two strings representing amino acid sequences.
Output: Any differences between two sequences. If there are no differences, output a message stating the sequences are equivalent.

- Traverse each input string from left to right, one character at a time.
- If the character at a given position in the first string is not the same as the character at the same position in the second string, store the position and the mismatching characters for later output.
- Output a list of any mutations found between two sequences. If there are no differences, output a message stating the sequences are equivalent.

Sample Pseudocode for Mutation Detection Algorithm
The below Pseudocode is to provide ideas.

```
count = 0
for each pos from 0 to len(aminoseq1)
    if aminoseq1[pos] != aminoseq2[pos] # not same character at each location in strings
        print aminoseq1[pos] + , + aminoseq2[pos] + , + pos
```

```
        write aminoseq1[pos] + , + aminoseq2[pos] + , + pos # sent to the output file
    count+=1
if count==0
    print no matches found  sequences are the same
    write no matches found  sequences are the same # sent to the output file
```

Required Deliverables

All of the deliverables specified below should be placed into a new folder named 'lab05' in your Bitbucket repository (`cs300f2017-bbill`) and shared with the instructor by correctly using appropriate Git commands, such as `git add -a`, `git commit -m ''your message''` and `git push` to send your documents to the Bitbucket's server. When you have finished, please ensure that you have sent your files correctly to the Bitbucket Web site by checking the `source` files. This will show you your recently pushed files on their web site. Please ask questions, if necessary.

1. A properly formatted and commented Python program that implements a Mutation Detection Algorithm and a snapshot of an output that your program produces.
2. A report to describe exactly where each of the differences occurred between each of the files (i.e., DNA, RNA and protein).
3. Documentation. Clearly written, please describe how your algorithm works to compare sequences. We do not care about the efficiency of this system at this point. Include a screen shot of your program to show how it shows the output.

You should see the instructor if you have questions about assignment submission.