Molecular Microbiology 2024

Protocol 7

Genome Assembly Quality Control and Annotation

This protocol is to assess the quality of a near-complete genome assembled from Oxford Nanopore sequencing data. It includes

- Bandage for looking at assembly graphs.
- CheckM for *estimating* genome completeness and contamination
- Prokka for annotating genomes.
- Artemis for visualising genomes.
- Interproscan for annotating protein sequences against the Interpro database(s).
- Custom scripts `plotGC.jl` and `plot_gene_lengths.jl` to aid in assessing genome assembly quality.

1. Connect to the server `dnaseq1a.bio.au.dk` using SFTP, for example using CyberDuck.

2. In the directory *genome* there will be two directories containing the output of your assemblies: one for flye and one for unicycler. In each of these directories there is a `.gfa` graph file showing the final assembly. For unicycler download the file *assembly.gfa*, for the flye assembly download *assembly_graph.gfa*.

3. Open *Bandage*. Go to *File -> Load graph...*, and navigate to one of the `.gfa` files you saved in the previous step.

4. First look at *Graph information* in the upper left-hand corner. *Total length* gives the total length of the assembly in bases - does this match your expectation?

5. Under *Graph drawing* select *Entire graph*, *Style: single*, and then click on *Draw Graph*. You will now see a graphical representation of your genome assembly, with each coloured line representing a contig. Click on each contig one-by-one and take note of the following characteristics:

- What ID number does it have (look under *Selected node* in the right-hand panel)?
- Is it linear or circular?
- How long is it?
- What is its depth (coverage)?

6. Now open the terminal in VS code, log in to `dnaseq1a.bio.au.dk` and change into the genome directory.

   ```
   ssh molmicroX@dnaseq1a.bio.au.dk
   ```

7. Now change into each assembly directory and use the following command to plot the GC content across each contig in each assembly:

   ```
   plotGC.jl \
   ```

```
--input_file assembly.fasta \
--output_file assembly_GC.pdf \
--windowsize 10000 \
--freq 1000
```

- `plotGC.jl` - This is the name of the program to be run (custom written for this class).
- `--input_file assembly.fasta` - This specifies the name of your input file, a `fasta` file containing the contigs you would like to plot the GC of. This parameter can probably remain unchanged unless the assembly's filename has been changed.
- `--output_file assembly_GC.pdf` - This specifies the name of your output file, a `PDF` file with the GC plots along each contig.
- `--windowsize 10000` - This specifies the size of the sliding window that each GC is calculated at. A higher number here will make for a smoother plot as GC will be averaged over a greater area.
- `--freq 1000` - This specifies the frequency that the GC is sampled along the genome - in this case every 1000 bases. Small numbers produce more fine-grained data and take longer to run, large numbers produce coarser data and run faster.

8. Download the PDF files produced by `plotGC.jl`. Open them and take a look at the plots. Does the GC% match your expectation? For all contigs? Are there any parts of the genome that have an unusually high low GC% compared to the rest, that may indicate a contaminated assembly or recent horizontal gene transfer event?

9. Return to your `genome` directory and then run **CheckM** to estimate the completeness and degree of contamination of your genome. Note that CheckM is designed to operate on all files ending with a given file extension (in this case `.fasta`) within a given folder, so you don't need to specify the input file directly, just the directory containing the input file(s). This will take about three minutes to run.

```
checkm taxonomy_wf \
--threads 3 \
--file checkm_results.txt \
--extension fasta \
domain Bacteria assembly_directory checkm_output_folder
```

- `checkm taxonomy_wf` - This is the command to run checkm using the "taxonomy workflow" (`taxonomy_wf`), where you manually specify the taxonomic level at which to carry out the contamination/completeness assessment. The other option is to determine this automatically using the "lineage workflow" (`lineage_wf`), but `lineage_wf` needs more memory than we have available on our servers.
- `--threads 3` - This parameter specifies the number of processors CheckM will use.
- `--file checkm_results.txt` - This is the name of the output file. You can change this if you like - especially if you run CheckM more than once, you might want to give each iteration a meaningful name.
- `--extension fasta` - CheckM is designed to analyse genomes in

batches, with input specified as a directory containing genome assembly files rather than a single genome assembly. The extension parameter specifies the file extension at the end of your filename as a way to distinguish your genome assembly fasta files from any other files that might be in the target directory. If you have renamed your scaffolds file to have an extension other than `fasta` (for example, `fa`, or `fna`), then you will need to modify this parameter.

- `domain Bacteria assembly_directory checkm_output_folder` - This specifies the taxonomic level and taxon (in this case "domain Bacteria", but this could also be "order Enterobacteriales" or "genus Escherichia" etc.), the name of your input folder `assembly_directory`, and the checkm output folder where checkm will place its working files, `checkm_output_folder`. To view available taxa, type `checkm taxon_list`. To search this list for a taxon of interest, use `checkm taxon_list | fgrep search_term`, e.g. `checkm taxon_list | fgrep Acidovorax` to search for taxa containing the word `Acidovorax`.

10. Look at your **CheckM** results using `cat`. Note the "Completeness" and "Contamination" percentages, and see if you can see how these numbers were derived. Make a note of these numbers, or save `checkm_results.txt` on your own computer, for use in your Genome Report. Note that you might need to deactivate word wrap (for example *View -> Word Wrap* in Sublime Text) to view the CheckM results correctly.

    ```
    cat checkm_results.txt
    ```

    You will see the following headings in your `checkm_results.txt` file:

    - `Bin Id` - the name you have given your scaffolds fasta file
    - `Marker lineage` - the name of the taxon you specified in the previous step.
    - `# genomes` - the number of genomes in CheckM's database used to find single-copy genes. The higher this number, the more precision you can expect from your completeness and contamination estimates.
    - `# markers` - the number of single-copy genes CheckM checks for
    - `# marker sets` - markers are found in co-located "sets"
    - `0 1 2 3 4 5+` - the number of copies found for the single copy genes. Zero copies of a single-copy gene indicates incompleteness. More than one copy of a single-copy gene indicates contamination.
    - `Completeness` - estimated genome completeness
    - `Contamination` - estimated genome contamination
    - `Strain heterogeneity` - if multiple copies of single-copy genes are very similar to one another, then this may indicate a mixture of very closely related strains. This is usually only relevant for metagenomes.

11. Re-run CheckM at several different taxon levels to try out some different accuracy and precision tradeoffs.

12. Looking at the results from GC plotting and CheckM, think about whether all contigs belong in your assembly or whether some should be removed. If necessary remove some contigs and try re-running CheckM. Then choose one assembly to continue to the annotation step with.

13. Before we begin the annotation, it's a good idea to create a "genus database" to base the annotation on. This is a handful of genomes closely related to your genome (typically within the same genus) that Prokka can compare to your genome and borrow the annotations from. This way your annotations will be consistent between close relatives, making comparison simpler. You probably have a good idea of what close relatives of your strain have publicly available genomes from your perusal of NCBI's genome database. You can choose to download these these genomes through the NCBI website and then upload them to the server. Download about five Genbank-formatted genomes from NCBI (the ones with files ending in `.gbff.gz`) then upload them to the server using *CyberDuck*. Place them in a new directory within your home directory called `relatives`.

14. Now unzip the relative genomes and use the following prokka commands to make your genus database, replacing `Genus_name` with your organism's genus name. The genus name is for your own use only - if you have genomes from multiple genera then just pick a name that makes sense.

```
cd relatives
gunzip *.gbff.gz
prokka-genbank_to_fasta_db *.gbff > Genus_name.faa
```

15. There will be a lot of very similar protein sequences in these genomes, and we don't really need them all - they will just slow Prokka down. To get rid of the redundancy we'll use a program called **cd-hit** to make clusters of similar proteins and then pick a representative sequence. Don't forget to change `Genus_name` to the name of the genus you're working with (from step 5).

```
cdhit -i Genus_name.faa -o Genus_name -T 1 -M 8000 -g 1 -s 0.8 -c 0.9
```

    - `-T 1` - number of CPUs to use, in this case 1.
    - `-M 0` - memory limit, in this case 8000 MB
    - `-g 1` - by cd-hit's default algorithm, a sequence is clustered to the first cluster that meet the threshold (fast cluster). If set to 1, the program will cluster it into the most similar cluster that meet the threshold (accurate but slow mode)
    - `-s 0.8` - length difference cutoff, at 0.8, the shorter sequences need to be at least 80% length of the representative of the cluster
    - `-c 0.9` - sequence identity threshold, with 0.9 equivalent to 90% sequence identity to be considered in the same cluster

16. Make a blast database using those representative protein sequences.

```
makeblastdb -dbtype prot -in Genus_name
```

17. Now copy that BLAST database (made of three files, Genus_name.phr, Genus_name.pin, and Genus_name.psq) into the directory where **Prokka** knows to look for it. Your genus database is now set up.

```
cp Genus_name.p* /usr/prokka/db/genus/
```

18. To confirm that it's set up, type the following command to see what databases **Prokka** can see. Your genus of interest should be included under "genera".

```
prokka --listdb
```

19. You are now ready to annotate your genome. Use the `cd` command to change to the directory where your assembled contigs fasta file is and then use `prokka` to annotate your genome. This will take about ten minutes to run.

```
cd ../genome

prokka \
--outdir prokka_annotation \
--prefix output_filename_prefix \
--locustag T \
--genus Genus_name --species species_name --strain strain_ID \
--usegenus \
decontaminated_genome_assembly.fasta \
--mincontiglen 500 --centre A \
--cpus 3
```

- `--outdir prokka_annotation` - Prokka will make a new folder to dump its output into - in this case `prokka_annotation`, but you can change this if you like. Note that if you run `prokka` for a second time you will have to change this - `prokka` will (wisely) not overwrite a previous annotation.
- `--prefix output_filename_prefix` - Here it's `output_filename_prefix`, but this is the string of letters you want to be on the beginning of every output file (i.e. output_filename_prefix.gbk, output_filename_prefix.fna etc.), so it should be the name of your organism - a logical name for your organism would be the genus name then, `MM2020` for Molecular Microbiology 2020, followed by your group number, i.e. `Bacillus_MM2020_1` for group 1.
- `--locustag T` - This is the prefix for all of your locus IDs (i.e. genes IDs). Here it's `T` which means that your genes will be number `T_0001`, `T_0002` etc.
- `--genus Genus_name` - This is your genus name. This is used to identify your genus database (so it must match the `Genus_name` you specified in step 6) and will also be written in your output files.
- `--species species_name` - This is the species name - it doesn't change anything, but this will be listed in your output files. If this is uncertain then just write `sp.`.
- `--strain strain_ID` - This is the strain ID for your specific strain (i.e. the series of letters and numbers that uniquely identify it) - `MM2020_X`, where `X` is your group number, would be fine here.
- `--usegenus` - This flag tells prokka to use the genus database that you made and to find out what that genus database is called from the –genus flag.
- `decontaminated_genome_assembly.fasta` - This is the file with your assembled scaffolds - change this to whatever you called your decontaminated scaffolds file (if you carried out any decontamination).
- `--mincontiglen 500 --centre A` - This sets them minimum contig (scaffold, in our case) length to annotate (500 base pairs here) and sets the name of the "sequencing centre" (not important)

- `--cpus 3` - This restricts the number of CPUs (threads) you will use to two. You will see there are other command line parameters for Prokka in the manual or by typing `prokka -h`.

20. If you `cd` into the output directory (specified by you – `prokka_annotation` from above) you will see a number of files that make up your annotated genome:

- `err` – errors and warnings from the annotation procedure
- `faa` – fasta file with amino acid sequences for all proteins in genome
- `ffn` – fasta file with nucleotide sequences for all genes in genome
- `fna` – fasta file with all scaffolds
- `fsa` – same as fna file, but with extra information required for NCBI submission
- `gbk` – full annotated genome, genbank format
- `gff` – full annotated genome, gff format
- `log` – prokka's output from when it was run
- `sqn` – file for Genbank submission
- `tbl` – feature table file (also related to Genbank submission)
- `txt` – summary of the genome - here you will find the total number of protein-coding sequences (CDS) and various kinds of RNA genes (rRNA, tRNA, etc.), you will need this numbers for your report

Download the `prokka_annotation` folder to your own computer to carry out subsequent steps.

21. If you have a Sanger-sequenced 16S rRNA gene for your genome you should confirm that your genome 16S rRNA sequence is near-identical to the Sanger sequence. Open the `ffn` file on your own computer in **VS Code** or another text editor, and search for "16S ribosomal RNA" (ensure that it is the ribosomal RNA gene you have found, and not a sequence encoding a ribosomal protein). If you have multiple 16S rRNA genes in your `ffn` file you should check them all to make sure they all come from the same organism. Copy the description line and gene sequence to a new file, and copy your Sanger description line and sequence to the same file to make a `fasta`-formatted file with the two 16S rRNA gene sequences. Open a web browser and go to the EBI muscle alignment web tool https://www.ebi.ac.uk/jdispatcher/msa/muscle. Copy and paste the two sequences into the input box and click *Submit*. Are the sequences nearly identical? Some differences, especially towards the 3' end of the Sanger, are an acceptable consequence of degrading sequence quality towards the end of the read and not necessarily cause for concern. If the sequences seem completely different then one of them may be a reverse complement (from the opposite strand), if so you can get the reverse complement of the sequence using a Reverse complement tool.

22. Look at the `txt` file from your prokka output to see how many protein-coding genes your genome has. How does this compare to the close relatives? How about the coding density (number of genes per megabases of the genome) - is this also comparable? A lot fewer or less genes than expected can be a sign of an assembly containing errors.

23. Plot the lengths of the genes from your prokka output using the following

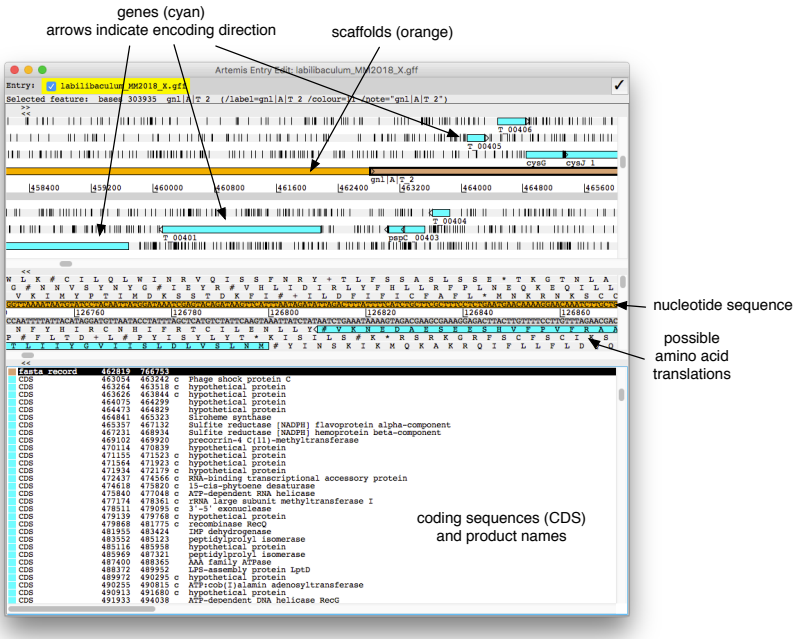
command:

```
plot_gene_lengths.jl \
--input_file gene_file.ffn \
--output_file output_file.pdf
```

- `plot_gene_lengths.jl` - This is the name of the script.
- `--input_file gene_file.ffn` - This is the name of the input file, a fasta file containing all of the genes from the genome, each in its own entry. `gene_file.ffn` should be changed to the file you are using.
- `--output_file output_file.pdf` - The PDF output containing a histogram with gene lengths.

Re-run this command for several of your relatives and your own genome. How does the gene length profile compare? Is there any evidence of sequencing errors causing genes to be missed, split up, or merged here?

24. Open **Artemis** on your own computer. Click `OK` when asked to set the working directory. Go to *File -> Open File Manager...* and navigate to the prokka annotation folder on your computer. Double-click the `gff` file. Right-click on the lower panel and select *Show Products*. You will see a graphical representation of your genome in the upper panel and a list of gene products in the lower panel. To search for specific genes, right-click on the lower panel and select *Goto->Navigator*, then select *Goto feature with this qualifier value*, enter a search term, and click *Goto*. Try some terms like *dehydrogenase*, *chemotaxis*, or *transporter*. Note that Artemis will treat all your scaffolds like one single scaffold in its graphical display, ordered from largest to smallest (i.e. in an arbitrary order). You can always see which scaffold you are on by clicking on the orange-coloured bar, and be aware that gene synteny around the beginnings and ends of scaffolds is not real.



25. It is sometimes useful to try an alternative annotation tool to determine protein function. We will use *Interproscan*, one of many only annotation tools. This tool takes a long time to run, so run it with `screen` so it will keep running after you log off the server. Make sure you're in your prokka output directory, then use the following command:

```
screen -L interproscan.sh -i output_filename_prefix.faa \
-f TSV \
-cpu 2
```

- `-i output_filename_prefix.faa` - This is the fasta file containing the protein sequences inferred from your genome. Change `output_filename_prefix` to whatever you used in step 10.
- `-f TSV` - This specifies that your output will be in tab-separated value (tsv) format, a way of making a table in a text file.
- `-cpu 2` - This specifies the number of CPU threads to run in parallel.

The output file will appear with the name `output_filename_prefix.tsv` - download this file to your own computer and open it in a text editor or Excel to see the matches.