Biol 591 Introduction to Bioinformatics (Fall 2003)
Problem Set 8: Markov models

This problem set assumes that you already have a working version of `MakeMarkov.pl`, the object of the notes for Monday, Dec 1. If you haven't yet worked through those notes and the accompanying study questions, that's step 1. The end product of the program will be a file containing a third-order Markov model based on the training set `6803PHX.nt`.

PS8.1: Download and run the program `Run_orfs_through_model.pl` (do I have to say anymore that you will probably need to change the **Files** section?). You will no doubt get some warning messages. For now, just note the problems and go on (it will take a couple of dozen clicks – don't lose patience).

1a. The reported nucleotide frequencies differ from what we've used in the past for *Anabaena*. Confirm this and offer ***two*** reasons for the difference.

1b. Why did those warning messages arise? Should they have? Use `Display_hash.pl` to look at the model produced by `MakeMarkov.pl`. Do you find the source of the problem? What *biological* conclusion do you draw?

1c. Bring up the output of the program into Excel. Insert a first row and use it to label each column. To find out what each column represents, you may have to refer to the program that made the file.

1d. Sort the table by score. To do this, click on the upper left corner of the table (above **1** and to the left of **A**), click on **Data**, then **Sort**. Choose **Score** from the pulldown menu under **Sort by**. Don't have **Score** on that menu? Then go back to **1c**.

1e. Qualitatively, what does the score mean? Very negative scores mean what? Very positive scores? A score of zero? To answer this, you may need to cogitate on the part of the program that calculates the score.

1f. Scan the distribution of scores to get a crude idea of what you have. One way to do this quickly is to select the **Score** column, click on the **Chart Wizard** icon (a tiny bar graph in the toolbar), select **Line** (counterintuitively; this plots the contents of the column against the row number), press **Finish**). Describe the distribution. Is it what you expect? Explain. In particular, where did the 38 most negative scores come from?

1g. PageDown through the table, noting what kind of genes you see at different scores. Any generalizations?

1h. There are other methods besides Markov models to detect anomalous genes, such as codon bias (used by Mrazak et al; see link to this article on unit web page). Compare the results of this method to our results, using the same training set and collection of orfs (see `6803orfs_codon_bias.xls`, available from the web page). The file shows many scores, and Mrazak et al uses several in combination to decide on whether a gene is anomalous. In the end, they come up with a yes/no answer (indicated in the file as PA/blank). See article for explanation. Which method looks better to predict alien genes: our score generated from the Markov model or Mrazak et al's procedure derived from codon bias analysis? How to decide?

PS8.2: Let's fix those anomalous 38 hypernegative scores. The situation is really analogous to the problem we faced with PSSMs. Initially, if a column in a PSSM didn't happen to have a particular nucleotide, then the scoring matrix gave a score of 0 to that nucleotide. In other words, it considered that if the nucleotide didn't occur in the training set, then it would be considered an impossible occurrence, and test sequences that DID have the nucleotide in that position would get trashed.

This is clearly an unreasonable procedure, given a finite training set. Maybe that nucleotide didn't happen to appear at the position in the 100+ sequences of the training set because it is rare, but it ***might have*** appeared if the training set were a bit larger. A test sequence shouldn't be thrown away because it contains a rare (but not impossible) nucleotide!

2a. Examine `Run_orfs_through_model.pl` and determine what the program would do if it encountered a sequence not represented in the training set. What subroutine of the program is responsible for assigning the score?

2b. Consider `Find_motif.pl` (available from the PSSM unit web page). How does ***it*** deal with a nucleotide/position not in its training set? In what section of the program does it deal with it?

2c. Does `Run_orfs_through_model.pl` have a similar subroutine? (hint: yes). Modify that subroutine so that it employs the same solution used by `Find_motif.pl`.

2d. Rerun the program. Any warnings? Bring it up into Excel. Any changes?

PS8.3: Markov models are characterized by their "order".

3a. What order Markov model is used by `Hamlet.pl`?

3b. Describe in concrete terms what a third-order model is and how the program uses it to produce pseudo Hamlet.

3b. Modify `Hamlet.pl` so that it produces and uses a ***fourth-order*** model. Compare the output of the original program to your modified program. Can you understand in what way the output is different?

PS8.4: Is our method to detect anomalous genes sensitive to the choice of genes in the training set? Do an experiment (computational) to find out.

PS8.5: It's always a good idea to check the quality of the data coming into the program. How do we know all the genes coming into the program are really genes? It may be that ribosomal RNA genes or other noncoding RNA genes might be part of the mix, which we should know about. Or maybe some garbage sequence somehow snuck in. Modify the subroutine that reads the FastA-formatted sequence to check whether the sequence is a bona fide orf. (Copy and rename the subroutine something like Read_orf_sequence, because we still may want to read FastA-formatted sequences that are NOT open reading frames).

PS8.6: If we were to write a program to find anomalous genes as judged by codon bias, we would need to know which codons coded for which amino acids. Well, we're NOT going to write that program, but let's at least do the first step, teaching Perl how to translate codons to amino acids. The file `aa_info.txt` (available from the unit web site) has in it a list of amino acids and their codons. Write a program that reads this file and makes a hash (%aa_of) such that $aa_of{"CCC"} gives you "Pro" and so forth.

Hints:

1. *You need to be able to read the file and extract two pieces of information: (a) the three-letter amino acid code and (b) the list of codons. I suggest that you do this in two separate steps, with one regular expression capturing you the three-letter code and a second regular expression capturing the list of codons.*
2. *To get an indeterminate number of codons use the following syntax:*

   something... `=~ /(`*something*`)/g`