October 2016

Assignment 1, Genome Informatics Module, Computational Biology MPhil

This assignment is to be carried out individually.

Due by **11.45pm Tuesday 1st November**. This is a fixed deadline and there will be no extensions as per the MPhil handbook.

Style: This assignment should be written in the style of a scientific paper, with sections for abstract (max. 250 words), introduction, results, discussion and methods. Your report should include the answers to the questions below.

Page limit: There is a page limit of 5, this is the absolute maximum and any additional pages will not be read. Please keep answers short, and use graphs and tables where appropriate. It is in your interests to be lucid and concise, and to think about the most informative way to present your findings - i.e. summarise results rather than show all the details.

Submission: write-ups and code must be submitted via the Moodle website. You must submit all code written along with your write-up. The code must be documented and will not count against your page limit. **EVERYTHING ELSE WILL**.

If you have any technical difficulties with the assignment please email Dr Alastair Crisp (eadc2@cam.ac.uk).

Assignment Description

Next generation sequencing (NGS) technology is currently developing rapidly, with throughput increasing and costs decreasing dramatically each year. The AB Solid, Roche 454, and Illumina HiSeq sequencers all produce huge numbers of 'short' sequences in parallel. For example, Illumina HiSeq sequencers can generate up to 500 gigabases of sequence per run. As more and more researchers use NGS technology, progressively larger quantities of data are being generated. The challenge for bioinformaticians and computational biologists is how to process this data quickly and effectively.

Every person has been provided with the same NGS data, 1129046 x 100bp genomic Illumina reads, from a bacterium.

The reads are in FASTQ format and are spread across two files labelled 1 and 2. Pairs of reads come from a single piece of DNA of a known size range (here the mean size is approximately 500 bp). File '1' contains the forward reads and file '2' contains the reverse reads. Read x in file '1' is paired with read x in file '2'. The read pairs are oriented –1–>gap<–2–

The first part of the assignment involves *de novo* assembling these short reads using Velvet, a short-read assembler developed at the EBI. Part two will involve identifying the bacterium, and aligning its genes to *E. coli*. Part three will involve investigating differences in gene content between the two species.

We have installed several useful programs, including Velvet and Exonerate, on a server (subliminal.maths.cam.ac.uk) so you can use them during the assignment. You are welcome to use any other software you want, and writing your own code will score extra points (but please don't try building your own assembler). The data files can also be found on subliminal in '/local/data/public/genome_informatics/assignment_1/' (see later for details of how to log on to subliminal).

1 QC & Assembly of the short reads using Velvet

a. Parse the provided files into an appropriate format for the velvet assembly software.
b. Assemble the short reads using velvet. Do not run multiple copies of velvet simultaneously or the server may run out of memory. You can run velvetg repeatedly on the same hashed data, without having to rerun velveth. n.b. velvet is currently compiled with a maximum k-mer length of 31 (it can be quickly and easily recompiled if you want to investigate higher values).
c. Use the node-coverage distribution to estimate the values of expected coverage, and coverage cutoff. Try a range of combinations of these values to see how they influence the N50. Explain what these parameters do and what are the 'optimal' parameters for the dataset.
d. Use the 'optimal' parameters to produce a FASTA file of the assembled contigs for use in the rest of this assignment. What are the N50, max-contig length, and what fraction of the reads are used in the final assembly?

2 Alignment for identification

a. What is this bacterial species? Align your genome to databases of other genomes to find out. Rather than looking at every contig in the assembly it should be sufficient to search for sequences similar to a few of the larger contigs. Using NCBIs BLAST web-interface would be one simple way to do this. Comment on the potential weaknesses of your approach.
b. Download the genes for the species from EnsemblBacteria. Align these genes to the genes of the closely related species *E. coli* K-12 (file supplied). Any aligner may be used.
c. Parse the alignment to get the names of the *E. coli* genes that correspond to those in the original species. You may find multiple genes aligning to the same one and genes with no alignment. Why is this?

3 Differences in gene content between the two species

There are a very different number of genes in the FASTA files from the two species.
a. What percentage of the original species genes are found in *E. coli* and vice versa? How similar are the genes to each other?
b. Do you think this difference in gene number is real, or an artifact of annotation - e.g. one species has been poorly annotated, too many genes have been predicted in one species or both?
c. What are the most common functions of the genes in the original species? This can be determined using enrichment analysis on the list of *E. coli* gene names e.g. at `http://geneontology.org/`.

Help and Guidance

Connecting to the server

You can connect to the server using secure shell (SSH)
From the linux command line you connect with the following command.
ssh <username>@subliminal.maths.cam.ac.uk
Windows users can use PUTTY `http://www.chiark.greenend.org.uk/~sgtatham/putty/`
There are plenty of HOW-TO guides online.

Using (or being) nice

To promote the sharing of resources when multiple users are trying to use the same server you can add:
nice -n x
where x is a number from 1 to 19 (1 highest priority) and the server will assign CPU time based on the priorities. This means you can run your programs using all 64 cores of subliminal (where possible) without worrying about preventing other people using the server and your programs will automatically fill the avaliable CPUs if other programs finish. This only works well if EVERYONE uses it. I would suggest you all use the same priority, say 5.

SFTP - SSH file transfer protocol

SFTP can be used to move files to and from the server.

GNU Screen

Screen is a terminal multiplexer which allow processes to continue running even after the client disconnects from the server. You can then reconnect to the screen session when you next login. After SSHing into the server, type 'screen' to start a new screen session. There are numerous screen tutorials online to help you use its full functionality.

top

The 'top' command displays what is currently running on subliminal. On the line starting 'MiB Mem' there is a total and a used figure. On the next line is a cached figure.
If $used - cached \approx total$ DO NOT start another copy of velvet. Velvet uses progressively more memory as it runs and if you use all the memory you will crash the server.

Bioinformatics Software

Programs are found in '/local/data/genome_informatics/programs/' and are not currently in $PATH.

Velvet (Zerbino & Birney 2008)

Velvet consists of 2 programs velveth and velvetg. Velveth is a simple hashing program that prepares the data for velvetg, a de Bruijn graph based assembler. Velvet can handle a number of input formats including FASTQ. For paired-end reads, each read MUST be next to its mate read. In other words, if the reads are indexed from 0, then reads 0 and 1 are paired, 2 and 3, 4 and 5, etc. Read orientation, Velvet expects paired-end reads to come from opposite strands facing each other.

The Velvet manual is available at `http://www.ebi.ac.uk/~zerbino/velvet/Manual.pdf`.

Recommended Workflow:

1. Run velveth - specifying read type and kmer length
2. Run velvetg, do not specify any parameters
3. Graph the stats.txt file, estimate the expected kmer coverage and kmer coverage cut off.
4. Rerun velvetg, specify -exp_cov and cov_cutoff parameters. N.B. you can rerun velvetg with different parameters WITHOUT rerunning velveth.

To run velveth:

$ velveth directory hash_length [-file_format][-read_type] filename
hash_length = an odd integer
-file_format = -fastq
-read_type = -short or -shortPaired
e.g. $ /local/data/genome_informatics/programs/velvet_1.2.10/velveth assembly_1 31 -fastq -short short_reads.fq
e.g. $ /local/data/genome_informatics/programs/velvet_1.2.10/velveth assembly_2 31 -fastq
-shortPaired short_reads_paired_end.fq
Typing 'velveth' will give you a full list of options.

To run velvetg:

/velvetg directory [options]
-exp_cov <floating point>- expected kmer coverage of unique regions.
-cov_cutoff <floating-point>- kmer coverage cut off (removes low coverage nodes from the graph).
-ins_length <integer>- average distance between two paired end reads.
e.g. $ /local/data/genome_informatics/programs/velvet_1.2.10/velvetg assembly_1 -exp_cov 25 -cov_cutoff 2
e.g. $ /local/data/genome_informatics/programs/velvet_1.2.10/velvetg assembly_2 -exp_cov 15.5 -cov_cutoff 3.5 -ins_length 500
Typing 'velvetg' will give you a full list of options.