Phylogenetic Trees – Part II (Maximum Likelihood)

Exercise 1: The Newick format

Phylogenetic trees are most commonly represented using the Newick format. In this exercise we will become familiar with its grammar rules and learn how to use it to depict phylogenies.

Part 1: The grammar

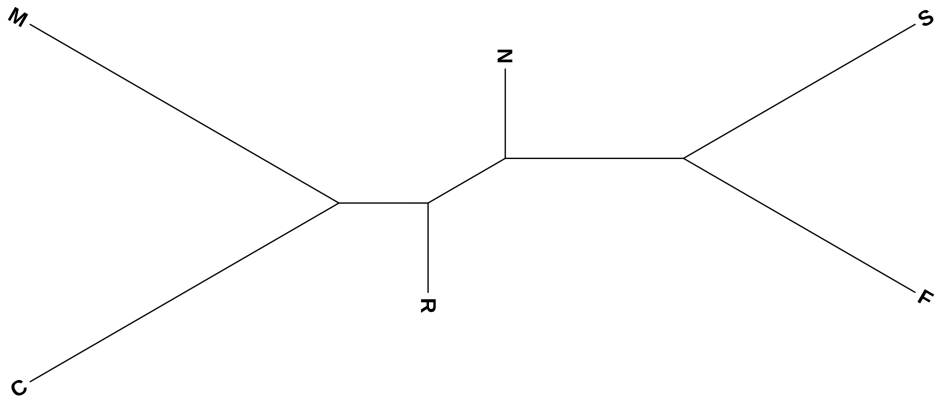
Look at a short explanation of this format on Wikipedia at http://en.wikipedia.org/wiki/Newick_format and after going through the example on the web page, try drawing the tree represented by the following Newick string:

((A,(B,(C,D))),(E,F));

Is this tree specified as rooted or unrooted? (if in doubt, learn more about this concept here: https://en.wikipedia.org/wiki/Phylogenetic_tree#Types). You can test this by uploading our example to the online tool iTOL (https://itol.embl.de/upload.cgi) and letting it visualize the tree (Tip: you can directly write your Newick string in the "Tree text" field).

Part 2: Specifying branch length

The example tree above does not include any information about the length of the branches, that is, they are not scaled. As mentioned in the Wikipedia example, the Newick format allows to add branch information by adding a number representing the distance after the node's name, separated by a colon ":". To test if you understood the concept, try writing the Newick code for a tree with branch lengths proportional to the ones in the tree depicted below. To visualize your tree, you can use the online tool iTOL (select Display mode "unrooted" in the top panel).



Part 3: Common errors in Newick representations

Can you identify the errors in the following Newick strings? Be aware that iTOL might still load them.

a) (A,(E,D)),(C,(B,F));
b) (A:1,D:6,([E:1.2,F:1.2]:1,B:2):0.21,(C:4,G:2.2):2):1);

Exercise 2: Reconstructing phylogenies with maximum likelihood methods

Small theory recap

Phylogenetic trees are used to represent the evolutionary relationships between a group of related sequences/species/genes. For tree construction, several computational methods are available. These are as follows:

1. **Distance-based methods**: Neighbor Joining, UPGMA, etc. These methods require a distance measure between the sequences and build trees based on a matrix of pairwise distances.
2. **Maximum Parsimony**: The tree based on this method minimizes the number of evolutionary steps required to produce the sequences.
3. **Maximum Likelihood**: This method uses an expected pattern of mutational changes from one DNA base to another with probability calculations to find the most likely arrangement of branches that generates the set of sequences.

$$L = p(\text{data} \mid \text{tree, branch lengths, model})$$

The ML algorithm searches different trees and branch lengths to find the $\mathbf{L_{max}}$. It works as follows:

LOOP OVER

Generate tree topology → Optimize branch lengths → Retain if result improved

In this session we will explore the maximum likelihood method for constructing a phylogenetic tree to investigate evolutionary relationships.

Part 1. First steps with RAxML

RAxML (Randomized Accelerated Maximum Likelihood) is a freely available tool for Maximum Likelihood based inference of phylogenetic trees, which we will use for the following exercises. This tool has already been installed in your Renku lab environment. Open the Terminal and acquaint yourself with the arguments available for RAxML by typing:

```
raxmlHPC -help
```

Can you identify which command line arguments are required (not-optional) for RAxML to be executed? You can also check out the RAxML manual for inspiration (https://cme.h-its.org/exelixis/resource/download/NewManual.pdf).

Part 2. Using RAxML on the MFS-1 dataset

In the session “Phylogenetic Trees: Part I”, we used the *MAFFT web server* to compute an alignment for our MFS domain proteins and reconstruct UPGMA and NJ trees with this alignment. To ensure comparability between all trees, we will use the same alignment to compute our RAxML tree. You can find the alignment file “mfs_domain_proteins_aligned_taxnames.fa” (and the newick files for the UPGMA and NJ trees) by going to Bio334 → 05_maximum_likelihood_phylogenies → exercises → files in your Renku lab environment (Tip: if you don’t see the Bio334 folder, type `git clone https://github.com/meringlab/Bio334.git` into the Terminal). Note that we also replaced whitespace characters in all sequence identifiers with “_” because RAxML truncates identifiers with whitespaces. Using this modified alignment file, now try performing a maximum likelihood inference with the following parameters:

- Input file: mfs_domain_proteins_aligned_taxnames.fa
- Output file extension: mfs.auto.txt
- Substitution model: PROTGAMMAAUTO
- Random seed: 123

Unlike UPGMA and NJ, ML computations are very expensive, so even this small tree might run for a few minutes. While waiting for the results, look at the RAxML manual (see link above) to find out more about the substitution model we used.

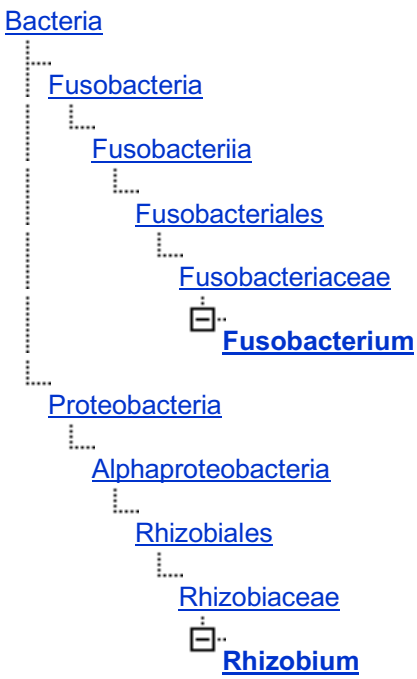
Part 3. Visualize the RAxML results on iTOL and root the tree

When the computations are finished you will find the final tree in RAxML_bestTree.mfs.auto.txt. Visualize this tree with iTOL.

You should now see a rooted tree. Note that the root has been **randomly selected**. A more realistic view is the un-rooted tree, which you can select at the top of the "Basic" tab. In order to root, we need to click on a branch and select *Editing* > *"Tree Structure"* > *"Reroot the tree here"*

Before you apply any rooting, take a moment to think where it would be best to place the root of the tree. You can help your hypothesis by looking at the Taxonomy of the species using the "Common Tree" feature of the NCBI (https://www.ncbi.nlm.nih.gov/Taxonomy/CommonTree/wwwcmt.cgi).

By entering and adding a few of the species names, you will obtain a rough idea of which domain they belong to. Click on the plus to visualize the intermediate levels, as has been done for "Fusobacterium" and "Rhizobium":



After rooting the tree, try to compare the resulting tree to the UPGMA and NJ trees (the files mfs_domain_proteins_aligned_taxnames_upgma.newick and mfs_domain_proteins_aligned_taxnames_nj.newick, respectively), but make sure to also root these on the same branch. What are the main differences you can see: How do the branch lengths compare in the ML version of the tree? Do the clades agree?

Exercise 3: HIV Case Study

Data Set Description

In this exercise you will analyze the phylogenetic relationship between HIV-related viruses from humans and monkeys: The "Pol" gene, which is present in the genome of all these viruses, encodes for polypeptides important for the viral life cycle. The file Pol21.fa (Tip: go to Bio334 ➞ 05_maximum_likelihood_phylogenies ➞ exercises ➞ files) contains a data set consisting of 21 different Pol-polyprotein sequences from HIV1, HIV2, chimpanzee SIV, *Sooty mangabey* SIV, and HTLV-1.

Part 1. Sequence alignment

As every phylogenetic tree building software requires a multiple sequence alignment (MSA) as an input, we will first generate the MSA using *MAFFT web server* (https://mafft.cbrc.jp/alignment/server/). Visually check the alignment on the results page (Is it compact? Can you spot a sequence that aligns worse than the others?), and save the alignment as Pol21.aligned.fa by downloading the link saying "Fasta format".

Part 2. Tree reconstruction with RAxML

Next, for constructing the maximum likelihood tree, run the following command:

```
raxmlHPC -s Pol21.aligned.fa -n pol21.txt -m PROTGAMMAHIVBF -f a -N 10 -x 123 -p 123
```

NOTE: be aware that copying the command can introduce symbol artifacts which sometimes are very difficult to spot, for best results quickly re-type the command yourself

While the program is running, can you identify what input parameters are we using?

-s
-n
-m
-f
-N
-x
-p

Which substitution model is used? Why could this parameter be particularly important for HIV sequences? More information on this topic here: https://doi.org/10.1371/journal.pone.0000503

How does the -f parameter affect the algorithm RAxML is using and why does RAxML provide different algorithms? Finally, what are seed numbers and what role do they play in stochastic algorithms? (google is your friend :-).

NOTE: The execution of this command can take a few minutes more, feel free to take a break.

Once the program stops running all the output files have been created in the RAxML directory. Open RAxML_info.pol21.txt to identify the number of bootstraps performed for our tree. What is bootstrapping? What makes bootstrapping necessary and what are its limitations?

Part 3. Tree visualization with iTOL

Load the Maximum Likelihood tree with bootstrap support measures (i.e. RAxML_bipartitions.pol21.txt) in iTOL. This time, we will focus on visualizing the bootstrap measure that was added in this exercise:

a) Play with some options in the "Basic" tab. How can one increase the branch thickness to make the tree more readable?

b) Now to visualize the bootstrap values. In the “Advanced” tab, choose to “Display” the Bootstraps/Metadata section. Explore the options of the new sub-menu. Can you color the branches according to their bootstrap support? What do red branches symbolize in this representation?

Part 4. Rooting the tree with an outgroup

Now we will learn to make a rooted tree with the help of an outgroup. We know from other sources that the lineage leading to HTLV branched off before any of the remaining viruses diverged from each other. Therefore, the root of the tree connecting the organisms being investigated must be located between the HTLV sequence (the "outgroup") and the rest (the "ingroup"). This way of finding a root is called "outgroup rooting".

a) Use the re-root option and root it after selecting the HTLV branch. You may want to also adjust the horizontal scaling factor (tab “Advanced”) to see the branching within clades more clearly.
b) Can you describe the evolutionary relationship depicted by the tree? What are the sister clades of HIV1? Which HIV type is the virus from *Sooty mangabey* most closely related to?
c) What do high bootstrap values mean?
d) How can you asses the robustness of the tree using the bootstrap values? What likely divisions do the high bootstrap values support?