UPPSALA
UNIVERSITET

AutoPhylo

an automatic pipeline for phylogenetic analysis

Erik Lundin

Degree project in biology, Bachelor of science, 2010
Examensarbete i biologi 15 hp till kandidatexamen, 2010
Biology Education Centre
Supervisors: Sandra Baldauf and Ding He

Summary

The task of constructing a molecular phylogenetic tree consists of finding homologous sequences, making a multiple sequence alignment, perhaps removing gaps and ambiguous positions in the alignment and finally phylogenetically inferring the tree with various evolutionary models. Often there is a need to refine the tree by removing inappropriate sequences. For each step of this process there is a tool to accomplish the task.

Starting with a sequence of interest, BLAST (Basic Local Alignment Search Tool) is used to find homologous sequences from various databases. Then multiple sequence alignment programs such as MUSCLE or Clustal can be used to align the sequences. In the alignment there are often regions of gaps and ambiguous positions that can be identified and removed with programs such as GBlocks. Finally, using the alignment a phylogenetic tree can be reconstructed using selected methods and models. Depending on the scientific goals and data, this process generally needs to be repeated several times in order to "refine" the tree. To construct a tree of correct phylogeny, "true" homologous positions in an alignment must be used. To further complicate tree reconstruction there are technical problems such as long branch attraction (where fast evolving sequences cluster together even if they are unrelated) and horizontal gene transfer (where cells that can be unrelated exchange genes) that could mislead the phylogeny.

At present there is no effective program that is sophisticated enough to correct these kinds of problems without careful manual examination. However, many of these steps are simple and repetitive. It is the goal of bioinformatics to automate as many of these simple tedious steps as possible, in order to allow large amounts of data to be processed quickly and accurately. This report describes the construction of a tool that streamlines the phylogenetic tree reconstruction process. The tool, named AutoPhylo, identifies and retrieves sequences from NCBI (database collection) or a user-defined local database via BLAST searches. These sequences are then used to construct a tree that can be examined with a graphical user interface (GUI). The GUI allows the user to identify and remove unwanted sequences in order to refine the tree. The sequences are retrieved in groups that have one or more queries that limits the selection to specific species, genes or others valid NCBI queries. Some tests were applied to show that the program is useful. The program can be obtained from http://code.google.com/p/autophylo/.

1. Introduction

One way to show the evolutionary relations between species is to use a phylogenetic tree. In the tree, a terminal node will be an extant species, while an internal node represents the hypothetical **LCA** (last common ancestor) of all the species (terminal nodes) above it **(2)**. This will show how different groups are related, and which groups share the most recent common ancestor. Before molecular data were available, the morphology (and/or physiology) of species was used to construct the trees. Nowadays, the amount of molecular data available and the use of computers have shifted the focus towards DNA, RNA and amino acid sequences. The use and development of bioinformatic tools help to ease and speed up the process of handling sequence data. Nonetheless, there will always be a need to construct trees using morphological data, especially when dealing with fossils that leave very little or, more often, no molecular data behind (in 2007 there was a publication about the discovery of collagen from Tyrannosaurus rex **(17)**).

One of the main things one wants to find in trees is the monophyletic groups. A **monophyletic** group is a group that contains an LCA and all of that LCAs descendants. For example, plants are a monophyletic group since the LCA of living plants was a plant and all of that LCAs descendants are plants. However, “Algae” is not a monophyletic group since the LCA of green algae and brown algae was not an alga. In addition, plants descended from green algae are not considered algae. There are two non-monophyletic groupings: paraphyletic and polyphyletic groups. A **paraphyletic** group excludes some taxa that are descendants of the LCA. An example of a paraphyletic group would be the commonly used group “reptiles”, since it does not contain the birds, and crocodiles are more closely related to birds that to the other reptiles. **Polyphyletic** groups are collections of unrelated taxa such, as the use of the group "warm-blooded" which includes birds and mammals but not their LCA or any of its other descendants. Reptiles would not be a polyphyletic group since it contains all of its ancestors up to the LCA while the warm-blooded grouping does not contain the LCA. **(2)**

Phylogenetic trees can be drawn rooted or unrooted. If the tree is unrooted only groups are shown, but the direction of evolution can not be determined. Meanwhile, in a correctly rooted tree the evolutionary history can be traced while an incorrectly rooted tree can give a false picture of the history. For example, a tree of amniotes that is rooted within birds would not show the true history of amniotes. The most common way to get a rooted tree is to use an **outgroup**, which is a group that is known not to be in the group of interest and share a LCA with it, that will be at the root of the tree (an outgroup for amniotes could be amphibians). **(11)**

When constructing organismal phylogenies, the problem of different kinds of homology is an important issue. If a gene in a species is duplicated into Gene 1 and Gene 2, these two genes are **paralogs**. If the species then gives rise to descendants Species A and Species B, then Gene 1 in Species A and Species B are **orthologs,** as are the Gene 2s. When collecting the sequences to use for a tree of species one wants to make sure that the sequences are orthologs. If they are not orthologs the true **species tree** (a tree showing evolution of species) would be different from the true **gene tree** (a tree showing evolution of genes). In a tree of amniotes where the bird sequences are paralogous to the rest of the sequences, the birds would end up at the root in the true gene tree, unlike where they appear in the true species tree. **(11)**

The most common way for genes to transfer is from parent to offspring. This process is known as **vertical gene transfer**. However, genes can also cross between unrelated species by a

process called **horizontal gene transfer**. This can happen in several different ways. In bacteria, DNA can be transferred between cells, for example in the case of the F factor using a pilus. Plasmids can carry different genes that are then transferred to the other cell. The new host cell will then have gained genes from another species that can be quite unrelated. Another means of DNA transfer is viruses, which sometimes pack a part of the host chromosomes instead of the virus DNA. In addition, bacteria may pick up naked DNA by a process called transformation. Horizontal gene transfer can confuse the relationships in a tree since the gene sampled may actually belong to another species than the one that it was sampled from, making the true gene tree different from the true species tree. **(13)**

Due to technical problems a tree can be incorrect even if orthologous sequences are used. Sequences that are not closely related can be clustered together and/or attracted towards the root of the tree. This is called **LBA** (long branch attraction) and happens with fast evolving sequences. If branch length reflects evolutionary change, a fast evolving sequence will be a long branch. LBA might be caused by the limited number of states a position can change to due to mutation. A base can only change into one of the three other bases. When sequences change rapidly, the same mutation can arise at the same sites in different species. This leads to saturation. If LBA happens the tree can give a false image of the evolutionary history. The two main ways to counteract LBA is to use additional taxa in order to break the attraction or to use a more phylogenetic method that makes corrections for possible LBA. **(11) (21)**

1.1. The making of a tree

1.1.1 Obtaining the sequences

To construct a molecular tree the first thing that is needed are the sequences. Today sequences are stored in databases all over the world. There are three main collections of public databases hosted by **NCBI** (National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov/) in the USA, **EMBL** (European Molecular Biology Laboratory, http://www.ebi.ac.uk/embl/) in England and **DDBJ** (DNA Data Bank of Japan, http://www.ddbj.nig.ac.jp/) in Japan. These databases share mostly the same data, but the most commonly used one is NCBI. In NCBI there are databases for proteins, structure, nucleotide, taxonomy, biomedical literature and other things. In addition to the tree main fully public databases, there are also several other important "semi-public" databases such as the **JGI** Genome Portal (http://genome.jgi-psf.org) funded by The U.S. Department of Energy Joint Genome Institute.

The huge (and exponentially increasing) number of sequences in these databases makes the task of finding the sequences of interest extremely difficult without the help of bioinformatic tools. **Entrez (14)** unifies the searching of NCBI databases where limits can be applied, for example searching for only turtle data one can use the limit "turtles[ORGN]". To find sequences similar to a given sequence the tool BLAST (Basic Local Alignment Search Tool) can be used.

BLAST works by dividing the query sequence into "words", *i.e.,* strings of sequence of a defined length. For every word in the query, the database is searched for words in target sequences that have a good enough match (identical or similar) to the query. Once a match is found, the hit is extended (in both directions) while recording the score (a measure of the quality of the match) for every position. When the score drops to a given amount below the

highest score found, BLAST stops extending. The local alignment obtained is called an **HSP** (High Scoring segment Pair). If the HSPs e-value is below a threshold it is kept, where the **e-value** is a measurement of how likely it is to find a sequence with the same score (or with a bigger score) by chance, for the given database. If the score is above a threshold the BLAST algorithm then aligns the HSP using the Smith-Waterman algorithm (see below). At the end of the search, all the HSPs with an e-value below a default or user-defined threshold are kept. Currently there are five main types of searches that can be performed with BLAST (Figure 1). **(3)**

BLASTN	Nucleic acid sequence against nucleic acid sequence
BLASTP	Amino acid sequence against amino acid sequence
BLASTX	Translated nucleic acid sequence against amino acid sequences
TBLASTN	Amino acid sequence against translated nucleic acid sequences
TBLASTX	Translated nucleic acid against translated nucleic acid

Figure 1: The different types of BLAST searches.

For some species there are not that many sequences available. For such species there might be EST sequences available. **EST**s are short parts of **cDNA** (complementary DNA, reverse transcription of mRNA) that have a length of around 200 to 500 nucleic acids. **(22)**

1.1.2. Aligning the sequences

Having obtained a set of sequences, these now need to be aligned. During the time since the LCA of the genes under analysis lived, various substitutions (mutations) have occurred and there can be **indels** (insertions or deletions, since one can not know if a sequence part was lost in one species or gained in another) **(22)**. As a result, it is unlikely that one will be able to just lay the sequences next to each other so that they match. Some parts of a sequences have change little during the time since the LCA. Such parts are called **conserved regions (1)** and should align well with each other. Conservation can occur if mutations reduce the function of the protein. Other parts of the sequence, for example loops in protein structure, might change a lot without loss of function and therefore might align badly.

In comparing sequences, it is important to note that some substitutions are more likely than others. Nucleic acids are made up of residues or nucleotides that are of two types: pyrimidines (C,T and U) and purines (A and G). Since these two classes of nucleotides have quite different structures, substitutions between nucleotides of the same type are more likely than substitutions between pyrimidine and purine. Similarly, in proteins some amino acids have common features and can be exchanged for each other without changing the proteins' function. So some substitutions are more likely and and indicate that the sequences are closely related, while other substitutions are less likely and can indicate that the sequences are distantly related. To account for the different types of mismatches in an alignment, different scoring matrices are used. For proteins these are mainly the substitution matrices PAM and BLOSUM that are derived from empirical data. **(15)**

To align two sequence one generally uses **dynamic programming** algorithms such as Smith-Waterman (local alignment, aligning parts of the sequences) and Needleman-Wunsch (global alignment, aligning the whole sequences). In dynamic programming, the two sequences are placed at the x-axis and y-axis of a grid where the position (0,0) is given the score zero. Now for every position (x,y), right to left and top to bottom, the score F is calculated as the maximum of the three possible:

$F(x-1,y-1) + S(x,y)$; $F(x-1,y) - G$; $F(x,y-1) - G$

where S(x,y) is the mismatch/match score for the position pair (for a protein this might come from a PAM or BLOSUM matrix) and G is the cost of a gap (there can also be a score for extending a gap), see figure 2. The score for the match of the first positions of the sequences is at position (1,1), this enables the beginning of one sequence to be free (unaligned). **(15)**

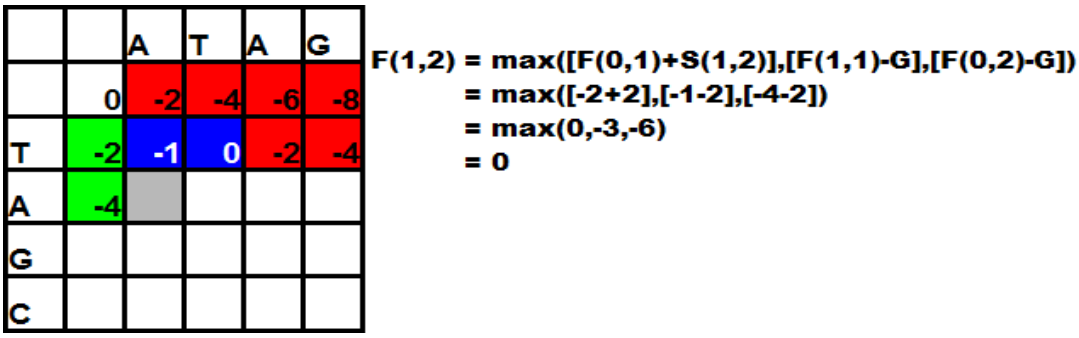


Figure 2: Calculation of the alignment of the two sequences TAGC and ATAG using the dynamic programming algorithm Smith-Waterman. The score for a match is 2, a mismatch is -1 and a gap is -2. Here the score for the position in grey is calculated.

By remembering from which position the F used to calculate the score came from, different paths are formed through the grid. By selecting a position in the grid, an alignment up to that position is gained. In **Needleman-Wunsch** the path starts from the bottom right corner of the grid and works toward the top left corner (figure 3a). This gives an alignment over the whole sequences. **Smith-Waterman** also uses the value zero to calculate F (this gives no position to remember, so the path ends there) so no value will be below zero. The path starts at the position with the highest value and follows that path until it ends at a position with value zero (figure 3b). This give an alignment over a part (that can be the whole) of the sequences. **(15)**

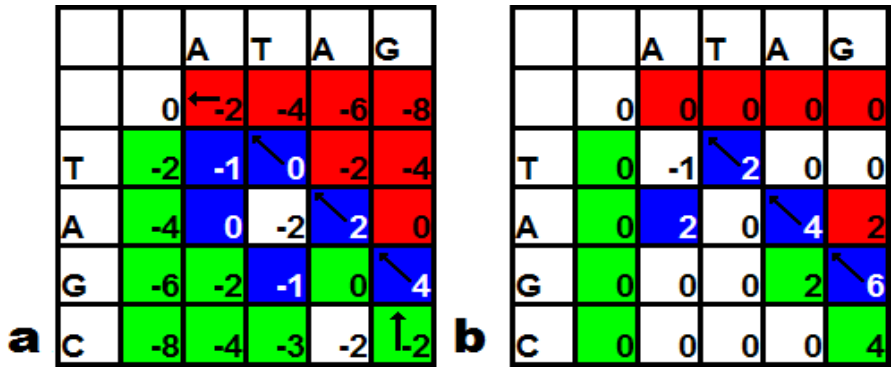


Figure 3: The dynamic programming alignment for TAGC and ATAG. Here the colour mean that position (x,y) was calculated with the value from position (x-1,y-1) [blue], (x,y-1) [green] and (x-1,y) [red]. The score for a match is 2, a mismatch is -1 and a gap is -2. a, is Smith-Waterman; b, is Needleman-Wunsch. The alignment path is marked with arrows in the figures.

If dynamic programming is used for three sequences, the grid becomes a cube, and so on , giving an extra dimension of positions to calculate the matches for each added sequence. This makes the dynamic programming approach computationally expensive and unrealistic. To speed up computations, less accurate but faster algorithms are used. The most common of such algorithms are progressive alignment methods, now most often using iterative refinement. One way to score a multiple alignment is to use the sum of the scores for all pairwise alignments **(22)**.

Progressive alignment begins with pairwise alignments of all sequences. These pairwise alignments are used to construct a guide tree, most often using a distance-based algorithm (see below). Then sequences are added to the alignment according to the guide tree (figure 5). If a gap is added in the already aligned sequences it is added to all sequences. Because of this, any mistakes early in the alignment will stay because gaps are not removed or changed. A popular program for progressive alignment is **Clustal** that take into account that different scoring matrices are suited for different evolutionary distances. **(19)**

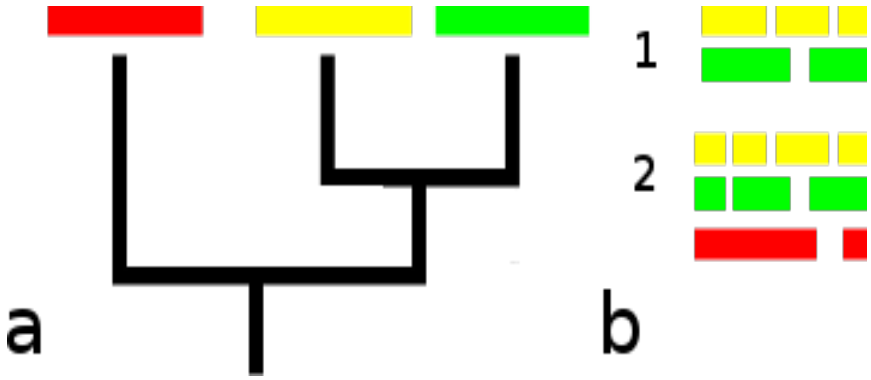


Figure 4: A simple example of progressive alignment. The guide tree (a) show that first the yellow and green sequences are aligned (b1). Then the red sequence is added (b2). Note how the gaps introduced when adding the last sequence are in either the last sequence or in all the previous (b2).

Iterative refinement is a major improvement of simple progressive alignment, because it allows the gap positions to be "refined". The program **MUSCLE** works by first constructing a guide tree with the help of a distance-based method. The distance when making the guide tree is a *k*mer distance, which is proportional to the fraction of sub-sequences of length k that the sequences share. From the guide tree an alignment is calculated. This alignment is used to give new pairwise distances that are again used to make a guide tree. If the branching order in the new guide tree differs from the old, a new alignment is made. In the refinement part of MUSCLE (figure 5) the sequences are divided into two sets by using the sequences in a sub-tree and the remaining sequences. These sets are then aligned against each other to give a new multiple alignment. If the score of the new alignment is better, it is kept; otherwise it is discarded. This refinement process is repeated until the score of the alignment converges or until a user defined limit is reached. **(6)**

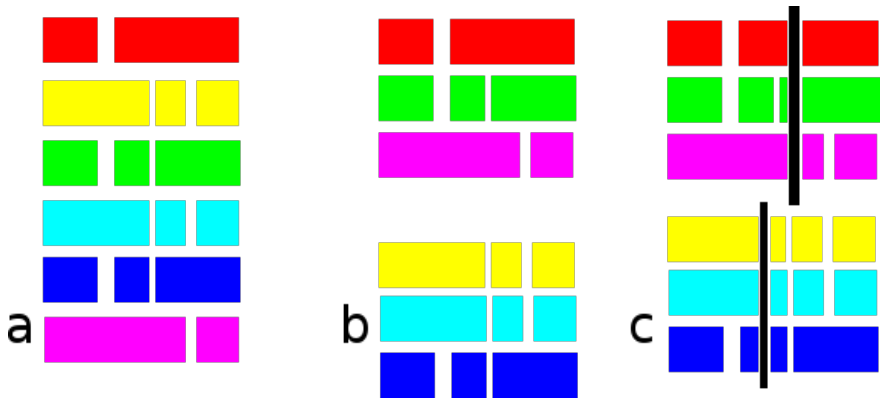


Figure 5: Refinement part of MUSCLE in step n. The set (a) is divided into two sets (b). The sets are then aligned against each other (c), note the new gaps marked with black lines. The alignments in two sets are then joined to one set.

1.1.3. Removing regions

Regions that are divergent or badly aligned can affect the resulting tree topology. Removing such regions might improve the tree, when positions that are more likely to show the true evolutionary history of the sequences are used. A program that automatically removes bad positions is **GBlocks**. GBlocks selects blocks in the sequence according to some criteria with different thresholds. **(4)**

1.1.4. Constructing the tree

Given a model, the most accurate way of constructing a tree is to test every possible tree for how well the alignment fits. For trees with just a few sequences this is possible. However, for large data sets testing every tree will not be practical or even possible. Instead methods that only test a subset of all possible trees in a reasonable time are used. There are different types of algorithms to use, such as distance-based and maximum likelihood.

Distance-based (12) methods try to construct a tree based on the pairwise distance between sequences using an evolutionary model. An evolutionary model is a set of rules about how sequences evolve. The evolutionary model is important, because if it is too simple or unrealistic, the tree can suffer, for example from long branch attraction. One method is to use the pairwise distances to find the sequences that have the least distance from each other. Those sequences are grouped and treated as a sequence when the pairwise distances are computed again. This is repeated until all sequences are grouped and there is a tree. A program that use a distance-based method is **QuickTree (8)**.

Maximum likelihood methods are based on finding the tree with the highest likelihood given the alignment, that is, how likely it is that the tree would give the alignment. To find the best tree ideally one must test every possible tree but this is not practical. Instead, one uses algorithms that test a subset of all possible trees. This is usually done by things like tree-rearrangement, where a tree is rearranged into some child trees. Among those child trees, the tree that has the highest likelihood is selected, and the process is repeated until there are no better trees to be found. **(16)**

The most commonly used method to test if the tree is probable is bootstrap analysis. Bootstrap is formally defined as “random sampling with replacement”. In **bootstrap**, positions in the alignment are chosen at random (a position can be chosen more than once) to build a random test set of the same size and general composition as the original data set. From the test sets trees are constructed. For every branch a bootstrap value is given that is equal to the percentage of trees in the test set that have a branch that divides the tree in the same way. **(12)**

1.1.5. Why construct trees

To understand why a species looks and behaves like it does, knowing its evolutionary history helps. Why do fungicides not work on oomycetes? because they are not fungi! Many questions can be understood by knowing which are the closest relatives. By constructing trees, information is gained on what the groups are and how the groups are related. Features in one species can be shared in relatives, and a useful protein may exist in other variants in the relatives.

1.2. The eukaryotes

Eukaryotes are a monophyletic group of organisms. One of the important distinguishing features of eukaryotes is that they have a membrane-bound organelle that contributes most of the ATP production for the cell, although there are a few important exceptions to this (see below). This organelle is called the mitochondrion, and originates from a freely living bacterium among the α-proteobacteria. This bacterium is thought to have been engulfed by the early eukaryote, and have survived in eukaryotic cells as an ATP factory. The bacterium lost most of its genes, leaving a tiny genome that is not able to produce all the proteins that are needed to survive. Instead, most of the proteins are produced from genes residing in the nucleus. Many of these proteins are thought to have originated by movement of genes from the original mitochondrion into the nucleus. The proteins synthesized from these genes are now synthesized in the cell cytoplasm and after their synthesis they are transported into the mitochondrion. **(10)**

1.2.1. Finding the tree of the eukaryotes

There have been many attempts to find the true tree of the eukaryotes. The earliest molecular trees were constructed using SSU rRNA. These painted a picture of a tree with unicellular eukaryotes as a base of the tree and the multicellular eukaryotes at the crown **(18)**. This appeared reasonable at first, given that the most “ancient” branches at the base of the tree were organisms that appeared to have no mitochondria. This suggested that mitochondria might have been acquired late in eukaryotic evolution. This “base-crown” tree turn out to be caused by long branch attraction of the “base” sequences arising from their fast evolution, probably due to the organisms' parasitic lifestyle. In fact, one of the eukaryote groups in the “base” was shown to belong to the fungi **(7)**. In addition, it has been shown that all these basal species have some kind of an organelle that was derived from a mitochondrion.
Currently there is a project at the Department of Systematic Biology at Uppsala University to find the root of the eukaryote tree by using proteins of mitochondrial origin **(20)**. That approach has the advantage that the entry of the mitochondrion is a later event than the eukaryotes' split from the rest of life. Previous attempts to root the eukaryote tree usually used archaea as the outgroup.

1.3. Programming

To write a program is to tell the computer what to do. For every type of processor there is an assembly language. In assembly language it is possible to write fast code, but there are no abstractions to make the coding easier. For most programming tasks higher level languages are used, since they give the programmer abstractions that lets the programmer focus on the problem to be solved. One such high level programming language is Python (http://www.python.org).

For Python a lot of code have been developed. One useful library for python is Biopython **(5)** that helps the user to with bioinformatical tasks such as doing BLAST searches in NCBI databases and parsing various file formats. This makes it easier for the programmer, letting the programmer focus on the problem.

Environment for Tree Exploration (ETE) **(9)** is another useful tool written in Python. ETE can be used to represent the tree as a data structure that ETE can manipulate. This also includes a graphical interface where the user can select parts of the tree to remove.

To build a graphical interface for the user there are several alternatives available. One such alternative is Qt (http://qt.nokia.com) that is usable in Python with PyQt (http://www.riverbankcomputing.com/software/pyqt).

1.4. Aims

The purpose with this degree project was to develop a program that is able to fetch sequence data from the main sequence database at NCBI, a local database or the specialized EST database at NCBI. The sequences should be automatically aligned and a tree constructed from the alignment. It should then be possible to prune the trees manually to get rid of long branches or paralogous sequences followed by automatic re-alignment and tree re-construction with the remaining sequences. This program was developed within the project to find the root of the eukaryotic tree **(20)** and will help with the processing of large amounts of data (see group file in appendix B).

2. Results

2.1. AutoPhylo

2.1.1. Starting a project

AutoPhylo can start with a new project or a previously existing project. To every project there are customizable groups with members associated. Membership is designated in this report as organisms that are used as legitimate entrez queries. E.g., the group Amoebazoa can have the members Dictyostelium[ORGN], Physarum[ORGN] and Acanthamoeba[ORGN]. To every group it is possible to assign a colour that makes the tree visually convenient to work with. The groups can be added by hand or loaded from a file (see appendix B for an example) containing the groups (figure 6).

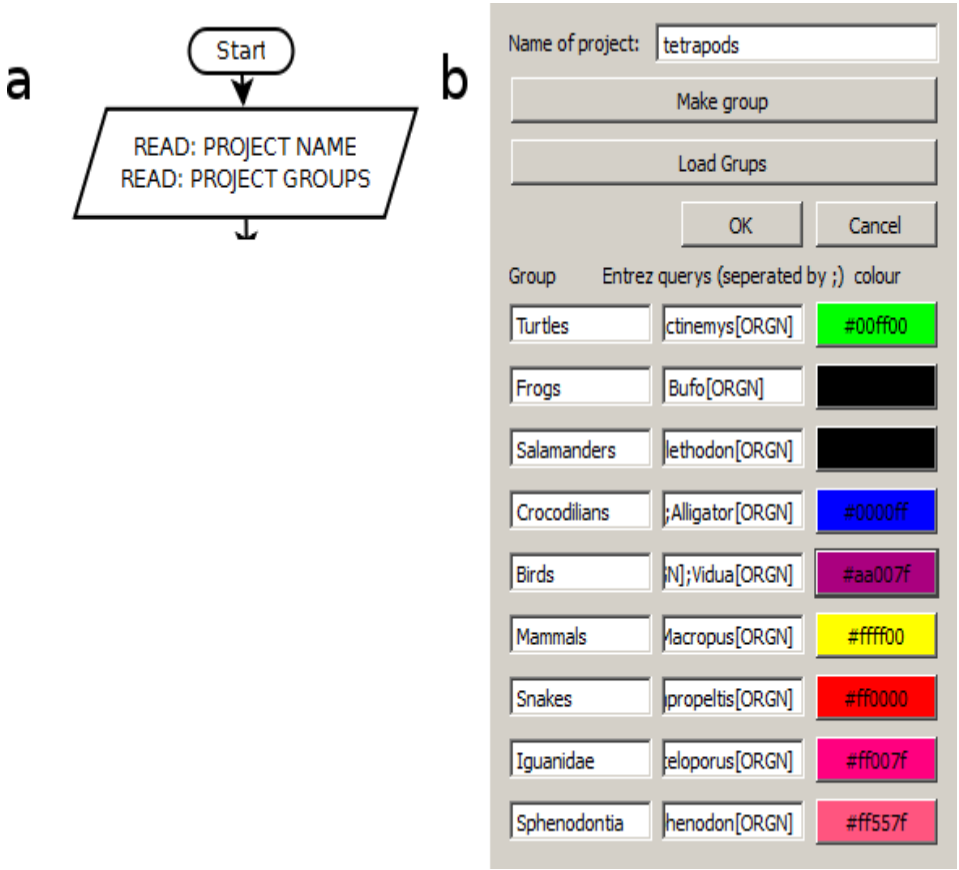


Figure 6: a, The new project part of the flowchart for AutoPhylo. b, A new project for AutoPhylo with the name Tetrapods with several groups with associated colours. Every group contains one or more entrez querys, for example the group Frogs have the member Bufo[ORGN] and the colour black.

2.1.2. Finding sequences for the tree

A tree within a project in AutoPhylo is started from a seed sequence that is used to find similar sequences (hopefully homologous sequences) for all designated species using BLAST (figure 7).

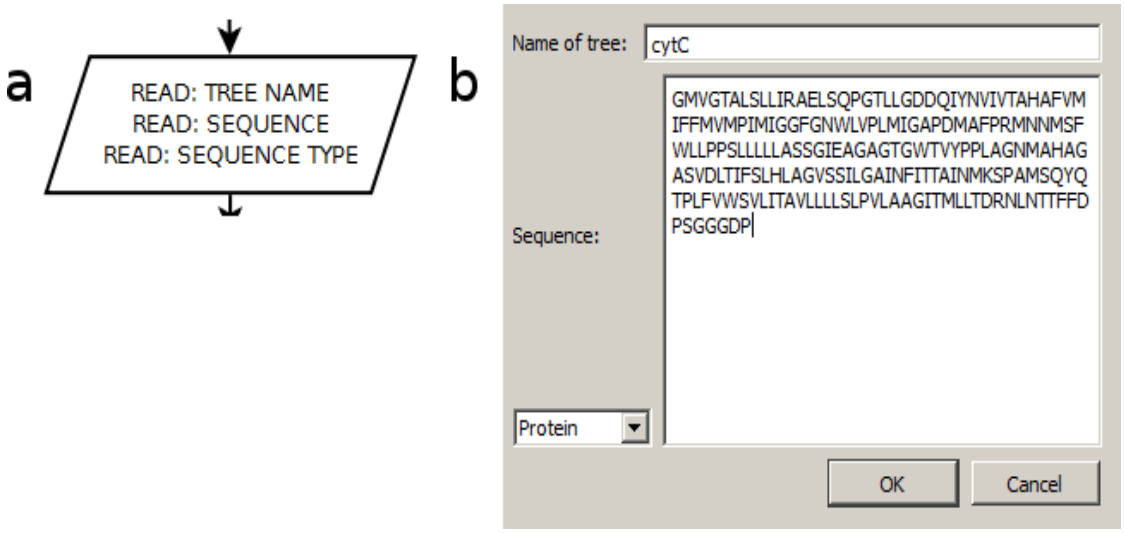


Figure 7: a, The new tree part of the flowchart for AutoPhylo. b, To construct a tree, a seed sequence is used, in this example a cytochrome c protein is used to make a tree named cytC.

For every member AutoPhylo will do the following. It will blast on NCBI using the seed sequence and the member as an entrez query. If no sequences are found AutoPhylo will BLAST on a local database (in this case, downloaded JGI databases) that are supplied by the user. The user has to program the interaction with the database. If AutoPhylo still has not found any sequences it will search in NCBI's EST databases again using the member as an entrez query. Since ESTs generally are very short sequences, the program will try to concatenate the selected ESTs with the method described below if it does not find ESTs.

The EST BLAST hits are sorted according to where they start on the query sequence. Then the first in the list is selected as current and if the next does not intersect with it, the current is saved and the next is made the current. If the next does intersect with the current the one with the lowest e-value is made current. This is repeated until all ESTs have been tested. Then the part of the saved ESTs that are aligned by BLAST are concatenated (figure 8). If any member still lacks sequences, the user is informed of which members these are.

2.1.3. Constructing a tree from the sequences

Now AutoPhylo use the retrieved sequences to construct a tree. First the sequences are aligned, then then bad positions are removed (this is optional) and then the tree is constructed. The constructed tree is then displayed for the user (figure 9). In the graphical user interface the user can choose to remove sequences (figure 10) and reconstruct the tree with the remaining sequences.

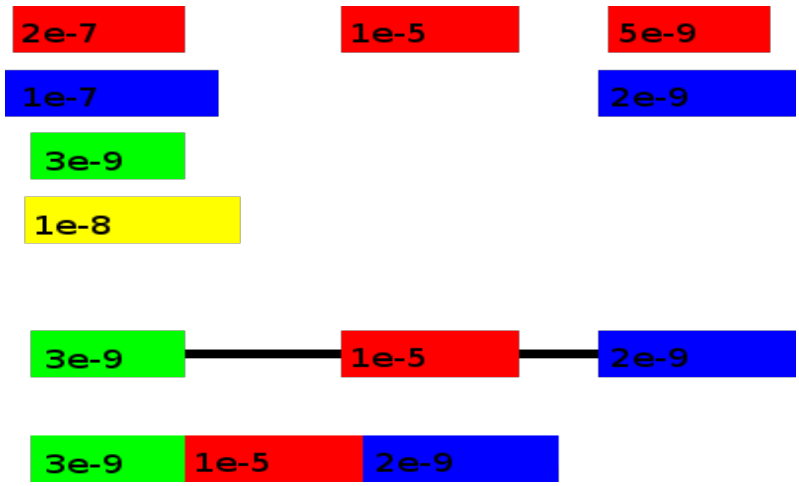


*Figure 8: How AutoPhylo choose ESTs. The ESTs are sorted according to where on the query sequence they have a hit. The values are e-values (1e-1 = 1*10^{-1}). The first EST (blue with e-value 1e-7) is selected as current and it intersects with red (e-value 2e-7). The blue has a better e-value than the red and stays current. Next it intersects with yellow (e-value 1e-8) and yellow is selected as current. Current (yellow) intersects with green (e-value 3e-9) and green is selected as current. Now green does not intersect with the next EST so green is saved and the next (red with e-value 1e-5) is selected as current. This current also does not intersect with the next EST so it is saved and the next (blue with e-value 2e-9) is selected as current. The current has a better e-value than the next and stays current. Now there are no more ESTs so current is saved. Now the saved ESTs' aligned with the query sequence are concatenated (gaps are removed) in order and so a sequence (green 3e-9, red 1e-5 and blue 2e-9) is constructed.*

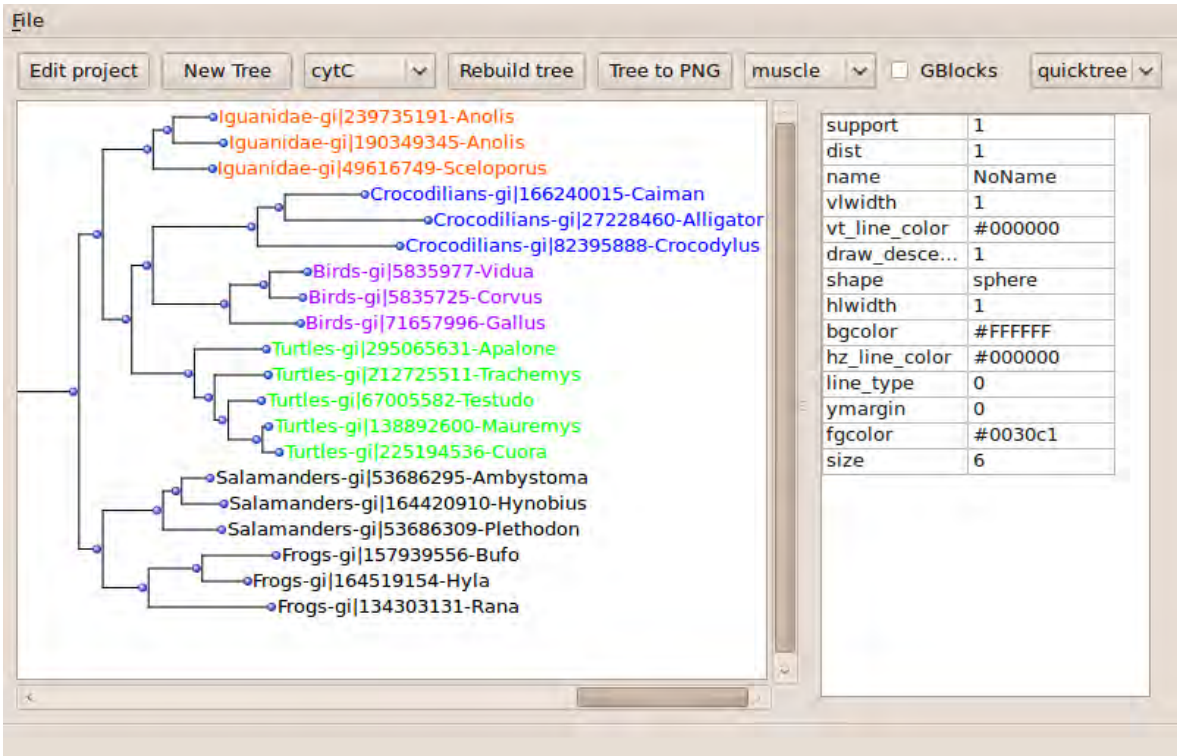


Figure 9: The AutoPhylo program with a tree for the cytochrome c protein with the groups Frogs, Salamanders, Iguanidae, Turtles, Crocodilians and Birds.

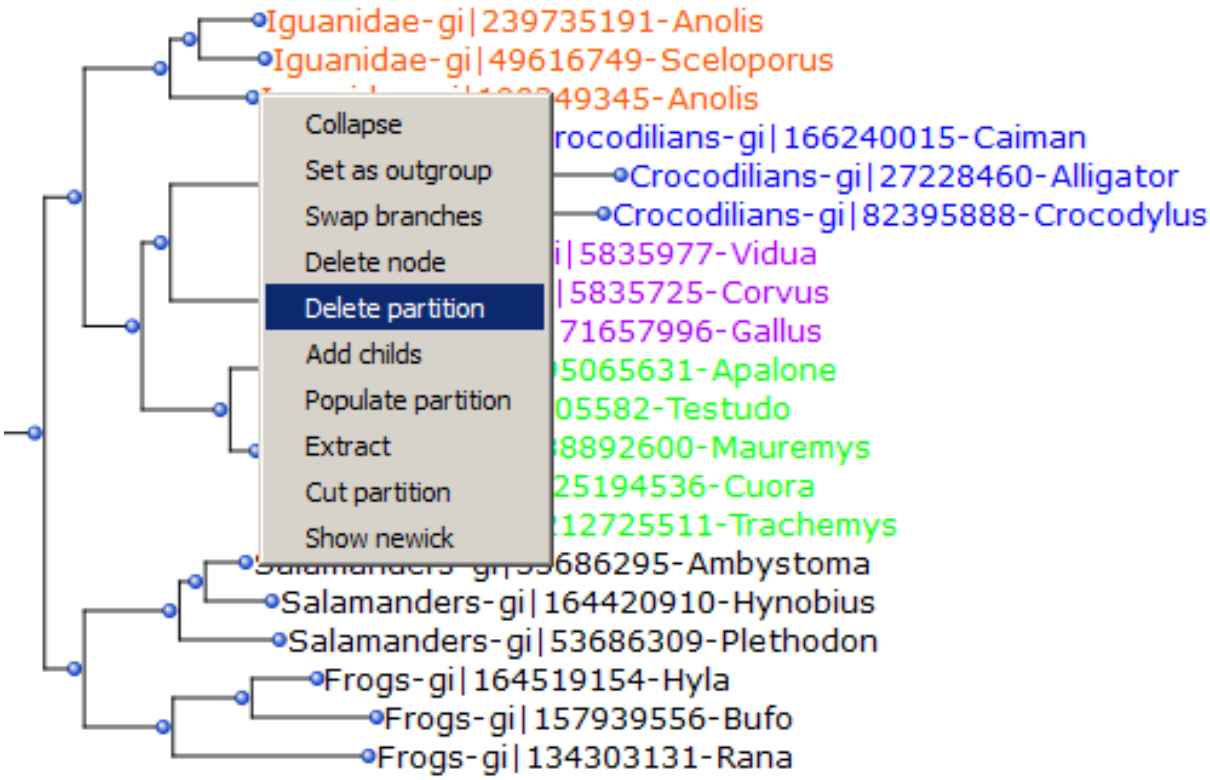


Figure 10: Right clicking on a node in AutoPhylo. Here the command Delete partition (remove the node and all of its descendants) is highlighted.

2.2. Testing autophylo

Any program that is to be used must be tested to make sure that it does what it is supposed to do under different conditions. A variety of tests were performed to test various parts of the program to make sure that they worked as they where supposed to. The tests shown here used the following parameters: The e-value cut-off for both the first NCBI BLAST and the EST BLAST was set at $1*10^{-5}$, sequences were aligned with MUSCLE using no additional parameters, GBlocks was not used to edit the alignment and QuickTree with boot parameter set to 100 was used in all tests. The sequences were retrieved from the NCBI non-redundant (*nr*) database for the first NCBI BLAST and from *est* database for the EST BLAST.

2.2.1. Fetching sequences

The first part to test was to see if the fetching of sequences worked as it was supposed to do. The first test was to BLAST the cytochrome b protein (GI number: 295442711) using the corresponding sequence from the flowerback box turtle (*Cuora galbinifrons*) as query with the target group set to human (member: homo[ORGN]). To succeed in this test, AutoPhylo had to get the same sequences as the BLAST on NCBIs homepage. The number of hits were limited to the top (best) ten.

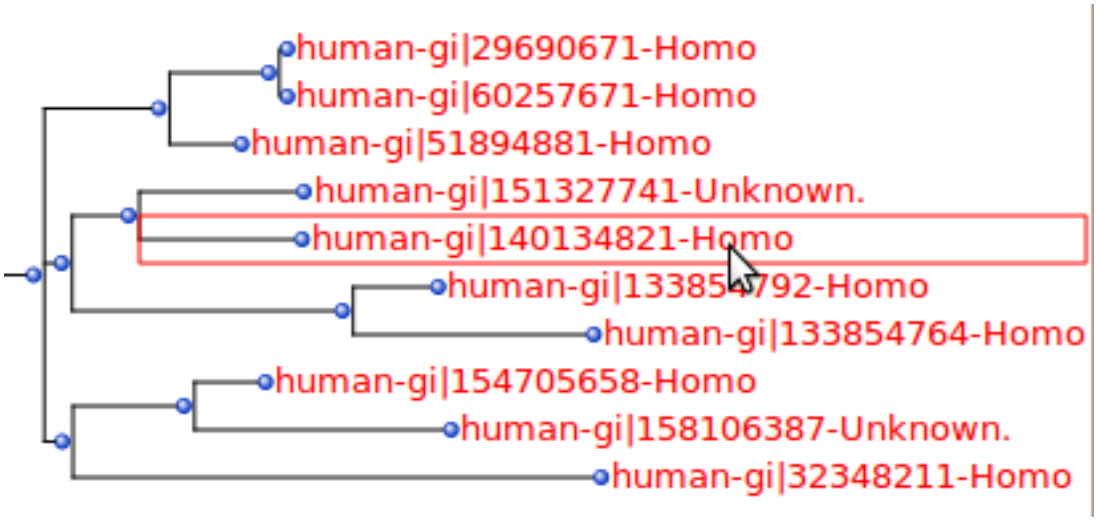


Figure 11: The sequences obtained with AutoPhylo using the sequence with GI number 295442711 for the entrez query homo[ORGN]. E-value cut-off of $1*10^{-5}$ was used and only the first 10 sequences were retrieved from the nr database.

Using the NCBI BLASTP on the *nr* database the first ten hits were 133854792, 32348211, 51894881, 133854764, 154705658, 151327741, 29690671, 140134821, 158106387 and 60257671. As seen in figure 11 the same sequences were found with AutoPhylo.

Next to test was for a seed sequence for which there was no BLAST hit in the NCBI *nr* database for the members, but there were hits in the local databases. Here the local databases were made out of sequences from JGI and the [ORGN] designation was used to find the correct database (if one existed). One protein that did not exist in NCBI (with a e-value below 1e-5 in *nr* database) is mitochondrial aspartyl-tRNA synthetase for the group fungi with the member Phycomyces[ORGN], using a seed sequences (GI number: 6325153) from baker's yeast (*Saccharomyces cerevisiae*).

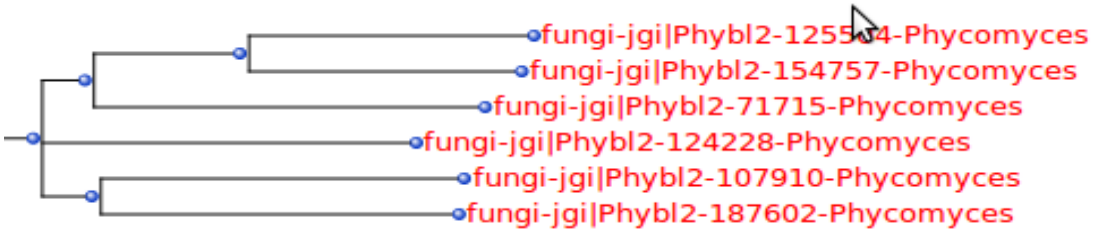


Figure 12: The sequences obtained with AutoPhylo using the sequence with GI number 6325153 with the entrez query Phycomyces[ORGN] and a e-value threshold of 1e-5. These sequences are obtained from a local database with Phycomyces sequences from JGI.

As seen in figure 12 Autophylo found 6 sequences that had an e-value under 1e-5 in a local JGI database for Phycomyces.

The last data fetching test was to try the EST data fetching. Here there had to be more than one member in the group (or more than one group) since ESTs are concatenated to one sequence and one sequence can not be used to make a tree (trees are automatically constructed).

The same query sequences as above was used for the group EST (members: Eimeria[ORGN], Caligus[ORGN], Euglena[ORGN], Physarum[ORGN], Lepeophtheirus[ORGN] and Acanthamoeba[ORGN]). For every member there was no hit with an e-value below 1e-5 in NCBI *nr* database and no hits in the local database. This forced AutoPhylo to try searching for EST data.

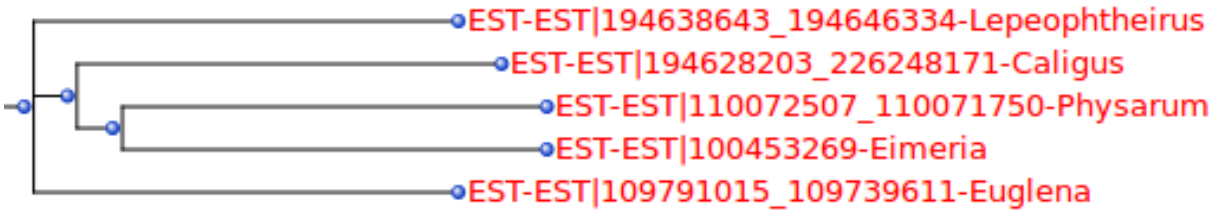


Figure 13: Sequences obtained with AutoPhylo for the group members Eimeria[ORGN], Caligus[ORGN], Euglena[ORGN], Physarum[ORGN], Lepeophtheirus[ORGN] and Acanthamoeba[ORGN] using the sequence with GI number 6325153 as seed sequence. The terminal nodes represented sequences constructed by concatenating ESTs using a method described in figure 8. The e-value cut-off for the EST´s was 1e-5.

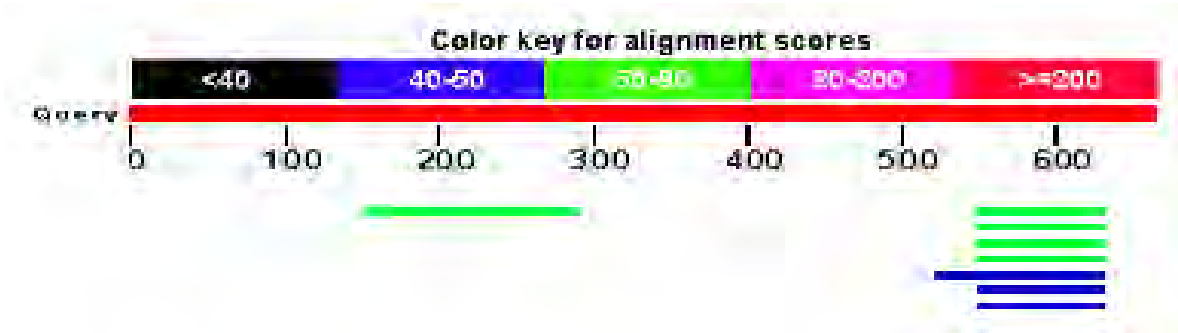


Figure 14: NCBI BLAST result graphics for sequence with GI number 6325153 using entrez query Caligus[ORGN] on database est. The e-value cut-off was 1e-5. As is seen some stretches in the sequence have multiple hits.

As seen in figure 13 AutoPhylo found ESTs for all members. All but one of the sequences in the figure were concatenated from two ESTs. One such is Caligus where there was more than one EST to choose from (figure 14). In the Caligus case AutoPhylo choose the best hit when there was two or more ESTs that overlapped.

2.2.2. Tree construction and manipulation

The program also should construct the same tree as the one that would be reconstructed by using each tool separately. In this example, the sequences obtained for the group EST above were aligned and a tree was constructed that should give the same result as above. This was done by using MUSCLE (no extra parameters) and QuickTree with the boot parameter set to 100.

Figure 15: EST data aligned with MUSCLE (no parameters) and tree constructed with QuickTree (boot parameter set to 100). To visualize, FigTree was used. The numbers in the names are GI numbers of the ESTs

Comparing figure 13 with figure 15, it is clear that they have the same topology, showing that the using of alignment program and tree construction program worked.

The most crucial part of AutoPhylo is the tree refinement function, which allows the possibility of removing sequences in a graphical interface and then reconstructing the tree with the remaining sequences. To test this, cytochrome c protein sequences for some groups were used, and a tree was constructed (see figure 14a). Three sequences (GI number 156788429, 833945 and 4261873) were removed using the graphical interface and the tree was reconstructed.

Comparing the trees in figure 16, one group (Sphenodontia, brownish red) had switched place with another group (turtle, green). In addition the snake group (red and brown, containing snakes and their closest related lizards) switched places in the tree. This shows that the tree indeed was reconstructed from the remaining sequences. If the tree was not reconstructed the topology would have stayed the same but with the sequences that were removed missing.

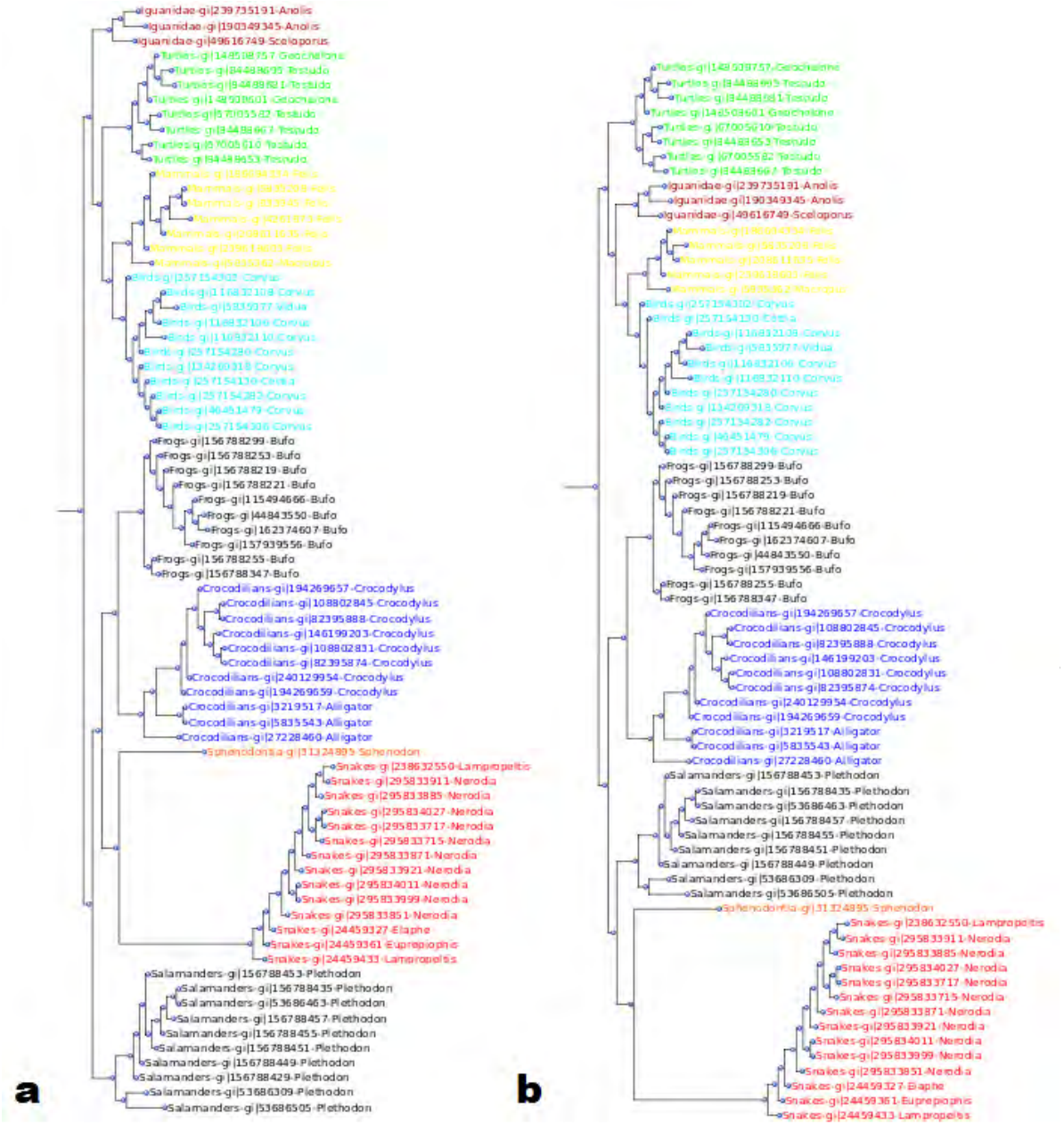


Figure 16: Trees of the cytochrome c protein for some groups. Tree b is the starting tree. Tree a is the tree after the removal of three sequences (GI numbers 156788429, 833945 and 4261873) and reconstruction from the remaining sequences.

3. Discussion

The program AutoPhylo was constructed to automate the steps of finding sequences, align them, optionally remove gaps, construct trees, deleting selected sequences and rebuild the tree. This is done for a single gene and is the initial step, and the most time-consuming part, when constructing large multigene data sets. The program allows for the tree to be refined in a graphical interface to remove unwanted sequences, easing the process of tree reconstruction. The current version of AutoPhylo, bachelor release (named after the degree that it was developed to obtain), described in this report, is still under development and should be considered more of a “proof-of-concept” than a finished product.

Any program requires a lot of testing by different users to find bugs and bad design choices. The different tasks that the users try to accomplish with the program will also show how to develop the program in the future. However, there are currently some issues that have presented themselves during the limited time that the tool has existed.

It should be possible to load previously acquired sequences to use for those studies where entrez queries are not sufficient for finding the sequences to use. This also is useful when the user already has the sequences and want to use the graphical interface to refine the resulting tree.

When adding a program, such as an aligner program, to AutoPhylo, the code needs to be changed in more than two places. In addition, this also requires programming skills. This could be changed with a system with files specifying the parameters the program uses and their value ranges.

To visualize and manipulate the tree, pre-existing code was used. This made it possible to develop the program in the short amount of time that was available for the project. However, the used code is not optimal for AutoPhylo with the extra options in the node menu and unimportant data given for a node, and the need to extend parts of the code to be able to use it in AutoPhylo. A solution to this would be to construct tree visualization and manipulation code specific to AutoPhylo that would fit perfectly, but that would have required more time.

The memory system for trees in this release only shows the latest tree. If the user want to compare older trees with the new tree, the user has to open project specific files in order to find older trees. It would be more useful if the system could be more flexible so that every refinement of the tree could be compared with the other trees.

It should be possible to show the alignment for any sub-tree. This would enable the user to see more easily which sequences should be deleted, based on the alignment. This would be useful if there are some sequences that are incomplete or misaligned. Incomplete sequences give shorter branches, so if the user always deletes the longer branches there is a possibility that the user will end up with incomplete sequences. For some species where there are more than one sequence, the user want to choose the most complete and with the shortest branch (the most evolutionary conserved sequence).

These issues show that there are many possibilities for the programs future beyond simple bug fixing. The bachelor release laid the foundation for this program and gave something to extend in any direction that may be needed. The program can be obtained from http://code.google.com/p/autophylo/.

4. Materials and Methods

4.1 Creating AutoPhylo

4.1.1. Programming

AutoPhylo was written in the programming language Python, using the version 2.6 but should work with older 2.x versions. To connect to NCBI and handle the data BioPython was used. For the graphical user interface pyQt was used. To draw the tree and manipulate it, ETE was used.

4.1.2. Databases used

The main database that BLAST uses is the non-redundant *nr/nt* (nucleotide) and *nr* (protein) databases at NCBI. From JGI protein sequences were downloaded for selected genus to be used. For the EST BLAST the *est* or *est_others* (*est* without human or mouse ESTs) databases at NCBI was used

4.1.3. Programs

In AutoPhylo four programs was used to align sequences, remove bad positions and construct a tree. To align sequences the programs MUSCEL and Clustal was available. To remove bad positions in the alignment GBlocks was used. For the tree construction QuickTree was used.

Acknowledgements

Coordinator Karin Carlson that informed me I could change course when I by mistake applied for the wrong course.

Course administrator Elsbeth Scholtes for helping me with the course change.

Supervisor Sandra Baldauf for introducing me to an interesting subject.

Co-supervisor Ding He for helping me with the project.

References

(1) Alberts B, Johnson A, Lewis J, Raff M, Roberts K, Walter P. 2002. Molecular Biology of the Cell. 4th ed. Garland Science, Taylor & Francis Group, LLC. USA.

(2) Baldauf SL. 2003. Phylogeny for the faint of heart: a tutorial. TRENDS in Genetics. 19: 345 – 351.

(3) Bottu G, Ranst MV, Lemey P. 2009. Sequence databases and database searching. In: Lemey P, Salemi MS, Vandamme AM (eds.). The Phylogenetic Handbook. A Practical Approach to Phylogenetic Analysis and Hypothesis Testing. 2nd ed. pp. 33 – 67. Cambridge University Press. Cambridge.

(4) Castresana J. 2000. Selection of Conserved Blocks from Multiple Alignments for Their Use in Phylogenetic Analysis. Molecular Biology and Evolution 17: 540-552.

(5) Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, Friedberg I, Hamelryck T, Kauff F, Wilczynski B, de Hoon MJL. 2009. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 25: 1422-1423.

(6) Edgar R. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 32: 1792 – 1797.

(7) Gribaldo S, Philippe H. 2002. Ancient Phylogenetic Relationships. Theor Popul Biol 61: 391 – 408.

(8) Howe K, Bateman A, Durbin R. 2002. QuickTree: building huge Neighbour-Joining trees of protein sequences. Bioinformatics 18: 1546 – 1547.

(9) Huerta-Cepas J, Dopazo J, Gabaldón T. 2010. ETE: a python Environment for Tree Exploration. Doi: 10.1186/1471-2105-11-24.

(10) Lewis B. 2007. What is a cell?. In: Lewin B, Cassimeris L, Lingappa VR, Plopper G (eds). CELLS. pp. 3 – 29. Jones and Bartlett publishers. USA.

(11) Page RMD, Holmes EC. 1998. Molecular Evolution. A Phylogenetic Approach. Blackwell Science, Cornwall.

(12) Peer YV, Salemi M. 2009. Phylogenetic inference based on distance methods. In: Lemey P, Salemi MS, Vandamme AM (eds.). The Phylogenetic Handbook. A Practical Approach to Phylogenetic Analysis and Hypothesis Testing. 2nd ed. pp. 142 – 180. Cambridge University Press. Cambridge.

(13) Prescott LM, Harley JP, Klein DA. 2002. Microbiology. 5th ed. McGraw-Hill. USA

(14) Romiti M. 2010. Entrez help. http://www.ncbi.nlm.nih.gov/bookshelf/br.fcgi?book=helpentrez&part=EntrezHelp. Retrieved 2010-05-26.

(15) Rosenberg MS. 2009. Sequence alignment. Concepts and history. In: Rosenberg MS. Sequence Alignment. Methods, Models, Concepts, and Strategies. pp. 1 – 22. University of California press. USA.

(16) Schmidt HA, Haeseler A. 2009. Phylogenetic inference using maximum likelihood methods. In: Lemey P, Salemi MS, Vandamme AM (eds.). The Phylogenetic Handbook. A Practical Approach to Phylogenetic Analysis and Hypothesis Testing. 2nd ed. pp. 181 – 209. Cambridge University Press. Cambridge.

(17) Schweitzer MH, Suo Z, Avci R, Asara JM, Allen MA, Arce FT, Horner JR. 2007. Analyses of Soft Tissue from *Tyrannosaurus rex* Suggest the Presence of Protein. Science 316: 277 – 280.

(18) Sogin ML. 1991. Early evolution and the origin of eukaryotes. Curr Opin Genetics Dev 1: 457 – 463.

(19) Thompson JD, Higgins DG, Gibson TJ. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22: 4673–4680.

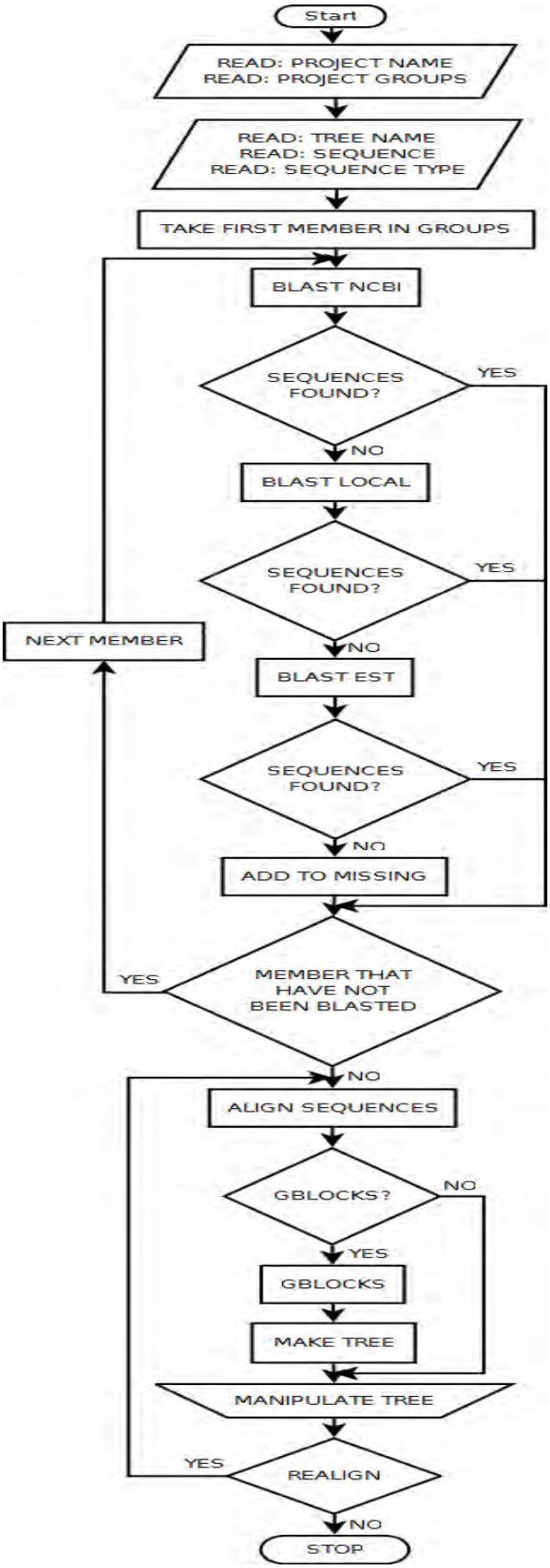
(20) Vetenskapsrådet. Vetenskapsrådets projektdatabas - Roten på det eukaryota trädet. http://vrproj.vr.se/detail.asp?arendeid=53792. 2010-05-26.

(21) Yang Z. 2006. Computational Molecular Evolution. Oxford University Press. King´s Lynn, UK.

(22) Zvelebil M, Baum J. 2008. Understanding bioinformatics. Garland Science, Taylor & Francis Group, LLC. USA.

Appendix A

Flowchart for AutoPhylo



Appendix B

Group file to use with AutoPhylo. Contains the groups used in a project to find the root of the Eukaryotic tree **(20)**. The format is name followed by a tab and some members (optional) separated by ;. There can also be a colour (hexadecimal RGB format, also optional) following the members (with a tab separating the members and the colour). The layout in this appendix is due to space issues, the group Jak shows the layout..

```
AAA

Anaeromyxobacter[ORGN];Aquifex[ORGN];Bacillus[ORGN];Burkholderia[ORGN];Chla
mydia[ORGN];Chlorobaculum[ORGN];Chloroflexus[ORGN];Cytophaga[ORGN];Deinococ
cus[ORGN];Escherichia[ORGN];Helicobacter[ORGN];Mycoplasma[ORGN];Nostoc[ORGN
];Planctomyces[ORGN];Pseudomonas[ORGN];Rhodopirellula[ORGN];Rhodopseudomona
s[ORGN];Rickettsia[ORGN];Dolibacter[ORGN];Streptomyces[ORGN];Thermotoga[ORG
N];Treponema[ORGN]      #000000

Alv

Babesia[ORGN];Theileria[ORGN];Plasmodium[ORGN];Eimeria[ORGN];Toxoplasma[ORG
N];Tetrahymena[ORGN];Paramecium[ORGN]

Ani

Apis[ORGN];Branchiostoma[ORGN];Brugia[ORGN];Caenorhabditis[ORGN];Caligus[OR
GN];Danio[ORGN];Drosophila[ORGN];Homo[ORGN];Ixodes[ORGN];Lepeophtheirus[ORG
N];Monosiga[ORGN];Schistosoma[ORGN];Strongylocentrotus[ORGN];Trichoplax[ORG
N]

Amo   Dictyostelium[ORGN];Physarum[ORGN];Acanthamoeba[ORGN]

Exc   Euglena[ORGN];Leishmania[ORGN];Trypanosoma[ORGN];Naegleria[ORGN]

Fun

Phycomyces[ORGN];Schizosaccharomyces[ORGN];Neurospora[ORGN];Aspergillus[ORG
N];Malassezia[ORGN];Yarrowia[ORGN];Saccharomyces',
'Pichia[ORGN];Lodderomyces[ORGN];Ustilago[ORGN];Laccaria[ORGN];Batrachochyt
rium[ORGN]

Hap   Emiliania[ORGN]

Jak   Reclinomonas[ORGN];Jakoba[ORGN]     #FF0000

Pla

Chlamydomonas[ORGN];Volvox[ORGN];Micromonas[ORGN];Ostreococcus[ORGN];Arabid
opsis[ORGN];Vitis[ORGN];Oryza[ORGN];Chara[ORGN];Physcomitrella[ORGN];Picea[
ORGN];Selaginella[ORGN];Chlorella[ORGN]

Str

Aureococcus[ORGN];Phytophthora[ORGN];Thalassiosira[ORGN];Phaeodactylum[ORGN
]
```