**SOP for Antimicrobial Resistance Identification By Assembly (ARIBA)**

ARIBA is a tool that identifies genes by running local assemblies. The name of the programme must not be confused with the its usage. ARIBA can be used to identify genes (coding and non-coding) from raw reads using various databases or a user-defined database which must be in fasta format.

The input is a FASTA file of reference sequences (can be a mix of genes and noncoding sequences) and paired sequencing reads.

ARIBA reports which of the reference sequences were found, plus detailed information on the quality of the assemblies and any variants between the sequencing reads and the reference sequences.

#Before we begin it is good to create three (3) directories. This will make our work organised and avoid mixing up results.

#To make a directory, log in to your home directory and use the command below:

*mkdir card*

*mkdir virulencefinder*

*mkdir plasmidfinder*

#Now change directory to the directory of your choice for example

*cd card*

#While in the directory of your choice, carry out the commands and instructions below below

#Please read up the documentations here: <https://github.com/sanger-pathogens/ariba/wiki>.

#Generally speaking ariba has four (4) major executables called tasks:

1. *ariba getref*
2. *ariba prepareref*
3. *ariba run and*
4. *ariba summary*

#Some variations exist for MLST calling, where you have two (2) major executables

1. ariba pubmlstget and
2. ariba run

#Let’s now look at how to use ariba to call AMR, virulence and plasmids genes.

**AMR prediction**

# the first step is to get a suitable reference for AMR gene prediction

*ariba getref reference\_name output\_name*

*ariba getref card card\_output*

# This will output two files – card\_output.fa and card\_output.tsv which will be used in the next step to prepare the downloaded card reference.

*ariba prepareref -f card\_output.fa -m card\_output.tsv prepared\_card\_output*

# The prepared reference will now be used to execute “ariba run” command. You will need to use the prefix of your fastq file as a prefix for your output. This will ensure that your output folders have unique identifiers.

ariba run prepared\_card\_output/ ../fastqs/G18503182\_1.fastq.gz ../fastqs/G18503182\_2.fastq.gz G18503182\_output

# ../fastqs/ before the fastq files indicates the path to the files. If the path is not specified you will get an error message and your analysis will not proceed.

# The output\_dir (G18503182\_output) contains seven (7) files, and the most important of these files is the *report.tsv*

# Below is the meaning of columns in report.tsv file:

| **Column** | **Description** |
| --- | --- |
| 1. ariba\_ref\_name | ariba name of reference sequence chosen from cluster (needs to rename to stop some tools breaking) |
| 2. ref\_name | original name of reference sequence chosen from cluster, before renaming |
| 3. gene | 1=gene, 0=non-coding (same as metadata column 2) |
| 4. var\_only | 1=variant only, 0=presence/absence (same as metadata column 3) |
| 5. flag | cluster flag |
| 6. reads | number of reads in this cluster |
| 7. cluster | name of cluster |
| 8. ref\_len | length of reference sequence |
| 9. ref\_base\_assembled | number of reference nucleotides assembled by this contig |
| 10. pc\_ident | %identity between reference sequence and contig |
| 11. ctg | name of contig matching reference |
| 12. ctg\_len | length of contig |
| 13. ctg\_cov | mean mapped read depth of this contig |
| 14. known\_var | is this a known SNP from reference metadata? 1 or 0 |
| 15. var\_type | The type of variant. Currently only SNP supported |
| 16. var\_seq\_type | Variant sequence type. if known\_var=1, n or p for nucleotide or protein |
| 17.known\_var\_change | if known\_var=1, the wild/variant change, eg I42L |
| 18. has\_known\_var | if known\_var=1, 1 or 0 for whether or not the assembly has the variant |
| 19. ref\_ctg\_change | amino acid or nucleotide change between reference and contig, eg I42L |
| 20. ref\_ctg\_effect | effect of change between reference and contig, eg SYS, NONSYN (amino acid changes only) |
| 21. ref\_start | start position of variant in reference |
| 22. ref\_end | end position of variant in reference |
| 23. ref\_nt | nucleotide(s) in reference at variant position |
| 24. ctg\_start | start position of variant in contig |
| 25. ctg\_end | end position of variant in contig |
| 26. ctg\_nt | nucleotide(s) in contig at variant position |
| 27. smtls\_total\_depth | total read depth at variant start position in contig, reported by mpileup |
| 28. smtls\_nts | nucleotides on contig, as reported by mpileup. The first is the contig nucleotide |
| 29. smtls\_nts\_depth | depths on contig, as reported by mpileup. One number per nucleotide in the previous column |
| 30. var\_description | description of variant from reference metdata |
| 31. free\_text | other free text about reference sequence, from reference metadata |

# The report.tsv file can further be summarised with another ariba executable – ariba summary thus:

*ariba summary G18503182 report.tsv*

#G18503182 is the output prefix for the .csv file that will be generated from report.tsv file.

#This command should generate a phandango.csv file, phandango tree (not a true phylogenetic tree though) file and a .csv file. However, for single analysis, you will only have a .csv file. Phandango files will only be generated for multiple .tsv files

# Summary of results can be prepared in a number of formats, depending on the “--preset” flag. This is explained in task summary here: <https://github.com/sanger-pathogens/ariba/wiki/Task:-summary> and some more explanation on the workflow and how some of the parameters are determined here:  <https://github.com/sanger-pathogens/ariba/wiki/The-assembled-column-from-ariba-summary>

**#Virulence genes and Plasmid replicons prediction**

# The steps described for AMR prediction above is also applicable here. The only difference being that whereas for AMR “card” or “resfinder” database will be used, for virulence genes and plasmids “virulencefinder” and “plasmidfinder” databases will be used respectively.

**#Virulencefinder:**

*ariba getref virulencefinder virulencefinder\_output*

**#Plasmidfinder:**

*ariba getref plasmidfinder plasmidfinder\_output*