Comparative Genomics and Visualisation Assignment 2015

Notebook preparation

The following line imports `pylab`, making useful MatLab-like functions available, and allowing inline graphics. As with any other iPython cell, you can run the cell by pressing `Shift-Enter`.

```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Introduction

All questions in this assignment can be answered using either the notebooks, or example code, from the activities in the Comparative Genomics and Visualisation 2015 course repository, and using the Life Sciences server for the course. Most modifications to the in-class exercise code require no more than changing the locations of input files.

Data download

To complete the exercises in this assignment, you will need to download data from the public databases at NCBI. To do this, run the `get_data.sh` script in the `assignment/data` directory. You can do this at the command-line with:

```
cd data
./get_data.sh
```

This script will download sequence and annotation files for four bacteria in each of the genera:

- *Dickeya*: `NC_014500`, `NC_013592`, `NC_012880`, `NC_012912`
- *Klebsiella*: `NC_016612`, `NC_011283`, `NC_017540`, `NC_013850`
- *Pseudomonas*: `NC_022808`, `NC_007492`, `NC_004578`, `NC_010322`
- *Staphylococcus*: `NC_004661`, `NC_007795`, `NC_017337`, `NC_013893`

These files will be placed in the `assignment/data` directory, and directories specific to each of the data file types `.faa`, `.fna`, `.gbk`, and `.gff` will also be created.

Part 1

In the `assignment/data` directory you will find the FASTA sequence file named `draft_genome.fasta`, representing assembled contigs from a bacterial genome sequencing project.

1a: Using bulk summary statistics of %GC content and genome size, produce a plot of genome length against %GC content, and determine to which of the four bacterial genera *Dickeya*, *Klebsiella*, *Pseudomonas* and *Staphylococcus* the draft genome is likely to belong.

1b: Comment on the relationship between the length of the draft assembly genome, and the lengths of the genomes in the genus to which you think it belongs.

NOTE: The in-class exercise `ex01_gc_content.ipynb` may be useful here. A copy of that file has been made available in this directory, along with the helper code in the `bs32010.ex01` module.

Part 2

2: Using Average Nucleotide Identity (ANI) measures, determine whether the draft genome belongs to the same species as any of the other members of the genus you identified in part 1.

You will need to inspect the `.fna` files corresponding to the genus you identified in part 1, to determine the species name in each case. You should copy these `.fna` files, as well as `data/draft_genome.fasta`, to a new directory (e.g. `assignment/my_sequences`) for the ANI analysis - otherwise it may take a long time to calculate.

NOTE: ANI analyses involve multiple pairwise genome comparisons, so may take a long time to run, depending on the computing power available to you.

NOTE: A suitable %ANI threshold for assigning species was given in the lecture slides

NOTE: You can use the script `average_nucleotide_identity.py`, the in-class exercise `ex03_ani.ipynb`, or the program `JSpecies` (http://imedea.uib-csic.es/jspecies/, or any other convenient method, to calculate ANIb, ANIm, and TETRA. The `average_nucleotide_identity.py` will likely be easiest.

NOTE: The `average_nucleotide_identity.py` script can be run from an iPython cell, as below.

```
# Help text for the average_nucleotide_identity.py script
!./average_nucleotide_identity.py -h
```

```
usage: average_nucleotide_identity.py [-h] [-o OUTDIRNAME] [-i INDIRNAME] [-v] [-f] [-s] [-l
```

```
                                        [--noclobber] [-g] [--gformat GFORMAT] [--gmethod GMET
                                        [--scheduler SCHEDULER] [--maxmatch] [--nucmer_exe NUC
                                        [--makeblastdb_exe MAKEBLASTDB_EXE] [--blastall_exe BL

optional arguments:
  -h, --help            show this help message and exit
  -o OUTDIRNAME, --outdir OUTDIRNAME
                        Output directory
  -i INDIRNAME, --indir INDIRNAME
                        Input directory name
  -v, --verbose         Give verbose output
  -f, --force           Force file overwriting
  -s, --fragsize        Sequence fragment size for ANIb
  -l LOGFILE, --logfile LOGFILE
                        Logfile location
  --skip_nucmer         Skip NUCmer runs, for testing (e.g. if output already present)
  --skip_blastn         Skip BLASTN runs, for testing (e.g. if output already present)
  --noclobber           Don't nuke existing files
  -g, --graphics        Generate heatmap of ANI
  --gformat GFORMAT     Graphics output format [pdf|png|jpg|svg]
  --gmethod GMETHOD     Graphics output method [mpl|R]
  --labels LABELS       Path to file containing sequence labels
  --classes CLASSES     Path to file containing sequence classes
  -m METHOD, --method METHOD
                        ANI method [ANIm|ANIb|ANIblastall|TETRA]
  --scheduler SCHEDULER
                        Job scheduler [multiprocessing|SGE]
  --maxmatch            Override MUMmer to allow all NUCmer matches
  --nucmer_exe NUCMER_EXE
                        Path to NUCmer executable
  --blastn_exe BLASTN_EXE
                        Path to BLASTN+ executable
  --makeblastdb_exe MAKEBLASTDB_EXE

  --blastall_exe BLASTALL_EXE
                        Path to BLASTALL executable
  --formatdb_exe FORMATDB_EXE
                        Path to BLAST formatdb executable



# Using the average_nucleotide_identity.py script on the nucleotide sequences
# in the ./my_sequences directory, to produce a table of ANIm scores.
#
# (uncomment the line below to run an ANIm analysis on the files in ./my_sequences)
#!./average_nucleotide_identity.py -i my_sequences -o assignment_ANIm -m ANIm -v
```

NOTE: The graphical output of the `average_nucleotide_identity.py` script will not be available to you on the Life Sciences server, as a more recent `matplotlib` version is required.

Part 3

Use suitable whole genome alignments to investigate whether the draft genome has undergone rearrangement with respect to any of the complete genomes in the genus you identified in part 1. Produce graphical output to demonstrate the extent of rearrangement (or lack thereof). Comment on your figures, and your choice of alignment program.

NOTE: The `whole_genome_alignments_A.md` exercises may provide a useful template.

NOTE: The `nucmer_to_crunch.py` script in this directory will be useful in generating `.crunch` output from `NUCmer` comparisons that can then be visualised using `ACT`. This can be run from an iPython notebook, as below:

```
# Help text for the nucmer_to_crunch.py script
!./nucmer_to_crunch.py -h

usage: nucmer_to_crunch.py [-h] [-o OUTFILENAME] [-i INFILENAME] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -o OUTFILENAME, --outfile OUTFILENAME
                        Output .crunch file
  -i INFILENAME, --infile INFILENAME
                        Input .coords file
  -v, --verbose         Give verbose output
```