Reference-based SNP phylogeny

Learning Objectives

In this tutorial you will learn how to

1. List the two main pathways for processing short read data
2. Describe how to select a reference sequence for reference-based mapping
3. Describe the critical steps taken when creating a phylogeny based on comparison of reads to a reference sequence
4. Apply this knowledge to
    a. running the command line tools for processing a single sample
    b. running Nextflow to process a batch of samples and produce a maximum-likelihood tree

Pathways for Processing Reads

There are 2 standard pathways for processing short read whole genome sequencing data:

1. ***De novo* assembly**
   This is a reference-free methodology that uses similarity in the sequence of reads to overlaps reads and join them together to form contiguous pieces of DNA, also known as contigs. Paired-end information can be used to stitch some of these contigs together to form scaffolds. Sometimes within the sequence of scaffolds runs of the character N may be seen. These are the regions between the original contigs that have been joined to make scaffolds. N's have been inserted to represent the unknown sequence between the original contigs.
   The process for *de novo* assembly in the GHRU pipeline consists of the following steps:
    a) QC of reads pre-trimming using FASTQC
    b) Trimming using TRIMMOMATIC
    c) QC of reads post-trimming using FASTQC
    d) Read correction using lighter
    e) Read merging using flash
    f) Read assembly using SPAdes
    g) QC of assembly using QUAST

2) **Mapping**
   This is a reference based methodology where reads are aligned to a reference sequence and changes between the sample sequence and reference sequence can be observed. Therefore selection of a reference sequence is crucial. In order to maximise the amount of diversity that can be observed, selection of a complete genome that is as close as possible to the majority of samples in the analysis is advisable. This will be covered in more depth below

   Having selected a reference sequence, each pair of read fastq files is aligned to the reference using a mapper such as bwa or bowtie. This produces a sam file which can be compressed and sorted so that the reads are in order with respect to the reference sequence to produce a binary bam file. The aligned reads are next processed with a variant caller (examples include GATK, bcftools and freebayes) to produce a variant call file (VCF) that can be compressed into a binary bcf file. This software that produces this file can be parameterized so that it will contain a line per position in the reference

sequence. Each line will describe the base at each position as either reference (REF) or alternative (ALT) type. This is the variant calling process. However sometimes the variant calling is not certain and so the calls must be filtered so that only high quality SNPs are retained. Therefore at each position a base will either be REF type (one of G,A,T or C), ALT type (one of G,A,T or C), a gap since no reads cover this position (represented by a - character) or an uncertain or low quality base (represented by an N). Once the high quality positions have been determined a pseudogenome for each sample can be produced. This will be exactly the same length as the reference sequence and each base will be either G, A, T, C, - or N.
Once all samples pseudogenomes have been produced they can be joined (otherwise known as concatenated) to produce a pseudogenome multiple alignment.
This can be used to generate a phylogenetic tree.

Mapping process

Reference selection

There are several possible mechanisms to select a reference sequence

Using mash to select a reference based on genome identities

Mash is a software that summarises a genome into a sketch based on kmers. See the following links for the paper and documentation.

The methodology can be applied as follows

1) Download a link to all genome sequences
   `wget ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/assembly_summary.txt`
2) Select only those which belong to the species of interest and write out to a file
   `grep 'Neisseria gonorrhoeae' assembly_summary.txt > ngo_genomes.txt`
3) Select only the complete genomes and latest versions.
   `awk -F "\t" '$12=="Complete Genome" && $11=="latest"{print $20}' ngo_genomes.txt > ngo_complete_genomes_directories.txt`
4) Edit each this file to make a download.sh file that fetches each file
   Each line will need to have `wget` prepended and '/' plus the name of the genome and `_genomic.fna.gz` appended

   For example
   `ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/845/GCF_000006845.1_ASM684v1`

   will become
   `wget ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/006/845/GCF_000006845.1_ASM684v1/GCF_000006845.1_ASM684v1_genomic.fna.gz`

   This can be achieved as follows:
   `awk 'BEGIN{FS=OFS="/";filesuffix="genomic.fna.gz"}{ftpdir=$0;asm=$10;file=asm"`

```
_"filesuffix;print ftpdir,file}' sau_complete_genomes_directories.txt >
ngo_complete_genomes_directories.txt
cat ngo_complete_genomes_directories.txt | while read line; do echo "wget
$line"; done > download_sau_complete_genomes.sh
```

Run this script using the command `sh download_sau_complete_genomes.sh`

5) Once downloaded, make a kmer sketch of each of the fasta file as follows
   `mash sketch *.fna.gz -o ngo_genomes.msh`
6) Now sketch several of the fastq files from the samples. An example command is
   `mash sketch -m 3 ERR2172273_1.fastq.gz -o ERR2172273`
   This sketches the R1 file and will output the sketch as ERR2172273.msh. The argument -m 3 ensures that only kmers seen three times will be included in the sketch. This is to remove low abundance kmers caused by sequencing errors that do not represent true kmers within the sample genome sequence.
7) The distance between the sample and all the reference genomes can now be calculated using the following command
   `mash dist ERR2172273.msh ngo_genomes.msh | sort -gk3`
   By piping it to sort and sorting on the third column the best match will be reported at the top
8) Steps 6 and 7 can be combined and all fastq files sketched as follows. This is best performed and executed in a small script file e.g sketch_and_dist.sh

```
for f in *_1.fastq.gz; do
  prefix=${f%%_1.fastq.gz}
  mash sketch -m 3 $f -o  $prefix
  mash dist ${prefix}.msh ngo_genomes.msh > ${prefix}.mash.dist.tsv
done
```

9) Once this has finished the results can be summarised using the following two lines
   Combine best matches into a single file

```
for f in *.mash.dist.tsv; do
  cat $f | sort -gk3 | head -1 >> all_mash_dists.tsv
done
```

Print out the best match from each sample and count them

```
awk '{print $2}' all_mash_dists.tsv | sort | uniq -c | sort -g
```

10) Based on the genome which has the best match most frequently, this genome is a good candidate for the reference genome used when mapping.
    Bear in mind however, that if your samples are from across the diversity of a species, in other words not from a single monophyletic lineage, then there may not be a clear 'winner'. In this case it may be best to select a reference that has been widely used in peer-reviewed scientific papers.

Once you have selected a genome uncompress the genome. e.g

```
gunzip ref_genome_seq.fna.gz
```

Check that the sequence does not contain any plasmids by opening the sequence with vi and searching for the fasta header '>'. If there is more than one header select the second one and delete to the end of the file ('dG' in vi).

Using a reference genome of the same sequence type

Look in the literature for a reference genome that matches the ST of the sample or is a single or double locus variant
A quick way to determine the ST of samples is to assemble them and upload them to Pathogenwatch (https://pathogen.watch). The ST will be included in the sample report.

Use a reference genome that is used often in the scientific literature

Search the literature for genomes and their sequence type. The MLST databases at https://pubmlst.org/ can be searched for SLV and DLVs.

In all cases once you have a final uncompressed fasta reference sequence, make sure that there is a single chromosome in the reference sequence (search for the '>' found at the start of a FASTA header line). Remove plasmid contigs if required.

Mapping reads to a reference sequence

1. First the reference sequence must be indexed so that the mapping step is fast. Since we are using the mapper bwa mem, the command will be

   ```
   bwa index reference.fasta
   ```
2. Reads can then be mapped to a reference using bwa mem to produces a sam file. Samtools is then used to make a bam file and sort it.
   These commands can be combined into a single command as follows

   ```
   bwa mem reference.fasta <R1 READ FILE> <R2 READ FILE> | samtools view -bS
   -F 4 - | samtools sort -O bam -o <SAMPLE ID>.sorted.bam -
   ```

   The -F 4 argument in the samtools view command only retains those reads that have mapped to the reference sequence and ensures that unmapped reads are discarded.

3. Variants are called by piping the bam file through bcftools mpileup and then bcftools call. This will produce a bcf file (binary variant call file) which has one row per position in the reference

   ```
   bcftools mpileup -a DP,AD,SP,ADF,ADR,INFO/AD,INFO/ADF,INFO/ADR -f
   reference.fasta <SAMPLE SORTED BAM FILE> | bcftools call --ploidy 1 -m -Ob
   -o <SAMPLE>.bcf
   ```

   This calls variants based on a haploid model and writes out one line per position in the reference genome.
4. This bcf file needs filtering now to label low quality positions with something other than PASS in the 'FILTER' column. We will use the label; 'LowQual'
   The exact filterings are critical. An example of some reasonable quality filter are:
   - %QUAL<25
     quality score assigned by the variant caller
   - FORMAT/DP<10
     minimum depth of coverage

- MAX(FORMAT/ADF)<5 || MAX(FORMAT/ADR)<5
  at least 5 reads in each direction
- MAX(FORMAT/AD)/SUM(FORMAT/DP)<0.9
  The consensus base must be present in 90% of the reads
- MQ<30 || MQ0F>0.1
  Mapping quality scores are assigned by mapper (e.g bwa) - a read that maps in multiple places will cause the score to drop. MQ0F is the proportion of reads that have a mapping score of 0

The command to perform this is:

```
bcftools filter -s LowQual -e'%QUAL<25 || FORMAT/DP<10 ||
MAX(FORMAT/ADF)<5 || MAX(FORMAT/ADR)<5 ||
MAX(FORMAT/AD)/SUM(FORMAT/DP)<0.9 || MQ<30 || MQ0F>0.1' <SAMPLE>.bcf -Ob
-o <SAMPLE>.filtered.bcf
```

5. From this filtered bcf, a pseudogenome can be produced. This is a FASTA file where the sequence is the exact length of the reference. Each base will either match the reference, be an alternative base (i.e a SNP) , a - representing no data (i.e no coverage of that position in the sample) or an N representing a low quality base where the exact base can not be determined.

   This is the only step where there is not open source software readily available to perform the step. A python script is available to make this file
   `filtered_bcf_to_fasta.py -r reference.fasta -b <SAMPLE>.filtered.bcf -o <SAMPLE>.fas`
6. Once the pseudogenome has been produced, all genomes can be concatenated into a single whole genome FASTA alignment

   ```
   cat *.fas > aligned_pseudogenome.fas
   ```

Phylogenetic Tree generation

To produce a tree from the whole genome alignment the aligned FASTA file should be fed into a tree-calculation algorithm. For a simple quick algorithm, a neighbour-joining algorithm can be used. For example use fasttree and the -noml option.
A more robust but slower algorithm is Maximum Likelihood using software such as RAxML or IQTREE.

An example of a command to produced a bootstrapped phylogeny is

```
iqtree -s aligned_pseudogenome.fas -m GTR+G -alrt 1000 -bb 1000 -nt AUTO -ntmax
4
```

In this command a GTR + gamma model is used to model the evolution of the sequence and 1000 bootstraps are performed to give confidence values to the branched observed in the final consensus tree. The arguments `-nt AUTO -ntmax 4` optimize the number of threads used to speed up the process up to a maximum of 4 (equivalent to 4 CPUs).

On a whole genome alignment this process can be very slow and use a lot of RAM memory. Therefore it is recommended that first of all the alignment is reduced to only those positions where there is variation since

those positions that are invariant will contain no information for tree generation. The snp-sites software can be used to achieve this. For example

```
snp-sites aligned_pseudogenome -o aligned_pseudogenome.variants_only.fas
```

For organisms where recombination is an issue, it is often best to remove recombing regions from the genome since they will result in artificially long branch lengths. This can be achieved using a program called gubbins but only if all the samples are from a single lineage. Trying to do this on samples from across the diversity of a species will either be incredibly slow or result in much of the genome being removed from the alignment. However when dealing with samples from just a single lineage (e.g single sequence type), gubbins can be applied as follows:

```
run_gubbins.py -t hybrid aligned_pseudogenome.fas -p aligned_pseudogenome
snp-sites aligned_pseudogenome.filtered_polymorphic_sites.fasta -o
aligned_pseudogenome.gubbins.variants_only.fas
```

Step by Step tutorial

We will use samples from this paper describing sustained transmission of high-level azithromycin-resistant *Neisseria gonorrhoeae* in England.

Requirements

To install the software for this the easiest way is via conda. On a Linux machine this can be accomplished as follows

Install conda

```
1. wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
2. chmod +x Miniconda3-latest-Linux-x86_64.sh
3. ./Miniconda3-latest-Linux-x86_64.sh
```

Accept the licence and all the defaults offered during the installation process

Add default channels to install software from via conda

```
1. conda config --add channels defaults
2. conda config --add channels bioconda
3. conda config --add channels conda-forge
```

Install the software and a python script for pseudogene creation

```
1. conda install mash bwa bcftools samtools gubbins snp-sites iqtree pysam
2. wget https://gitlab.com/cgps/ghru/pipelines/snp_phylogeny/raw/master/bin/filtered_bcf_to_fasta.py
3. chmod +x filtered_bcf_to_fasta.py
4. sudo mv filtered_bcf_to_fasta.py /usr/local/bin/
```

Single sample

1. Download the paired fastq raw read files derived from a single sample included in the study

   ```
   wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR217/003/ERR2172273/ERR2172273_1.fastq.gz
   wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR217/003/ERR2172273/ERR2172273_2.fastq.gz
   ```

2. To determine which genomes will Make a mash sketch for all complete genomes using the steps described above

   Sketch one of the fastq files

   ```
   mash sketch -m 2 ERR2172273_1.fastq.gz -o ERR2172273
   ```

-m 2 excluded low frequency kmers produced from sequencing errors, -o just makes sure the sketch name is a bit prettier. It will be ERR217273.msh

FInd the distance between this sketch and the reference genomes sketch made earlier.

```
mash dist ERR2172273.msh ngo_genomes.msh | sort -gk3 | head
```

This command calculates the distances between the sample ERR2172273 and the reference genomes, sorts by the 3rd column which contains the distances and shows the first few lines (using head).

The top hit is

GCF_900186935.1_49677_G01_genomic.fna.gz

The identity of this genome can be seen using

```
zcat GCF_900186935.1_49677_G01_genomic.fna.gz | less
```

From the header line it can be seen that this genome is from sample from the National Collection of Type Cultures from Public Health England: NCTC13799

Details for this sample can be found here. You will see that it is a sample possessing high level resistance to Azithromycin from England, so it makes sense that this is the closest match and confirms the methodology.

3. Now that a suitable reference genome has been found, it should be uncompressed and an index made for the mapper.

```
gunzip GCF_900186935.1_49677_G01_genomic.fna.gz -c > NCTC13799.fna
bwa index NCTC13799.fna -p NCTC13799
```

This will make several files that make up the index that bwa will use.

4. The reads can now be aligned to the reference genome using the mapper bwa

```
bwa mem NCTC13799 ERR2172273_1.fastq.gz ERR2172273_2.fastq.gz >
ERR2172273.sam
```

This produces a sam file but it is more efficient to work with a compressed binary sam file with only mapped reads. This can be made as follows:

```
samtools view -bS -F 4 ERR2172273.sam > ERR2172273.bam
```

Following this most downstream processes will require a sorted bam file where the mapped reads are ordered with respect to the reference genome.

```
samtools sort ERR2172273.bam -O bam -o ERR2172273.sorted.bam
```

All the commands can be combined into a single line to speed up the process and save on disk space since intermediate files are not saved.

```
bwa mem NCTC13799 ERR2172273_1.fastq.gz ERR2172273_2.fastq.gz | samtools
view -bS -F 4 - | samtools sort -o ERR2172273.sorted.bam -
```

5. Once the reads have been mapped, variants are called by initially converting to VCF format. This translates the mapped reads to a format where the bases from reads at each position in the reference genome are encoded in a single row. Originally this used a format called pileup, see here for a description. This has now been superseded by the Variant Call Format (VCF) but the bcftools command still refers to the older format. The -a argument specifies what fields each VCF row is annotated with. The description of these can be found in the VCF specification document

```
bcftools mpileup -a DP,AD,SP,ADF,ADR,INFO/AD,INFO/ADF,INFO/ADR -f
NCTC13799.fna ERR2172273.sorted.bam > ERR2172273.vcf
```

This VCF file will not include a genotype call or a quality score for this genotype. To call variants and populate a field named GT (genotype) the bcftools call command is used specifying a haploid genome and a compressed binary output (bcf).

```
bcftools call --ploidy 1 -m -Ob -o ERR2172273.bcf ERR2172273.vcf
```

As with the previous mapping command this can all be achieved in a single line to avoid the unnecessary intermediate file

```
bcftools mpileup -a DP,AD,SP,ADF,ADR,INFO/AD,INFO/ADF,INFO/ADR -f
NCTC13799.fna ERR2172273.sorted.bam | bcftools call --ploidy 1 -m -Ob -o
ERR2172273.bcf
```

The contents of the binary file can be viewed for learning or debugging purposes using the command

```
bcftools view ERR2172273.bcf | less
```

6. As described in the mapping section above the variants called within this VCF file must be filtered to leave only high quality variants

```
bcftools filter -s LowQual -e'%QUAL<25 || FORMAT/DP<10 ||
MAX(FORMAT/ADF)<5 || MAX(FORMAT/ADR)<5 ||
MAX(FORMAT/AD)/SUM(FORMAT/DP)<0.9 || MQ<30 || MQ0F>0.1' ERR2172273.bcf -Ob
-o ERR2172273.filtered.bcf
```

This will produce another binary VCF file that has one of two entries in the FILTER column for each row representing a position in the reference genome. This will be either PASS or LowQual.
As previously the output can be viewed if desired using the following command

```
bcftools view ERR2172273.filtered.bcf | less
```

7. The VCF file can not be used to generate a phylogeny since a multiple fasta file is required as an input. A simple Python script can convert the VCF file to a pseudogenome sequence.

   Run the script specifying the reference sequence and filtered bcf file as inputs using the -r and and -b arguments respectively and the output file as sample name and the fas extension using the -o argument

   ```
   filtered_bcf_to_fasta.py -r NCTC13799.fna -b ERR2172273.filtered.bcf -o
   ERR2172273.fas
   ```

Multiple samples

In order to convert this into a phylogenetic tree, at least 3 samples are needed and they need to be combined into a multiple alignment file for input into a tree drawing algorithm. Obviously doing this for a large number of samples by hand would be tedious so again we can use the power of the Nextflow workflow executor to perform the multiple tasks for each sample.

Requirements

For this part of the tutorial you will need a Linux server with Nextflow and Docker installed.
The docker image used can be obtained by typing the command

```
docker pull bioinformant/ghru-snp-phylogeny:1.0
```

Running Nextflow

1. Download the nextflow files necessary to run the pipeline and extract them to a directory called snp_phylogeny_nextflow_files

   ```
   wget -qO-
   https://gitlab.com/cgps/ghru/pipelines/snp_phylogeny/-/archive/master/snp_phylogeny-m
   aster.tar.gz  | tar xvz --one-top-level=snp_phylogeny_nextflow_files
   --strip-components 1
   ```

2. Fetch the input files for the pipeline.

   ```
   wget https://snp_phylogeny_tutorial.cog.sanger.ac.uk/accessions.txt
   wget https://snp_phylogeny_tutorial.cog.sanger.ac.uk/ngo_reference.fna
   ```

   These are the a list of accession numbers from the paper described above and the NCTC13799 reference genome

3. It is possible to run this nextflow script as a standalone script or using a small utility script to make this a less verbose command.
   To run these files as using 'vanilla' nextflow, the command would be as follows:

```
nextflow run snp_phylogeny_nextflow_files/snp_phylogeny.nf \
--accession_number_file accessions.txt \
--reference ngo_reference.fna \
--output_dir snp_phylogeny_output \
--adapter_file snp_phylogeny_nextflow_files/adapters.fas \
--depth_cutoff 100 \
--remove_recombination \
--tree \
-resume -ansi -profile standard
```

Stepping through each of these commands. First off, please note that the `\` characters are just ways to write a single command line over several lines so that it is easier to read. It would work just as well on a single line without the \ characters

The first line specifies to run the nextflow workflow found in the `snp_phylogeny_nextflow_files` directory (downloaded and extracted in step 1) and named `snp_phylogeny.nf`

The second line specifies that the fastqs will be downloaded from the short read archive and points to a file called `accessions.txt` that is a simple text file with one accession file per line.
If the fastqs were instead found locally, this would be specified using
`--input_dir <PATH TO INPUT DIR> --fastq_pattern <PATTERN TO MATCH FASTQS>`
for example
`--input_dir fastqs --fastq_pattern "*_R{1,2}.fastq.gz"`

The third line specifies where to find the fasta format genome reference file

The fourth line specifies the path to the output directory

The fifth line specifies where to find a file containing adapter sequences for the trimming step

The sixth line specifies to downsample reads to 100x coverage

The seventh and eighth line are optional.

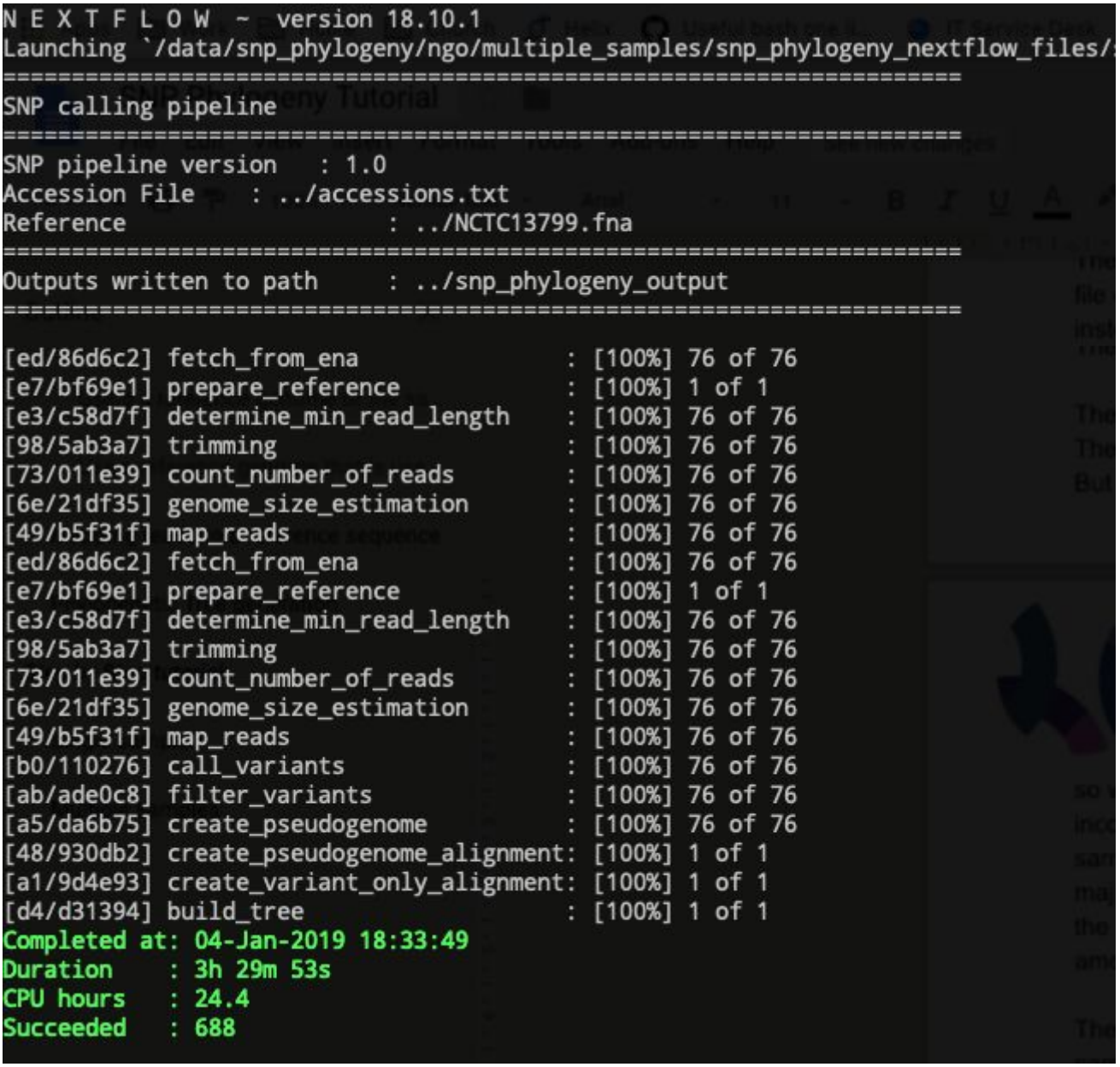
The seventh line will use gubbins to remove recombination. This is not always necessary or desirable. But in this case we are analysing *Neisseria gonorrhoeae* and this species is highly recombinogenic. Therefore without recombination removal, the genetic distances reported on phylogeny creation may be incorrect dues to horizontal gene transfer and recombination artificially increasing the distance between samples. In addition, it is appropriate to remove recombination since the samples are all from one major clonal lineage. If you were trying to remove recombination from samples spanning the diversity of the species the process of recombination detection and removal may be intractable, taking a very long time to complete and result in the removal of large amounts of informative variation.

The eight line line will remove invariant positions using snp-sites. These are sites that are identical in all samples including the reference and therefore will not be informative in terms of phylogenetic distance

The ninth lines are not workflow specific but tell nextflow to resume if the workflow was interrupted and to output progress using ansi (regularly updated info text rather than scrolling progress). The profile argument tells nextflow to run locally rather than in the cloud using the standard profile found in the

config file (called `nextflow.config` and found in the `snp_phylogeny_nextflow_files` directory downloaded and extracted in step 1).

When you run this you should see output as follows



4. This is quite a complex command and this can be simplified using a small bash wrapper script which is again found in the `snp_phylogeny_nextflow_files directory` downloaded and extracted in step 1.

This bash script makes an assumption.

The input, output and work directories will all be in a single parent directory specified with `-p`

The equivalent to the nextflow command in step 3. is

```
snp_phylogeny_nextflow_files/run_snp_phylogeny.sh \
-w snp_phylogeny_nextflow_files \
-a accessions.txt \
-p . \
-r NCTC13799.fna \
```

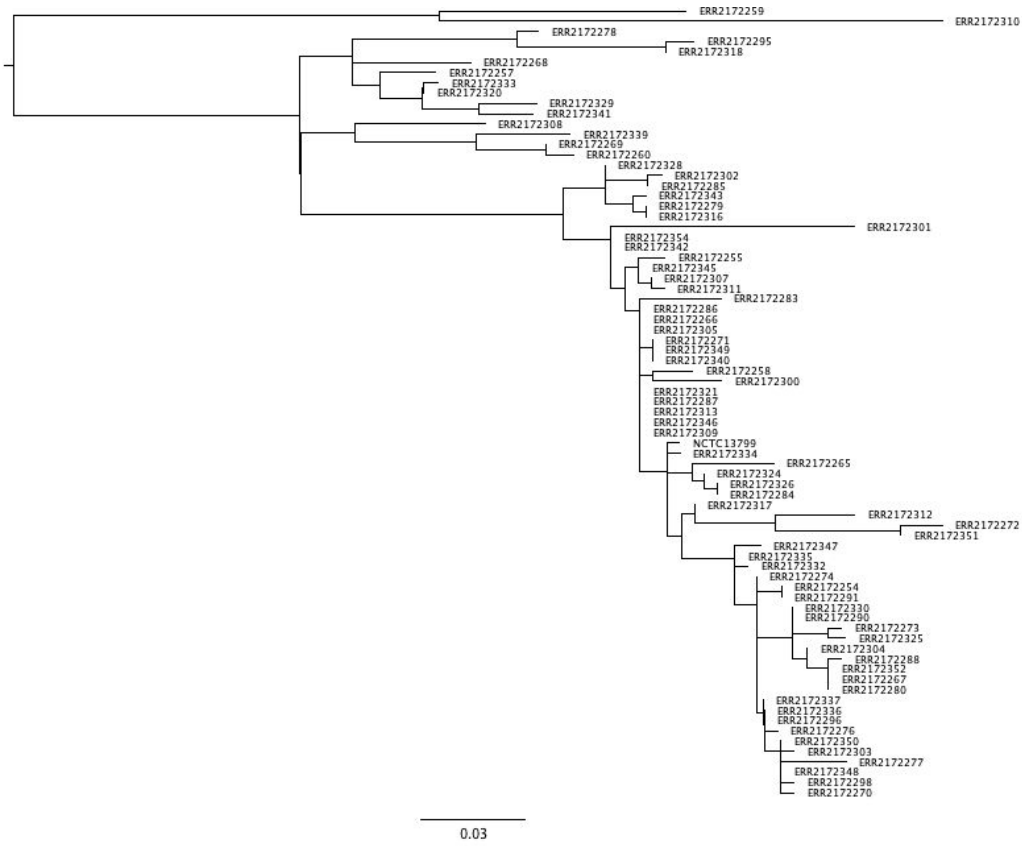
```
-rr \
-t
```

`-w` specifies where to find the nextflow nf and config files
`-a` specifies where to find list of accession numbers. Without this it would assume that the fastq files are found in a directory called `fastqs` within the parent directory (specified by `-p`) with a fastq pattern of `*{R,_}{1,2}*.fastq.gz`. This will recognise bot the R1.fastq.gz and _1.fastq formats but if you have something different this can be overridden using `-fq`
`-p` specifies a parent directory where outputs will be written into a subdirectory called `snp_phylogeny_output` and temporary files into a subdirectory called `work`
`-r` specifies where to find the reference sequence
`-rr` specifies the `--remove_recombination` nextflow command
`-t` specifies the `--tree` nextflow command

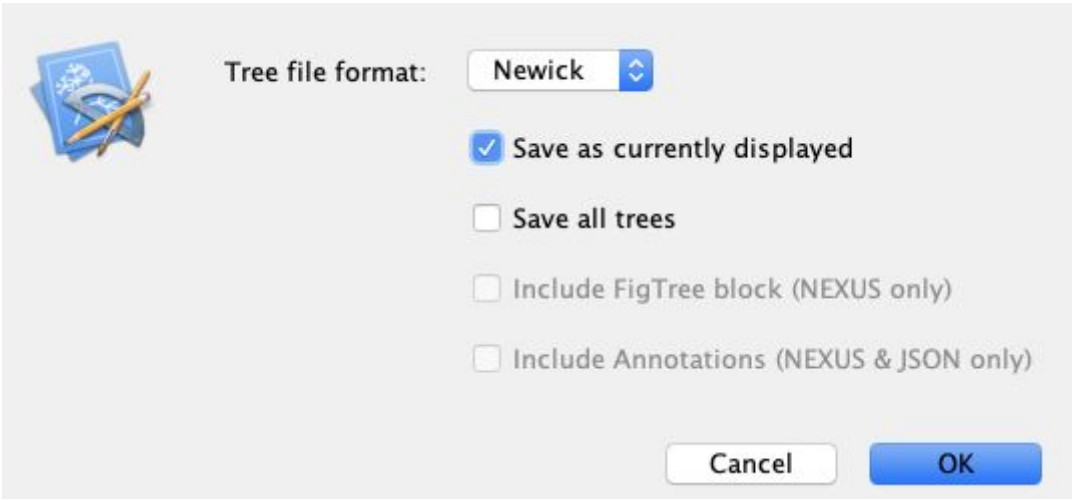
If you append the `-d` argument, this will result in a dry run where nextflow is not run but instead the full nextflow command that would have been run is displayed. This can be useful for debugging.

5. The final output from this workflow will be a consensus tree named `aligned_pseudogenome.gubbins.variants_only.contree`. This is a bootstrapped consensus tree where the values at the internal tree nodes represent a value for the statistical confidence for each internal branch. This is generated by resampling the tree a 1000 times and the number is the percentage of times the branch point in the consensus tree is observed in the 1000 resamples. The higher the number the more confidence in that branch point.
This tree is in newick format which can be viewed by several tree viewing packages such as FigTree or Dendroscope. Download the newick tree file, install one of these programs and open the file from the program. In both cases you should make visualisation of the tree easier to interpret by mid-point rooting the tree and arranging the leaf nodes in descending order.
Options for this can be found in the Tree menu in FigTree and the Edit and Layout (Ladderise Right) menus in Dendroscope.

   Once you have the tree visualised in this way it should appear as follows



Once you have the tree in this view save it in this view by exporting the tree in Newick format. In FigTree ensure you check the box to save the current view



6. This tree can be combined with metadata in and visualised on the website https://microreact.org . A CSV file in the required format, where the leaf names match the id column in the file can be downloaded from here.

   If you do not have an account already sign up on the microreact website and upload the tree you have generated and the metadata CSV file. You may see an error saying the leaf names in the newick tree and the metadata CSV do not match. Ini this case you will need to add a line for the sample in the tree that is missing from the CSV file. This will be the reference sequence for which we have no metadata data. A single row in the metdadata matching the name in the id column will be sufficient.

   You should see a tree that appears similar to the one shown in the link below

https://microreact.org/project/HL_AziR_GC/d66f9203

Version Control Table

Title	Reference-based SNP phylogeny			
Description	A document describing how map multiple samples to a reference genome and optionally remove recombination and/or generate a maximum likelihood tree			
Created By	Anthony Underwood			
Date Created	7th January 2019			
Maintained By	Anthony Underwood			
Version Number	**Modified By**	**Modifications Made**	**Date Modified**	**Status**
1.0	Anthony Underwood	First version	7th January 2019	First Live Version