Instruction to apply rabies-tailored ARTIC network bioinformatic pipeline

Adapted from https://artic.network/ncov-2019/ncov2019-bioinformatics-sop.html
Last updated: 9th May 2023 (new changes highlighted)

Remember: these instructions give the generic structure for commands but require editing to make them specific to your data, they are not a straight copy and paste! I have tried to indicate what you will need to edit.

Make a new directory for analysis

This instruction organises output files into an 'analysis' directory in your home directory, within which each run should have its own subdirectory. If you wish to change the location of the directory, modify the path accordingly (**/path/**analysis/run_name). The commands below will only create these directories if they do not exist already i.e. will not overwrite an existing output directory for the named run.
run_name should be replaced with the name of your sequencing run.

```
mkdir -p analysis/run_name
cd analysis/run_name
```

Activate the ARTIC environment:

All steps should be performed in the artic-rabv conda environment:

```
source activate artic-rabv
```

Basecalling with Guppy

If you did basecalling with MinKNOW, you can skip this step and go to Demultiplexing.
Run Guppy basecaller on the MinION output folder from your sequencing run:
For fast mode basecalling:

```
guppy_basecaller -c dna_r9.4.1_450bps_fast.cfg -i /path/to/reads -s
run_name -x auto -r
```

You need to substitute /path/to/reads to the folder where the FAST5 files from your run are. The default locations for MinKNOW output folders are:
Mac: /Library/MinKNOW/data/run_name
Linux: /var/lib/MinKNOW/data/run_name
Windows c:/data/reads
This will create a folder called run_name with the basecalled reads (fastq files) in it.

Demultiplexing

For the current version of the ARTIC protocol it is essential to demultiplex using strict parameters to ensure barcodes are present at each end of the fragment (the --require_barcodes_both_ends option)

```
guppy_barcoder --require_barcodes_both_ends -i run_name -s
output_directory --barcode_kits "EXP-NBD104"
```

Modify argument –-barcode_kits according to kit used for run e.g.
- if you used native barcodes 1-12 and 13-24 it should be "EXP-NBD104 EXP-NBD114"
- 96 native barcode kit would be "EXP-NBD196"

Note: the output files from this command (or from MinKNOW) will be labelled according to their barcode id i.e. NB01, or barcode01.

The following steps must be done for every barcode

Read filtering
This step is performed for each barcode in the run.
We collect all the fastq files (typically stored in files each containing 4000 reads), filter out reads below the expected amplicon length (less than 350) and concatenate them into a single file.
To collect and filter the reads for barcode03, we would run:

```
artic guppyplex --skip-quality-check --min-length 350 --directory
output_directory/barcode03 --prefix run_name
```

You will now have a files called: run_name_barcode03.fastq

Run the MinION pipeline: medaka
For each barcode you wish to process (e.g. run this command 12 times for 12 barcodes), replacing the file name and sample name as appropriate:

```
artic minion --no-frameshifts --medaka --medaka-model
r941_min_fast_g303 --normalise 200 --threads 4 --scheme-directory
~/Github/artic-rabv/primer_schemes/ --read-file <xx.fastq>
<rabv_ea/V1> <samplename>
```

Replace the read-file, samplename and primer scheme (the bit after fastq) as appropriate.
E.g. To process a sample sequenced with barcode02 (and therefore output from Minknow labelled as barcode02.fastq), prepared using the east Africa primer scheme (rabv_ea) but a sample name of exampleRun1_GBR_RV921…

```
artic minion --no-frameshifts --medaka --medaka-model
r941_min_fast_g303 --normalise 200 --threads 4 --scheme-directory
~/Github/artic-rabv_ea/primer_schemes/ --read-file barcode02.fastq
rabv_ea/V1 exampleRun1_GBR_RV921
```

Relevant primer schemes:
rabv_ea/V1 – East Africa primers
rabvSEasia/V1 – Philippines primers
rabvPeru2/V1 – Peru primers

Double check the filepath to the artic-rabv folder. If this does not match the location on *your* laptop, modify as necessary.

Output files
samplename.rg.primertrimmed.bam - BAM file for visualisation after primer-binding site trimming
samplename.trimmed.bam - BAM file with the primers left on (used in variant calling)
samplename.merged.vcf - all detected variants in VCF format
samplename.pass.vcf - detected variants in VCF format passing quality filter
samplename.fail.vcf - detected variants in VCF format failing quality filter
samplename.primers.vcf - detected variants falling in primer-binding regions
samplename.consensus.fasta - consensus sequence

Gather consensus files together into one fasta file
To put all the consensus sequences in one file called my_consensus_genomes.fasta , run

```
cat *.consensus.fasta > <my_consensus_genomes>.fasta
```

Choose an informative name for the output fasta!

Sequence alignment
MAFFT can be used for alignment (it should already be installed on your laptop). To align sequences in your fasta file, run

```
mafft <my_consensus_genomes>.fasta > <my_consensus_genomes>.aln.fasta
```

This runs mafft with default parameters. Try and distinguish between unaligned and aligned fasta files in the filename e.g. I always add a .aln to the output file to annotate it as an alignment.

To visualise genomes in Tablet
Open a new Terminal window and type:

```
tablet
```

Go to "Open Assembly"
Load the BAM (binary alignment file) as the first file.
Load the reference file (e.g. in artic/artic-rabv/primer_schemes/rabv_ea/V1/rabv_ea_reference.fasta or whatever is the relevant one for your primer scheme) as the second file.
Select Variants mode in Color Schemes for ease of viewing variants.