

Lecture #7: Genome Alignment

We will begin the comparative genomics unit by discussing genome alignment, the building block of comparative genomics. We motivate genome alignment by discussing several applications and begin our discussion of algorithms for genome alignment. In the next lecture we will conclude our discussion of algorithmic details and add pointers to useful software in this area.

1 Genome Alignment

1.1 Motivation

Genome alignment is the heart of comparative genomics. Just as a high quality reference *assembly* of a single genome allows different investigators to compare their results against the same coordinate system, a high quality reference *alignment* between the genomes of two or more different organisms allows us to set up a consistent set of coordinates for comparing multiple organisms. Not only does this allow us to transfer experimental annotations of one genomic region to another region in a different species¹, it also enables the use of bioinformatics to identify functional and constrained elements at many different levels of resolution, from single base pair alterations in promoters to syntenic blocks of conserved gene sequence and order to multi-megabase pair inversions, transpositions, insertions, and deletions.

Here is a short list of some of the major applications of genome alignment, illustrated by seminal papers:

1. Syntenic blocks btwn human & mouse. See Figure 3:
<http://www.nature.com/nature/journal/v420/n6915/full/nature01262.html>
2. Ultraconserved elements (Bejerano et al. 2004). See Figure 1:
<http://www.sciencemag.org/cgi/content/full/304/5675/1321>
3. Motif identification (Kellis et al. 2004). See Figures 1 and 6:
<http://www.nature.com/nature/journal/v423/n6937/full/nature01644.html>
4. Genomic evolutionary rate profiling (Cooper et al. 2005). See Figures 1,5:
<http://www.genome.org/cgi/content/full/15/7/901>
5. ENCODE project: <http://www.genome.gov/10005107#1>

The National Human Genome Research Institute (NHGRI) launched a public research consortium named ENCODE, the Encyclopedia Of DNA Elements, in September 2003, to carry out a project to identify all functional elements in the human genome sequence. The project is being conducted in three

¹Keeping in mind, of course, that such annotation transfer must be done cautiously as it's not 100% correct in many circumstances.

phases: a pilot project phase, a technology development phase and a planned production phase.

ENCODE Project Target Selection Process and Target Regions

For use in the ENCODE pilot project, defined regions of the human genome - corresponding to 30Mb, roughly 1 percent of the total human genome - have been selected. These regions serve as the foundation on which to test and evaluate the effectiveness and efficiency of a diverse set of methods and technologies for finding various functional elements in human DNA.

Prior to embarking upon the target selection, it was decided that 50 percent of the 30Mb of sequence would be selected manually while the remaining sequence would be selected randomly. The two main criteria for manually selected regions were: 1) the presence of well-studied genes or other known sequence elements, and 2) the existence of a substantial amount of comparative sequence data. A total of 14.82Mb of sequence was manually selected using this approach, consisting of 14 targets that range in size from 500kb to 2Mb.

The remaining 50 percent of the 30Mb of sequence were composed of thirty, 500kb regions selected according to a stratified random-sampling strategy based on gene density and **level of non-exonic conservation**. The decision to use these particular criteria was made in order to ensure a good sampling of genomic regions varying widely in their content of genes and other functional elements. The human genome was divided into three parts - top 20 percent, middle 30 percent, and bottom 50 percent - along each of two axes: 1) gene density and 2) level of non-exonic conservation with respect to the orthologous mouse genomic sequence (see below), for a total of nine strata. From each stratum, three random regions were chosen for the pilot project. For those strata underrepresented by the manual picks, a fourth region was chosen, resulting in a total of 30 regions. For all strata, a “backup” region was designated for use in the event of unforeseen technical problems.

It is no exaggeration to say that the annotation of the human genome – the determination of the functional role of every single base pair – is dependent upon constructing reference alignments between humans and other organisms.

1.2 Biological Issues

1. *Scale*: genomic sequences range from 10^6 – 10^{12} base pairs in length.
2. *Evolutionary distances*: sequences which are very close together (such as different strains of the same microbe, e.g. *E. coli* K12 vs. *E. coli* O157) present a somewhat different problem from aligning sequences which are quite different (such as mouse and human).
3. *Repeats and Paralogs*: both within and between genomes you will find many regions which are all highly similar to each other, at all kinds of different scales from small

repeats to full proteins to huge repetitive elements. There are many reasons for this; here is a noncomprehensive list:

- (a) the way DNA polymerase works (can lose track of where it is while polymerizing)
 - (b) some organisms, such as yeast, have had an ancestral fusion of chromosomes which resulted in massive gene duplication.
 - (c) transposable elements have their own evolutionary niche (a variant of Dawkins' Selfish Gene)
 - (d) it's often harder to develop a new protein than to reuse/modify an old one (Ohta's duplication hypothesis)
 - (e) often multiple copies of a protein under different kinds of regulatory control are needed (e.g. backup ribosomes for intense periods of protein synthesis).
4. *Shufflings and Inversions*: In addition to the issues leading to sequence duplication, sequences can also be reordered, either through shuffling or inversion. Shuffling occurs when (say) a sequence of similar elements (A, B, C) in organism 1 has the order (C, B, A) in organism 2. Inversion occurs when a genomic region is cut out and flipped around, such that the strand orientation has been flipped. These occur when (for example) two double stranded breaks in the chromosome happen near each other, and the repair mechanism mistakenly reintegrates the ends.

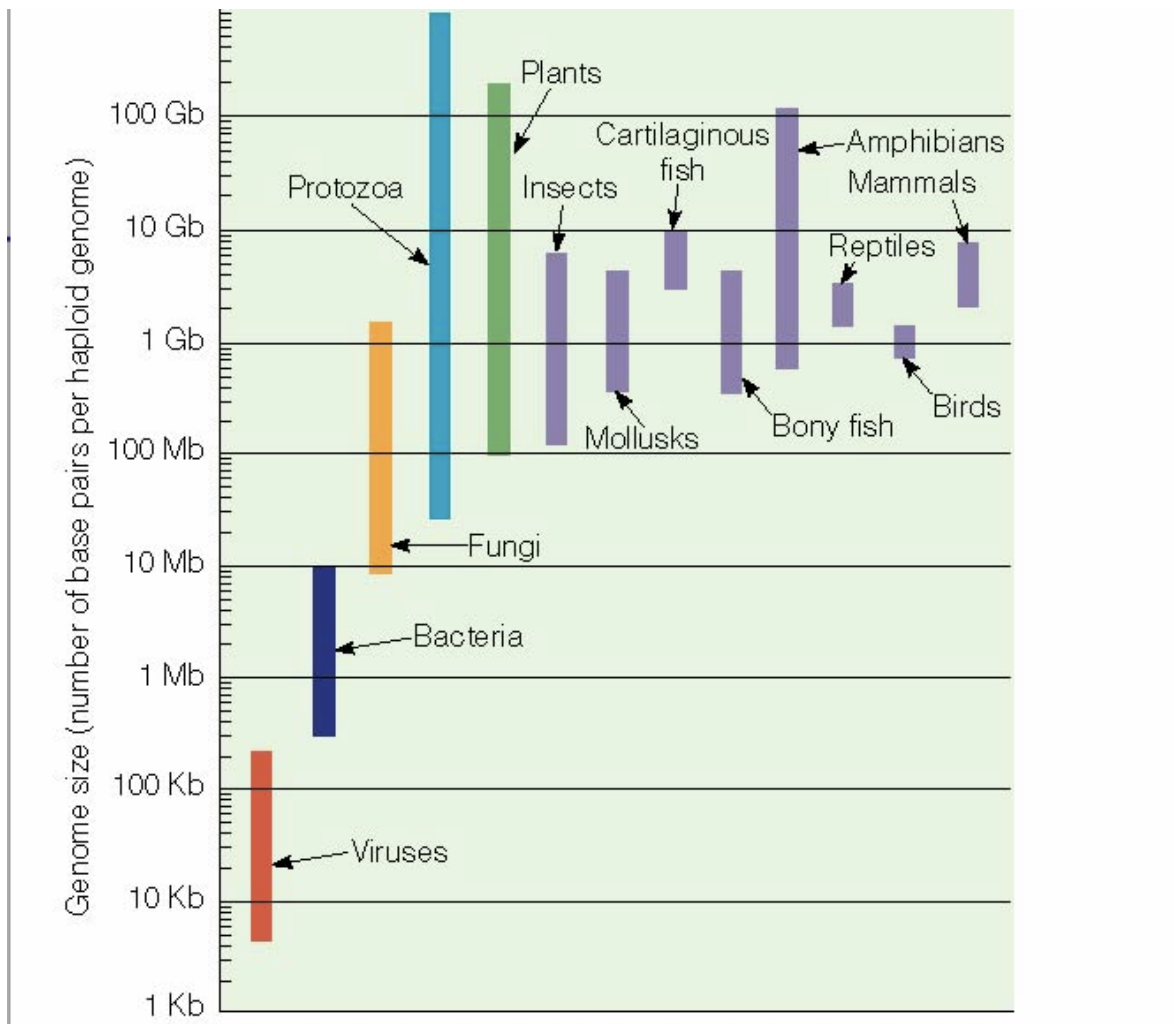


Figure 1 | Genome Size distributions of organisms within taxonomic groups. Note that our impression of overall organismal complexity only weakly correlates with genome size. This is partially resolved by consideration of non-repetitive elements in the genome.

As genome sequencing proceeds ever more rapidly, these issues become more rather than less relevant. For example, aligning amphibian genomes on the order of 10^{12} base pairs to mammalian genomes may require new heuristics to deal with the likely extreme sparsity of similar regions.

1.3 Mathematical Basics

Here we discuss the first stage of genome alignment, the process of seeding and local alignment. In the next lecture we will cover algorithms for chaining these local alignments together to obtain a global alignment of two very large sequences. For organisms with multiple chromosomes, we calculate this global alignment for all pairs of chromosomes and then color code the results as in Figure 5.

1.3.1 Overview

Given two sequences, say human chromosome 2 and mouse chromosome 1, the first step is to find short matches of extremely high identity between the sequences. These seeds will be used to build up local alignments: larger, less exact matches, called anchors. These anchors are then chained together to get a global genome alignment.

Genomic regions of interest contain islands of similarity, such as genes

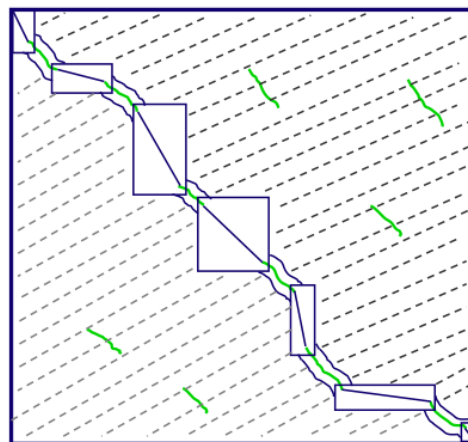
1. Find local alignments
2. Chain an optimal subset of them
3. Refine/complete the alignment



Systems that use this idea to various degrees:

MUMmer, GLASS, DIALIGN, CHAOS, AVID, LAGAN, TBA, & others

Figure 2 | Genome alignment overview, verbally.



1. Find local alignments
2. Chain $-O(N \log N)$ L.I.S.
3. Restricted DP

Figure 3 | Genome alignment overview, visually.

1.3.2 Seed Generation

Seed generation² is the location of short, exact or nearly exact matches between two sequences of lengths N and M . This can be accomplished quickly by indexing one of the two sequences in an appropriate data structure, such as a lookup table or suffix tree. Seeds help to reduce the search area of the local alignment algorithm to just the regions that are likely to be similar, rather than attempting the full-bore dynamic alignment of both sequences, which takes $O(NM)$ time and is intractable for long genomic sequences.

²This discussion is adapted from Brudno and Dubchak's chapter in the Handbook of Computational Molecular Biology.

Once generated, nearby seeds are joined together; the presence of several seeds in close proximity is stronger evidence of homology than a single seed. Individual seeds are then extended to find alignments which have slightly less than exact matching, and which may incorporate gaps, but are still highly conserved. This is essentially the same set of heuristics that we discussed earlier, which BLAST uses to search a database for homologs to a query protein. Note that often these regions of homology will in fact be protein-coding sequences (albeit untranslated).

There are several ways to generate seeds:

1. *Lookup table*: The simplest is a lookup table. For the first sequence, all k -mers of one sequence (the database) are indexed in a table together with their locations. Then the k -mers of the second sequence (the query) are used to retrieve the locations at which the particular k -mer of the query is present in the database sequence. Thus for each k -mer we have a list of pairs of coordinates, where the first coordinate corresponds to the start position of the k -mer on the database sequence and the second to the start position on the query sequence.
2. *Degenerate Seeds*: An obvious extension is to use degenerate seeds, which permit some degree of mismatch between sequences. This approach gains sensitivity but results in a slower or memory intensive seed matching process which loses many of the benefits of a hash.
3. *Spaced Seeds*: A more sophisticated approach is the use of spaced seeds, initially implemented in the PatternHunter program. For example, a 110101 seed requires 4 (1st, 2nd, 4th, and 6th) out of 6 positions to match. The other two positions may not match. This pattern is referred to as a (4,6) spaced seed. The point of spaced seeds is to reduce the correlation between adjacent words in a sequence, as two adjacent k -mers no longer share $k - 1$ positions.

For exact matching seeds of length k it is sufficient to introduce a mutation every $k - 1$ positions in order to prevent the algorithm from finding a seed. However, for the (9,15) spaced seed 111001010011011 it is necessary to have a mutation every 7 base pairs in order to prevent a single seed from being found. Thus, since they can tolerate a more frequent mutational rate³ while still finding regions of homology, spaced seeds are better for evolutionarily distant organisms. Exact seeds work well for very close sequences (such as human and chimp, or two strains of bacteria).

1.3.3 Seed Match Visualization

We can visualize such seed matches with a dotplot:

³Note that there are also known mutational hotspots in genomes. I am not currently aware of a seeding algorithm which uses a biological model of mutation to optimize seed selection, though it seems that one could make progress in this area.

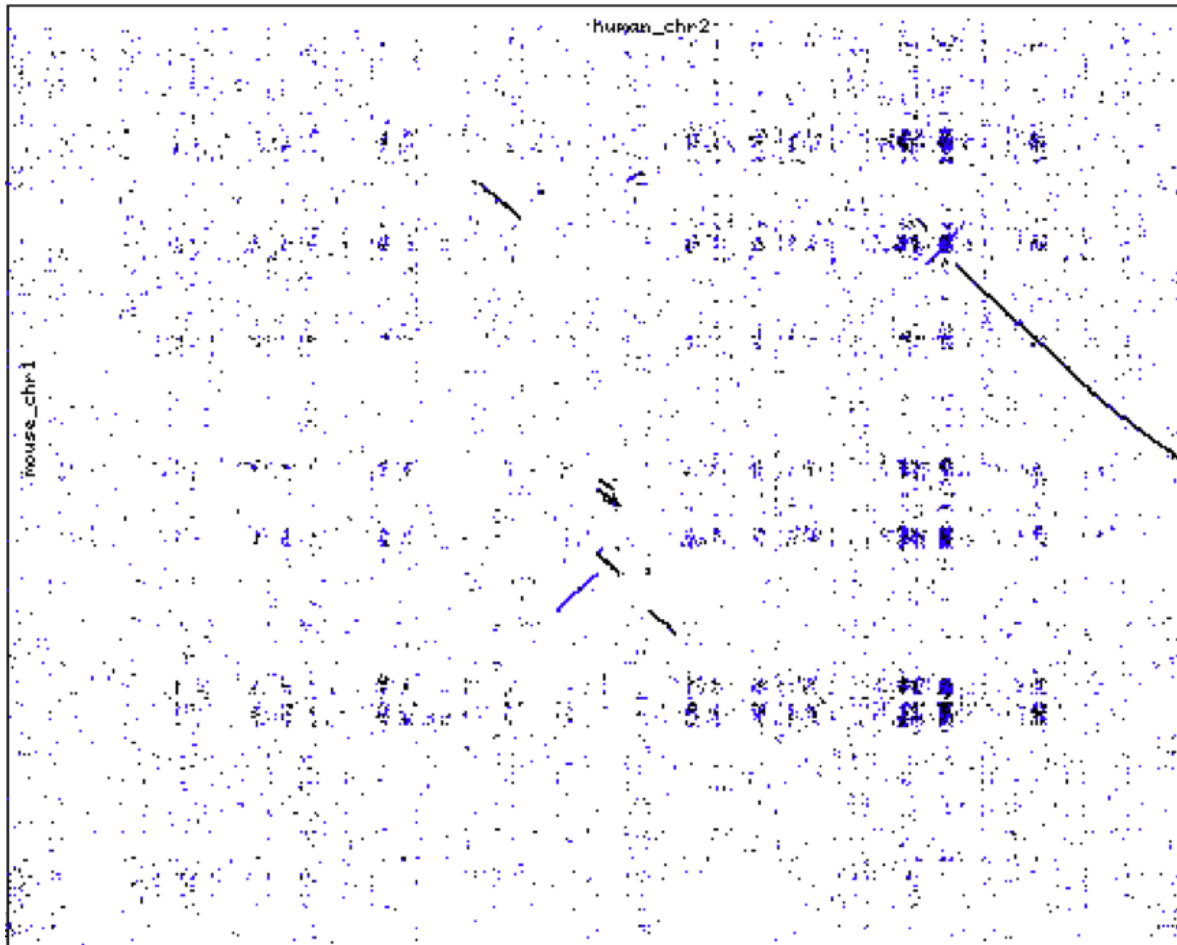


Figure 4 | Dotplot comparing regions of similarity between mouse and human. Human chromosome 2 is about 245 MB and mouse chromosome 1 is about 183 MB. Each dot represents a seed match between two sequences, in which BLAST matched a k-mer and extended it to a significantly high scoring alignment. Black dots correspond to forward alignments. Blue dots correspond to alignments between one chromosome and the reverse complement of the other. One can think of this as the calculation and superposition of two different dot plot matrices, the forward matrix and the reverse matrix.

If we look at this from the perspective of a banded chromosome diagram (an ideogram), we have:

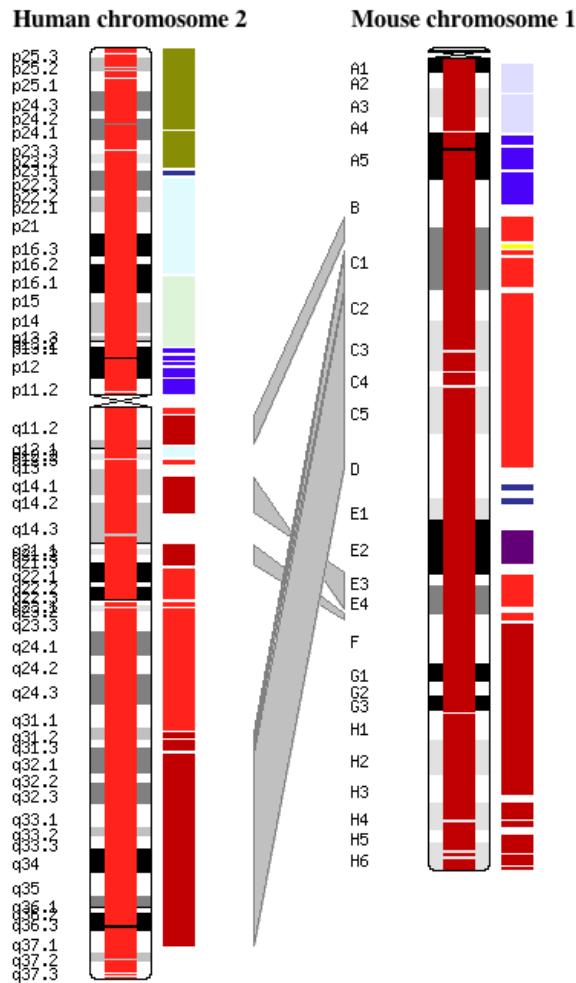


Figure 5 | Visualization of the dotplot at the chromosome level. Note that the long black region of the dotplot corresponds to the large forward alignment between the two sequences, roughly at the end of human chromosome 2 and the middle of mouse chromosome 1. Note that these are the same regions in human chromosome 2 which have been colored a dark red color in the second column on the left hand side; these are the regions which align to column 1 on the right hand side, which is colored dark red throughout. Similarly, the light red coloring of human chromosome 1 in column 1 on the left correspond to the light red regions in column 2 on the right hand side. Essentially we can color code each region according to which part of the other organism it is homologous to. This is similar to Figure 3 of the mouse genome paper: <http://www.nature.com/nature/journal/v420/n6915/full/nature01262.html>

You can see this online at:

http://www.sanger.ac.uk/Projects/M_musculus/publications/fpcmap-2002/syndata/hm.2.1.html

1.3.4 Extending Seeds to Build Local Alignments

Given a seed match, we use a two stage process (ungapped extension and then banded/gapped alignment) to extend that seed in either direction. This is shown in Figure 6; this should seem familiar as it is the BLAST heuristic which we discussed earlier:

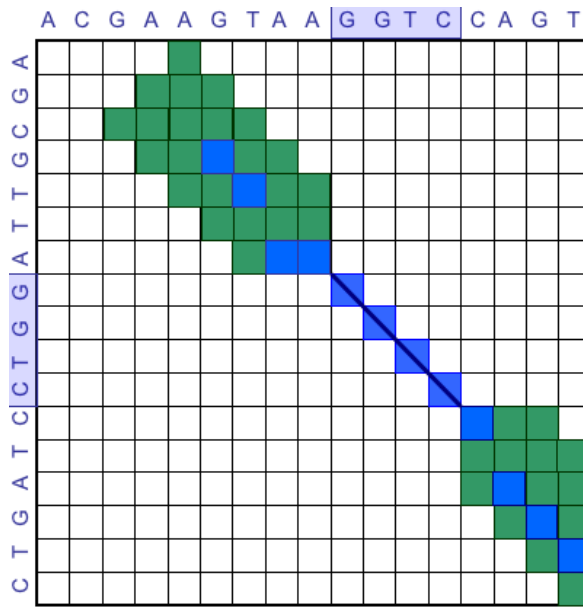


Figure 6 | Given a seed match (blue), we first extend it on either side with an ungapped alignment. If the score is above a certain threshold, we extend further with a banded gapped alignment, in which we limit the number of possible gaps. The result is a local alignment between two sequences obtained from seed matching.

In the next lecture we will discuss algorithms for chaining these local alignments together to build a global genome alignment. We will also discuss open issues in genome alignment and discuss software for this purpose.