COSC 494/594 – Bioinformatics Computing
Lecture #2

DNA sequencing has been around since the late 1970s (Sanger won his second Nobel Prize by inventing DNA sequencing, a method still used today), but a "big" genome was not completed until 1995 by The Institute for Genomic Research (TIGR) headed by Dr. Craig Venter. "Big" in biology context was a free-living organism and the overall size was 1.83 million nucleotides (see Table 1.1 in the text).

Behind this project was the TIGR assembler, developed by a CS PhD student at the University of Maryland in the early 1990s. The text does not talk about these software tools, but we will later in the semester. The same person who wrote the TIGR assembler was involved in writing the Celera Assembler that some of you are aware of, and he still works on genome assembly tools to this day.

The text, as will this course, looks at genome sequences and the steps (often heavily dependent on computing) required for deciphering the underlying biology. It takes a practical, or "hands-on", approach and we'll try and do the same throughout the semester.

First, though, lets consider how we store genomes on a fundamental level. At one point in time (not too long ago based on this instructor's memory working with TIGR in the late 90s) genome data was distributed on magnetic tape and later on CD. Now, most if not all data are available on the internet (e.g., GenBank in the U.S).

There are five general types of genomes we may consider this semester:

Prokaryotic - bacterial genome such as E. coli
Eukaryotic - mice, people, mosquitoes
Viral - SARS, AIDS, flu
Organelle - chloroplast, mitochondrion
Metagenome - environmental samples (ocean, volcanoes, soil)

Back then, though, hard drive space was expensive and only one strand of DNA was stored. By convention, this strand is listed 5' - > 3'.

Luckily, we can always determine the opposite strand of DNA within a computer program. For an illustration, consider the following sequence:

5' ATGCATGC 3'

The complementary DNA sequence based on what we learned in lecture # 1 would be:

3' TACGTACG 5'

In this class, we will stick with the 5' -> 3' convention, however, and produce the other strand of DNA using these well-tried steps:

1. Reverse the string (e.g., CGTACGTA)
2. For each character, replace A ⇔ T and G ⇔ C (e.g., GCATGCAT)

Note either step can be performed first. This is called the "reverse complement" in bioinformatics and although it is simple, it will be an important practical concern throughout the semester.

Modern sequencing projects are usually broken into two separate types:

1. Hierarchical – A large genome is broken into smaller physical "genomes", and each smaller "genome" is processed independently using computing.
2. Whole Genome Shotgun (WGS) – A large genome is broken into smaller chunks and processed using very large computers to do the work.

The rationale behind sequencing is driven by the repetitive nature of the genome. For example, the instructor has worked in a maize genome project completed using ~20K 150-200,000 nucleotide pieces and is working on mosquito genome projects that were done using WGS. In general, biologists are more confident in hierarchical approaches.

We will leave the dirty details of sequencing and assembly to later in the semester, once we have a firm grasp of sequence alignment algorithms. Today, we will begin to look at the next step: annotation. In simpler terms, what is the "good stuff" in a genome that biologists are initially interested in?

A large part of this effort is statistical methods, which we will begin to introduce today with simple probabilistic models of DNA sequences. Two quick notes/observations:

1.) Treating DNA as a 2D object (string/vector/sequence) is a very powerful abstraction. Most of what we will cover this semester treats DNA molecules as strings, and we will derive some nice mathematical/computational properties using this abstraction.

2.) DNA is not, however, a 2D object (see notes from lecture #1). Although we lose information representing a 3D object as a 2D object, for the most part this is not a major concern in practical bioinformatics. More throughout the semester.

State-of-the-art annotation of important features in genomes is also fundamentally based on sequence alignment (can't wait until next week, right?). But it is also based on statistical properties to a great extent, most of which are outside the scope of this course. We'll cover the simple ones today that will be a great foundation for the next few weeks.

First, we need to go over some terminology for later in the semester about strings/sequences.

Definition 1: a "DNA sequence" is a finite string from the alphabet $\sum = \{A,C,G,T\}$.

Definition 2: a "genome" is the set of all DNA sequences in a cell

Definition 3: Elements of a sequence s are denoted as $s = s_1 s_2 \ldots s_n$. Each individual nucleotide is represented by s_i, where i is the position of the nucleotide in the string (1 based in this example).

Definition 4: If we would like an interval from i to j, $K = [i, j]$, this substring is denoted as $s(i : j)$

Definition 5: Individual nucleotides can be represented as $s(i) = s_i$.

Example 1.2 from the text

Given a DNA sequence s = ATATGTCGTGCA

$s(3 : 6)$ = ATGT and $s(8) = s_8$ = G

Other alphabets exist throughout the course, for example RNA (replace T with U in the DNA alphabet) and protein (20 characters, see page 7).

There are generally two simple models used to model DNA sequences. Like most models, they are not perfect but are often good enough, esp. for purposes of this course. For example, DNA is often called "pseudorandom" in that there are seldom long-range correlations in most large DNA sequences but various biological processes produce sequences that are not truly random. Today we will discuss these models at some depth.

The simplest model of sequence data is the *multinomial model*, which assumes that DNA sequences are random and therefore nucleotides are independent and identically distributed ("i.i.d" assumption). The core of this model is determining the probability of seeing a given character from our alphabet by simply counting. We will call this p, where for DNA:

$p_{DNA} = (p_A + p_C + p_G + p_T)$ and $p_A + p_C + p_G + p_T = 1$

The simplest case of course is where $p_A + p_C + p_G + p_T = 0.25$; however, this very rarely occurs in nature for a variety of reasons.

In general the probability of an arbitrary character x is, $p_x = p(s(i) == x)$

To calculate the probability of a query sequence Q under the multinomial model, where the length of Q is n (or $n = |Q|$ by convention) is simply:

$$P_q(Q) = \prod_{i=1}^{n} P(Q(i))$$

Note this is equivalent to multiplying all of the probabilities together. This is either called the probability of a given sequence (P) or its likelihood (L). Further, remember that:

$$\log\left(P_q(Q)\right) = \log\left(\prod_{i=1}^{n} P(Q(i))\right) = \sum_{i=1}^{n} \log(P(Q(i))$$

This is often easier to compute than the very small number obtained by multiplying the individual probabilities.

The multinomial model, however, does not take into consideration local correlations. *Markov sequence models* utilize Markov chains, in which the probability of observing a symbol depends on the symbols preceding it in the sequence. An order 1 model depends on the preceding nucleotide, order 2 preceding two nucleotides, etc. A multinomial model can be viewed as a Markov chain of order 0 (no dependence).

We begin by discussing the basic properties of a Markov chain. This model is simply a set of states (here the letters of the alphabet) and transition probabilities between states (see Fig 1.1 for an example in the text). We represent the transition probabilities as a matrix T.

Markov models can be used to generate sequences. In fact, a bacterial-like random sequence generated from a higher order model trained from a given genome (order 5 or 6) is nearly statistically indistinguishable from the real genome! To perform such generation we utilize additional probabilities denoted by π for the initial state to enter. More on this in the homework assignment.

More formally, we can represent transition probabilities for an order 1 model as:

$$P_{xy} = p(s_i+1 == y \mid s_i == x)$$

Therefore the probability for a sequence s can be written as is:

$$P(s) = \mathrm{P}(S_n|S_{n-1})\mathrm{P}(s_{n-1}|s_{n-2}) \ldots \mathrm{P}(s_2|s_1)\pi(s_1)$$

or

$$P(s) = \pi(s_1)\prod_{i=2}^{n} p(s_i \mid s_{i-1})$$

There are a variety of simple calculations we can perform based on these statistics. Two mentioned in the text are base composition, which is the relative frequency of bases in a

given sequence, and GC content, which is the sum of p_G and p_C. Because all of the probabilities must add to 1 (see previous discussion) AT content is 1 – GC.

Why does this matter? Well, some bacteria genomes harbor "alien" DNA from another species that may significantly differ in these simple statistics. Also, from a practical perspective, modern molecular biology is based on bacteria making lots and lots of copies of specific pieces of DNA (say human). When biologists obtain a sequence from an experiment, it is important to know if it is human or bacterial. This is still important given past claims of bacterial gene transfer into the human genome (see Venter et al., 2001).

The text has a clearer example of an ancient virus inserting into *H. influenzae* on the top of page 13.

k $P_q(Q)$-mer frequencies and motif recognition are also briefly introduced in this chapter. We may discuss them later in the course but for now look at example 1.5 in the text. Significant patterns are found using an *odds ratio*, but that will be discussed during the gene finding lecture.

Begin to familiarize yourself with GenBank (www.ncbi.nlm.nih.gov). It will be used for your first homework assignment.