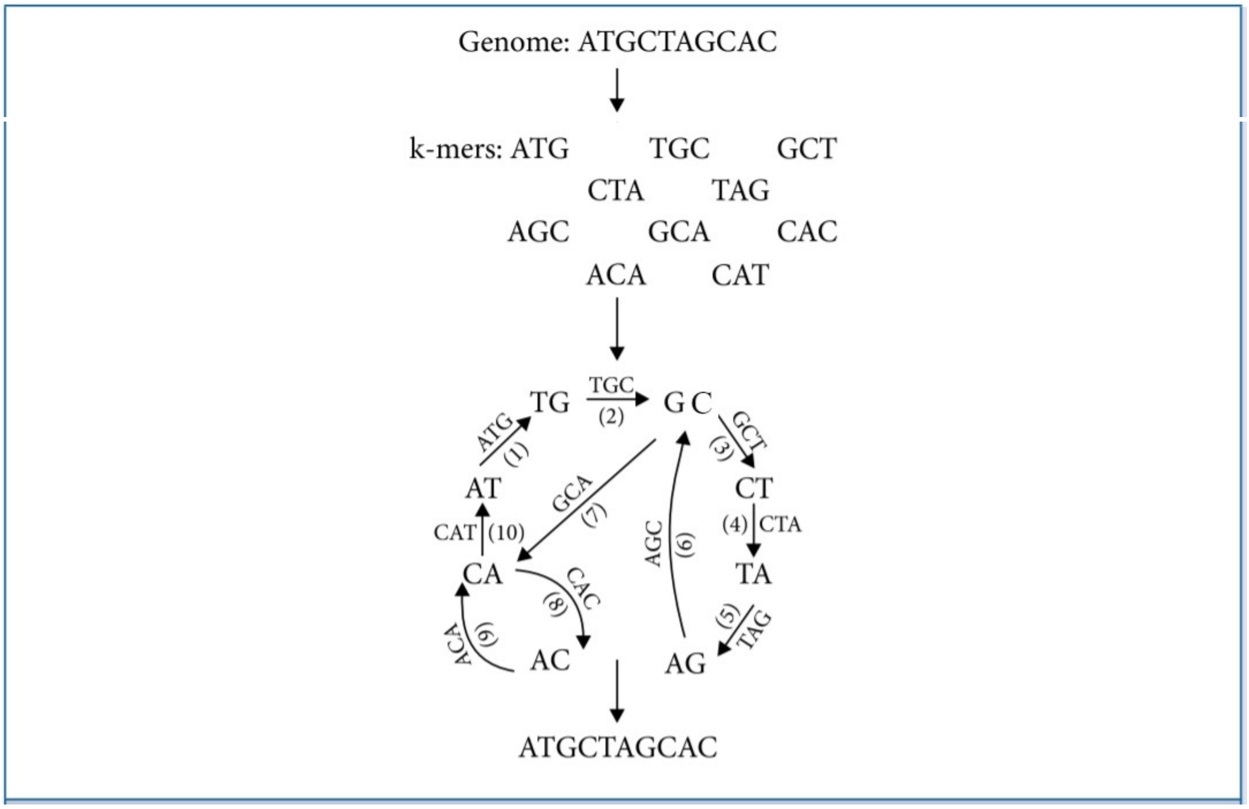
Lab 13: De-novo Genome Assembly

Introduction to Bioinformatics programming.



DNA sequencing involves taking a series of reads and aligning them with each other to get the full length of the original sequence. This process involves finding the overlap of potentially millions of short sequences. Doing this by brute force would computationally infeasible. To get around this problem a solution from graph theory is employed. For this lab we will practice performing an assembly on a short sequence inside R. Note that real genome assemblies involve much larger datasets and usually occur outside the R environment. This practice implementation will provide a window into what takes place on a smaller scale.

Part 0: Create a new R Script called [YourLastName]Assembly.R

Part 1: Write 3 functions.

1) Kmerize will take a DNA sequence in the form of a string and an integer n reflecting the size of the kmer. It will return a list of strings representing overlapping kmers of length n - k. Where n is the length of the input DNA Sequence
2) kmerGraph will take a list of kmers as a vector of strings. It will use the igraph package to produce de Brujn graph of the unique kmers. The graph will contain an edge for every pair of kmers where the ending k - 1 nucleotides are the same as the beginning k -1 nucleotides of another kmer. For example if k = 5 then the sequences “ATTCA” and

“TTCAG” would result in an edge connecting the first to the second. The function should return the graph object.

3) getAssembly should take the graph object and find its eulerian path. If there is no eulerian path it should return nothing. Once the path is fount it should extract the node names and merge them back together. The sequence should be identical to the original input sequence.

Part 2: Test your functions on different DNA sequences using different length kmers. Is there a minimum kmer length for finding a single unambiguous assembly? Record your answer as a comment at the end of your code.

Part 3: If you are successful in assembling your kmers from a single continuous sequence write two additional functions.

1) readWriter will take an input DNA sequence and two integers, the first readSize will determine the size of the simulated read that will be taken from the sequence. The second integer, numReads will determine how many reads are taken. The function will take the specified number of reads from the sequence at random locations along the sequence. It should return the reads as a character vector.
2) readWronger will take the vector of reads and a number errorRate between 0 and 1. This number will be the probability that a given nucleotide was called incorrectly by the sequencer. It should find the total number of bases across all of the reads and switch the base on a random sampling of read loci determined by the given error rate. The function should return the error laden reads as a vector.
3) Modify your kmerize function such that it can take either a single sequence or a string of sequences and return a list of kmers.
4) Observe how the change in error rate changes the properties of your graph, what is the relationship between the error rate and the number of nodes in your graph.

Part 4: Save your script and upload to the Mycourses Drop Box.