Series 3

Genomics and bioinformatics - Week 3

October 3, 2011

1 Learning objectives

The final goal of the programming exercises you receive each week is for you to be able to download some interesting data files from UCSC, format them, extract the relevant data from them and apply some statistics that you have seen in this course. At the end of the fourteenth week, you should be able to answer many different types of biological questions by fetching the data needed and using some existing software necessary for the analysis or writing the code necessary for the analysis yourself.

2 Today's exercise

We are going to create a contig from a bunch of reads. The reads have been specially created by the assistants for this problem and will be given to you on the Moodle website. Your goal is to generate the largest contig possible. We will consider a simplified ideal case where:

- The reads don't contain any errors.
- The reads have the same length (60bp).
- The reads are all from the same strand.
- The full chromosome of our imaginary species is small.

2.1 Hamiltonian path problem

1. Read the file `reads.fastq` and load all the reads it contains inside a list. It may help you to take a look at `reads.fastq` in your text editor first.
2. Write a function that takes two reads as input and returns the overlap sequence; if they don't overlap, return `False`. The function should also have a `min_overlap` parameter that specifies how many base pairs must match between both reads at a minimum for it to be considered an overlap.
3. Using the the list of reads you just loaded and the function you just wrote, you can now generate a list of vertices and a list of edges as defined for the Hamiltonian problem. Indeed, the reads will act as the vertices and the overlaps as the edges. As you know, an edge exists if a vertex connects to an other vertex. In our case, a link between two reads exists if a read overlaps another read.
4. Use the following function (brute force longest path algorithm) to extract the largest contig possible, and display it on the screen. Record the time it takes to compute (import the `time` library).

2.2 Eulerian path problem (de Bruijn algorithm)

1. Write a function that for a given read returns a list of all the l-mers contained in the read, e.g. f(AATGCG) -> [AATG,ATGC,TGCG] for l=4.

2. Taking l = 59bp, and using the same reads library as before, generate the list of vertices and the list of edges as defined for the Eulerian problem.

3. Use the following function (Hierholzer's algorithm) to extract the largest contig possible, and display it on the screen. Record the time it takes to compute, and compare with the Hamiltonian case.

3 Python tricks

1. You don't need to use the indexes of a vector to iterate over its elements. Example:
2. Use `;` to put several statements on the same line. Else it is not needed.
3. dir(object) tells you all the existing methods for this object. If you are using `ipython`, you can type `object.` and press Tab to display the same information.
4. Comprehension lists

4 R tricks