Project 2: Annotating Viral Genomes

Due Date: 12:01am Saturday March 30th, 2013. + 8 hour grace period.

Project Goals:

The purpose of this project is to introduce you to the way biological systems organize data within genome sequences, and to some of the tools developed to decipher this organization. There is no universal organization scheme that all organisms share, so we will focus on something specific: finding and comparing genes within viral genomes. Again, there are two projects defined in this document: an applications project and an implementation project. Pick exactly one. Again, credit for each project is 50% project and 50% report. Report descriptions for this project will be released later.

Note: If you are registered for CSE308/408, please do the implemention project. If you are registered for BioE 308/408, please do the applications project.

Background:

Genomes are complex collections of information that control the construction and regulation of genes. Not only do genomes encode proteins, they also interact with the feedback loops that monitor and regulate the expression of proteins. These are the mechanisms involved in the "decision processes" that drive bones to grow when they are fractured but not otherwise, or make tumors attract blood vessels in order to more rapidly metastasize.

Biologists and medical researchers, who seek to understand how a biological system functions in health and malfunctions in disease, are thus committed to decoding the patterns of external inputs, to discovering how the inputs cause reactions in biological systems, and unraveling how the reactions change gene expression to respond to the inputs. This difficult process is aided immensely by computational tools that help sift through enormous genomes to find hints – fleeting similarities in nucleotide sequences, or a marker suggesting the beginning or end of a gene - that suggest something about genes and how they are expressed.

But based on our current assembly technologies, we know that genomes can contain multi-megabase contigs. These sequences of nucleotides have no markers, no signposts, no guides to suggest where the genes are encoded, and what other sequence regions effect expression. Herein lies the fundamental purpose of Genome Annotation: assign markers that guess at the meaning of the invisible subsequences within the genome: the locations of genes, the position of promoter and repressor binding regions, and the regions that do not code for genes. We will only have time to scratch the surface of this complexity.

Despite the immense scale of our current genome sequencing technologies, we have sequenced less than 1% of known organisms, which is in itself a tiny fraction of an immense cosmos of unknown microorganisms. Among the genomes that we have sequenced, we have already observed immense variation. There are no global rules that dictate the start of a gene, for example. But among some groups, similarities nevertheless exist, and they can be exploited to at least find clues to the way organisms actually function on the molecular level.

Applications project

You are still trapped on a space ship with famed (though still ailing) biologist J. Craig Venter. Fortunately, with your understanding of sequence assembly, and Venter's generous tissue sample, you have managed to assemble the nonhuman reads from into a coherent set of contigs. In project 1, MegaBlast suggested that the space virus is related to a virus that has been sequenced before.

The space virus has continued to spread, however, and you fear that by the time your damaged ship limps into communications range with another vessel, you will all have left this mortal coil. Based on a vote among the surviving passengers, you decide to press on, annotating the nonhuman contigs you have isolated, in the hopes of identifying a recognizable gene that can be targeted by an inhibiting drug[1].

Your Task: (some of these questions depend on other parts; put your answers in your report.)

1) Using Glimmer and Genemark, predict the presence or absence of genes in, or shared among, the provided contigs. Which contigs should be joined to get each whole gene? (40%)
2) Use megaBLAST to confirm your predictions, finding genes homologous to those you identified. What are the names of these genes? (40%)
3) Using ClustalW, align the closest BLAST homolog to your contig (20%)

Software tools at your disposal:

1) Glimmer and Genemark. These gene identification tools use Markov Models to predict the position of start and stop codons, which are markers for the beginnings and ends of genes. While they do not always agree, they have been shown to be fairly accurate (>98% on open reading frames).
   Glimmer is located here: http://www.ncbi.nlm.nih.gov/genomes/MICROBES/glimmer_3.cgi
   Genemark is located here: http://exon.biology.gatech.edu/heuristic_hmm2.cgi
2) MegaBLAST. While Glimmer and Genemark can isolate novel genes, learning what role these genes play in the viral lifecycle will require you to search for related genes. While in Project 1 you searched for similar genomes, megaBlast can also be used to search for related genes, and thus help you predict which genes exist in your space virus.
   MegaBLAST is located here:
   http://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastn&BLAST_PROGRAMS=megaBlast&PAGE_TYPE=BlastSearch&SHOW_DEFAULTS=on&LINK_LOC=blasthome
   Make sure to use the nr/nt database.
3) ClustalW. This alignment tool is critical for comparing predicted genes to known genes.
   ClustalW is located here: /proj/cse308.s13/Project2/ClustalW

Recommended Pacing (Take special note if you are also taking IPD):

[1] Note: this is not unlike the approach scientists would take after having isolated a new virus: They would sequence its genome and try to isolate genes that can be targeted for inhibitor (drug) design.

I recommend finishing tasks 1 and 2 before the week of the 2nd. Task 3 and the Report will take some time, but can be completed in the last week, since you have worked with ClustalW before.

Hints:

There are no genes on the complementary strand. Also, note that Glimmer and GeneMark can be wrong on the length of a gene, even though they have the general position right – one end or the other might be predicted wrong. Use the lengths of closely matching genes found with megaBLAST to adjust your estimate of the actual gene length.

Applications Project Report (3-5 pages, single spaced, 1 inch margins):

Section 1) Some regions within the space virus' genes may be similar to known genes. Explain, algorithmically, how ClustalW and its affine gap scoring approach can help you find these regions. (30%) Why are the alignments identified by ClustalW optimal? (10%)

Section 2) Glimmer and Genemark find genes even in genome sequences no one has ever seen before. Explain how, algorithmically, they achieve this goal, using Markov and Interpolated Markov Models, and how these models work, and differ from each other(40%)

Section 3) Glimmer and Genemark are extremely sensitive gene finding tools as long as they are trained on the proper genomes. What happens when you run Glimmer with incorrect training data, and why does this happen? (20%)

Note: explaining something algorithmically means that an intelligent person who understands computers should be able to implement the algorithm from your description. There should never be a point in your description where it is unclear what to do.

Implementation Project: Implement nucleotide sequence alignment.

Genome annotation, in the post-genomic era, is impossible without sequence alignment algorithms. These technologies enable putative genes, identified by the likes of Glimmer and Genemark, to be compared to known genes, and hopefully identified. Later, in Project 3, we will see how these algorithms also can reveal evolution in genes.

Sequence alignment is a way to arrange two or more nucleotide sequences to reveal sequence similarities of an evolutionary nature. The sequences are arranged in order (in other words, one nucleotide is never moved in front of another) by adding gap characters, represented with an underscore ("_") or a dash ("-"), to superpose similar sequence regions. For example:

Unaligned (original sequences before alignment):
ACTACGATCGA
ACTTGAAGATGGA
Aligned (Sequences after alignment, where gaps have been added):
ACT__ACGATCGA
ACTTGAAGATGGA

Note that sequence alignment does not strictly demand that nucleotides be identical in order to be aligned with each other. Instead, it seeks to align as many as possible. The way these alignment decisions are made is actually by considering all possible alignments and choosing the best scoring individual, based on substitution rules (e.g. how positively do you score for aligning similar nucleotides, how negatively do you score for mismatch?) and gap penalties (e.g. how negatively do you score for opening a new gap, or extending an existing gap?).

The cornerstone algorithm for the alignment of nucleotide and protein sequences is the Needleman-Wunsch and Smith-Waterman dynamic programming algorithms. These similar algorithms guarantee an optimal alignment of two sequences in O (n^2) time, while remaining flexible to different substitution rules and gap-scoring schemes. A superior description of this algorithm and related algorithms is in Jones and Pevzner, from Chapter 6.3 to 6.10. Here, the description of the Manhattan Tourist serves as an excellent analogy for the sequence alignment problem.

Implementation breakdown and grading:
There is one minor step and two major steps in sequence alignment
0) Parse fasta files and output alignments to file or stdout. (20% of project)
1) Construct an m by n matrix of values, to score alignments with an affine gap penalty. (40% of project)
2) traverse the matrix backwards, in order to determine the highest scoring alignment. (40% of project)

Testing:
Compare your results against ClustalW. Do not start with large datasets, but rather make up short sequences – like 10 nucleotides – and compare your alignment against the alignments that ClustalW generates. ClustalW will not necessarily be identical, but will give a good idea of what to expect.

A hint for Debugging, De-stressing:
Software development is hard because debugging is hard: you cannot know how long it takes to find a bug until you've found it. This is true from the lowliest missing semicolon error to the meanest distributed-memory pointer-arithmetic spaghetti. The only time guarantee you have on finding a bug is that finding a bug in a large amount of code is a much harder (i.e. exponentially harder) than finding a bug in a small amount of code, because of the limits of human memory. Hence, breaking your code into small pieces and creatively testing each piece is inevitably more efficient. Practice this. It can seem annoying, but it pays immense dividends. Programming is like mountain climbing: you stop routinely to clip in so accidental falls are short. And survivable. In courses, you are always programming something you have never programmed before: a mountain climber would never free climb a route they never climbed before – don't do that with your code, either.

Recommended Pacing:
Read Chapters 6.3 to 6.10 of Jones and Pevzner for a second description of sequence alignment. We will cover sequence alignment in detail in class. Take this weekend to copy your old fasta parsers from the previous assignment to a fresh assignment, and write an output function that prints out two sequences in tandem (like Clustalw). Next, try to finish the matrix construction part by the end of the week of the 19st, and then the backwards traversal part by the end of the week of the 26th. Lectures will reflect this recommended pace. The report and any bugs should fall into line by the last week.

Implementation Report

Section 1) If the sequence alignments you have computed are representative of a population of organisms, and not just any bunch of sequences, what does this tell us about the relationship between specific genome locations and reproductive fitness? (20%) What happens if the set of sequences is not representative, and just a set of sequences? (20%)

Section 2) Aligning genes is only a first step to understanding the how a genome influences a biological system. What other parts of a gene affect the larger system, through mechanisms like gene regulation (20%), and if you wanted to detect them, experimentally (10%) or computationally (10%) how would you? Explain your procedures in detail.

Section 3) Genes can be recognized by transcription factors in many ways. Virii, with their immense populations, could theoretically evolve the widest range of such recognition mechanisms. Yet many of the most common virii use similar mechanisms; similar enough, at least, that Glimmer and Genemark accurately identify their genes. Why? (20%)

Groups Rosters

	Roster	Contig Set for Applications Project
1	Bicher, Wallace	/proj/cse308.s13/Project2/contigData/set1.contigs
2	Guo, Chu	/proj/cse308.s13/Project2/contigData/set2.contigs
3	Dodd, Schardt	/proj/cse308.s13/Project2/contigData/set3.contigs
4	Gu, Romero	/proj/cse308.s13/Project2/contigData/set4.contigs
5	Feinstein, Wu	/proj/cse308.s13/Project2/contigData/set5.contigs

NOTE: This data is not identical to Project1 data.

Sample sequences for Implementation projects:

/proj/cse308.s13/Project2/sampleSequences/Homologs.fasta
/proj/cse308.s13/Project2/sampleSequences/HomologsMedium.fasta
/proj/cse308.s13/Project2/sampleSequences/HomologsShort.fasta

These sequences are all short, but some far shorter than others. Efficiency will not be a major problem with this project, but expect that your code will be tested on larger data. Always make sure your code works correctly on small data before you test on larger data.

Project submission:
Package your report and code (for implementation projects) or your report and aln file (for applications projects) into a .tar.gz and email it to chen@lehigh.edu. To download your file to your computer or laptop, on the Mac, use Fetch(http://www.lehigh.edu/mac/software.html), and on the PC, use SSH (http://www.lehigh.edu/helpdesk/ssh32/ssh32.html). Lehigh has site licenses for these file transfer clients, which will allow you to use them for free if you install the versions from these links. See previous emails for details on using these file transfer clients.

Extra Credit

Full extra credit (50%) is possible by successfully using the sequence alignment algorithm developed in the implementation project to compare putative genes predicted in the applications project with known genes. Total extra credit including collaboration cannot exceed 50%.

Up to 25% extra credit is possible by documenting your collaborative activities with your teammate and with other members of the class, depending on the degree and significance of your collaboration. Please be aware of syllabus guidelines on allowed and disallowed forms of collaboration.