**18.417 - Introduction to Computational Molecular Biology** **PS 1**
**Bonnie Berger and Manolis Kamvysselis** September 6, 2001

# Problem Set 1
## Due Date: Thursday, September 20

In this and subsequent problem sets, you will need to run some commands on Athena from the course locker, which you can get access to with the command

`athena% add 18.417`

If you don't want to type this every time you work on 18.417, put the line

`add 18.417`

followed by a newline in your `~/.environment` file.

1. Suppose we sequenced a small bacterium and found the following really short gene: `ATGGCAAGAAGCGCAACAACGGCGTGTAAGAGTTAA`

   Hand-translate the sequence, assuming the bacterium uses the same genetic code as we do (*Hint:* "Why can't we all just ... *get along?*")

   There are 64 codons ($4^3$) out of which 61 encode amino acids. However, only 20 amino acids exist. Many amino acids therefore can be encoded in more than one ways. Write the most divergent sequence you can construct that translates to the same amino acid sequence, using the genetic code in reverse. What percent identity do you find at the nucleotide level? At the amino acid level?

   How many possible sequences translate to the same protein? R.A.Bowen has written a reverse-translator that you may find helpful for this. You can use it through the web at:

   `http://arbl.cvmbs.colostate.edu/molkit/rtranslate/`

   (Optional:) Why does the last nucleotide in a codon triplet have very little influence on the amino acid residue the codon translates to?

2. Score the alignment

   $$\begin{array}{l} GA{-}CGGATTAG \\ GATCGGAATAG \end{array}$$

   Give a match a score of $+1$, a mismatch 0, and a gap $-1$.

3. Find the optimal global alignment (and the resulting score) between GAGC and CCG, with the scoring system used in class (i.e., $+1$ for a match, $-1$ for a mismatch, and $-2$ for a gap).

4. Find the best local alignment between ATACTCTCCTAAG and GACTCG-TAACGTAT, with the scoring system used in class. Also turn in the entire scoring matrix which the dynamic programming algorithm would compute (writing this out should take less than 15 minutes). If you need to break ties between local alignments with the same score, choose the longer sub-sequences.

5. **Getting genomes:**(Optional) This is just an example to show you how to get access to the genomes yourself, if you want to. Go to the webpage `http://www.ncbi.nlm.nih.gov/Entrez/` using netscape or some other browser. Click on the "Genome" link. In the search field at the top of the page that takes you to, enter "GAL4" and hit return. This will take to a page with a list of matches. Click on the second match, and then click on the link `NC_001148`.

6. **Playing with genomes:** Once you've `add`'ed the `18.417` locker, run the command `idle -e /mit/18.417/share/problem_code/ps1.py &`. This will bring up a python file that you can run by holding down the control key and pressing the F5 key. It computes how many of each of the nucleotides used in the genes in the first chromosome of Baker's Yeast, S. Cerevisiae. It uses the functions in `/mit/18.417/share/problem_code/Sequence.py`.

   If you know the name of a gene on this chromosome, you look at it with commands like these:

```
print Chromosome.genes['FUN26'].gene
print Chromosome.genes['FUN26'].protein
print Chromosome.genes['FUN26'].CDS().data
print Chromosome.genes['FUN26'].CDS().reverse_complement().data
print Sequence.translate(Chromosome.genes['FUN26'].CDS())
```

7. Write a script similar to the ps1.py script that counts how often each codon is used in the genes on the first chromosome. (*Hint:* For each coding sequence, instead of counting individual nucleotides, modify your code to count triplets of nucleotides (codons). Now group the 64 possible codons by the amino acid they translate to. What do you notice? Is S.cerevisiae indifferent to which codon it uses to encode each amino acid? Extra points for noticing a correlation between the codon preferences you discover and the frequency of G's and C's that you computed in the last question. :)

8. **Alignment implementations:** Have a look at the program in `/mit/18.417/share/problem_code/alignment.py`. The code in `/mit/18.417/share/problem_code/tests/alignment.py` shows how to

use it. Currently this gives the best *global* alignment between two sequences. Modify it so that it gives the best *local* aligment.

(Optional) Genes which are descended from a common ancestral gene are called *orthologous.* Find two genes that probably come from common ancestor (e.g., a human and a mouse gene that generate proteins with identical functions) and use your program to align their coding sequences. Within the aligned codon triplets, which of the three nucleotides is the least likely to match? If you didn't already know the frame in which the sequences are translated, how might you guess it from the alignments?