CMPSC 300
Introduction to Bioinformatics
Fall 2017

Lab 5: Genomic Regions Associated with Parkinson's Disease
Save this lab assignment to: `labs/lab5`

Objectives

- To become familiar with tools that can be used to manipulate sequences in a variety of ways and make basic comparisons between sequences.
- 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.
- Understand how genetic information is computationally decided and appreciate the important complications in working with sequences (introns/exons, start codons, template/nontemplate strand orientation, etc.).

Reading Assignment

Chapter 2 in Exploring Bioinformatics textbook.

Existing Tools

Part 1: Obtaining the mRNA Sequence of CFTR

Mary and her husband Tom would like to start a family. However, Mary's mother has cystic fibrosis (CF) and Mary carries an allele for this fatal genetic disorder. Tom was adopted at a very young age and little is know of his biological parents. Mary and Tom have sought the assistance of a genetic counselor, who has recommended genetic screening for Tom. The genetic counselor would like to send Tom to a genetic testing laboratory to take a DNA sample from Tom. The lab may then sequence Tom's CFTR gene and use their software to look for mutations in the DNA sequence and their effects on the protein. In this laboratory, you will examine real CFTR sequences and arrive at a genetic diagnosis.

For the purposes of this exercise, we will limit ourselves to the analyses of the sequence corresponding to the spliced CFTR mRNA; analysis of the much longer CFTR genomic DNA sequence is complicated by the interruption of the coding sequence by introns (more on that in a later chapter).

1. Go to the NCBI website and search the Gene database for CFTR. Open the Gene record for the human CFTR gene, then scroll down to find the link for the Genbank record. Right click to open the Genbank record in a new file (with the entire 189-kb sequence).

2. In the Gene record, you should also be able to find an accession number for the CFTR mRNA that links you to this much shorter sequence. Open the mRNA record and scroll to the bottom. Copy the entire mRNA sequence, including numbers and spaces, and save it in a text file (gedit).

   a. How many nucleotides make up the mRNA sequence transcribed from the CFTR gene?

Part 2: Tools for Manipulating DNA

3. Use Google to perform a search for the Sequence Manipulation Suite (SMS). This is a set of Bioinformatics tools written in JavaScript that can be run in any web browser.

4. Notice in the left-hand column the many useful tools that are brought together in a single web interface. Some are very simple text manipulations but are very useful in working with real sequences. For example, you many receive a sequence that is not in FASTA format but the programs you are planning to use require FASTA format. Use the Filter DNA tool to covert the mRNA sequence you saved into FASTA format.

5. Paste the resulting FASTA-formatted mRNA sequence into a text editor, change the comment line to something more meaningful, and save the file.

6. Click on the Reverse Complement tool. Notice that from the pull-down menu you have the option to "reverse" (that is, inverse), "complement", or "reverse-complement" your sequence.

   b. Which option would give you the sequence of the template strand of the DNA from which your mRNA sequence was transcribe?

   c. Use the tool to generate the template strand and write out the first five nucleotides.

Part 3: Translating the CFTR mRNA

7. Click on the Translate tool. Notice there is some complexity here: the drop-down menus below the input box allow you to choose the reading frame and also the strand to translate. You could either translate the strand as written (which SMS calls the "direct" strand) or generate its inverse complete and translate that ("reverse" in SMS) which altogether would give the six reading frames.

   d. Paste your FASTA-formatted mRNA sequence into the input box. Write the first three amino acids (single letter abbreviations okay) for each of the six reading frames.

8. You can quickly see that as soon as we start looking at real sequences, things get complicated and you may have no clue what to consider as the true amino-acid sequence for CFTR. To make things easier to visualize, click the Translation Map option and use it to visualize your mRNA sequence. Note that here and in the Translate output, stop codons are indicated by asterisks (*) in the amino-acid sequence. We would expect the CFTR coding region to take up most of the mRNA and not be broken up by stop codons; this consideration should make it easy to identify the correct amino-acid sequence in the correct reading frame.

e. What is the reading frame (1, 2, or 3) and coordinate of the start codon? What is the coordinate of the stop codon (TAG)?

Hint check your answer by returning to the mRNA record on NCBI. Find the sequence titled "Features" and then the Feature titled "CDS" CDS stands for "conserved domain" this is the part of the mRNA that is translated into protein (the open reading frame). Confirm that your coordinates above agree with those listed for the CDS of the CFTR mRNA.

9. To facility sequence comparison, we need to generate two files one with just the coding region of the mRNA and one with just the CFTR amino acid sequence. Use the Range Extractor DNA tool to get a new FASTA-formatted sequence representing just these nucleotides (read the information on how to specify a range carefully and check your output to see if you got what you expected). This is your coding sequence.

   Now, use Translate, set for reading frame 1 and the direct strand, to get the corresponding amino acid sequence. If you have done all this correctly, you should have 1,480 amino acids in your final sequence.

10. To demonstrate one more SMS tool, try running One to Three on your amino acid sequence. Although the one-letter amino acid code is much easier to use in computer programming, many people find the less-cryptic three-letter code easier to work with.

Part 4: Detecting Mutations

11. We can use the alignment tools in SMS to look for mutations in CFTR. Make a copy of your CFTR coding sequence and create a substitution "mutation" by changing one base to a different base.

12. Go to the Pairwise Align Codons tool in SMS and paste your wild-type sequence into the first input box and your mutant sequence into the second. Dont worry about the parameters for now. Submit the sequences and examine the results. Now copy both sequences, including their comment lines, from the results page and paste them into the input box in the Color Align Conservation tool. On the results page you should see a black background wherever the two sequences have the same nucleotide and a white or gray background where they differ; can you find your mutation?

13. Now translate your mutant coding sequence and align the result with the wild-type amino acid sequence, this time using Pairwise Align Protein (again with Color Align Conservation)

    f. Describe your mutation. Did you make a missense mutation, a nonsense mutation, or a silent mutation? How do you know?

Part 5: Genetic Counseling for Tom and Mary

14. The file CFScreenting.txt has been uploaded to the lab5 directory in the shared repository. This file contains four FASTA-formatted sequences representing the coding regions of each of

Marys two CFTR alleles and the coding regions of each of Toms two CFTR alleles.

g. Compare each of Mary's alleles with the wild-type coding sequence you created in step 9. Describe the mutation(s), if any, that you find in each allele.

h. Translate each of Mary's alleles and compare them with the wild-type amino-acid sequence. What differences can you detect?

i. We know Mary is a carrier of the CF allele but does not have the disease. Summarize your findings for Mary's CFTR alleles: Describe the mutation(s) that have occurred, discuss how (if at all) they affect the CFTR protein, and explain how your genomic data fit with what Mary already knows, including which allele she must have inherited from each of her parents.

j. Repeat your analysis for each of Tom's two alleles. Is he a carrier of a CF allele? Summarize your findings as in question 3 and determine the probability that Tom and Mary will have a child with CF.

Developing Your Own Tool

Part 6: A Single Tool for DNA Mutation Detection

If you have to sequence DNA from people for genetic screening to identify mutations, tools such as SMS are not very practical. It is more efficient to develop a single tool that can manipulate, transcribe, translate and compare sequences to find any mutations. In this part of the lab you will develop a Python program that can manipulate sequences, transcribe and translate them and find mutations.

Required:

To keep your implementation simple you can assume that the sequences are of equal length. Your program should perform the following steps:

1. Open two input files in a FASTA format and one output file.
2. Save the input sequences into String variables and converts both input strings to upper case characters.
3. Manipulate the input sequences to prepare them for transcription.
4. Transcribe the sequences to produce mRNA sequences.
5. Produce amino acid strings.
6. Compare two amino acid sequences and report any mutations including any characters that differ between two comparison sequences and the positions at which the mutations occur. The mutation results should be displayed in the terminal and also saved in the output file. This step involves implementing a Mutation Detection Algorithm, which is outlined below.

Mutation Detection Algorithm

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

```
count = 0
for each c from 0 to len(aminoseq1)
if aminoseq1[c] != aminoseq2[c]
print aminoseq1[c] + , + aminoseq2[c] + , + c
write aminoseq1[c] + , + aminoseq2[c] + , + c  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 to      the output file
```

Optional:

1. Modify your program so that it can also compare two DNA sequences. Your extended program should ask the user to choose whether to compare DNA sequences directly, translate them before comparing, or both, and it should execute the user-defined comparison appropriately.
2. Modify your program so that it is not restricted to sequences of the same length. Requiring that the strings are of the same length is unrealistic because real mutations are often insertions and deletions of one or more nucleotides. If the sequences are of different length, they need to be aligned so that appropriate characters line up. The challenge is in figuring out how to align the sequences. Your solution to this step should:
    - Input and manipulate DNA sequences as before. Omit transcription and translation for now to avoid further complexity.

- Determine the lengths of the two sequences. If they are equal, compare the two strings as before.
- If the strings are of different length, determine the number of nucleotides deleted from or inserted into the second string relative to the first. This is the number of gaps that need to be inserted into the shorter sequences to align it with the longer sequence.
- Construct all possible alignments by placing the desired number of gaps at each possible position in the shorter sequence and determining which alignment has the fewest mismatches. Based on this alignment, output the position and the insertions/deletions found.

First develop a step-by-step algorithm to solve this problem (include this algorithm in your report). Then write a Python program to implement your algorithm.

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. An electronic version of the report containing the answers to the lab comprehension questions in red.
2. A properly formatted and commented Python program that implements a Mutation Detection Algorithm and a snapshot of an output that your program produces.

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