

Problem Set 1

CS 174 Bioinformatics

In this first assignment you will write some code for discovering genes in the yeast genome. The goal is to familiarize yourself with Python and how to do gene discovery in simple organisms.

Assignment

1. Create a file `ps1.py` which will contain your code for this assignment. Add some comment lines at the beginning of the file, indicating “CS174 Problem Set 1” and your name.
2. Download the following file containing the genetic code for translating nucleotide sequences to amino acid sequences:

- <http://www.ics.uci.edu/~xhx/courses/CS174/assignments/PS1/standard-genetic-code.py>

Copy and paste its contents into your script, to be used as a Python dictionary.

3. Write a function `readSeqsFromFile` that reads nucleotide sequences from a file in the FASTA format (the file name is the input to the function). A good starting point for information about this format is:

- http://en.wikipedia.org/wiki/Fasta_format

Don't worry, you don't need to know and implement all the details. In particular, for this assignment we will assume that the input file contains only straight DNA nucleotides (A, C, G, or T).

Your function should return a list of strings, one string for each sequence in the input file. For instance, a return value of `['ATC', 'TGAACCTG']` would encode two sequences of length 3 and 8, respectively.

4. Write a function `translateSeq(sq)` which takes as input a sequence of nucleotides and translates it to an amino acid sequence according to the genetic code from step 2. You can assume the length of the input is a multiple of 3. The result should be a string of amino acid codes.

For instance, calling `translateSeq('GTTCCGTAC')` should produce `'VPY'`.

Test this function on the following FASTA file containing two sequences:

- http://www.ics.uci.edu/~xhx/courses/CS174/assignments/PS1/two_genes_yeast.fa

Note that in this case the two sequences correspond to genes: Their *frame* is provided implicitly, i.e., they begin with the “start” codon (ATG) and end with a “stop” codon.

5. Now assume that the frame of a gene (where to start and end) is unknown. Write a function `findFrames` that identifies genes in a nucleotide sequence. It takes as its input a nucleotide sequence (a string over A, C, G, and T, as above), and outputs a list of tuples, one tuple per gene, denoting its offset and length in the input list.

For example, given the sequence `'GTTAATGCATGCATGACTATGGGCTAAAC'`, the result would be the list `[(4, 9), (12, 12)]`, indicating two genes: at offset 4 with length 9 (excluding the stop codon) and at offset 12 with length 12.

For your actual implementation, however, we define a gene to be a contiguous subsequence that contains *at least 200 amino acids* (after translation from DNA to amino acid sequence) – so make sure to filter out shorter subsequences.

6. Write a function `readFindTrans`. It should make use of the functions you already implemented, and write its results to the console (with the `print` command, for example). Given a filename as its input, this function should do the following:
 - (a) Read the nucleotide sequences from the specified FASTA file.
 - (b) For each sequence:
 - i. Use `findFrames` to identify all genes, in the forward string as well as in its reverse complement.¹
 - ii. For each gene, output its location and the corresponding amino acid sequence.
 - iii. Output sequence statistics: number of genes found in this sequence, length (in amino acids) of shortest and longest gene in the sequence.
 - (c) Output overall statistics: number of sequences, overall number of genes, and length (in amino acids) of shortest and longest gene across all sequences.

Test your implementation on chromosome 1 of the yeast genome:

- <http://www.ics.uci.edu/~xhx/courses/CS174/assignments/PS1/chr1.fa>

¹You can extend `findFrames` to do this internally, but it is probably easier to just generate the reverse complement within `readFindTrans` and call `findFrames` on that again.

7. Extend your script so that it can be called from the command line through the command `python ps1.py <filename>`. Check the Python manual to find out how to do this:
 - <http://docs.python.org/tutorial/modules.html#executing-modules-as-scripts>
8. Run your script on chromosome 1 of the yeast genome that you downloaded in step 6 and record the output into a plain text file called `ps1.txt`.
9. Upload both files `ps1.py` and `ps1.txt` to the assignment submission folder in the EEE dropbox "CS174 Problem Set 1"
10. Download the full yeast genome consisting of 16 chromosomes, 12 Megabytes unpacked:
 - http://www.ics.uci.edu/~xhx/courses/CS174/assignments/PS1/yeast_genome.fa.gz

Unpack the file (with *gzip/gunzip*) and try running your script on it. Output the genes discovered by your code into a text file with each line showing the chromosome, the start position, the end position, and the strand of one gene.

General remarks

- Keep your code legible so we can read and grade it, if necessary. For instance, variables should have meaningful names. Use comments to explain principles and ideas, but don't comment on every single statement.
- Try to have complexity in mind when writing your code (this will become more relevant later on).