Tree Building Guide

Tree building is kind of an art. Remember how we said that there were MANY different multiple alignment programs? There are many different options for every step in making a tree (phylogeny.fr is great because it describes these options and allows you to swap between them). This variability stems from the problem that making a perfect tree by trying all the options is computationally impossible. All programs must reduce the problem to being feasible, requiring assumptions and choices to be made. Each one makes different assumptions in its algorithms, and each performs well in different situations.

Below is a guide for making phylogenetic trees.

# What alignment to use?

**Goal:** Align sequences so that homologies are in line and all sequences are the same length with gaps.

**Purpose:** Allow for future analyses, gather information about the sequences as a whole (mutation points, level of divergence, etc.)

**Challenges:** Variable input, variable divergence, how do we know what is “correct” when alignments aren’t a real thing (they are a concept we use for analyses, they don’t exist in nature)?

*Alignment requires a lot of decisions!* Pairwise alignments are easier, but still…

Which is closer to “graffe”? Graft? Giraffe? Graf? Raffle?

String matching is a computer science problem that is usually approached by finding the minimal edit distance between two strings.

Giraffe Graf-- Graft-

| ||||| |||| ||||:

G-raffe Graffe Graffe

In biological systems:

ATCG-ATACGAGGTA

||| |||| | ||

ATCCAATACTA--TA

However, decisions are still needed to determine how much each change is “worth”. Are insertions and deletions weighted the same as substitutions? Are all substitutions equal (transition versus translation)? What about protein changes (Dayhoff versus BLOSSUM versus PAM)? Is one gap better than two (gap open penalties)? Do the sequences have to be the same length (global versus local)? Are substitutions weighted differently if they are close together versus further apart (gamma)? These are all addressed to different degrees in different programs and usually adjustable with options.

Additionally, multiple alignment makes this problem even more difficult. Typically, only conserved sequences will produce a good multiple alignment. Multiple alignment is also very much based on pairwise alignments, meaning the order of consideration of sequences has a heavy impact on the alignment output. It is impossible to calculate all options and calculate the score for each one, as the number of calculations scales exponentially.

Because of this, different heuristics and algorithms are used in computer science to make these problems solvable. Because these heuristics are built on different assumptions about the data (size, amount, variability), different programs are used for different data types. We will continue talking about the most common (sequence alignment), but know that different approaches are used for mapping short reads to a reference.

## Different ways to deal with challenges:

### Basic, string based solution (Clustal, MUSCLE):

These programs seek to maximize alignment score, more or less reducing the number of gaps and mismatches between strings, without biological guidance for the most part. These are fast algorithms, but basic solutions and not always accurate in the face of horizontal transfer, variable domains, etc.

### Incorporate biological information to guide alignments (T-coffee):

These programs seek to maximize phylogenetic information by penalizing mismatches between chemically different amino acids or nucleotides, decreasing number of gaps that are individual to a sequence, among other things (if you are curious for the details, look up the T-coffee paper!). These algorithms are more accurate, but take longer to compute.

### Probabilistic Concordance (ProbCons):

This program looks for the best model based on the consistency between all algorithms, including MUSCLE and T-coffee. Basically, it is assuming that multiple sources of independent evidence is more likely to occur in good alignments. It is currently no longer an option in Phylogeny.fr. You can use it by searching for “Prob-Cons server” and then pasting the alignment other websites give you into any of the tree builders on phylogeny.fr. This is a slow program.

All are explained by scrolling over the names on the phylogeny.fr page.

### For more on specific software:

https://en.wikipedia.org/wiki/List\_of\_sequence\_alignment\_software

### For more on algorithms:

http://www.amazon.com/Sequence-Alignment-Methods-Concepts-strategies/dp/0520271319

# What alignment curation to use (optional but important step)?

**Goal:** Improve the alignment by removing loci that are uninformative (random noise) and sequences that are missing too much information.

**Purpose:** Decrease computational time in tree building, improve final tree.

**Challenge:** Replicability due to variation in alignments, variation in sequence similarity, biological information such as shared domains.

Gblocks is a common option. It is a tool I like to use since it makes trimming and cropping sequences consistent every time. It reads the alignment produced and determines poorly aligned and highly divergent regions (i.e. regions with little or confusing phylogenetic signal) and removes them. However, if the sequences are too divergent, it will fail out. At this point, I would reconsider the dataset or choose not to use curation and move on. Information here: http://molevol.cmima.csic.es/castresana/Gblocks/Gblocks\_documentation.html

Additional manual curation may be required, to remove sequences with too much missing information or adjust anything that is really weird. Be sure you can justify anything you do and that you are consistent across all sequences and loci – needs to be replicable!

NOTE: Some tree building programs request or offer to take a model. JModelTest will look at your alignment and determine the likely pattern of evolution that it has (are there a lot of gaps, are there a lot of transversions, etc). These parameters help tree builders make decisions about where to put things. Phylogeny.fr can do this for you.

## How do I know an alignment looks okay?

This is a hard question. Generally, before attempting to make a tree based on an alignment, you want to look at the alignment. Think about what you expect from your data: do you expect to see large insertions (exon shuffling for instance?), fragments, etc? If not, a large number of –‘s may indicate a poor alignment and that more thought is needed before moving on. Removing poorly aligning sequences is often a requirement!

# What software to use to build tree?

**Goal:** Build a tree, summarize the alignment information, add statistical support to our findings.

**Purpose:** Needs to be done before we can make visualization

**Challenges:** Tree space is too huge and complex as to search everything. Different algorithms do different things to reduce the search space. For reference:

Given n sequences | # rooted tree for n >2: (2n-5)!/(2n-3(n-3)!)

i.e.; n = 5  105 rooted trees, which is searchable.

n = 10  34,459,425 trees, which is not really searchable.

Different solutions to heuristic tree search (tree search with assumptions that make it computationally possible) exist and are enumerated below.

## Distance based: Neighbor joining – i.e.: BioNJ, Trex

These algorithms assume that sequences with lower edit distance are closer related. Basically, you join the two closest related sequences with a node, and calculated the distance between that node and the next closest sequence. It doesn’t take into account homologies or really much in the way of biological information – just sequence similarity. Generally, this solution performs poorly for anything with complex evolution or horizontal gene transfer. Statistical support is gleaned from subsampling the sequences and rebuilding the tree, then comparing it to the original. A significant use of these algorithms is non-sequence data (any distance measure can be made into a tree).

## Character based I: Maximum Likelihood – i.e: see below

These algorithms assume sequences with more shared homologies are closer related. These algorithms generally start with two sequences connected with a node and iteratively add sequences to the tree and attempt to optimize its best location. As a result, ML is sensitive to order of sequences added. To get around this issue, trees are rebuilt hundreds to thousands of times with randomized input order. The branches that appear frequently are considered to have highest likelihood of being the correct relationship. By adding branches one at a time, the program limits the number of searches it has to do each time. It is still pretty slow with huge data sets.

### RAxML

Randomized Axelerated Maximum Likelihood

Derived from Phylip Starts a basic tree, adds a sequence.

Each time it adds a sequence from the alignment, it attempts to optimize its location in the tree. Randomizes the start set and the order of addition of sequences to obtain bootstraps.

Benefits: Constantly updated, pretty good with optimization

Cons: Limited in model of evolution used.

Requires .phy format

Tutorial: <http://bodegaphylo.wikispot.org/RAxML_Tutorial>

### Garli

Genetic Algorithm for Rapid Likelihood Inference

Similar to RAxML, but a little simpler to use.

Benefits: More flexibility in options, similar run time to RAxML

Cons: Not terribly common use, requires editing a runfile each time

Requires Nexux format

Tutorial: <http://bodegaphylo.wikispot.org/Maximum_Likelihood_%28GARLI%29>

### PhyML

Phylogenetic estimation using Maximum Likelihood

Benefits: Large number of substitution models, online execution - fast and easy(Phylogeny.fr ), allows for quick "SH-like" branch support as a first pass look to troubleshoot.

Cons: Bit harder to run on command line.

## Character Based II: Baysian – i.e.: MrBayes

These programs are similar to ML tree building in that they seek to join sequences that have shared homologies. They constrain space by determining a prior constraints, either by inputting a species tree to guide it, or calculating one based on thousands of calculations (monte carlo markov chains). This is a newer algorithm, and the exact parameters aren’t well established, leading to debate about the outputs. Most commonly, you will see these values mapped onto a ML tree. Statistical support is in the form of posterior probabilities, which are a Bayesian statistical concept.

NOTE: Many of these programs will test for a model of evolution before running the tree. However, some may ask you to choose a model. To find which best fits your data, you can input your alignment into programs such as JModelTest or MrModelTest.

## Output from Tree Building: The Newick File

Here is a short introduction to the format:

http://www.megasoftware.net/mega4/WebHelp/glossary/rh\_newick\_format.htm

# Tree Visualization

**Goal:** Make a pretty tree.

**Purpose:** Visualize Phylogenies for further analysis, make it easier to interpret data.

**Challenge:** Many options exist for this step because editing and outputting options vary between programs.

General options: FigTree, TreeDyn, Dendroscope

# Further analyses

## PAML

Phylogenetic Analysis Using Maximum Likelihood PAML is a program that intakes PHYLIP 4.0 format which can include tree data. This program can be used to make trees, but it’s older and clunkier, but it does allow for the analysis of selection parameters, omega, etc. Here is a good overview/tutorial: <http://abacus.gene.ucl.ac.uk/ziheng/data/pamlDEMO.pdf>

Mega

These same analyses can also be done in mega, which can be used to import trees and alignments, and will calculate selection, neutrality, etc. It is the more user friendly option of these two, and then one I have preferred. Tutorial: <http://www.megasoftware.net/tutorial.php>

CompPhy

New program that allows you to compare trees and easily co-edit them across the internet with collaborators. I have not used it, but the publication and website were very pretty.

In summary

What I do: Align with probcons (or whatever you choose), Gblock to remove noise consistently (using default parameters), model test, use PhyML to build a tree based on that model. These are all done online usingphylogeny.fr. Then I use Mega to import my alignment and tree, and test for selection. If I need to make a huge tree than phylogeny.fr cannot handle, I use CYPRESS (online job submissions) or RAxML on a cluster.