

CS262 Computational Genomics

Lecture 10: DNA Sequencing and Fragment Assembly

Lecturer: Serafim Batzoglou

Scriber: Sina Firouzabadi

February 11, 2009

→ Sequencing

The main goal of DNA sequencing is to discover the nucleotide sequence of a given DNA. Due to its wide research implications on disease and biological discovery in general, there have been both government-funded and private efforts ([Celera Genomics](#)) to push forward the sequencing of the human genome.

Before we start let's go over some terminology quickly

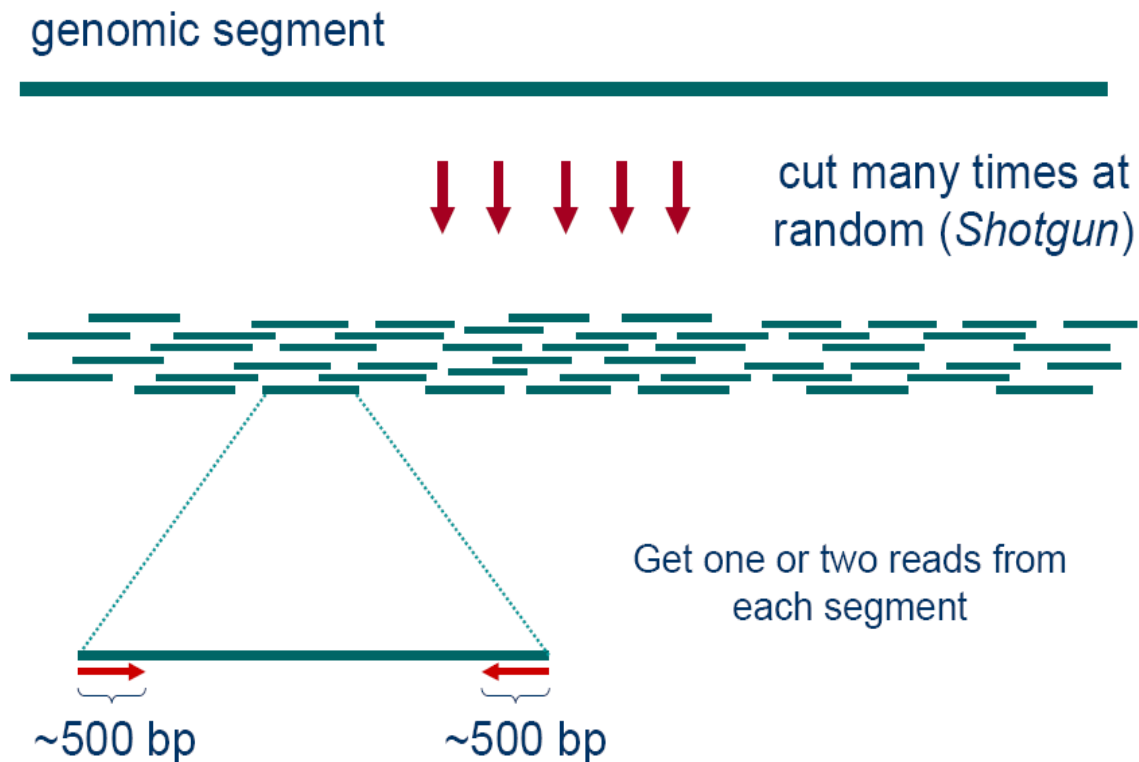
- **insert**: A fragment that was incorporated in a circular genome (such as plasmid), and can be copied (cloned) an arbitrary number of times.
- **BAC**: Bacterial Artificial Chromosome, a type of vector that incorporates inserts of length 100,000 to 250,000.
- **read**: A sequence that comes out of a sequencing machine. For a *Sanger* machine, in the year 2000 reads were 500 long and today they are 1000 long. Cost has decreased much faster. Other techniques are coming up. For instance, A company named *454 Life Sciences* now can produce reads of around 400 long and *Solexa* Machine gives something around 125 read lengths.
- **coverage**: the average number of reads that cover a position in the target DNA piece. Defining :

L = length of genomic segment
N = number of minimal tilingads
l = length of each read
C (coverage) = n l / L

As an example If we have sequenced a genome of size 3 billion letters. We have sequenced randomly positioned 60 million reads of length 1000. The coverage is 20 times, though you typically want a 10X coverage. It is provable that for coverage C the expected value of the proportion which is not covered $\exp[-C]$.

● Methods to sequence long regions :

The procedure is to take the entire genome and break it up into smaller pieces. Choose pieces at random and sequence the two ends of each piece. Now we have a puzzle with pieces that look as shown in the following figure. From this we want to construct the original genome. Note that we know the relative orientation of the reads with respect to each other, however, we do not know which strand they have been derived from. The genome is cut at random by shaking.



Now we have millions of pieces to put together—pieces consisting of a pair of reads whose distance we know within $\pm 20\%$ standard deviation and whose sequence we know within 1% error rate. What we want to end up with is a sequence for the entire genome, so we need to reconstruct the genome from these fragments by putting them together, in order.

Typically used procedure in the genome is the following :

1. Find all pairs of reads that overlap. Linked reads are called "mate pairs"
2. Choose some subset of those overlaps and merge the "good" pairs of reads into longer contigs (those from similar places in the genome)
3. Link these contigs to form supercontigs (You can put supercontigs together with mapping information to form ultracontigs)
4. Derive consensus sequence from supercontigs which are essentially sequence derived from multiple alignments of the reads composing the contigs.

Note: when reading from the genome, we don't know if it's from the forward or reverse complement of the DNA—so we treat every read as two equivalent words: both the forward and the reverse complement.

Sequence assembly is difficult due to repeats, which are significant in number for higher organisms. This is because higher organisms have more energy resources to spend in DNA replication without feeling the cost, and therefore incentive to keep their genomes concise is low. 50% of the human DNA is composed of repeats. Repeats may cause two far-away regions to be concatenated together, skipping the region in between during assembly.

Types of repeats :

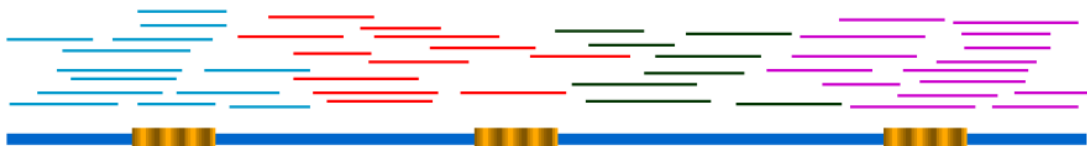
There are many different types of repeats -- usually, a repeat is simply DNA that has learned to use the cellular machinery to copy (and transpose) itself. Bacterial genomes has 5% repeats, where as mammals typically have 50% repeats.

- **Low-Complexity DNA** - (e.g. ATATATATACATA...) Repeat short words like ACA and ATA. They happen especially in oligomers and kilobases.
- **Microsatellite repeats** - $(a_1...a_k)_N$ where $k \sim 3-6$ (e.g. CAGCAGTAGCAGCACCAG) More copies of the same word. It is used for studying human population. This gives clues about genetic ancestry.
- **Transposons** – This is thought of as selfish DNA. They move by retro-transposition. The ones that replicate faster will survive.
- **SINE** (Short Interspersed Nuclear Elements) e.g., ALU: ~300-long, 106 copies
- **LINE** (Long Interspersed Nuclear Elements) ~4000-long, 200,000 copies
- **LTR retrotransposons** (Long Terminal Repeats (~700 bp) at each end) cousins of HIV
- **Gene Families** genes duplicate & then diverge (paralogs). This can give rise to pseudo genes.
- **Recent duplications** ~100,000-long, very similar copies that are entirely duplicated to a different region.

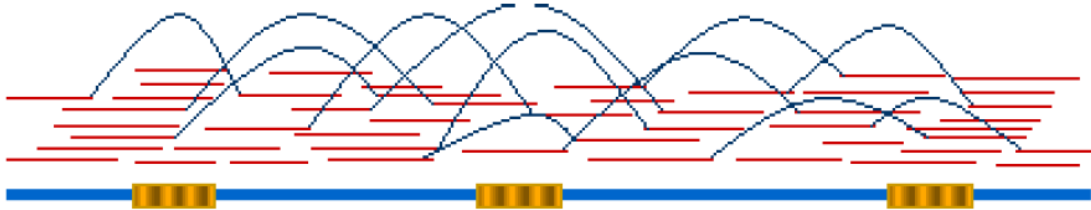
A repeat will be passed onto future generations. Any mutation could either cause it to become more fit or less fit in copying itself and the repeat will therefore “learn” to improve the replication of itself via evolution. An example of this is the ALU repeat family (specific to primates). Humans have about 1 million copies of the ALU repeat, which is 300 letters long. Some members of the ALU repeat are alive, while other members are dead. Entire family histories of the ALU repeat can be constructed.

To mitigate this repeat problem we have at least two options :

- ✗ Cluster the reads: If we can cluster reads to given regions we could have unique repeats in each region to allow us to distinguish between reads from different instances of a repeat



- x Link the reads: Use forward, reverse mate pairs from sequencing. Though reads are quite small, the inserts are much longer, hence, we can link reads which are quite far away.



→ Strategies for Whole-Genome Sequencing

Hierarchical Clone-by-Clone (a.k.a Clustering approach)

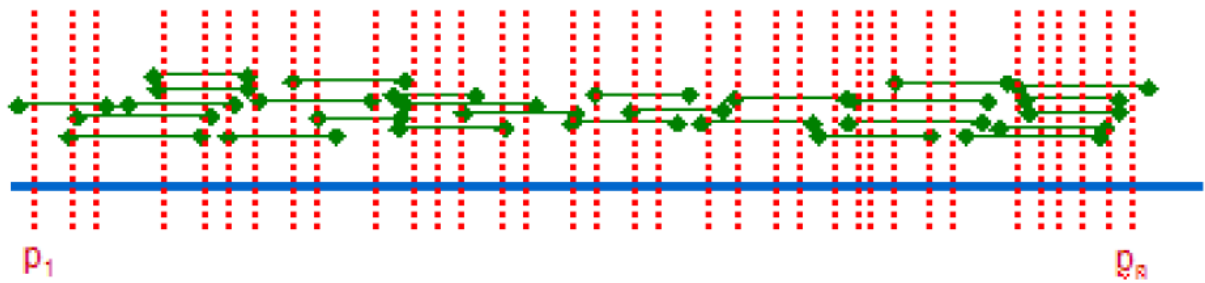
1. Break genome into many intermediate length pieces (150 bp or so)
2. Take each piece and clone into BAC's and create a library of BAC clones. The BAC clones should cover genome to a redundancy of 20-40x (in other words: the total length of clones is approx 20-40x the length of the original genome).
3. Map BACS into the original genome to get a physical map -- so that we know which one overlaps with another. There are two ways to do this: hybridization and digestion – discussed in detail below.
4. Select a minimum tiling path based on the physical mapping. This path is a minimum set of clones that cover the genome with smallest overlap between every successive clone. We want the tiling path to be minimum because we have to perform shotgun sequencing for each clone – expensive.
5. Put the shotgun derived sequences together based on the order established from the physical mapping

Physical mapping methods

1. Hybridization

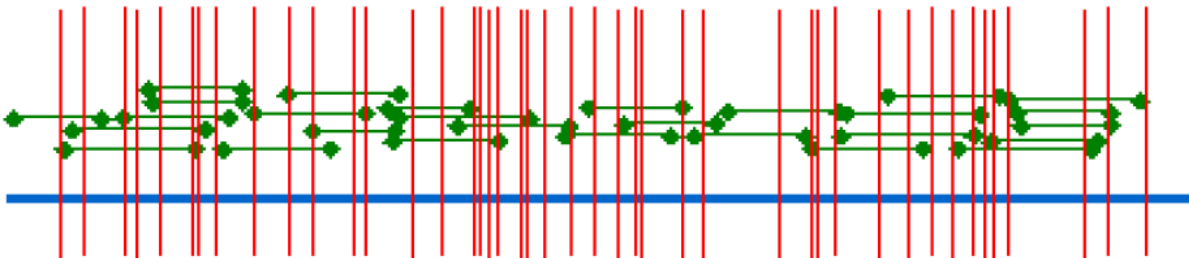
Short words, the *probes*, attach to complementary words

- Construct many probes
- Treat each BAC with all probes
- Record which ones attach to it
- Same words attaching to BACS X, Y □ overlap



2. Digestion

Cut the clones with restriction enzymes (sequence-specific scissors). The clones will all be cut in specific places according to their sequence. The product lengths are measured using gel electrophoresis. Two clones that overlap will contain many fragments of the same length. Now, using the overlap information we can infer the original ordering of the clones. Typically, you cut with 2 restriction enzymes, a and b, as well as a combination of a + b. (Double Digestion)



→ Fragment Assembly

We now move on to talk about what we do with the millions of reads that we've acquired. What we want to do is to take these pieces and put them together in a coherent order to form the genome in question.

These are the Steps to Assemble a Genome :

1. Finding overlapping reads

The goal of the overlapping step is to find, among all the reads we have, all the pairs of reads that are actually overlapping with each other in the genome. In other words, two reads would be overlapping if they're offset from each other only a small amount, so that both reads contain a section of DNA that is common to both. Since we have millions of reads, we cannot afford to use an algorithm that runs in longer than linear time (in the number of reads we have), in order to be efficient. Earlier algorithms for fragment assembly, such as PHRAP, required comparing every pair of reads, which resulted in $O(N^2)$ time.

Basic Linear-Time algorithm:

1. Using a value of $k \sim 24$, list all the words of length k in all the reads. Each read yields $2 * \text{ReadLen} - (k + 1)$ words. Note: Due to memory limitations, this task must be done in several passes, each time only taking k -mers that start and end in particular sequences
2. The k -mers are sorted according to read, position, word, orientation
3. Sort the k -mers according to word (lexicographically), read, orientation, position. Sorting can be done in $O(N)$ time, using radix sort
4. Read all overlaps off this list; this can be done easily because all overlaps will be adjacent to each other (since the k -mers are sorted lexicographically!)
5. Extend each overlap to full alignment between its two source reads and throw the pair away if not 98% similar (this can be done very quickly)

One problem we encounter here is repeats. Basically, very common k -mers that occur many times in the genome can increase our running time to quadratic from linear, because k -mers that occur N times cause $O(N^2)$ overlaps or read/read comparisons. In addition, they do not offer very useful information, since for overlaps we're interested in overlaps with unique regions, not overlaps stemming from repeats.

The solution to this is to discard all k -mers that occur too frequently. This threshold is variable, as we have to set it to balance the sensitivity/speed trade-off, according to the genome at hand and the computing resources available. What this means is that we will require that every overlap we do pay attention to include an uncommon k -mer, because otherwise we will not flag it as an overlap (since we've already discarded all the common ones).

The next step is to perform the error correction. We start by creating local multiple alignments from the overlapping reads, as shown below:

```
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
```

insert A

replace T with C

```
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
```

```
TAGATTACACAGATTACTGA
TAGATTACACAGATTACTGA
```

```
TAG-TTACACAGATTATTGA
TAG-TTACACAGATTATTGA
```

On the left side, we can see quite clearly that the gap in the bottom sequence should be filled in with an A, according to the other reads, and the T in the middle sequence is a sequencing error that should be a C. We can fix these and move on. However, the correlated errors (on the right) are probably caused by two versions of the same repeat. The solution is to disentangle the overlaps into two sets of the same overlap.

2. Merge Reads into Contigs

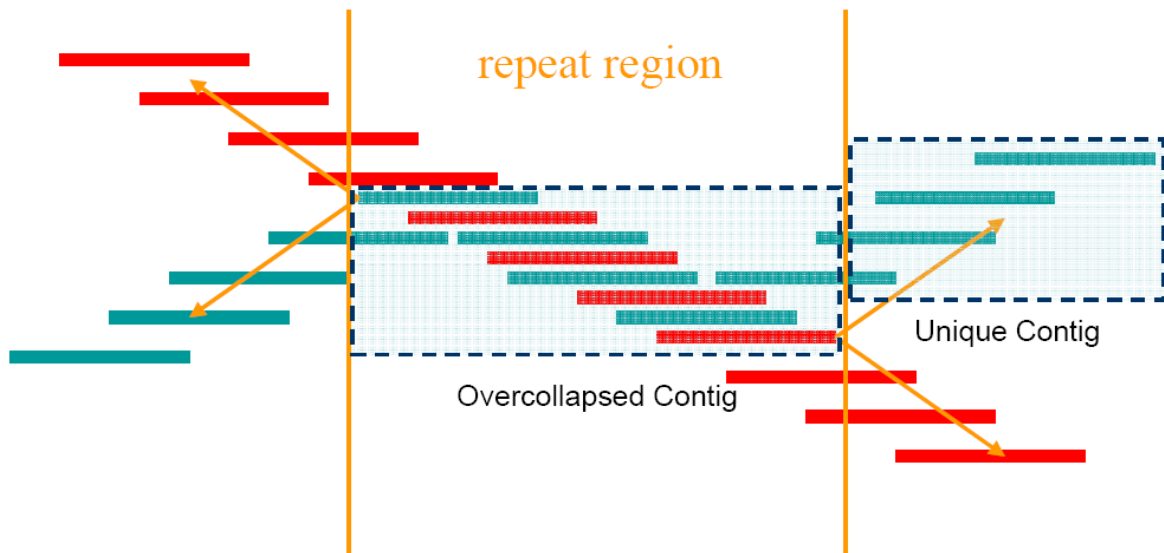
Summarize the overlap info into an overlap graph by forming one node for every read, where:

- Nodes are : reads $r_1 \dots r_n$
- Edges are : overlaps $(r_i, r_j, \text{shift}, \text{orientation}, \text{score})$

The main problem here is the repeated regions in the overlaps. In the following example, we see a set of overlaps that appear to all be related. However, the blue reads and the red reads actually come from different regions of the genome (although we can't tell this!). They appear to all be related because they both have overlaps with the same repeat region.

The problem this causes is that we want all the contigs we form to actually exist in the target genome, but in this case, if we formed a contig that crosses the repeat boundaries, we would be forming a contig that does not really exist!

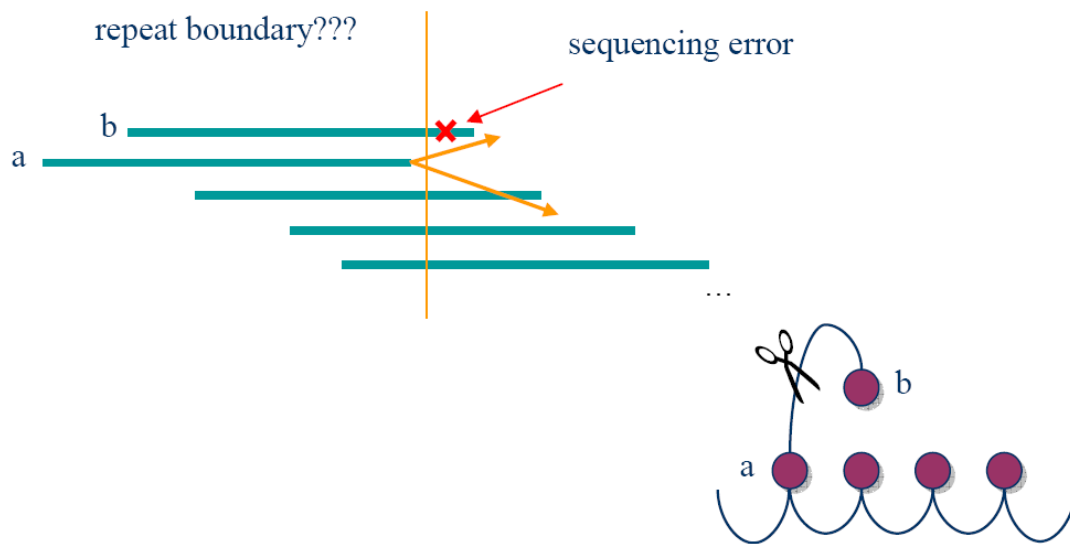
To avoid doing this in this case, we have to only look at reads up to potential repeat boundaries; here, we would form five contigs: four for the flanking regions, one for the repeat region.



Now the question is how to find the repeat boundaries. Detecting repeat boundaries like this one relies upon recognizing a read that is overlapping with two other reads that are very dissimilar to each other (like the red read at the bottom right of the repeat region), which is overlapping both with the blue read above it and the red read below it; the non-overlapping regions in those two reads do not match at all.

One should also have to be careful not to flag a repeat boundary where one doesn't exist. In this example, if we're not careful, we may see this as a repeat boundary, when in reality, we're dealing with a sequence error on read b that

makes it look like it's overlapping with an unrelated region.



The solution is to ignore 'hanging' reads that we see here (the overlap with b) when detecting repeat boundaries, because it does not indicate a true repeat boundary.