Genome Biology and Genomics

Tom Williams, School of Biological Sciences
(tom.a.williams@bristol.ac.uk)

Session 3: Genome Annotation practical

Overview: Genome annotation is the process of finding, labelling, and predicting the function of genes and other interesting features on genome assemblies.This practical gives you hands-on experience with structural and functional annotation of prokaryotic genomes, and an introduction to how to interpret functional annotations in terms of metabolic pathways and the functional capabilities of an organism.

*NB: The optional exercise is **not** mandatory, and the coursework will not require anything specific to the optional material*

Learning objectives: We're given a genome assembly: now what? Learn how to assess assembly quality and completeness; how to find genes and other sequence features; how to assign functions to predicted genes; how to interpret annotated pathways in terms of organismal capabilities, functions and ecological roles.

General notes

- These practicals are not assessed, but you may find it helpful to keep some notes on your progress in a format that makes sense to you. You could consider a Word processing document, a Jupyter notebook, or anything in between. The timed assessment at the end of the course directly tests the skills you are learning in these practicals, so you may find keeping detailed notes of use at that stage.
- Students on this course have a fairly broad range of background knowledge and existing experience with genomics, bioinformatics, working at the command line, and so on. To help to deal with this, the practical contains some activities labelled as "optional". If you quickly complete the main tasks and are looking for something to do, you may find these more interesting/challenging/time consuming/useful than the main activities. These are truly optional, and you should not feel nervous if you do not complete them.
- Text in the `Consolas font` indicates a command you might run in a bash session or job script. The convention `myprogram <input file>` means that <input file> should be replaced with the name of the input file. For example, if your input file was called hello.fasta, the above example indicates that you should run `myprogram hello.fasta` on the command line. You should not use the characters "<>" on the command line unless you mean to. Feel free to use informative names for your program outputs and inputs, not just the default ones in the example commands below.

- Most bioinformatics tools will print something helpful if you run them with -h, --h, -help, or --help arguments.
- All of these analyses can either be run on your laptop/some other machine (with a UNIX environment), or on the BluePebble cluster. If the latter, please use job submission scripts and do not run any jobs on the head node. If you run jobs on a remote machine (such as BluePebble), you will need to use scp or an alternative to copy files back and forth between your machine and the server.
- Please ask for help if you are stuck, or if you have questions about the underlying theory or rationale of what you have been asked to do. It's also OK to ask your more knowledgeable friend, or to try to solve issues yourself by googling for the answer.
- When in doubt, searching chunks of sequence against the NCBI RefSeq or GenBank databases may be a good way to understand what sort of data you are dealing with. This can be done interactively using a web browser at https://blast.ncbi.nlm.nih.gov/Blast.cgi.

Running jobs on BluePebble

If you plan to work through the practical on BluePebble, you should first log in in the usual way, then request an interactive job:

```
srun --nodes=1 --ntasks-per-node=1 --time=01:00:00 --pty bash -i
```

This will launch an interactive bash session on one of the compute nodes, so you can run commands directly. **Whatever you do, do not run jobs directly on the head (login) node!**

All of the needed tools are installed on BluePebble already. To use them, you need to load the appropriate module(s) using the `module add <insert name>` command. To see a list of available modules, type `module avail`

BUSCO: Loading BUSCO requires loading three modules in the correct order, and then activating a conda environment (that's bioinformatics for you…).

```
module add lang/python/anaconda/3.8-2021-TW
module add lang/perl/5.30.0-bioperl-TW
module add apps/blast/2.2.31+
source activate busco-5.0.0
```

Once you have finished using BUSCO, you should deactivate the environment:

```
conda deactivate
```

Prokka: Loading prokka requires loading three modules in the correct order. (If you get things wrong, use `module unload moduleName` to remove the offending module and start again):

module add lang/python/anaconda/3.8-2021-TW
module add lang/perl/5.30.0-bioperl-TW
module add apps/blast/2.2.31+

Installing software on your local machine

In general, this involves visiting the software website or github repository and following the installation instructions. As people following this route often know what they are doing, we have decided to give the precise commands for the tools installed on BluePebble and loaded via `module add` in the tutorial below; you will need to modify them slightly for a local install (e.g., you may need to change the path to the binary).

(NB: If you cannot find something, try Googling for it. The Github pages of software packages usually point to some documentation. Sources like Stack Overflow, Biostars can also be very useful).

BUSCO: https://busco.ezlab.org/busco_userguide.html
Prokka: https://github.com/tseemann/prokka

Problems with Prokka on BluePebble

In true bioinformatics style, the Prokka install on BluePebble continues to behave erratically and give unexpected errors. If you are attempting to use Prokka and having it fail, you can use the protein-coding gene prediction tool prodigal instead, as a reliable backup. Prodigal is the tool Prokka uses to actually find the coding sequences, so if you load the Prokka modules, prodigal will also be available. The “-i” flag is used to provide the input DNA sequences for gene finding.

```
prodigal -a my_protein_output.fa -i contigs_file -o genes_in_gff
```

Further (optional) reading

Description of *Naisuia deltacephalinicola*:
https://academic.oup.com/gbe/article/5/9/1675/555845

A short review of the current state of genome annotation methods:
https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1715-2

Review on the distinction between orthologues and paralogues:
https://link.springer.com/protocol/10.1007/978-1-4939-9074-0_5

Beginner's guide to eukaryotic genome annotation: https://www.nature.com/articles/nrg3174

Prodigal paper: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-119
Prokka paper: https://academic.oup.com/bioinformatics/article/30/14/2068/2390517

Part 1. Clone the course repository from Github

All of the data needed to follow the course practicals can be found in the Github repository for the course (https://github.com/Tancata/GenomeBiologyandGenomics). Use git to clone (create a local copy of) the repository in your home directory (or somewhere else convenient, such as a directory you might create for this course):

```
module load tools/git

git clone https://github.com/Tancata/GenomeBiologyandGenomics.git
```

This will create a directory, "GenomeBiologyandGenomics", containing the course materials. Later in the course, you may need to update the repository (as I update materials to fix errors that you might helpfully report to me, or post more materials for later practicals). You can do this by cd'ing into the GenomeBiologyandGenomics directory, and typing:

```
git pull
```

Part 2. Structural and functional annotation of a small prokaryotic genome (*Nasuia deltacephalinicola)*.

Nasuia is a bacterial endosymbiont of the leaf hopper, *Macrosteles quadrilineatus.* To my knowledge, it has the smallest known prokaryotic genome, at ~112kb (the *Nasuia* genome is about one-fortieth the size of the *E. coli* genome at 4.6Mb). It's a complete organism (sort of), at a genome size which will allow us to annotate and interrogate the genome in a reasonable amount of time, either on your laptop computer or on the Blue Pebble cluster.

BUSCO is a widely-used tool that surveys a genome assembly for the presence of “universally conserved” marker genes in order to estimate the assembly completeness (an assembly which only contains 5 out of 10 “universal” marker genes is ~50% complete).

1. **Use BUSCO to assess genome completeness in terms of the proportion of bacteria-specific marker genes identified. What does the result indicate? (Note that this is the complete, closed genome assembly of *Nasuia*).**

```
busco --in nasuia.fasta --mode genome --lineage bacteria_odb10 --out
output_dir
```

2. **Perform structural annotation of the *Nasuia* genome using Prokka. Both programs take the genome assembly (nasuia.fasta) as input:**

```
prokka --kingdom Bacteria nasuia.fasta
```

[OPTIONAL] Write a bacterial gene finder using Python (or your favourite other scripting language). Run it on the *Nasuia* genome. How many protein-coding genes does your method find compared to Prokka?

Prokka performs a basic gene functional annotation analysis using a database of characterised gene families.

3. **Inspect the output of your Prokka run. How many genes were identified in total? How many protein-coding genes? How many of those protein-coding genes have been assigned potential functions using the initial Prokka annotation?**

4. **View the gene annotations in their context on the genome using IGV viewer.** In IGV, you can load a FASTA genome (for example, a “final.contigs.fa” file) using Genomes => Load Genome from File. PROKKA annotations for such a file can then be loaded with “File => Load from File”, choosing the .gff file. How does the structure and coding features of the assembled genome compare to the published reference (https://www.nature.com/articles/s41586-020-2008-3)?

A list of annotated gene functions is useful, but does not give insight into the metabolic pathways possessed by an organism, or its functional/metabolic capabilities. GhostKOALA is a web server that assigns KEGG IDs to a set of protein sequences using a fast similarity search (Ghost) against a functionally annotated database of gene families. The list of KEGG IDs can then be used to populate metabolic pathway lists and maps, helping to make sense of what an organism might be able to do. (BlastKOALA is a similar tool, but GhostKOALA is quicker and is of sufficient accuracy for our purposes here).

5. **Use GhostKOALA (https://www.kegg.jp/ghostkoala/) to functionally annotate the protein set predicted from the *Nasuia* genome. What metabolic modules and pathways do you think *Nasuia* is capable of?** How does your list compare to the original description of *Nasuia* (https://academic.oup.com/gbe/article/5/9/1675/555845)?

Part 3. Genome annotation of a prokaryotic consortium

Had the course proceeded as scheduled this year, you would have already assembled contigs from a co-culture of an archaeon and a bacterium living together in a metabolic cross-feeding relationship (syntrophy). As a result, the set of contigs assembled from the sequence reads contains material from two different organisms.

We will now attempt to functionally annotate the archaeal and bacterial “partners” and understand how and why they might be interacting. You can obtain the assembly of the data from the Github repository (https://github.com/Tancata/GenomeBiologyandGenomics/session3).

1. **Use BUSCO to evaluate the completeness of the assembly. What do you think about the result: do you believe it? Why or why not?**
2. **Use GhostKOALA to annotate all of the proteins from the assembly. What group of Bacteria and Archaea do the two partners hail from? What kinds of metabolisms do they have?**
3. **Why do you think these two organisms were growing together in the culture originally?**