1 Phylogeny and Mapping

1.1 Introduction

Phylogenetics is essentially about similarity, and looking at patterns of similarity between taxa to infer their relationships. It has important applications in many fields of genome biology. For example, when annotating a gene in a new genome it is useful for identifying previously-annotated genes in other genomes that share a common ancestry. It is also becoming increasingly common to use phylogeny to trace the evolution and spread of bacterial diseases, and even as an epidemiological tool to help identify disease outbreaks in a clinical setting. Further analysis of genome sequences to examine recombination, molecular adaptation and the evolution of gene function, all benefit from phylogeny.

1.2 Learning outcomes

On completion of the tutorial, you can expect to be able to:

- Identify different approaches to constructing phylogeny from whole genome sequence data
- Map sequence data to a reference genome and identify variants
- Use SNP data to construct a phylogenetic tree
- Identify and remove recombination with Gubbins
- Visualise phylogenies in the context of the sample metadata

1.3 Tutorial sections

This tutorial comprises the following sections:

1. Introduction to phylogenetics
2. Phylogeny from gene sequences
3. Phylogeny from whole genome sequence data
4. Phylogeny and metadata

1.4 Authors and License

This tutorial was written by Jacqui Keane.

Some of the material has been adapted from the Wellcome Connecting Science Courses AMR-ASIA23, GenEpiLAC2023 and WWPG21.

1.5 Running the commands in this tutorial

You can follow this tutorial by typing all the commands you see in a terminal window on your computer. Remember, the terminal window is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

To get started, open a terminal window and type the command below followed by the `Enter` key:

```
cd ~/course_data/snp_phylogeny/data
```

1.6 Prerequisites

This tutorial assumes that you have the following software and their dependencies installed on your computer. The software used in this tutorial may be updated from time to time so, we have also given you the version which was used when writing this tutorial.

Package name	Link for download/installation instructions	Version
seaview	https://doua.prabi.fr/software/seaview	5.0.5
fastqc	https://www.bioinformatics.babraham.ac.uk/projects/fastqc	0.12.1
fastp	https://github.com/OpenGene/fastp	0.23.4
bwa	https://github.com/lh3/bwa	0.7.17
samtools	https://github.com/samtools/samtools	1.17
bcftools	https://github.com/samtools/bcftools	1.17
snp-sites	https://github.com/sanger-pathogens/snp-sites	2.5.1
gubbins	https://github.com/nickjcroucher/gubbins	3.3.0
iqtree	http://www.iqtree.org/	2.2.3
FigTree	http://tree.bio.ed.ac.uk/software/figtree/	1.4.4
Microreact	https://microreact.org	240

The easiest way to install the required software is using `conda`, a software package manager. These software have already been installed on the computer for you. To activate them type:

```
conda activate snp-phylogeny
```

After the software is activated type the following commands:

```
seaview &
```

```
fastqc -h
```

```
fastp -h
```

```
bwa
```

```
snp-sites -h
```

```
run_gubbins.py -h
```

```
iqtree -h
```

```
figtree &
```

This should return the help message for these tools or launch the GUI software in the background.

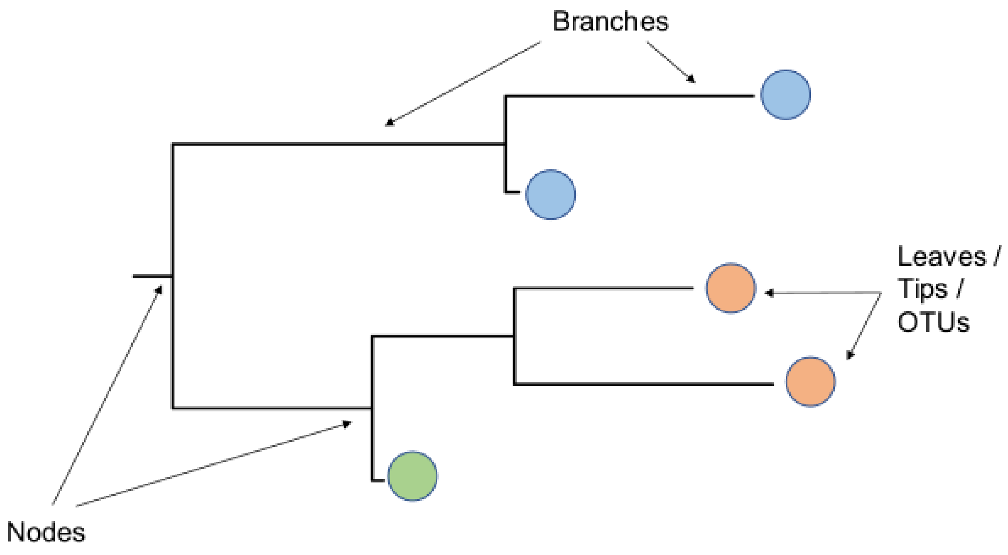
To get started with the tutorial, go to the first section: Introduction to phylogeny

2 Introduction to Phylogeny

Before we describe how to construct a phylogenetic tree, we will introduce some important concepts related to phylogenies.

2.1 What is a phylogentic tree?

A phylogeny, also known as a phylogenetic tree, depicts estimated evolutionary relationships between **taxa** - these can be species, strains or even genes. It consists of *nodes* and *branches*.



2.1.1 Nodes

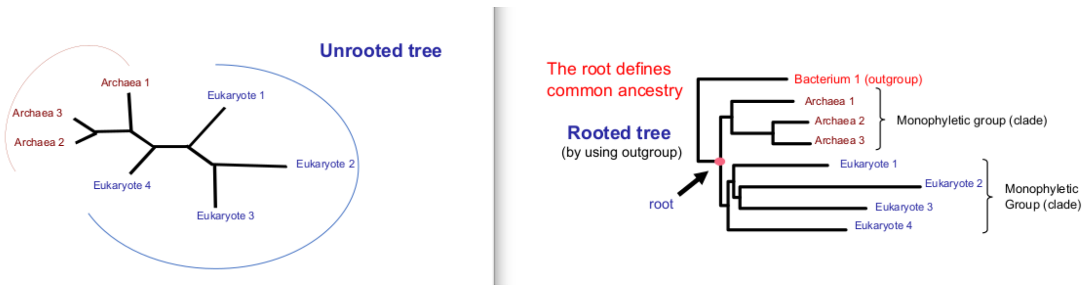
There are two types of *nodes* in a phylogenetic tree; external nodes, also called 'tips' or 'leaves' of the tree (i.e. taxa), and internal nodes which denote their hypothetical ancestors.

2.1.2 Branches

The *branches* in a tree are shown as lines connecting the nodes. They show the path of transmission of genetic information from one generation to the next. Branch lengths indicate genetic change i.e. the longer the branch, the more genetic change (or divergence) has occurred.

2.1.3 Root

A phylogenetic tree can be unrooted or rooted as shown below.



The *root* of a phylogenetic tree is the most recent common ancestor of all of the taxa in the tree. It is therefore the oldest part of the tree and tells us the direction of evolution, with the flow of genetic information moving from the root, towards the tips with each successive generation. Deciding upon an appropriate root position is critical for phylogenetic interpretation because the root tells us the direction of evolution and so affects statements that we make about patterns of relatedness.

There are two main approaches that can be used to root a tree, midpoint rooting and outgroup rooting. We will describe these approaches in more detail later in the tutorial.

2.1.4 Topology

The *topology* is the branching structure of the tree and the same topology can be drawn in lots of different ways. It is of particular biological significance because it indicates patterns of relatedness among taxa, meaning that trees with the same topology and root have the same biological interpretation.

2.2 Microbial phylogenetic trees

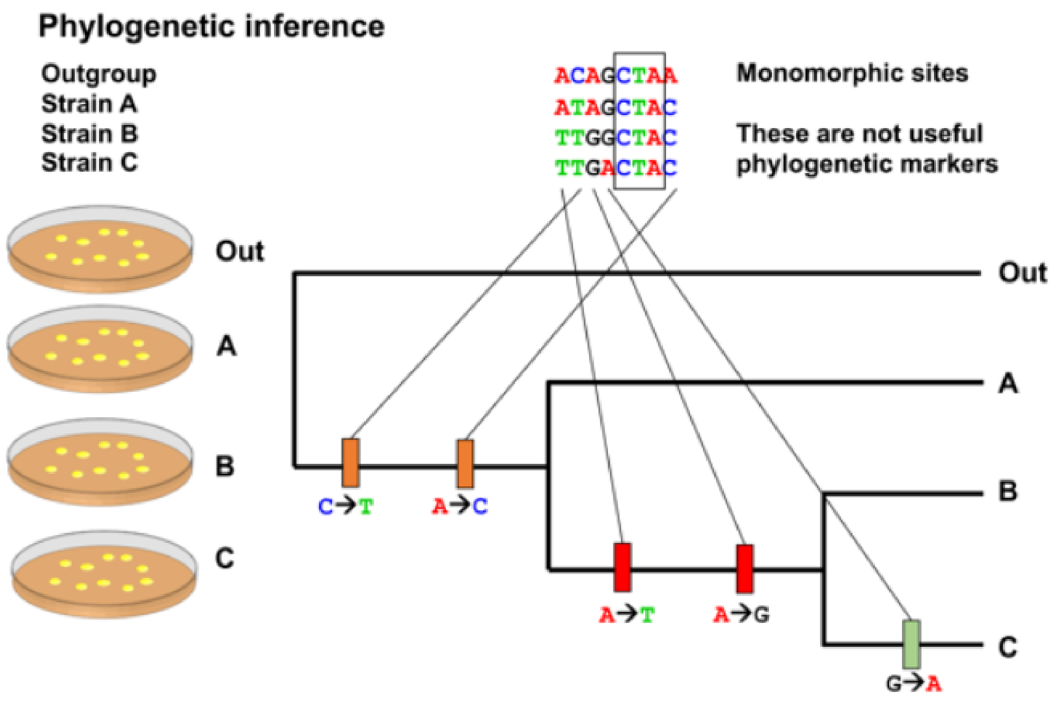
In the context of infectious diseases epidemiology, phylogenetic trees are commonly used to define evolutionary relationships between strains of the same bacterial species. This is possible because bacteria reproduce clonally. During clonal reproduction, bacterial progenitor cells replicate their DNA at high fidelity. Despite this, random errors in DNA replication may still occur, resulting in a clonal progeny that will inherit these genetic replication ‘errors’ (i.e. mutations) in their DNA and may not be strictly identical to their progenitor cells. Bacterial strains that have recently originated from the same progenitor cell are thus expected to share identical genomes or have diverged at most by only a few genetic differences (mutations). The number and pattern of shared mutations between bacterial strains can be used to reconstruct their genealogical and evolutionary relationships.

On a phylogenetic tree, isolated bacterial strains are depicted on the *leaves* of the tree, whereas the internal nodes of the tree denote their hypothetical ancestors. Groups of bacterial strains (taxa) that share the same common ancestor form a *monophyletic* group (also known as *clade*). A group of strains that descends from a common ancestor, but does not include all descendants, is called *paraphyletic*.

2.3 How are phylogenetic trees reconstructed?

Today almost all phylogenetic trees are inferred from molecular sequence data, most often from DNA sequences. This is because DNA is an inherited material; it can easily, reliably and inexpensively be extracted and sequenced; and DNA sequences are highly specific to bacterial species and strains.

The goal is to generate a multiple sequence alignment of the DNA sequences of the bacterial species or strains and use this to re-construct the phylogenetic tree. Polymorphic sites (that is, nucleotide positions that are variable across multiple strains) in multiple alignments are used to infer evolutionary relationships, whereas monomorphic sites (nucleotide positions with the same DNA base) are generally ignored. The figure below shows an example of a simple multiple alignment of eight sites from four strains, which include monomorphic (squared) and polymorphic sites. Genetic changes in the phylogenetic tree are showed as coloured vertical rectangles on the branch where they originated. The identification of genetic changes (alleles) that are unique and common to multiple taxa (strains) are used to group them into monophyletic groups (clades) in a hierarchical manner with the goal of constructing the most plausible genealogical relationships between strains and clades.



There are many computational methods for reconstructing a tree from sequence data, e.g. Parsimony, Maximum Likelihood. Most methods require a model of evolution (substitution model) to provide information on how the sequences evolved. We won't expand on this here as it is beyond the scope of this tutorial.

2.4 Data selection for phylogenies

The type of sequence data that is used to construct a phylogenetic tree can vary. A tree can be constructed using sequence data from:

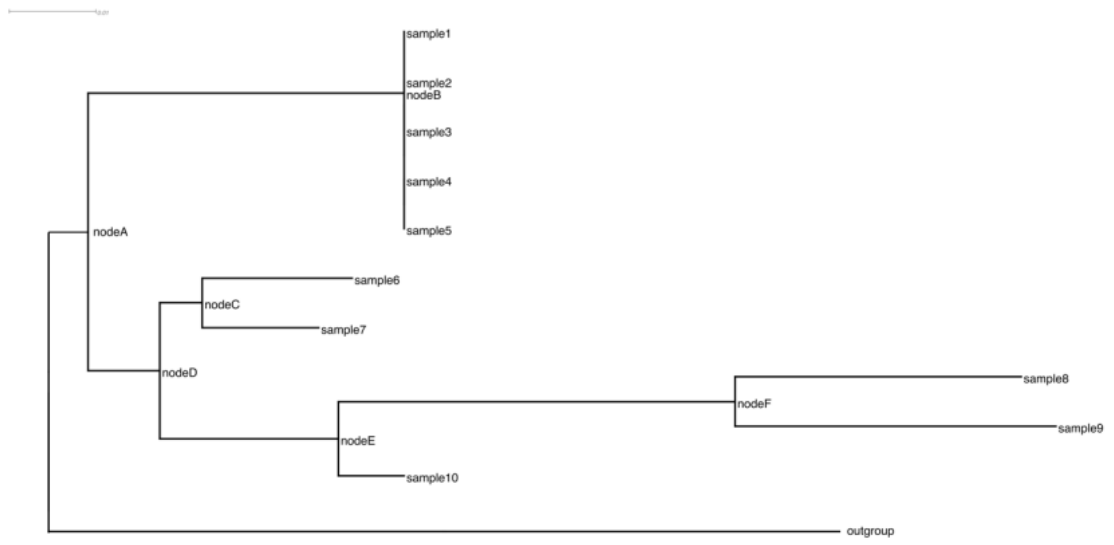
- a single gene
- Multi-locus sequence typing (MLST) data: a set of housekeeping genes for a species
- Core genome MLST (cgMLST) data: an expanded MLST scheme using all genes shared by all strains within a species
- Whole genome MLST (wgMLST) data: an expanded cgMLST scheme to include a selection of accessory genes

- entire genome reconstructed from the whole genome sequence data

The sequences to include and the type of data that you choose when constructing your phylogenetic tree will depend on your objective.

2.5 Exercises

Look at the following tree and answer the questions that follow:



1. How many nodes are in the tree?
2. What internal node corresponds to the most recent common ancestor of samples 8 and 10
3. Which term best describes a group that share the same common ancestor?

a. Polyphyletic
b. Monophyletic
c. Paraphyletic

Now let's construct our first phylogeny: Phylogeny from gene sequences

3 Phylogeny from gene sequences

For our first phylogeny we will make a tree using the sequences of the *ompA* gene from 16 strains of *C. trachomatis*. First, let's check that you're in the right place. Type the command below in the terminal window followed by the Enter key:

```
pwd
```

It should display something like:

```
/home/manager/course_data/snp_phlogeny/data
```

The gene sequences for the 16 strains are found in a multifasta file called `ompA.fa` in the `gene` directory. These nucleotide sequences have been constructed from sequence data of the 16 strains. We won't cover how the genes were constructed here but will learn more about this in a later tutorial.

Take a look at the file containing the gene sequences:

```
cd gene
```

```
cat ompA.fa
```

3.1 Introducing the dataset

The dataset used in this section has in part been derived from the following paper:

> **The Swedish new variant of Chlamydia trachomatis: genome sequence, morphology, cell tropism and phenotypic characterization**
> Unemo M, Seth-Smith HMB, Cutcliffe LT, Skilton RJ, Barlow D, Goulding D, Persson K, Harris SR, Kelly A, Bjartling C, Fredlund H, Olcén P, Thomson NR, Clarke IN. *Microbiology 2010. doi: 10.1099/mic.0.036830-0* PMID: 20093289

Chlamydia trachomatis is one of the most prevalent human pathogens in the world, causing a variety of infections. It is the leading cause of sexually transmitted infections (STIs), with an estimated 131 million new cases each year. Additionally, it is also the leading cause of preventable infectious blindness with tens of millions of people thought to have active disease.

Historically, the most commonly used tool for typing *C. trachomatis* isolates was serotyping using the MOMP (major outer membrane protein), which is encoded by the *ompA* gene. There are two biovars of C. trachomatis:

- the *trachoma* biovar includes ocular and urogenital strains, which cause the majority of trachoma and STIs, and are characterised by localised infections of the epithelial surface of the conjunctiva or genital mucosa;
- the *lymphogranuloma venereum (LGV)* biovar includes strains which are distinguished by their ability to spread systemically thorough the lymphatic system, causing genital ulceration and bubonic disease.

Based on MOMP serotyping, *C. trachomatis* has been subdivided into between 15 and 19 serotypes: the *trachoma* biovar includes ocular serotypes A to C and urogenital serotypes D to K, while the *LGV* biovar includes serotypes L1, L2 (including L2a, b and c) and L3.

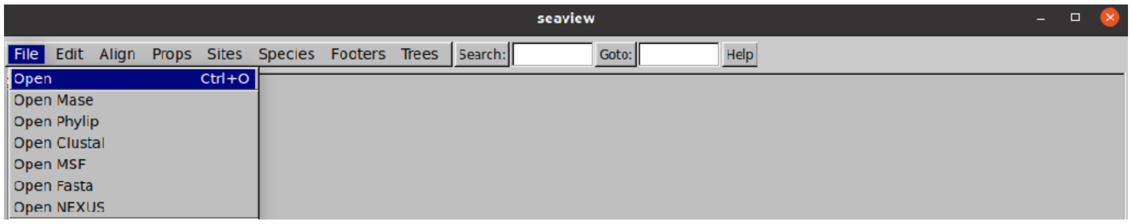
In 2006, a new variant of *C. trachomatis*, known as NV was reported in Sweden and triggered a European health alert. During this time it became the dominant strain circulating in some European countries and began to spread world wide. The reason for this was that it evaded detection by the widely used PCR-based diagnostic test. During this exercise we will place this New Variant of *C. trachomatis* (NV) in the context of several other *C.trachomatis* strains based on the ompA gene.

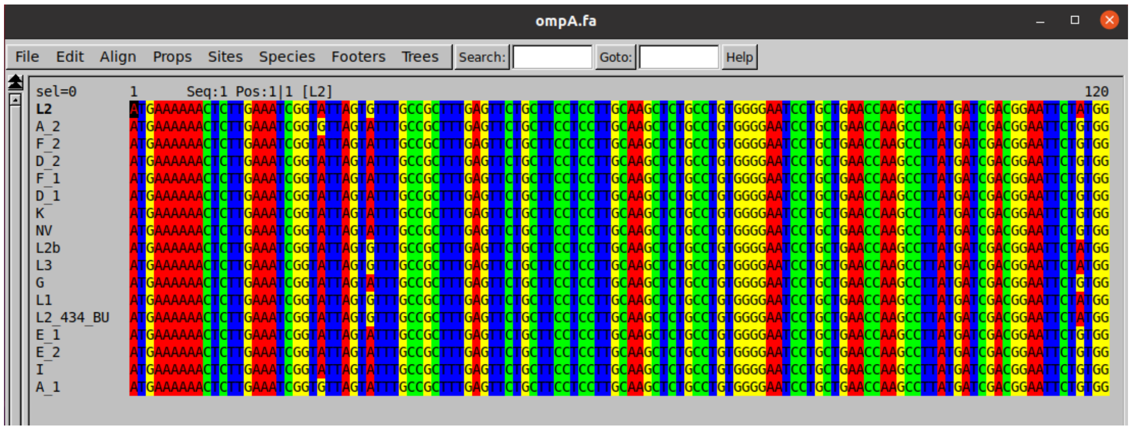
3.2 Viewing the alignment in Seaview

To view our gene sequences and produce a phylogeny we will use a program called `Seaview`. `Seaview` is a graphical user interface (GUI) that combines a number of the most popular alignment and phylogeny programs. Launch `Seaview`

```
seaview &
```

Now load the file `ompA.fa` by selecting 'File' and 'Open' from the main menu.





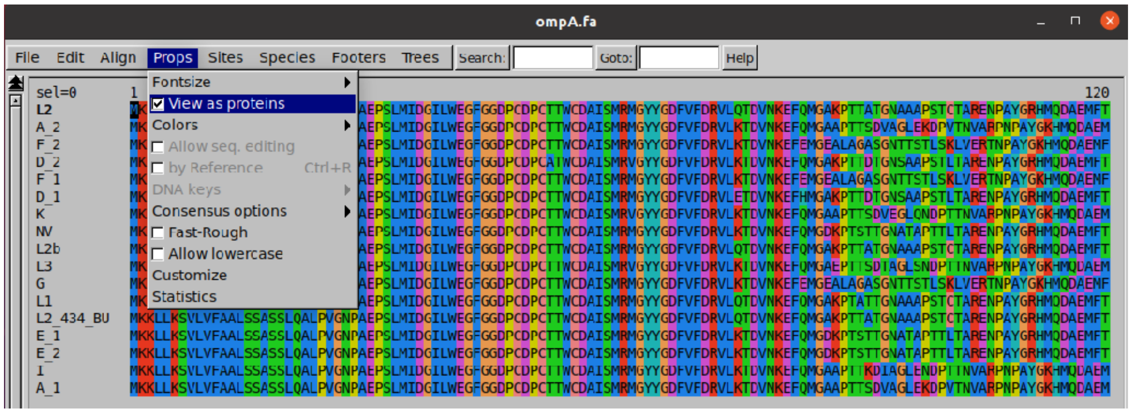
3.3 Multiple sequence alignment

Homology among DNA, RNA, or proteins is typically inferred from their nucleotide or amino acid sequence similarity. Significant similarity is strong evidence that two sequences are related by evolutionary changes from a common ancestral sequence. Alignments of multiple sequences are used to indicate which regions of each sequence are homologous.

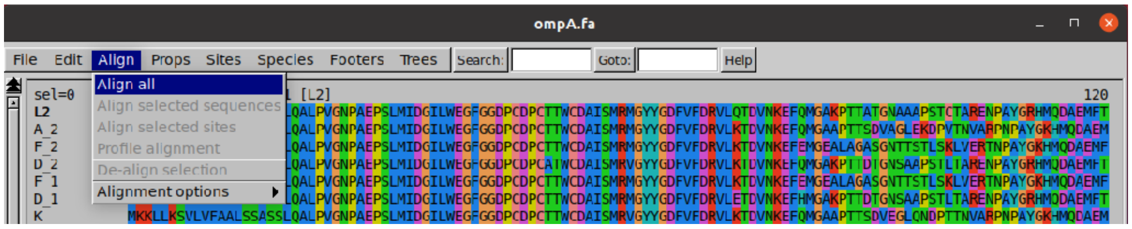
Therefore before we perform any phylogenetic analysis, we must make sure that the columns in our data represent homologous bases. With gene or protein sequence data, this usually means aligning the nucleotide or amino acid sequences using a multiple alignment program. Length differences of the sequences complicate multiple sequence alignment because these require the insertion of gaps into an alignment to ensure that homologous sites remain aligned. When possible, alignments should be checked by eye.

`Seaview` allows alignment using two programs, `clustal` and `muscle`. Generally `muscle` is faster, providing protein alignments that are of similar quality to `clustal`. It is usually better to align genes after translating them into amino acids, so we will do that here.

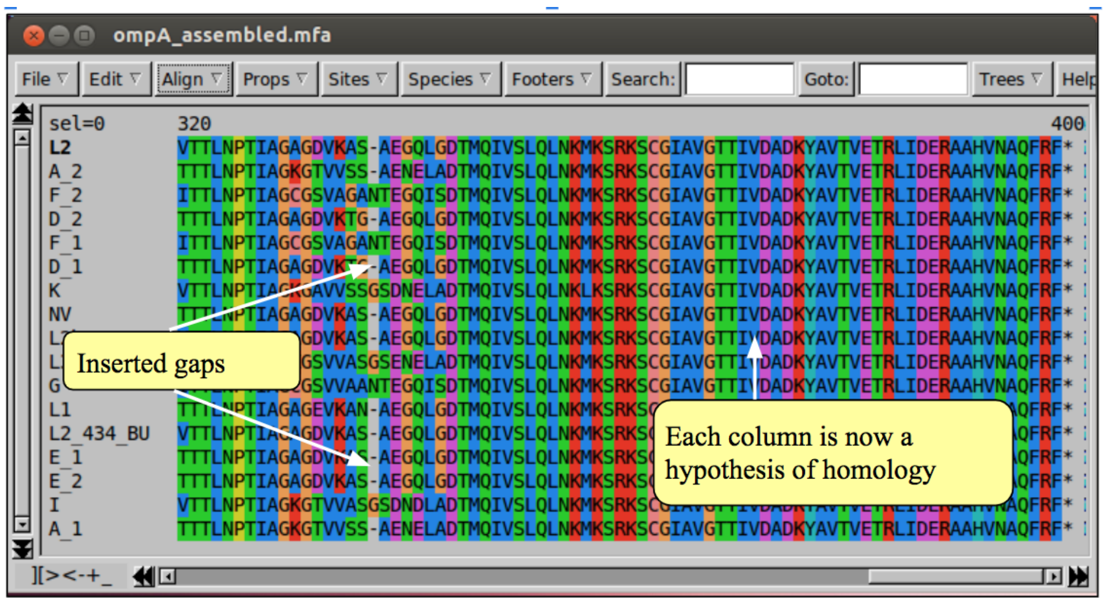
On the 'Props' menu tick 'View as proteins'.



To run the alignment, select 'Align' then 'Align all' from the main menu.

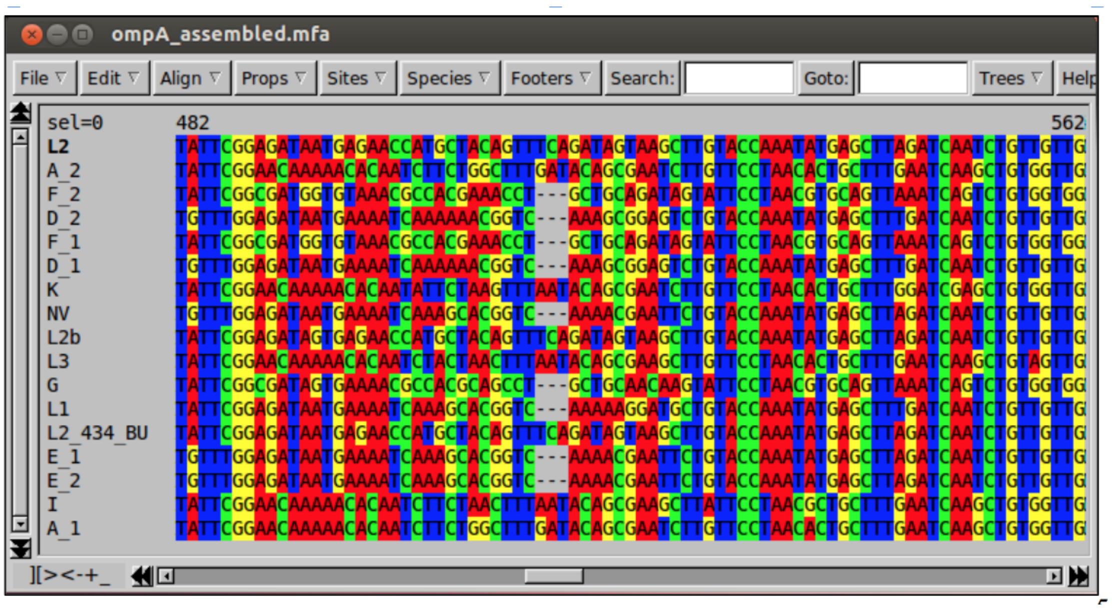


When the alignment process is complete, `Seaview` will have inserted gaps into the sequences so that homologous sites (or at least homologous according to the alignment program) are lined up in columns. Look at the alignment, it should now be clearer how the sequences differ from one another.



```
sel=0      320                                                                           400
L2         VTTLNPTIAGAGDVKAS-AEGQLGDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
A_2        TTTLNPTIAGKGTVVSS-AENELADTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
F_2        ITTLNPTIAGCGSVAGANTEGQISDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
D_2        TTTLNPTIAGAGDVKTG-AEGQLGDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
F_1        ITTLNPTIAGCGSVAGANTEGQISDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
D_1        TTTLNPTIAGAGDVK[illegible]-AEGQLGDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
K          VTTLNPTIA[illegible]GAVVSSGSDNELADTMQIVSLQLNKLKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
NV         TTTLNPTIAGAGDVKAS-AEGQLGDTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
L[illegible]         [illegible]GDVKAS-AEGQLGDTMQIVSLQLNKMKSRKSCGIAVGTTI[illegible]DADKYAVTVETRLIDERAAHVNAQFRF*
L[illegible]         [illegible]GSVVASGSENELADTMQIVSLQLNKMKSRKSCGIAVGTTI[illegible]DADKYAVTVETRLIDERAAHVNAQFRF*
G          [illegible]GSVVAANTEGQISDTMQIVSLQLNKMKSRKSCGIAVGTTI[illegible]DADKYAVTVETRLIDERAAHVNAQFRF*
L1         TT[illegible]NPTIAGAGEVKAN-AEGQLGDTMQIVSLQLNKMKSRKS[illegible]AQFRF*
L2_434_BU  VTTLNPTIA[illegible]AGDVKAS-AEGQLGDTMQIVSLQLNKMKSRKS[illegible]QFRF*
E_1        TTTLNPTIAGAGDV[illegible]-AEGQLGDTMQIVSLQLNKMKSRKS[illegible]QFRF*
E_2        TTTLNPTIAGAGDVKAS-AEGQLGDTMQIVSLQLNKMKSRKS[illegible]QFRF*
I          VTTLNPTIAGKGTVVASGSDNDLADTMQIVSLQLNKMKSRKSC[illegible]AQFRF*
A_1        TTTLNPTIAGKGTVVSS-AENELADTMQIVSLQLNKMKSRKSCGIAVGTTIVDADKYAVTVETRLIDERAAHVNAQFRF*
```

Now turn off protein view and you will see that the nucleotides are also now aligned.

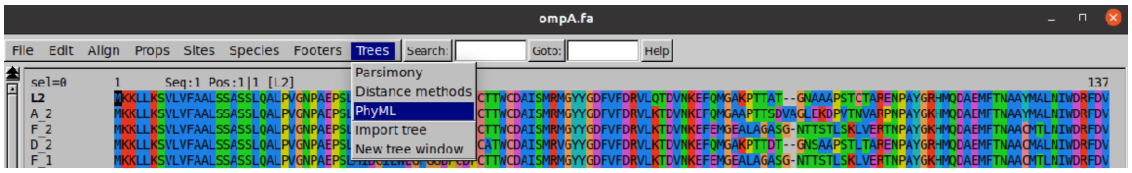


```
sel=0      482                                                                             562
L2         TATTCGGAGATAATGAGAACCATGCTACAGTTTCAGATAGTAAGCTTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
A_2        TATTCGGAACAAAAACACAATCTTCTGGCTTTGATACAGCGAATCTTGTTCCTAACACTGCTTTGAATCAAGCTGTGGTTG
F_2        TATTCGGCGATGGTGTAAACGCCACGAAACCT---GCTGCAGATAGTATTCCTAACGTGCAGTTAAATCAGTCTGTGGTGG
D_2        TGTTTGGAGATAATGAAAATCAAAAAACGGTC---AAAGCGGAGTCTGTACCAAATATGAGCTTTGATCAATCTGTTGTTG
F_1        TATTCGGCGATGGTGTAAACGCCACGAAACCT---GCTGCAGATAGTATTCCTAACGTGCAGTTAAATCAGTCTGTGGTGG
D_1        TGTTTGGAGATAATGAAAATCAAAAAACGGTC---AAAGCGGAGTCTGTACCAAATATGAGCTTTGATCAATCTGTTGTTG
K          TATTCGGAACAAAAACACAATATTCTAAGTTTAATACAGCGAATCTTGTTCCTAACACTGCTTTGGATCGAGCTGTGGTTG
NV         TGTTTGGAGATAATGAAAATCAAAGCACGGTC---AAAACGAATTCTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
L2b        TATTCGGAGATAGTGAGAACCATGCTACAGTTTCAGATAGTAAGCTTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
L3         TATTCGGAACAAAAACACAATCTACTAACTTTAATACAGCGAAGCTTGTTCCTAACACTGCTTTGAATCAAGCTGTAGTTG
G          TATTCGGCGATAGTGAAAACGCCACGCAGCCT---GCTGCAACAAGTATTCCTAACGTGCAGTTAAATCAGTCTGTGGTGG
L1         TATTCGGAGATAATGAAAATCAAAGCACGGTC---AAAAAGGATGCTGTACCAAATATGAGCTTTGATCAATCTGTTGTTG
L2_434_BU  TATTCGGAGATAATGAGAACCATGCTACAGTTTCAGATAGTAAGCTTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
E_1        TGTTTGGAGATAATGAAAATCAAAGCACGGTC---AAAACGAATTCTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
E_2        TGTTTGGAGATAATGAAAATCAAAGCACGGTC---AAAACGAATTCTGTACCAAATATGAGCTTAGATCAATCTGTTGTTG
I          TATTCGGAACAAAAACACAATCTTCTAACTTTAATACAGCGAAGCTTATTCCTAACGCTGCTTTGAATCAAGCTGTGGTTG
A_1        TATTCGGAACAAAAACACAATCTTCTGGCTTTGATACAGCGAATCTTGTTCCTAACACTGCTTTGAATCAAGCTGTGGTTG
```

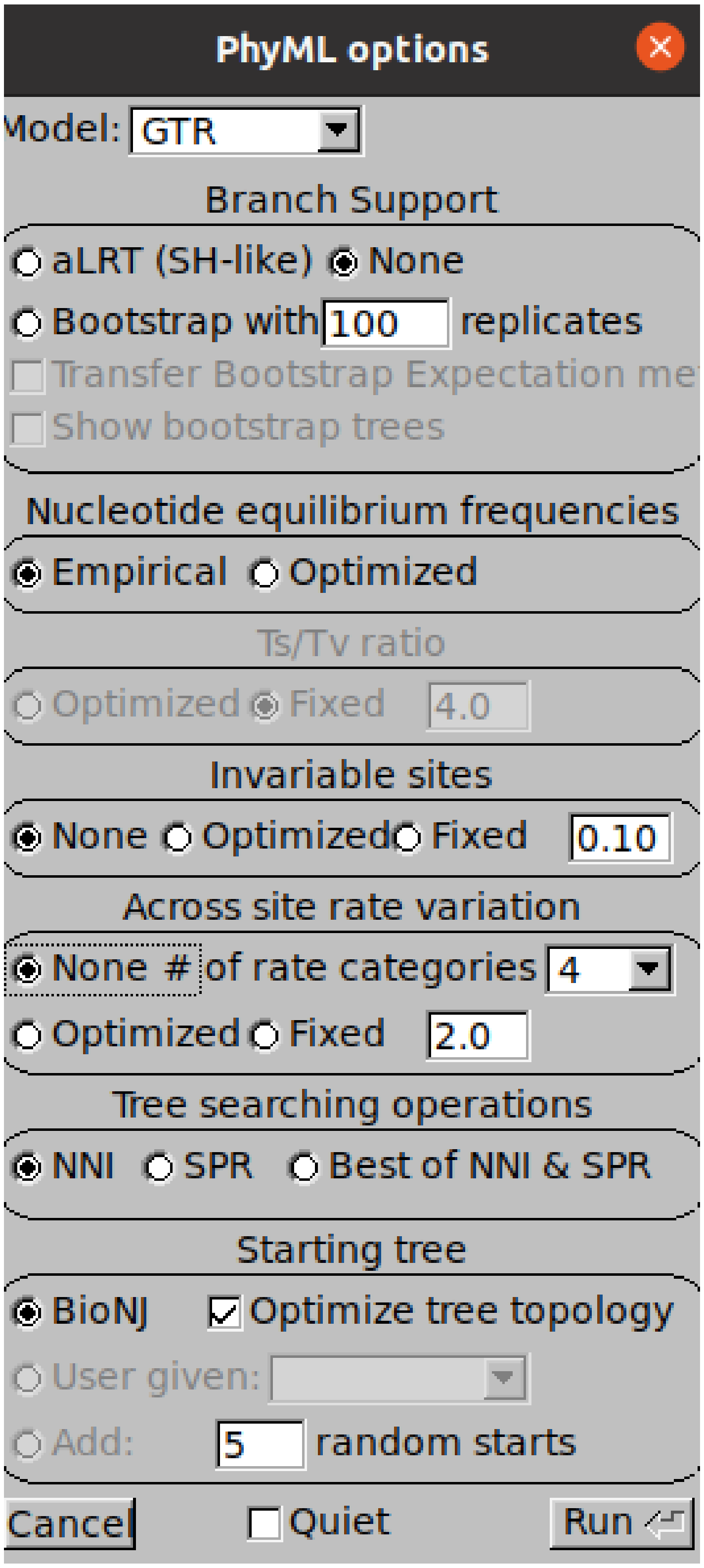
3.4 Phylogeny estimation using PhyML

To estimate the phylogeny, we will use a program called `PhyML`, which is included in `Seaview`. `PhyML` uses a maximum likelihood (ML) method to estimate the tree and includes a number of nucleotide substitution models ranging from the very simple (and could be unrealistic) to more complex ones. Evolutionary (or substitution) models are statistical models that describe the substitution and divergence of sequences over time.

To create a tree in `Seaview`, select 'Trees' and 'PhyML' from the main menu.



Choose the *GTR* (General Time Reversible) model for substitution rates, set all other parameters as shown below and click `Run`.



PhyML options

Model: GTR

Branch Support

aLRT (SH-like) [✓ None]

Bootstrap with 100 replicates

Transfer Bootstrap Expectation me

Show bootstrap trees

Nucleotide equilibrium frequencies

[✓ Empirical] Optimized

Ts/Tv ratio

Optimized [✓ Fixed] 4.0

Invariable sites

[✓ None] Optimized Fixed 0.10

Across site rate variation

[✓ None] # of rate categories 4

Optimized Fixed 2.0

Tree searching operations

[✓ NNI] SPR Best of NNI & SPR

Starting tree

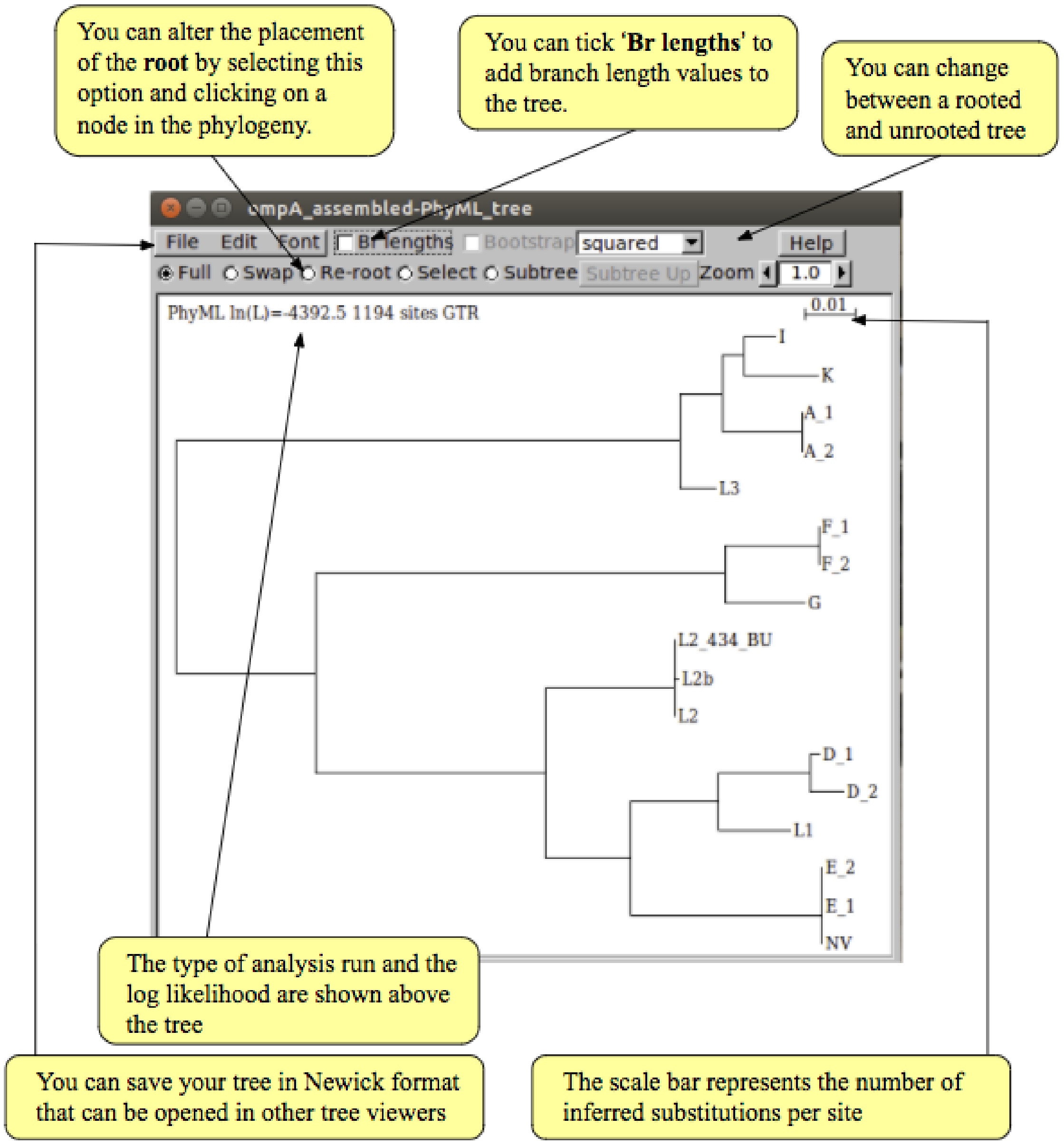
[✓ BioNJ] [✓ Optimize tree topology]

User given:

Add: 5 random starts

Cancel Quiet Run

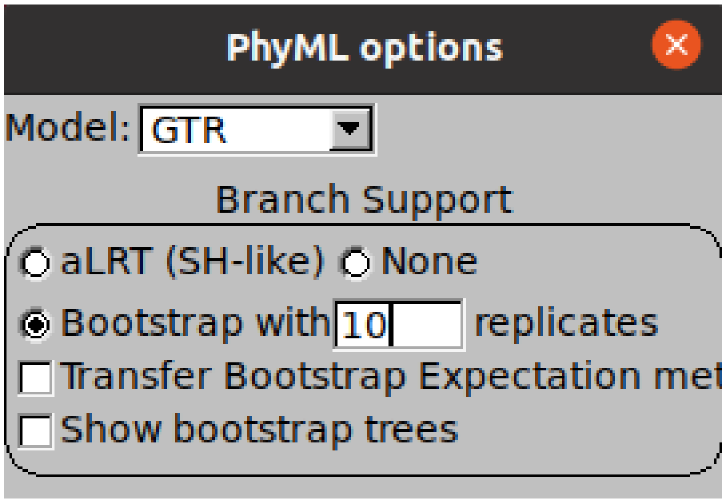
Once the run has finished, click ‘OK’ and you should see a phylogenetic tree as shown below. The tree created by `PhyML` includes the topology of tree (i.e., the relationships between sequences) and the branch lengths (i.e., the amount of change occurring in each lineage). Therefore, the tree is drawn as a *phylogram*, in which the length of branches is proportional to the amount of evolutionary change.


You can alter the placement of the **root** by selecting this option and clicking on a node in the phylogeny.
You can tick '**Br lengths**' to add branch length values to the tree.
You can change between a rooted and unrooted tree
ompA_assembled-PhyML_tree
File Edit Font Br lengths Bootstrap squared Help
Full Swap Re-root Select Subtree Subtree Up Zoom 1.0
PhyML ln(L)=-4392.5 1194 sites GTR
0.01
I
K
A_1
A_2
L3
F_1
F_2
G
L2_434_BU
L2b
L2
D_1
D_2
L1
E_2
E_1
NV
The type of analysis run and the log likelihood are shown above the tree
You can save your tree in Newick format that can be opened in other tree viewers
The scale bar represents the number of inferred substitutions per site

3.5 Phylogeny estimation with bootstrapping

Bootstrapping is a statistical technique to assess the confidence level around each phylogenetic node. The bootstrapping values indicates how many times out of 100, the same branch was observed when repeating the phylogenetic reconstruction on a re-sampled set of the data. Robust relationship should be repeatable, and subsequently observed in a large proportion of randomised data. Therefore, if you get 100 out of 100 times for a particular node, you can be more confident that the observed branch is not due to chance, but likely to be real.

To estimate a bootstrapped phylogeny for the *ompA* data, start by creating a new phylogeny as before and then click on 'Bootstrap' in the 'Branch Support' box, and enter '10' in the replicates box. The processing of the 10 replicates may take a few minutes, so you could move on while this is running. Ideally you would run more replicates (100 or 1000), but to save time we will only run 10 replicates here.


PhyML options

Model: GTR

Branch Support

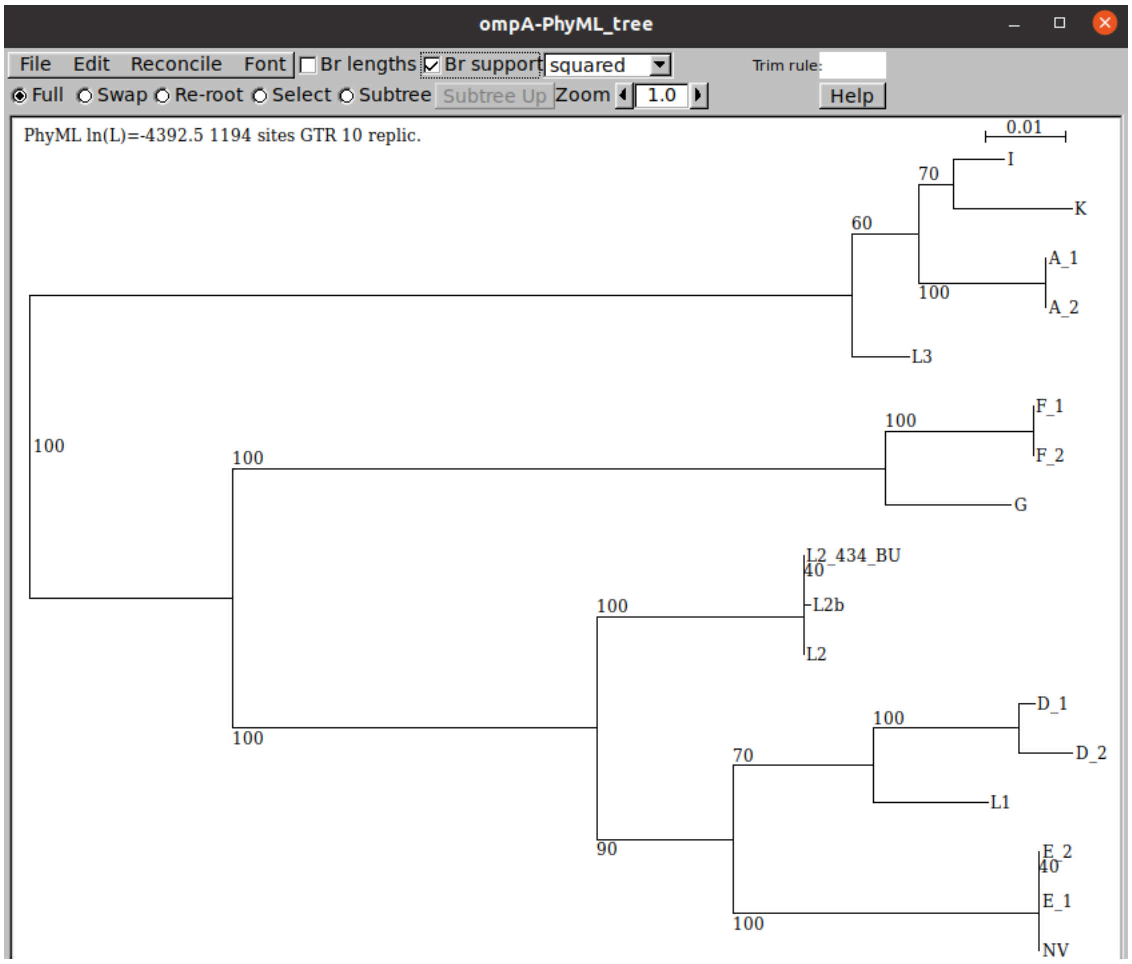
○ aLRT (SH-like) ○ None

● Bootstrap with 10 replicates

☐ Transfer Bootstrap Expectation met

☐ Show bootstrap trees

Once the search is complete, you can show the bootstrap values on the tree by ticking the **Br support** box.



PhyML ln(L)=-4392.5 1194 sites GTR 10 replic.

Each node in the tree now has an associated value out of 100, its bootstrap. Can you identify any nodes that are not robust? Unfortunately there is no generally accepted threshold for significant bootstrap robustness, so you must use your judgement.

WARNING!: bootstrap proportions are measures of robustness, or repeatability. A high bootstrap value indicates that a given node tends to occur in every analysis. This does not guarantee that the node is correct. For example, if the substitution model is inaccurate, it could produce the wrong answer in every estimation.

3.6 Phylogeny visualisation

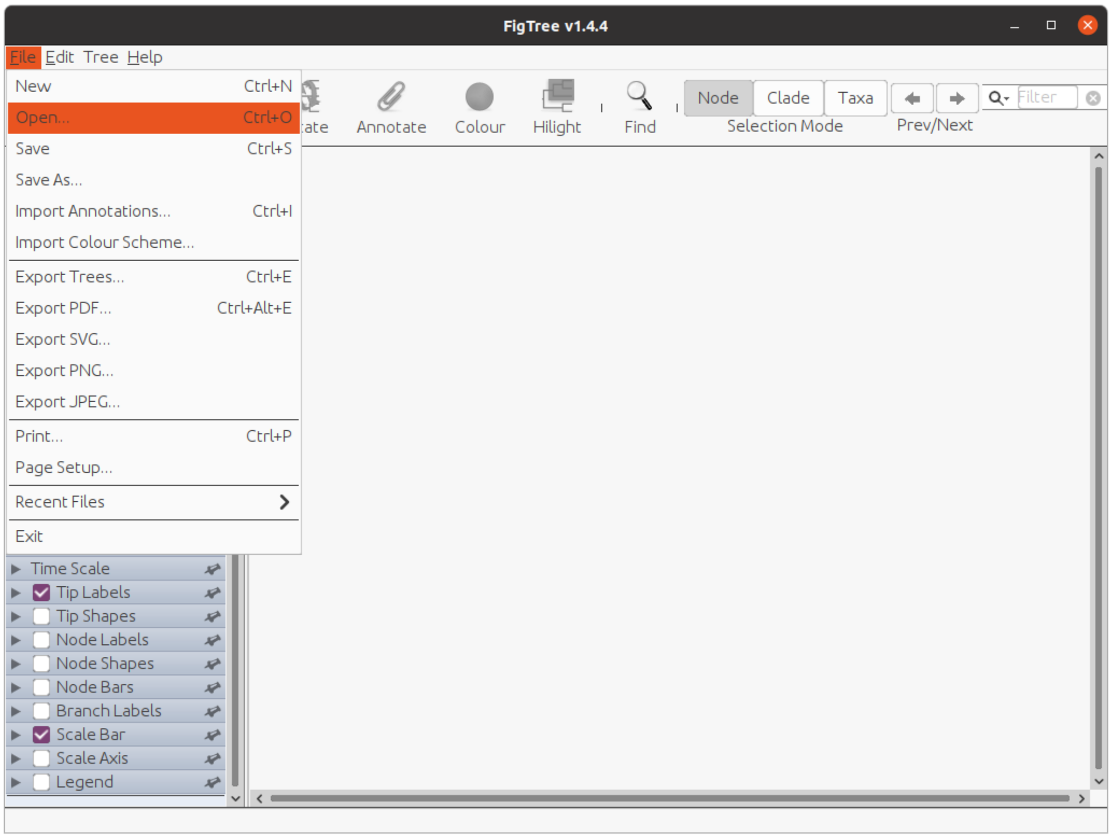
In addition to `Seaview` there are many other tools that can be used to visualise phylogenetic trees including `FigTree`, `Microreact` and `iToL`. As a short exercise, let's use `FigTree` to visualise our tree. `FigTree` is more versatile than the tree viewer in `Seaview`, allowing you to colour branches and taxa, redraw the tree in a number of ways and output the results in a large number of graphics formats including `eps` and `pdf`. It is particularly useful for preparing figures for manuscripts.

First, export the tree produced by PhyML by choosing the 'File' and Save unrooted tree' from the main menu. The file will be saved in newick format. A newick file is a standard text-based format for representing trees in computer-readable form using (nested) parentheses and commas.

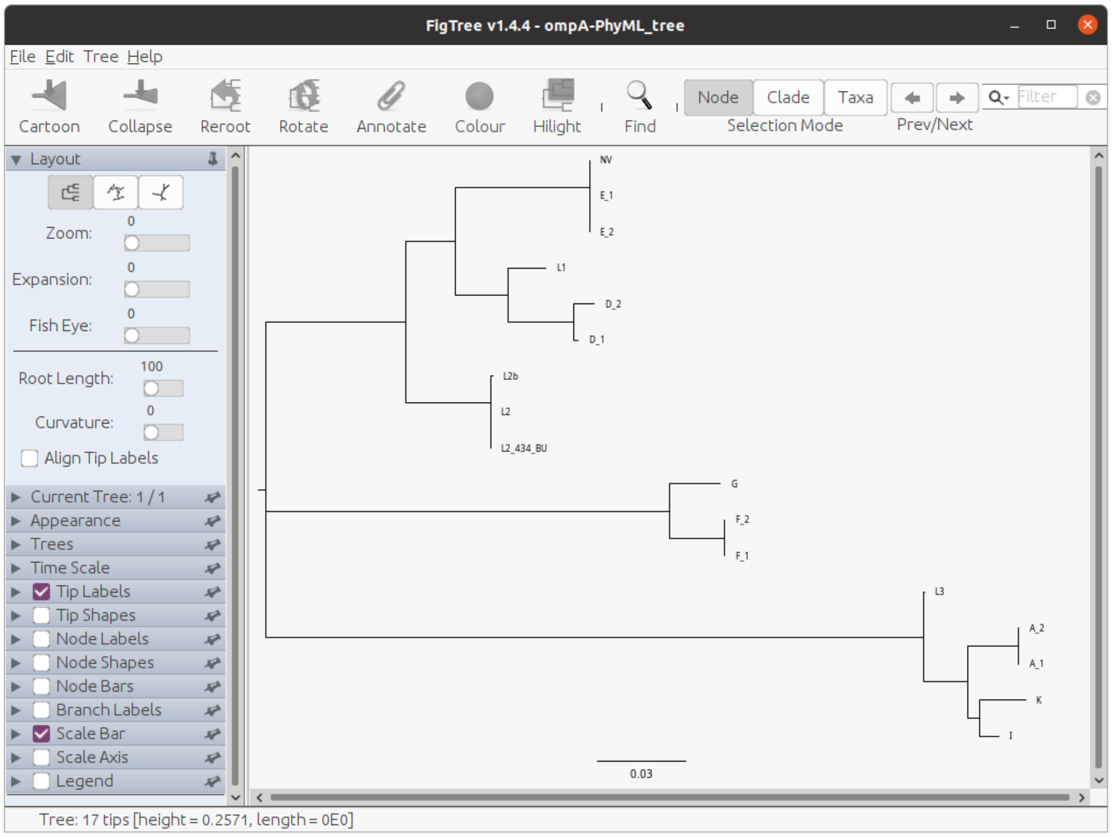
Open FigTree by typing 'figtree' in the terminal.

```
figtree &
```

Now open the exported newick tree file using `FigTree`.



If you have time explore the different functionality available in `FigTree`.



Tree: 17 tips [height = 0.2571, length = 0E0]

3.7 Exercises

1. From the trees that you have produced, which MOMP type would you suggest the new variant (NV) strain belongs to?
2. Do the ompA trees agree with the separation of *C. trachomatis* into trachoma (serotypes A to K) and LGV (L serotypes) biovars?

Now move on to the next section: Constructing phylogeny from whole genome sequence data

4 Phylogeny from whole genome sequence data

When we sequence a population, we aim to capture the variation (SNPs, indels, gene gain and loss etc.) in the samples and use it to infer the relationships between the samples. Two of the main approaches to capturing this variation and reconstructing the bacterial genomes are:

- De novo genome assembly and annotation
- Mapping and variant calling against a reference genome

Each approach has it's benefits and limitations. We will focus on mapping and variant calling in this tutorial. For mapping and variant calling, whether we are dealing with different bacterial isolates, with viral populations in a patient, or even with genomes of different human individuals, the principles are essentially the same. Instead of assembling the newly generated sequence reads de novo to produce a new genome sequence, it is easier and much faster to align or map the sequence reads to a reference genome. We can then readily identify SNPs and indels that distinguish closely related populations or individual organisms and may thus learn about genetic differences that may cause drug resistance or increased virulence in pathogens, or changed susceptibility to disease in humans. One important prerequisite for the mapping of sequence data to work is that the reference and the re-sequenced subject have the same genome architecture.

In this exercise, we will use sequence data from *Salmonella enterica serovar Typhi* samples to demonstrate the mapping and variant calling approach. Importantly, although the data is based on real sequence data, it has been edited to make it run more efficiently for the purpose of this tutorial.

Navigate to the directory that contains the sequence data:

```
cd ~/course_data/snp_phylogeny/data/typhi
```

Take a look at the directory containing the sequence data for the samples:

```
ls fastq
```

4.1 Introducing the tutorial dataset

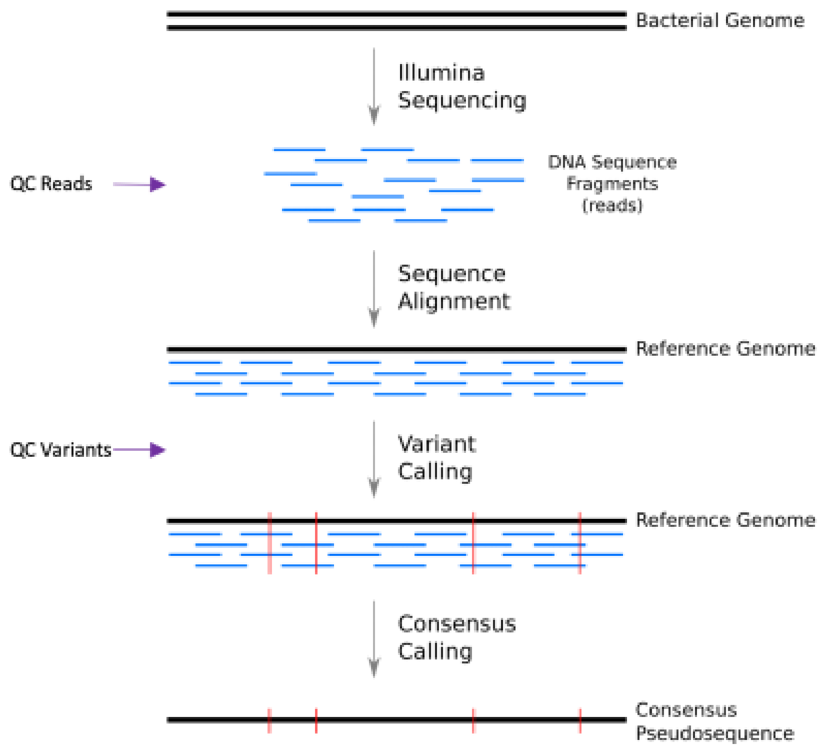
We will use data adapted from the following paper:

> **A genomic snapshot of Salmonella enterica serovar Typhi in Colombia**
> Guevara, Paula Diaz, et al.
> *PLoS Neglected Tropical Diseases2021. doi: 10.1371/journal.pntd.0009755*
> PMID: 34529660

Salmonella enterica serovar Typhi (*S. Typhi*) is the causative agent of typhoid fever, with between 9–13 million cases and 116,800 associated deaths annually. Typhoid fever is still a public health problem in many countries, including in Latin America, which has a modelled incidence of up to 169 (32–642) cases per 100,000 person-years. Several international studies have aimed to fill data gaps regarding the global distribution and genetic landscape of typhoid; however, in spite of these efforts Latin America is still underrepresented. This study provided the first enhanced insights into the molecular epidemiology of S. Typhi in Colombia, using whole genome sequencing data to investigate the population structure in Colombia and identify predominant circulating genotypes.

4.2 Overview of mapping and variant calling approach

The diagram below illustrates the steps involved when mapping and calling variants for a set of bacterial samples.



The first step once you have obtained your sequence data (FASTQ) is to QC the data. After QC, the sequence data is matched or aligned to a reference genome (FASTA) in a process called read mapping to produce a set of read aligments (SAM/BAM). These read alignments are inspected to identify differences between the aligned reads and the reference genome. This process is called variant calling and produces VCF files. In fact during this process we capture information about every position in the genome (variant and non-variant sites) in the VCFs. Each site in the VCF has a set of quality filters applied and any sites identified as low quailty (e.g. less than 4 reads aligned at that position) are marked as low quality in the VCF to produce a filtered VCF. We use this filtered VCF file in a process called consensus caling to reconstruct a consensus *pseudosequence* or *pseudogenome* for our sample (FASTA). In the *pseudogenome*, any sites marked as low quality will be represented as an N in the reconstructed sequence. These pseudogenomes (multi-FASTA) are then aligned and variation identified and used to reconstruct a phylogeny of our samples.

4.3 Exercise

Now let's analyse some data!

4.3.1 Prepare the data

First take a look at the sequence data provided.

```
ls fastq/
```

Check your understanding

1. How many samples have been sequenced?
2. How many fastq files are there?

We will use the chromosome sequence of *Salmonella typhi CT18* as the reference genome. This has already been downloaded from RefSeq. Take a look at the reference genome:

```
ls ref/
```

Check the size of the reference file:

```
assembly-stats ref/Styphi_CT18.fa
```

Now use bwa to index the reference genome. This creates a lookup table that bwa uses when matching the sequence reads against the reference genome.

```
bwa index ref/Styphi_CT18.fa
```

Check your understanding

3. How many sequences in the reference fasta file?
4. What are the names of the sequences in the reference fasta file?
5. What is the size of the reference?
6. What additional files did the indexing step produce?

4.3.2 QC the sequence data

An important first step in any analysis is QC of the data. We will the FastQC software to QC the data. First create a directory for the qc results:

```
cd fastq
mkdir qc_results
```

Run FastQC on the all the fastq files and store the results in the directory `qc_results`:

```
fastqc -o qc_results *.fastq.gz
```

This will create one html report for each of the fastq files. Take a look:

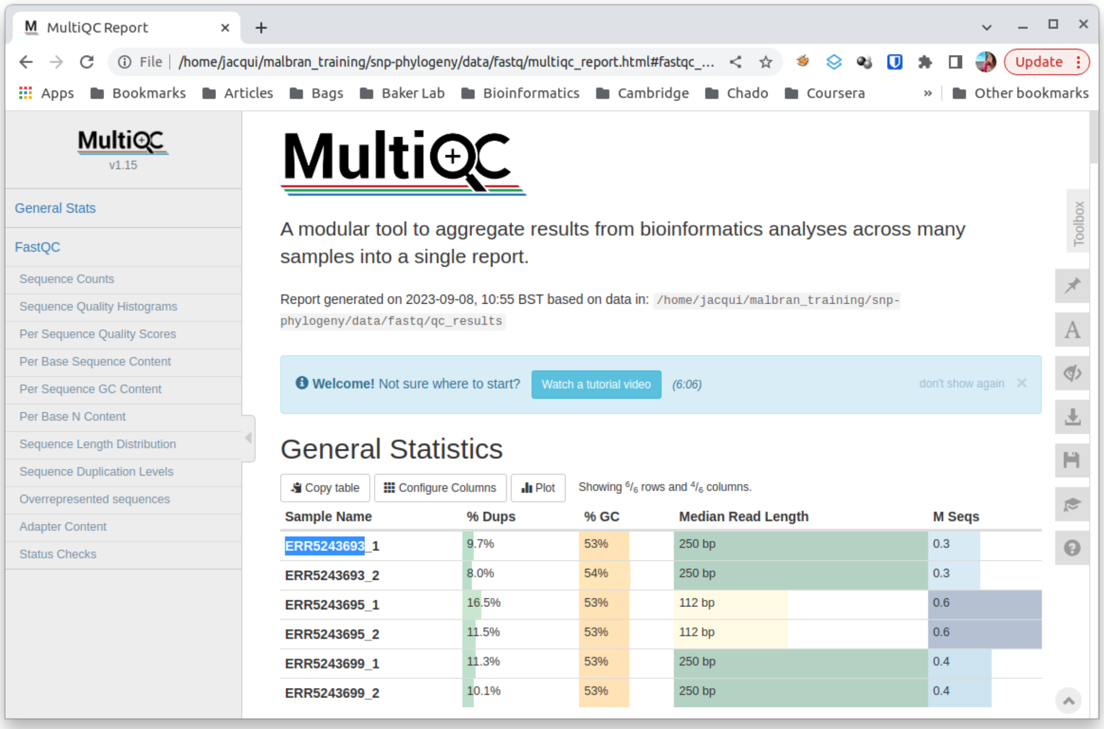
```
ls qc_results/
```

If we have lots of samples it will be difficult to manually inspect each file. Therefore we will use multiQC to collate all the QC reports into one file.

```
multiqc qc_results
```

Open the collated report `multiqc_report.html` in firefox.

```
firefox multiqc_report.html &
```



A modular tool to aggregate results from bioinformatics analyses across many samples into a single report.

Report generated on 2023-09-08, 10:55 BST based on data in: `/home/jacqui/malbran_training/snp-phylogeny/data/fastq/qc_results`

Welcome! Not sure where to start?

General Statistics

Showing $^6/_6$ rows and $^4/_6$ columns.

Sample Name	% Dups	% GC	Median Read Length	M Seqs
ERR5243693_1	9.7%	53%	250 bp	0.3
ERR5243693_2	8.0%	54%	250 bp	0.3
ERR5243695_1	16.5%	53%	112 bp	0.6
ERR5243695_2	11.5%	53%	112 bp	0.6
ERR5243699_1	11.3%	53%	250 bp	0.4
ERR5243699_2	10.1%	53%	250 bp	0.4

Check your understanding

1. What is the median read length for sample ERR5243693?
2. Which sample has the largest yield (most sequence data)?

4.3.3 Trim the reads to remove low quality and adapter sequence

If your sequence reads have a high level of adapter contamination and/or have low quality bases at the end of the reads you can trim the reads to remove these sequences. There are several software that can be used to do this including trimmomatic and fastp.

Use fastp to trim the reads for sample ERR5243693.

```
fastp \
   --in1 ERR5243693_1.fastq.gz --in2 ERR5243693_2.fastq.gz \
   --out1 ERR5243693_1.trim.fastq.gz \
   --out2 ERR5243693_2.trim.fastq.gz \
   --json ERR5243693.fastp.json --html ERR5243693.fastp.html \
   --detect_adapter_for_pe --cut_mean_quality 20 \
   --thread 2
```

Now repeat for the other samples and look at the output from fastp:

```
head -20 ERR5243693.fastp.json
head -20 ERR5243695.fastp.json
head -20 ERR5243699.fastp.json
```

Check your understanding

1. How much data (bp/base pairs) was lost due to trimming and adapter removal?

4.3.4 Map the data to a reference genome

When performing any data analysis it is good practice to arrange your data in a logical way rather than putting all files in a single directory. Let's create a directory to store the results of mapping the reads to the reference genome:

```
mkdir ../samtools
cd ../samtools
```

Use `bwa` to map the reads for sample ERR5243693 to the reference genome.

```
bwa mem -t 2 ../ref/Styphi_CT18.fa \
../fastq/ERR5243693_1.trim.fastq.gz ../fastq/ERR5243693_2.trim.fastq.gz \
> ERR5243693.sam
```

This may take a few minutes to run. Let's take a look at the options:

- `-t` : tells the program to use 2 CPUs
- `../ref/Styphi_CT18.fa` is the reference to match the reads against
- `../fastq/ERR5243693_1.trim.fastq.gz` and `../fastq/ERR5243693_2.trim.fastq.gz` are the fastq files containing our sequence reads (after trimming) for sample ERR5243693
- the output is SAM format and `> ERR5243693.sam` redirects the output to the file `ERR5243693.sam`

When complete, convert the sam file to a bam file with samtools:

```
samtools view -@ 2 -bh -o ERR5243693.bam ERR5243693.sam
```

The `-@` option tells the program to use 2 CPUs and the `-b` option specifies to write the output as a bam file, `-h` option means include the header information in the output. The `-o` option specifies the name of the output file `ERR5243693.bam`.

Sort the bam file and then index the sorted bam file:

```
samtools sort -@ 1 -o ERR5243693.sorted.bam -T ERR5243693.sorted ERR5243693.bam
```

```
samtools index ERR5243693.sorted.bam
```

Generate some statistics about the alignment:

```
samtools stats ERR5243693.sorted.bam > ERR5243693.stats
samtools flagstat ERR5243693.sorted.bam > ERR5243693.flagstat
samtools coverage ERR5243693.sorted.bam > ERR5243693.coverage
```

Now repeat for the other samples.

Check your understanding

1. What %reads mapped to the reference for each sample?
2. What %genome was covered for each sample?
3. What is the mean depth of coverage for each sample?

4.3.5 Call variants

Go through each position in the reference genome and look at reads aligned at that position and make a call about what the base is at that position for the sample. This information will be stored in a VCF file and if there are any differences then this will be marked as a variant (snp/indel) in this VCF file.

Again create a directory to store the results of the variant calling step:

```
mkdir ../variants
cd ../variants
```

Run this for sample ERR5243693 using bcftools:

```
bcftools mpileup --fasta-ref ../ref/Styphi_CT18.fa \
--min-BQ 20 \
--annotate \
FORMAT/AD,FORMAT/ADF,FORMAT/ADR,FORMAT/DP,FORMAT/SP,INFO/AD,\
INFO/ADF,INFO/ADR ../samtools/ERR5243693.sorted.bam | bcftools call \
--output-type v --ploidy 1 --multiallelic-caller - | \
bcftools view --output-file ERR5243693.vcf.gz --output-type z
```

Again this may take some time to run. Let's look at the options:

The `bcftools mpileup` command is passed the following options:

- `--fasta-ref ../ref/Styphi_CT18.fa` is the reference that the reads were mapped to
- `--min-BQ 20` is minimum base quality for a base to be considered
- `--annotate FORMAT/AD,FORMAT/ADF,FORMAT/ADR,FORMAT/DP,FORMAT/SP,INFO/AD,INFO/ADF,INFO/ADR` tells the program what tags to incude in the VCF

The VCF produced by the bcftools mpileup command is passed to `bcftools call` with the following options:

- `--output-type v` tells the program to output VCF format
- `--ploidy 1` tells the program that we are daling with haploid datasets
- `--multiallelic-caller` tells the program which variant calling algorithm to use

The VCF produced by the `bcftools call` command is passed to `bcftools view` with the following options which converts the VCF to a compressed VCF file:

- `--output-file ERR5243693.vcf.gz` tells the program the name of the output file
- `--output-type z` tells the program to create a gzipped VCF file as output

When it completes, index the gzipped VCF file that has been created:

```
tabix -p vcf -f ERR5243693.vcf.gz
```

Take a look at the VCF file that was produced:

```
bcftools view ERR5243693.vcf.gz | less -S
```

Notice that the VCF has information about every site in the genome.

Now generate some statistics about the VCF file:

```
bcftools stats ERR5243693.vcf.gz > ERR5243693.vcf.stats.txt
```

Now repeat for the other samples.

Look at the statistics for the variant calling:

```
less ERR5243693.vcf.stats.txt
less ERR5243695.vcf.stats.txt
less ERR5243699.vcf.stats.txt
```

Check your understanding

1. How many sites are in the VCF file for each sample?
2. Does this match to the size of the reference used in the read mappping step?
3. How many variant sites were identified for each sample?

4.3.6 Filter the VCFs

We want to identify calls where we have a high confidence that they are correct (and not due to sequencing errors and/or misalignment of the reads). We use criteria like read depth at a position, quality scores etc. to filter out low quality calls at each position.

Use bcftools to filter sites for sample ERR5243693.

```
bcftools filter \
--output ERR5243693.filtered.vcf.gz \
--soft-filter LowQual \
--exclude "QUAL<25 || FORMAT/DP<10 || MAX(FORMAT/ADF)<2 || MAX(FORMAT/ADR)<2 \
|| MAX(FORMAT/AD)/SUM(FORMAT/DP)<0.9 || MQ<30 || MQ0F>0.1" \
--output-type z ERR5243693.vcf.gz
```

The filtering step may take some time to run. Let's look at the options used in the command above:

- `--output` specifies the name of the output file
- `--soft-filter` tells the program to keep any filtered position in the file and mark them as `LowQual` rather than remove them completely from the file (hard filter)
- `--exclude` lists the filtering criteria to apply to each position
- `--output-type` specifies the type of output file to create, in this case `z` means a xompressed VCF file

When the bcftools filter command is complete, index the filtered VCF file:

```
tabix -p vcf -f ERR5243693.filtered.vcf.gz
```

Take a look at the VCF file and notice how some of the sites are marked as `PASS` or `LowQual` under the filter column.

```
bcftools view ERR5243693.filtered.vcf.gz | less -S
```

Check your understanding Looking at the filtered VCF for sample ERR5243693:
1. Does position 2 pass or fail?
2. Does position 244 pass or fail?
3. What is the reason for the failure at position 2527?

To view only the sites that passed the filtering step:

```
bcftools view -f PASS ERR5243693.filtered.vcf.gz | less
```

To generate some statistics about the filtered VCF file:

```
bcftools stats ERR5243693.filtered.vcf.gz > \
ERR5243693.filtered.stats.txt
```

Now repeat for the other samples.

Check your understanding

4. How many sites were marked as low quality in the filtering step?
5. How many variant sites were marked as low quality in the filtering step?

4.3.7 Call a consensus sequence for each sample

A pseudogenome is a reconstruction of what we think the genome is for the sample using the reference genome as a basis. To create it for a sample, you go through each position in the reference and determine what base is called (using the VCF from the previous steps) for the sample. Sometimes this will be the same as the reference, and sometimes it will differ from the reference (a variant). For positions that are flagged as low quality/filtered out (e.g. no reads covering the position) we use an N in the pseudogenome. This is because you cannot be confident what the base is at this position for the sample. In the end the length of the pseudogenome for your sample should be the same as the length of the reference.

To create a pseudogenome for sample ERR5243693 use the script *vcf2pseudogenome.pl*. This has already been installed on the computer using this command (it also has a dependency on *pysam* and *biopython*):

```
wget https://raw.githubusercontent.com/nf-core/bactmap/master/bin/vcf2pseudogenome.py
```

Create a directory to store the pseudogenomes:

```
mkdir ../pseudogenomes
cd ../pseudogenomes
```

The run the `vcf2pseudogenome.py` script:

```
vcf2pseudogenome.py -r ../ref/Styphi_CT18.fa \
-b ../variants/ERR5243693.filtered.vcf.gz -o ERR5243693.fas
```

Now repeat for the other samples.

Check your understanding

1. What is the length of the pseudogenomes? (Hint: Use assembly-stats)
2. Does it match the length of the reference?

4.3.8 Create a multiple sequence alignment of all pseudogenomes

Remember that to reconstruct the phylogeny of our samples we need a multifasta alignment of our sequences. We are going to use a reconstruction of the entire genome as our set of sequences.

```
cat *.fas > aligned_pseudogenomes.aln
```

Because of the way the pseudogenomes were constructed resulting in them all being the same length we do not have to perform a multi-sequence alignment to ensure all the bases are homologous.

Check your understanding

1. How many sequences in the multiple sequence alignment file of pseudogenomes?
2. What is the largest and mean sequence length? (Hint: Use assembly-stats)

4.3.9 Add the reference genome to the multiple sequence alignment

Sometimes you may want to include the reference genome in your tree. To do this, add the reference sequence to the multifasta alignment of your samples.

```
cat ../ref/Styphi_CT18.fa aligned_pseudogenomes.aln \
> ref_and_aligned_pseudogenomes.aln
```

At this point it would be useful to look at your alignment in a multiple sequence alignment viewer e.g. seaview.

```
seaview ref_and_aligned_pseudogenomes.aln &
```

Check your understanding

1. How many sequences in the multiple sequence alignment file containng the reference genome and sample pseudogenomes?

4.3.10 Mask repetitive regions (optional but good practice)

Bases called in repetitive regions may not be true variation (e.g. due to misalignment of reads) and may compromise the core phylogenetic signal. Therefore it is good practice to identify known repetitive regions and mask these out from your alignment.

To achieve this, either a file of known regions for the reference you mapped to will exist (see the literature) or one can be generated by matching the reference genome against itself (to identify repeat regions) with e.g. Mummer and Phast used to identify prophage. You can use the software remove_blocks to mask out any repetitive region in your multifasta alignment, this masking involves replacing the bases with Ns in this region. To save time we wont run this step here.

4.3.11 Draw a tree with iqtree

Now that we have a multiple sequence alignment, we can use IQ-TREE to build a maximum likelihood phylogeny. But first create a directory to store the iqtree analysis.

```
mkdir ../iqtree
cd ../iqtree
```

Calculating a phylogeny on whole genome sequences can be very time consuming. We can speed this up by only using the variable sites (SNPs). These are sites that differ in at least one of the samples. However, we need to be aware that only including variable sites can affect the evolutionary rate estimates made by phylogenetics software - therefore, we need to account for the sites we remove in our analysis.

We will use snp-sites to do this. You can view the options for snp-sites:

```
snp-sites -h
```

First, remove all the invariant sites and create a SNP-only multiple sequence alignment:

```
snp-sites -o pseudogenomes.snps.aln \
../pseudogenomes/ref_and_aligned_pseudogenomes.aln
```

Look at the new file that is created:

```
less pseudogenomes.snps.aln
```

We can also see how many invariant sites were removed (and what proportion of A, T, G, C they were) using

```
snp-sites -C ../pseudogenomes/ref_and_aligned_pseudogenomes.aln
```

Check your understanding

1. How many variant sites are identified?
2. How many invariant sites are identified?
3. Does this correlate with the expected number i.e. from the literature?

Now look at the options for IQ-TREE:

```
iqtree -h
```

And then construct the tree with IQ-TREE:

```
iqtree \
    -s pseudogenomes.snps.aln \
    -fconst $(snp-sites -C ../pseudogenomes/ref_and_aligned_pseudogenomes.aln) \
    -m GTR+G \
    -B 1000 \
    -T 2 \
    -mem 2GB
```

In the command above, we have used te following options

- `-s` to specify the multiple sequence alignment file `pseudogenomes.snpsites.aln`
- `-fconst` to ask IQ-TREE to take account of missing invariant sites $(snp-sites -C ref_and_aligned_pseudogenomes.aln calculates the specific values to pass to iqtree)
- `-m` to specify an evolutionary model, we want IQ-TREE to use -m GTR+G
- `-B` to perform 1000 ultrafast bootstraps
- `-T` to tell IQ-TREE to use a maximum of 2 CPUs (threads)
- `-mem` 2GB to tell IQ-TREE to use a maximum of of 2GB memory

Our maximum likelihood tree is labelled `pseudogenomes.snpsites.aln.treefile`. The treefile suffix is not always correctly identified by many tools, so we'll relabel this as something else:

```
cp pseudogenomes.snps.aln.treefile pseudogenomes.snps.aln.tre
```

We can look at the raw text file:

```
cat pseudogenomes.snps.aln.tre
```

But it is better if we visualise this using a tree view e.g.figtree

```
figtree pseudogenomes.snps.aln.tre &
```

4.3.12 Accounting for recombination with Gubbins

Variation due to recombination events can mask the core phylogenetic signal, therefore it is recommended to identify these regions in your alignment and mask them out. We can use gubbins to infer recombining sites by looking for increased SNP density that occurs in specific ancestral nodes. Look at the options for gubbins:

```
run_gubbins.py -h
```

Make a directory to store the results of running gubbins:

```
mkdir ../gubbins
cd ../gubbins
```

Run gubbins on your data:

```
run_gubbins.py -c 2 -p gubbins \
../pseudogenomes/ref_and_aligned_pseudogenomes.aln
```

The `-c` option tells the program to use 2 CPUs and the `-p` option tells the program to name all files with the prefix gubbins. This command can take a few minutes to run.

Note: Gubbins must be run on the entire alignment (not just SNPs) as it uses the spatial distribution to identify and filter out recombinant regions.

If gubbins takes more than 10 mins to complete, we have already run it for you - the files are available at:

```
ls -l ~/course_data/snp_phylogeny/data/typhi/gubbins_backup
```

Lets look at what gubbins has done:

```
ls -l *
```

You can explore these files. For example `gubbins.recombination_predictions.gff` is a gff file that contains a record of each recombination block identified, how many SNPs it contains, and what samples are affected.

head gubbins.recombination_predictions.gff

The file `gubbins.final_tree.tre` is a phylogeny in which recombination has already been accounted for.

You can visualise this in figtree.

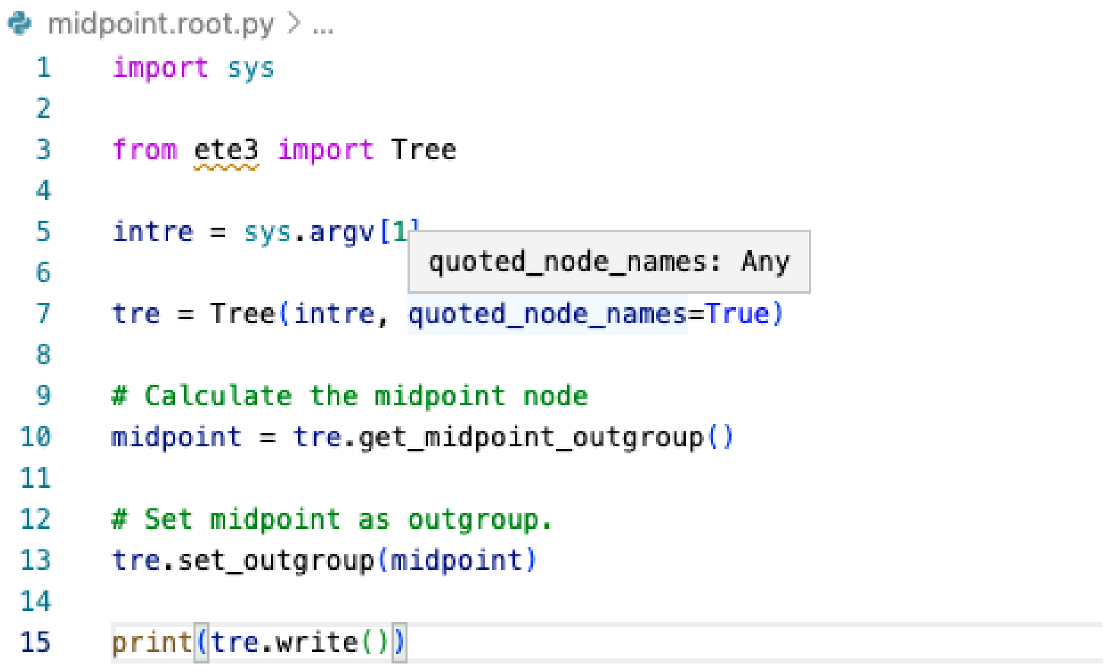
```
figtree gubbins.final_tree.tre
```

4.4 Root the phylogeny

The trees generated by iqtree and gubbins are unrooted, but we may want to apply some evolutionary direction to them. One strategy for rooting a tree is called *midpoint rooting*. Midpoint rooting involves locating the midpoint of the longest path between any two tips and putting the root in that location. Note that this does not necessarily infer the true root, and this should be used with caution.

To midpoint root our tree, we will use a simple script written in python that uses the ete package. You can examine the code:

```
less ../midpoint.root.py
```



```python
import sys

from ete3 import Tree

intre = sys.argv[1]

tre = Tree(intre, quoted_node_names=True)

# Calculate the midpoint node
midpoint = tre.get_midpoint_outgroup()

# Set midpoint as outgroup.
tre.set_outgroup(midpoint)

print(tre.write())
```

Run this script to midpoint root the tree.

```
python midpoint.root.py gubbins.final_tree.tre > gubbins.final_tree.midpoint.tre
```

Visualise this in figtree.

Another common strategy for rooting the tree is *outgroup rooting*. This is the preferred approach for bacterial datasets. Outgroup rooting involves including one or more sequences in the analysis that are more distantly related to our sequences of interest than they are to one another. These sequence are usually referred to as *outgroups*. The root estimate is then simply the point at which the outgroup(s) join the tree. The best possible outgroups are those available which are most closely related to our sequences of interest but still different enough. For examples, in this dataset we could use a *Salmonella paratyhi A sample* as an outgroup.

There are a few ways to do this. One is to obtain sequence data for the outgroup sample and incorporate it into the dataset from the beginning of the analysis and construct a pseudogenome for the outgroup. If no sequence data is available, you can take a complete reference genome for the outgroup and `shred` it to simulate sequencing reads for the reference. Just remember when calculating and reporting the number of variant sites for your dataset that you remove the outgroup from the alignment.

4.4.1 Clean-up!

Clean up any intermediate files that were generated during the analysis that you no longer require. This is always an important last step of any analysis as sequence data analysis files can use up large amounts of disk space. Let's look at what files were created in our analysis

```
cd ..
ls -alhrt */*
```

As you can see the data analysis has generated a lot of files! So let's remove any files that do not need to be kept long term:

```
rm fastq/*.trim.fastq.gz
rm fastq/*.fastp.html
rm fastq/*.fastp.json
rm samtools/*.sam
rm samtool/ERR5243693.bam*
rm samtools/ERR5243695.bam*
rm samtools/ERR5243699.bam*
rm variants/ERR5243693.vcf.gz*
rm variant/ERR5243695.vcf.gz*
rm variants/ERR5243699.vcf.gz*
```

Now go to the next section: Phylogeny and Metadata

5 Phylogeny and Metadata

Often it is useful to visualise metadata about your samples in the context of your phylogenetic tree. For example, location information can provide insight into the processes that drive their epidemiology. This can be used to infer whether single introductions of a pathogen have occurred followed by local evolution or whether it transmits frequently across borders. It can also indicate regions affected by antimicrobial resistance (AMR).

In this section, you will use `Microreact` to visualise the phylogeny you created in the previous section with some basic metadata. `Microreact` is software developed by the Centre for Genomic Pathogen Surveillance (CGPS) that allows you to upload, visualise and explore any combination of clustering (trees), geographic (map) and temporal (timeline) data. Other metadata variables are displayed in a table. You can specify colours and/or shapes to display on the map, tree and/or timeline. A permanent URL is produced for you to share your Microreact, or a .microreact file can be downloaded for sharing with collaborators.

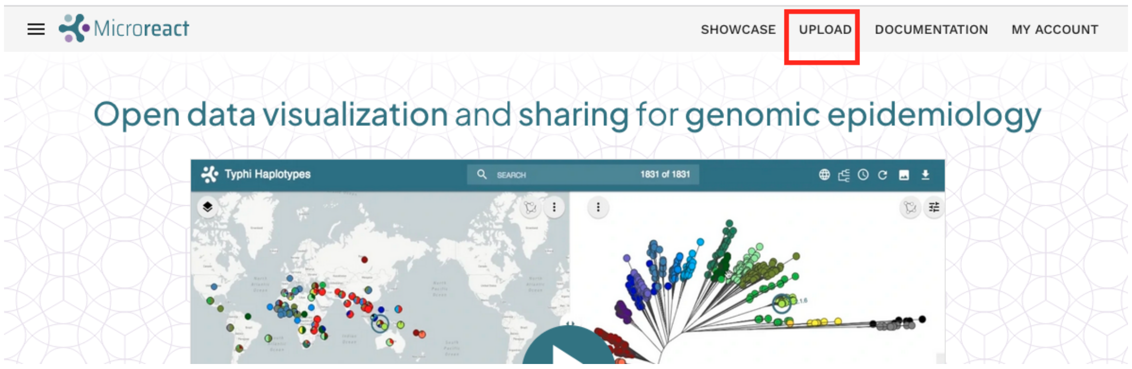
There are several alternatives including FigTree, Phandango and iToL but for the purposes of this tutorial we will demostrate the use of Microreact for metadata visualisation.

5.1 The dataset

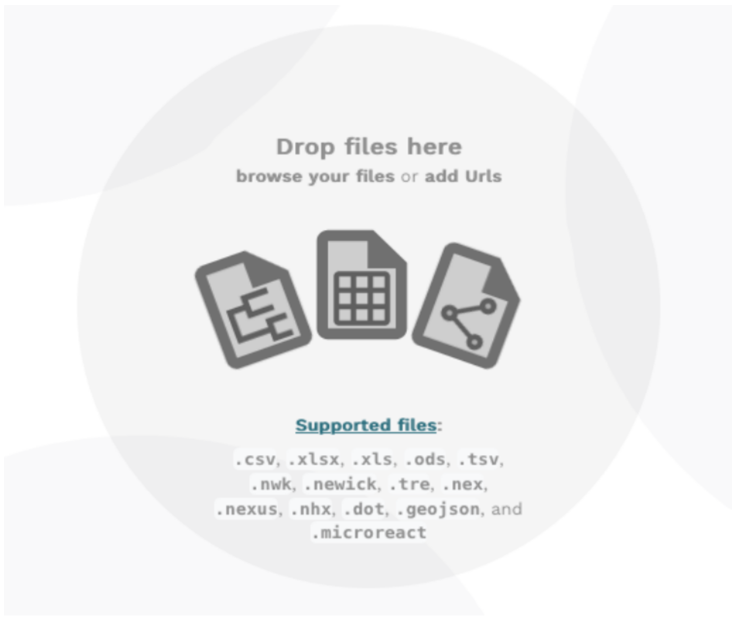
In this section we will make use of a maximum likelihood phylogenetic tree of the *S. typhi* isolates from the previous section and a metadata file with information on location, time of isolation and genotype for the samples.

5.2 Create a Microreact project

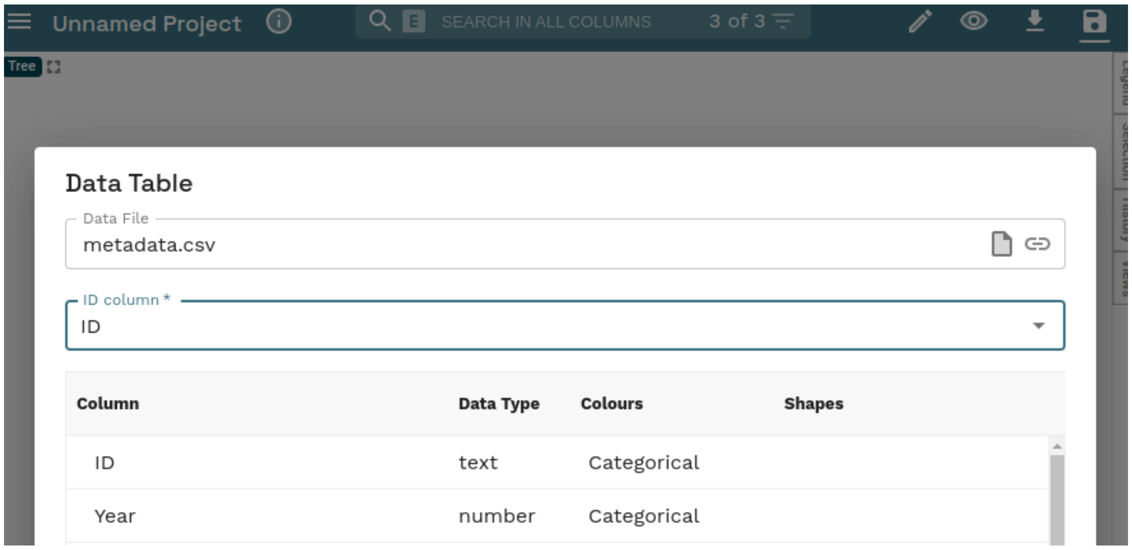
Start by opening up a new window in Firefox and typing https://Microreact.org in the address bar. Click on "Upload" as shown below and create a new project.



Open a file browser and navigate to the metadata directory (`/home/manager/course_data/snp_phylogeny/data/metadata`). Drag and drop the files "metadata.csv" and "tree.tre" from the file browser onto your Internet browser.


Drop files here
browse your files or add Urls
Supported files:
.csv, .xlsx, .xls, .ods, .tsv, .nwk, .newick, .tre, .nex, .nexus, .nhx, .dot, .geojson, and .microreact

Once the tree and metadata files are loaded you will be directed to a new window where files will be automatically detected as Data (CSV or TSV) file (metadata.csv) and Tree (Newick) file (tree.tre). In this new window click on 'Continue'. In the next window and make sure column 'ID' is selected as the 'ID column' and then click on 'Continue'.



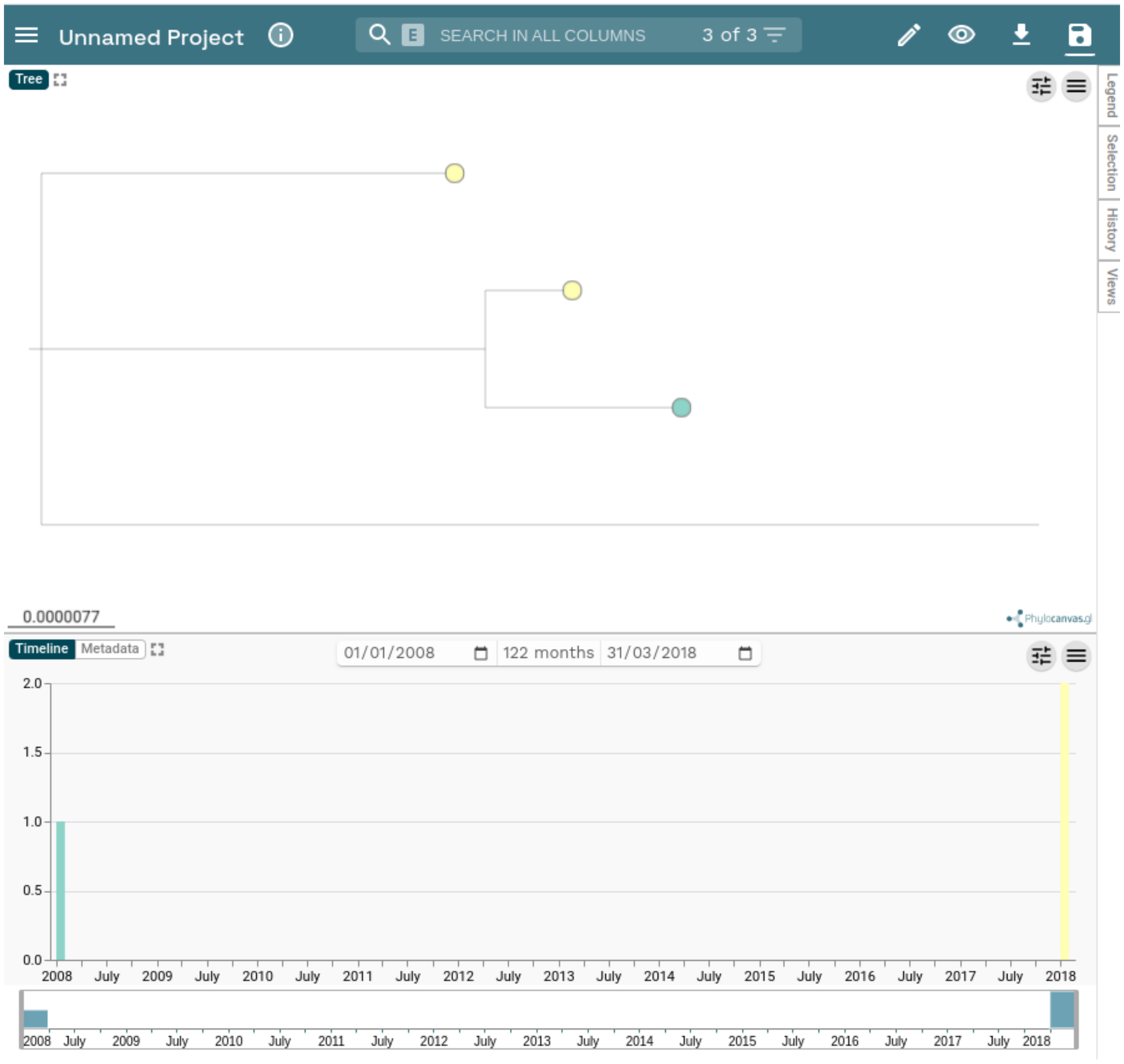
Data Table

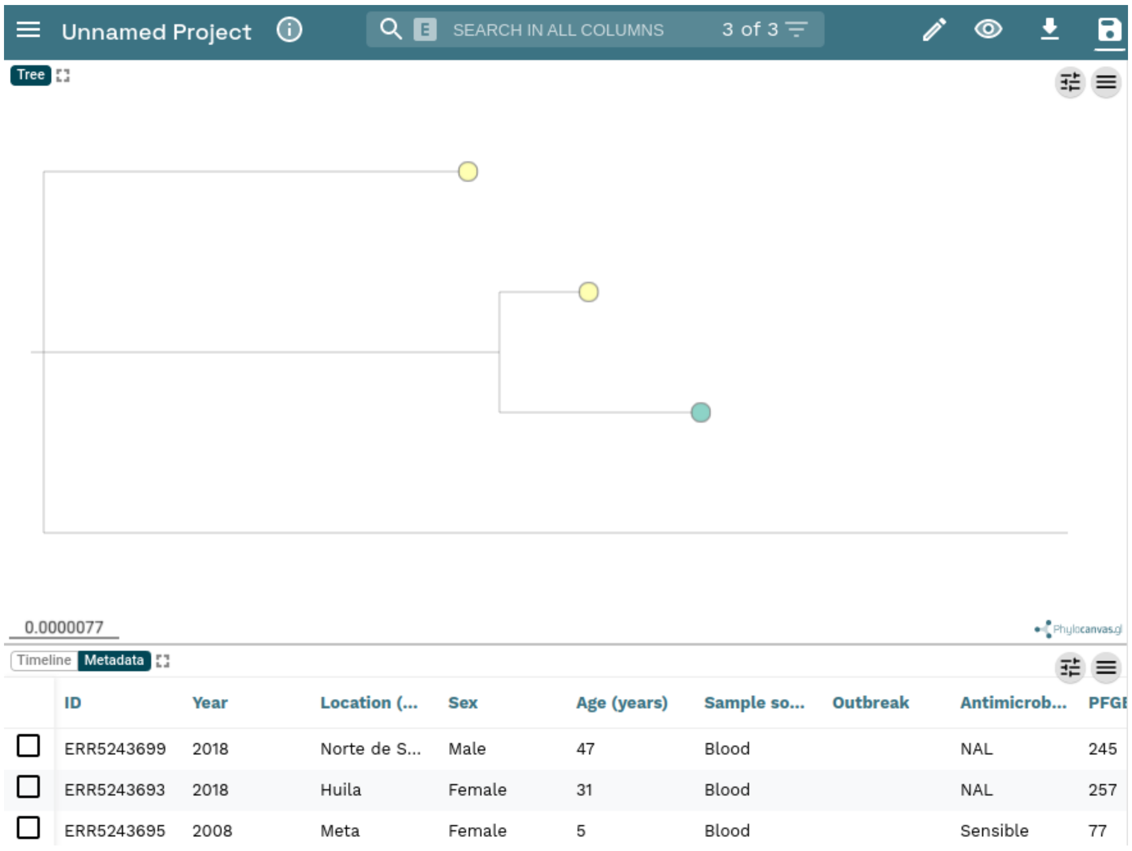
Data File: metadata.csv

ID column *: ID

Column	Data Type	Colours	Shapes
ID	text	Categorical	
Year	number	Categorical	

Once these forms are completed your data will be utilized to create a Microreact project. You should now have a view similar to the ones shown below. You should see a tree view and a timeline and metadata view.

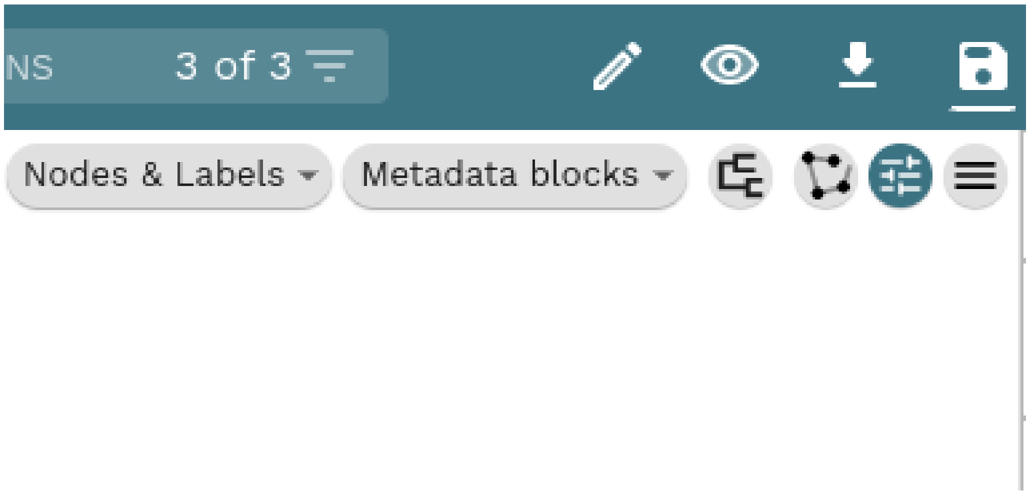

Unnamed Project
SEARCH IN ALL COLUMNS
3 of 3
Tree
Legend
Selection
History
Views
0.0000077
Phylocanvas.gl
Timeline
Metadata
01/01/2008
122 months
31/03/2018
2.0
1.5
1.0
0.5
0.0
2008
July
2009
July
2010
July
2011
July
2012
July
2013
July
2014
July
2015
July
2016
July
2017
July
2018



	ID	Year	Location (...	Sex	Age (years)	Sample so...	Outbreak	Antimicrob...	PFGE
☐	ERR5243699	2018	Norte de S...	Male	47	Blood		NAL	245
☐	ERR5243693	2018	Huila	Female	31	Blood		NAL	257
☐	ERR5243695	2008	Meta	Female	5	Blood		Sensible	77

5.3 Exploring the metadata

Clicking on 'Show Controls' button in the top right hand corner of the the tree window allows you to view different kinds of trees and change text/node size. Try changing layout of the tree.



There are lots more things you can do with `Microreact` but unfortunately we don't have time to cover them in this tutotial.

Congratulations, you have reached the end of the tutoial!