Final Project: Phylogenetic Tree Construction

Hazelyn Cates

EN.605.651.81.FA23

Dr. Qie

December 11, 2023

Introduction and Background

Phylogeny, which describes how organisms are evolutionarily related to each other, was formally introduced in the 19th century by Ernest Haeckel who in 1866 first used the term and developed an early phylogenetic tree that grouped blue-green algae and bacteria together due to their supposed lack of nucleus that was typically observed in cells [3a]. However, almost 30 years prior to Haeckel in 1837, Charles Darwin constructed his "tree of life", which consisted of a single main branch ("trunk") and from it many other connected branches going out [4a].

From the time of Darwin and Haeckel, the "tree of life" went through many contestations, but the year 1977 was a pivotal year in molecular biology when Carl Woese and George Fox published a paper in the journal *Proceedings of the National Academy of Sciences of the United States of America* that described the division of all organisms on Earth into three groups: eukaryotes, archaebacteria, and eubacteria [3a]. Carl Woese's role in this discovery was his sequencing and subsequent comparisons of ribosomal RNA (rRNA), which plays a vital role in the translation of mRNA to an amino acid sequence, of various organisms [3a]. Ribosomes consist of a small and large subunit, and upon studying the small ribosomal subunit, Woese and his research team (which included Fox) deduced that nucleotide fragments in the small subunit that are at least six bases long were unlikely to be changed over time due to random chance, and thus are likely homologous between species and therefore useful for looking at evolutionary relatedness [3a]. The phylogenetic tree of life that is most familiar and still serves as the basis of phylogeny today was proposed in 1990 by Carl Woese, Otto Kandler, and Mark Wheelis, with the three domains Eucarya, Archaea, and Bacteria [3a].

The Sanger sequencing method that Carl Woese and his research team used in the 1960s and 1970s has subsequently been replaced by quicker and more elegant next generation sequencing methods that begin with reverse transcribing the RNA into complementary DNA (cDNA), amplifying that cDNA via polymerase chain reaction (PCR) and then performing sequencing on the cDNA [2a, 5a]. Once the sequencing is complete, there are a myriad of phylogenetic analysis tools that are publicly available.

Given this, phylogenetic analysis and tree construction have multiple uses and purposes, with the overarching purpose of finding, determining, and classifying evolutionary relatedness of organisms based on genetic information rather than physical appearance. Other uses include but are not limited to: paralog versus ortholog identification and in relation to that, exploring biodiversity within a population or region, and where and when speciation (i.e. divergence) took place and to what extent, and the development of PAM (point accepted mutation) scoring matrices [1a].

Phylogenetic trees typically begin at a root, which is a presumed common ancestor to all of the other organisms represented. Divergence of groups is further represented by branches off of the main "trunk", and lineages can be traced backwards to a common ancestor (not necessarily the last common ancestor) from a given branch, with these groups referred to as "clades" [1a].

Problem Definition

This project focuses on the construction of phylogenetic trees given at least two DNA sequences. The implementation of solving this problem was done using Python and with the aid of the Biopython package.

Methods and Algorithms

In today's practice, there are various algorithms and tools available for phylogenetic tree construction. A popular and widely available tool is the publicly available software MEGA (Molecular Evolutionary Genetics Analysis), which can be downloaded onto a user's computer and available for five different operating systems [9b]. From user input data, MEGA can perform sequence alignment and evolutionary analysis with further investigation and visualization options of the processed data, with the option to save the results [3b]. For the sequence alignment prior to phylogenetic tree construction, MEGA allows the user to choose between the MUSCLE and ClustalW algorithms, which are both multi-sequence alignment programs that are independently publicly available online [5b]. For phylogenetic tree construction, MEGA has multiple options available: (1) neighbor joining (NJ), (2) minimum evolution (ME), (3) maximum parsimony (MP), and (4) maximum likelihood (ML) [3b].

Another publicly available and downloadable tool is Jalview, which can be used across four different operating systems and is functionally similar to MEGA, in that it can perform sequence alignment followed by further visualization and analysis, including phylogenetic tree construction [15b]. In Jalview, phylogenetic trees can be constructed from aligned or unaligned sequences, and alignment can be done using ClustalOmega, T-Coffee, MUSCLE and other tools [1b]. If the sequences were not aligned prior to tree construction, a tree can be generated directly from the set of sequences [1b]. Contrarily, if using aligned sequences, the phylogenetic tree is constructed based on sequence similarity, via PID (percent identity), substitution matrices, or sequence feature similarity methods [1b]. To generate the tree, Jalview has two clustering methods available: (1) UPGMA (Unweighted Pair-Group Method using Arithmetic averages) clustering which ultimately forms clusters using the least dissimilar average non-member sequence, and (2) NJ (neighbor-joining) clustering, which constructs trees with the shortest branches using a greedy algorithm approach [1b].

Typically, as stated above, phylogenetic tree construction begins with sequence alignment, either pairwise or multiple sequence alignment. However, regardless of if alignment is performed beforehand, the popular tree generation methods (i.e. algorithms), which were stated above, have their own approaches, advantages and disadvantages.

To start, the maximum parsimony (MP) method uses a character-based approach, meaning that when the sequences are simultaneously compared, the comparison is done one character (ex. "A", "T", "C" and "G" for DNA sequences) at a time [10b]. The goal of the MP method is to produce the most accurate phylogenic tree possible by including the fewest number of resultant branches over a set of evolutionary steps from a common ancestor between the

sequences being analyzed [12b]. More specifically, from a computational approach, the goal is to minimize the overall sum of the weights of the edges from parent to child/children all the way out to the furthest branch, which is referred to as the parsimony score [7b]. Interestingly, this search for the most optimal parsimony score is characterized as NP-hard [7b]. A large advantage for the MP approach is that it works very well when the sequences are similar but tends to decline in accuracy if the sequences diverge [10b]. An additional disadvantage of the MP method is its tendency to be subject to long branch attraction, which is when diverged species falsely appear to be closely related as a result of convergent evolution [8b].

Another aforementioned phylogenetic tree construction method that is also character-based is the maximum likelihood (ML) method [10b]. ML is a statistical method that in contrast to MP's approach, uses the probability that two sequences are evolutionarily related to construct the phylogenetic tree [2b]. Also in contrast to the MP approach, maximum likelihood continues to work well even if the sequences diverge, which increases the accuracy of the final tree [10b]. However, a disadvantage to the ML approach is that its run time is directly related to the size of the data and can become inefficient rather quickly [10b]. An example of an algorithm that implements maximum likelihood is PhyML, which was first put forth in 2003 [4b]. Since then, other versions of PhyML have been released, some of which use genetic-based algorithms (that is, algorithms whose goal is to mimic the process of natural selection), like TREEFINDER (2004), and GARLI (2006) [4b, 16b]. As of 2010, the PhyML algorithm has been updated to PhyML-SPR, which usess subtree pruning and regrafting (SPR) instead of nearest neighbor interchange (NNI), the latter which is susceptible to long-branch attraction [4b].

An additional phylogenetic tree construction method mentioned above is neighbor joining (NJ). As previously mentioned, neighbor-joining is a greedy approach, with the goal to minimize the branch length sums of the final tree, since the building of the phylogenetic tree using NJ is based on distance between the species being investigated [6b]. This algorithm is based on the principle of minimum evolution (ME), where a distance matrix is created from the input sequences, and for NJ, it is assumed that the final tree with the minimized branch lengths is the most accurate [13b]. Since the development of NJ in 1987, many different versions of NJ have been developed, all of which have improved the speed at which the tree is constructed [6b]. In 2021, researchers in China developed ENJ (extended neighbor joining) as an improvement to the NJ algorithm [6b]. Broadly speaking, ENJ starts with a distance matrix of the species in question and can join three true neighbor nodes to build a triple phylogenetic tree, which is more evolutionarily accurate [6b]. A disadvantage of NJ however is that the trees produced are not rooted, and the nodes and leaves within the tree are not ordered in any specific way, so it does not suggest any kind of evolutionary or ancestral relationships between the leaves [11b].

Last is UPGMA (Unweighted Pair-Group Method using Arithmetic mean). This method is a distance analysis method that uses a distance matrix and begins clustering by first finding two organisms with the smallest difference (aka distance) between any two sequences (DNA/RNA or protein) [14b]. This cluster now represents the two closest related organisms and forms a single cluster. To find the next distance value, the mean of the sums of the newly formed cluster intersecting with the remaining clusters (which at this point are still individual organisms)

is calculated and the process is repeated, decreasing the matrix by a column and row in each iteration as the clusters are formed. [14b]. Eventually, clusters will keep being added to the initial cluster to form a single resulting cluster, which is the phylogenetic tree rooted at a single node [14b]. However, UPGMA has the disadvantage of producing ultrametric trees, which occurs due to the fact that UPGMA method assumes a constant evolutionary rate, which is not biologically accurate [11b]. Because of this, UPGMA trees also suffer from long branch attraction, where leaves in a tree are falsely represented as being related [11b].

Implementation

For this project, the implementation was carried out using the programming language Python via the IDE Pycharm 2021.2. The package Biopython was used, which contains an extensive collection of tools for many bioinformatics tasks. More specifically, this program utilized from Biopython the *SeqIO, AlignIO*, and *Phylo* packages, and from the *Phylo* package the *TreeConstruction* module and from this module the *DistanceCalculator*, and *DistanceTreeConstructor* functions.

To begin, FASTA sequences were collected from the NCBI Nucleotide database [1d]. The gene of interest for this project is Tumor Necrosis Factor (TNF), which is considered orthologous between species. The TNF gene is a protein coding gene that codes for a cytokine in the TNF superfamily that gets secreted primarily by the immune cells macrophages [14c]. This cytokine promotes inflammation and receptor binding is what activates its functions, which can include cellular differentiation, cellular proliferation, angiogenesis, and apoptosis [14c]. Production of TNF is also linked to fever development and can contribute to insulin resistance and some autoimmune disorders [14c].

For this project, the TNF gene sequences of 10 species were chosen at random and the list of the 10 species and their accession numbers follows [2d-11d]:

NM_000594.4 Homo sapiens (humans)

NM_214022.1 Sus scrofa (wild boar)

NM_001082263.1 Oryctolagus cuniculus (European rabbit)

NM_012675.3 Rattus norvegicus (brown rat)

M13049.1 Mouse

AJ417565.2 Ictalurus punctatus (Channel catfish)

NM_173966.3 Bos taurus (cattle)

NM_001024860.1 Ovis aries (sheep)

NM_001003244.4 Canis lupus (wolf)

NM_001081819.2 Equus caballus (horse)

Before a phylogenetic tree can be built, the sequences must be aligned. This was accomplished using the online multiple sequence alignment tool MUSCLE [6c]. The results of the alignment were saved in a .clw file format (since this program asks for user input of the file name, any sequence alignment result can be used as long as they are in .clw format). In Python, this alignment file was read into a variable called "aligned_seqs" using the *AlignIO* package and the read function, specifying the file type as an argument in the function [5c, 13c].

Following multiple sequence alignment, the next step in building a phylogenetic tree is to calculate the distances between the sequences in order to build a distance matrix, which becomes the input for the tree-building algorithms. To do this, the *DistanceCalculator* module is called, and the argument is "identity", which specifies that the sequences in the multiple alignment are DNA/RNA, not protein, creating an object of that type called "calc" [2c, 5c]. This object "calc" is then used to call the ".get_distance" function in the *DistanceCalculator* module, which actually constructs the distance matrix using the aligned sequences that were read in as the argument [2c, 5c]. The resulting matrix is stored in a variable called "dist_matrix". With the matrix, a function written from scratch specifically for this program, called "four_point_condition" is called, which only takes the calculated distance matrix as its argument. This function determines if the distance matrix is additive, meaning that the distances in the matrix between the species can be represented as a weighted tree, where the edges between vertices add up to the distances represented in the matrix [11c].

The "four_point_condition" function begins by first importing the *math* package and the *permutations* module from the *itertools* package. A flag variable is set to keep track of if the matrix is additive, initially set to 0 to represent that it is not additive. Next, determining the four points is considered a combination problem, where n = number of sequences and r = 4. The *comb* function in the math package determines the number of permutations of 4 leaves (vertices) given how many sequences the user inputs (call this x), which is stored in a variable called "num_combos" [10c]. In order to keep track of the matrix labels, a variable called "names" stores the accession numbers of the x sequences. Then, the *permutations* function finds the permutations of the x sequences and these permutations get stored in an array called "combos", which is essentially a list of lists [7c]. Once the permutations are determined, a for loop that iterates the length of "num_combos" executes. The goal is to satisfy the following formula [4c]:

$$\max\{d(a, c) + d(b, d), d(a, d) + d(b, c)\} \geq d(a, b) + d(d, c)$$

Where a = the first label, b = the second label, c = the third label and d = the fourth label of the given permutation sequence. From here, the above formula is calculated in two parts, where the max value to the left of the inequality sign is calculated separately and stored in a variable called "val_1", and the sum to the right of the inequality sign is stored in the variable "val_2". If "val_1" is greater than or equal to "val_2", then the flag is set equal to 1 and the for loop continues to the next permutation of 4. Once this formula is no longer satisfied, the flag is set to 0 and the loop breaks. An if-else statement then checks the condition of the flag, where if the flag

is set to 1, the user is informed their matrix is additive, but if the flag is set to 0, the user is informed their matrix is not additive.

However, it is important to note that many matrices generated from sequence alignments are not additive. And because of this, heuristic tree-building methods are often utilized instead, like UPGMA and neighbor joining (both of which are implemented in this program), which approximates distance metrics which helps the algorithm recognize any types of patterns or similarities within the data [11c, 12c]. Therefore, in this case, a matrix being additive or not does not impact tree construction since the methods used in this program are heuristic, but rather serve to provide information to the user. However, an important note is that the neighbor joining method tends to work fairly well even when the matrix is not additive, and recall from above, NJ trees do not suffer from long branch attraction [11b].

Following the "four_point_condition" function call, the trees are constructed. This begins by creating an object of the *DistanceTreeConstructor* module called "construct" [1c, 3c, 5c]. The program asks the user which tree(s) they want to build using single character input. The program also outputs the distances of the leaves and the inner nodes for each tree using the ."get_terminals" and ".get_nonterminals" functions on the tree object [9c].

To construct the NJ tree, the *nj* function on the "construct" object is called, whose argument is the distance matrix "dist_matrix" calculated earlier. The resulting tree is printed to a separate window using the *draw* function in the *Phylo* module [5c, 8c]. The same procedure is then carried out for creating a UPGMA tree where the *upgma* function is used on the "construct" object, taking the distance matrix "dist_matrix" as the argument. Again, like the NJ tree, the *draw* function in the *Phylo* module prints the resulting tree to a separate window [5c, 8c].

To continue, in the interest of comparison, other methods were utilized to construct phylogenetic trees using the same 10 sequences as above. To compare to the Python implementation, tree construction was done on the software MEGA (described above) and directly from MUSCLE following multiple sequence alignment.

As stated above, the 10 FASTA sequences are stored in a .txt file that can be easily uploaded to both MEGA and MUSCLE. Starting with MUSCLE, the .txt file was uploaded, multiple sequence alignment was performed with the default parameters, and from the output, the "Phylogenetic Tree" tab was selected to display the tree. MUSCLE can display the cladogram or the real branch lengths. An important note is that MUSCLE uses a neighbor joining approach for tree construction and the resulting tree can be seen in figure 3 under Results.

Next, in the software MEGA, under the "phylogeny" option, there are five types of trees that can be constructed: NJ, UPGMA, minimum evolution (ME), maximum parsimony (MP) and maximum likelihood (ML). Sequence alignment must first be conducted in MEGA before building a phylogenetic tree, and MEGA requires that the sequence input be in .fasta file format. This FASTA file can be opened in MEGA with the options "Align" or "Analyze". Selecting "Align" pulls up the color-coded sequences, and from here, all sequences can be selected and aligned by MUSCLE (which can be selected from the toolbar). From here, the user can choose to alter the alignment parameters but, in this case, the default parameters were kept. With the

alignment complete, the results can be exported to MEGA format and once saved, can be used for phylogenetic tree construction. All default parameters were used for all five trees, and the resulting five trees can be seen in figures 4 through 8 in the Results section.

Results

The following figures show the outputs of the various methods described in the Methods and Algorithms section:

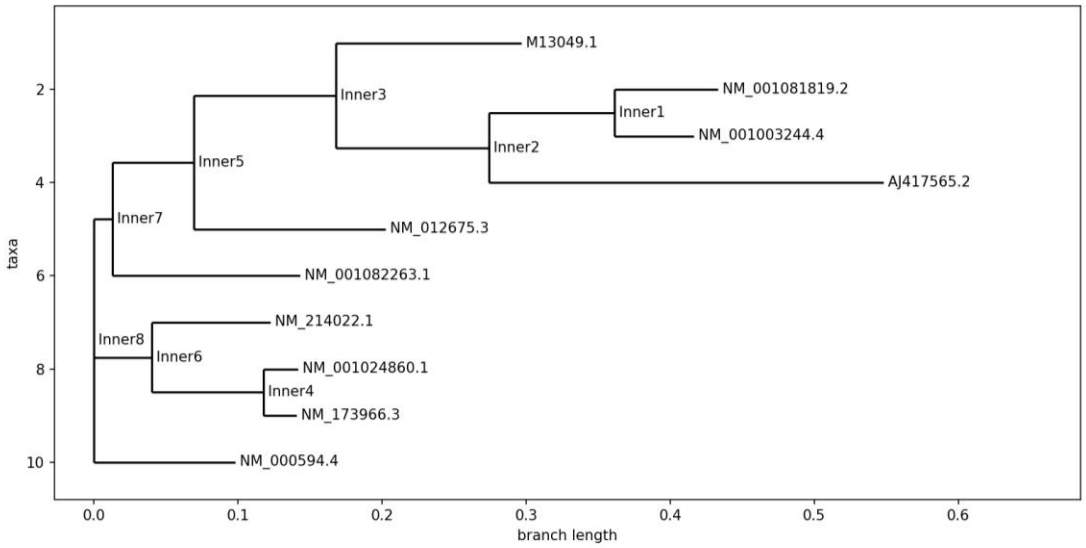

Figure 1: NJ Tree from Python implementation

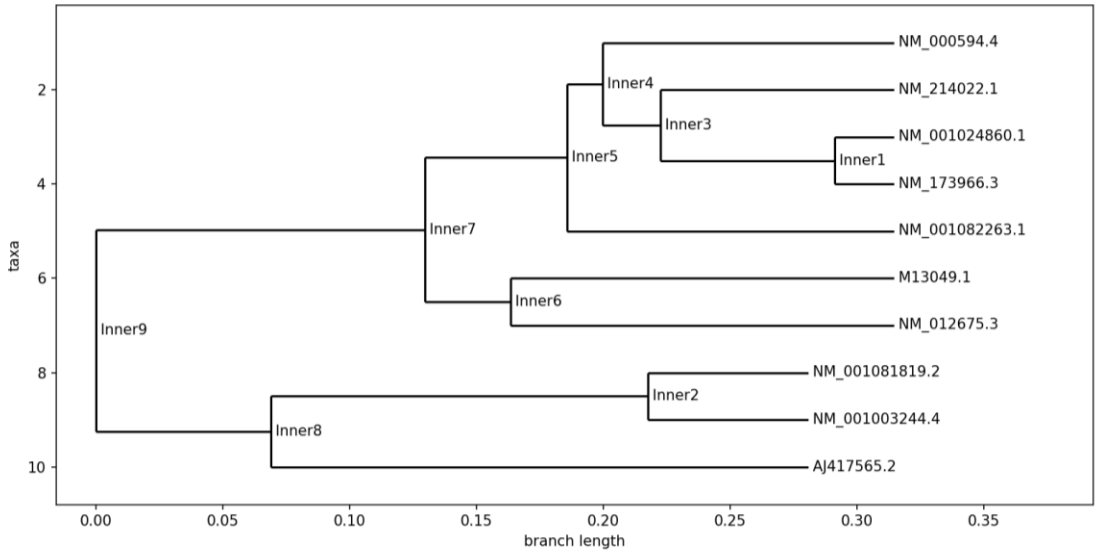

Figure 2: UPGMA Tree from Python implementation

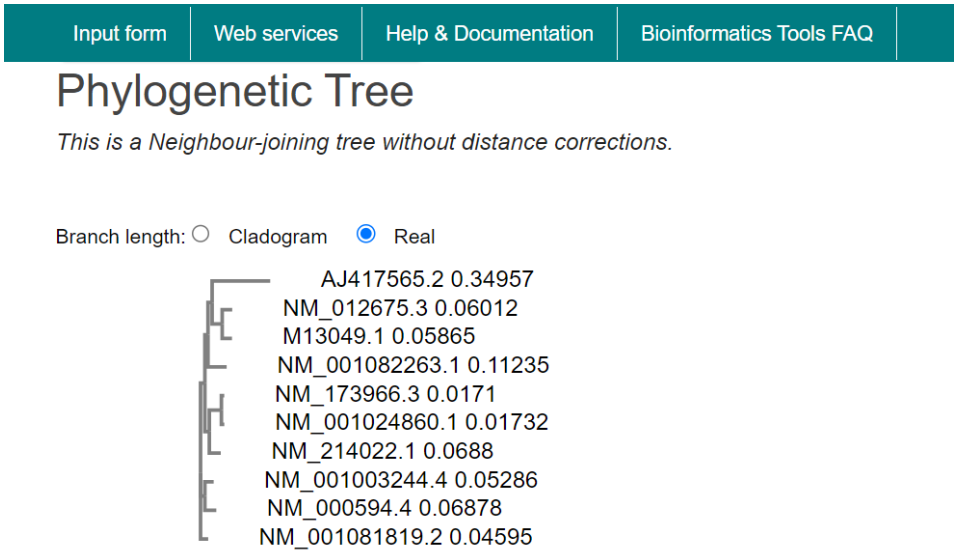

Figure 3: NJ phylogenetic tree constructed from EMBL-EBI MUSCLE software

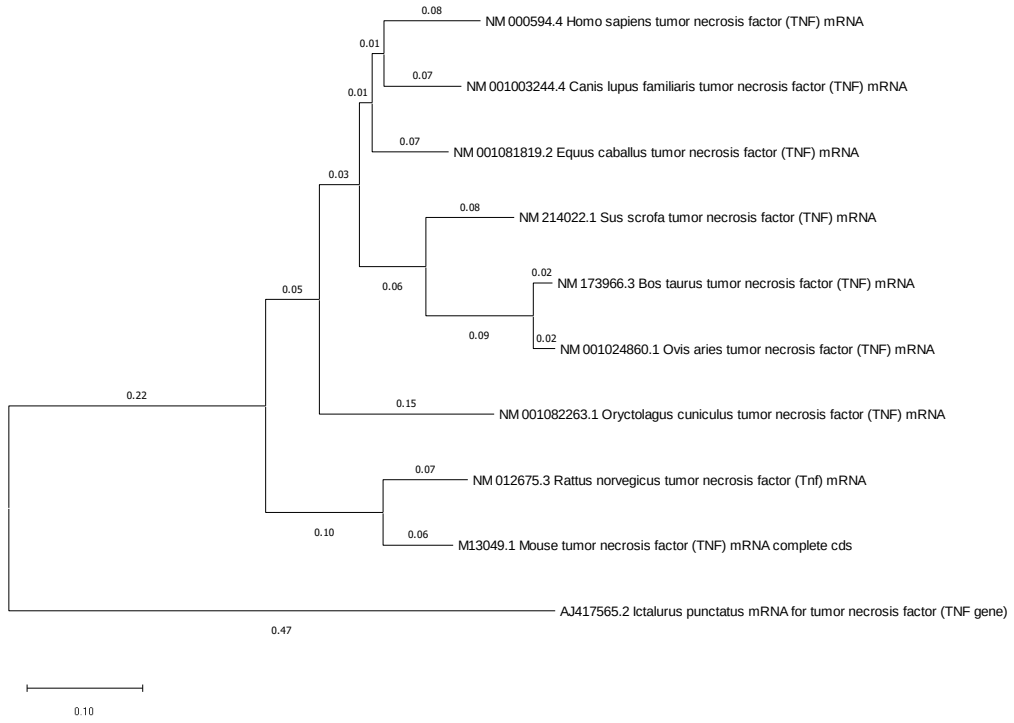

Figure 4: ML tree from MEGA

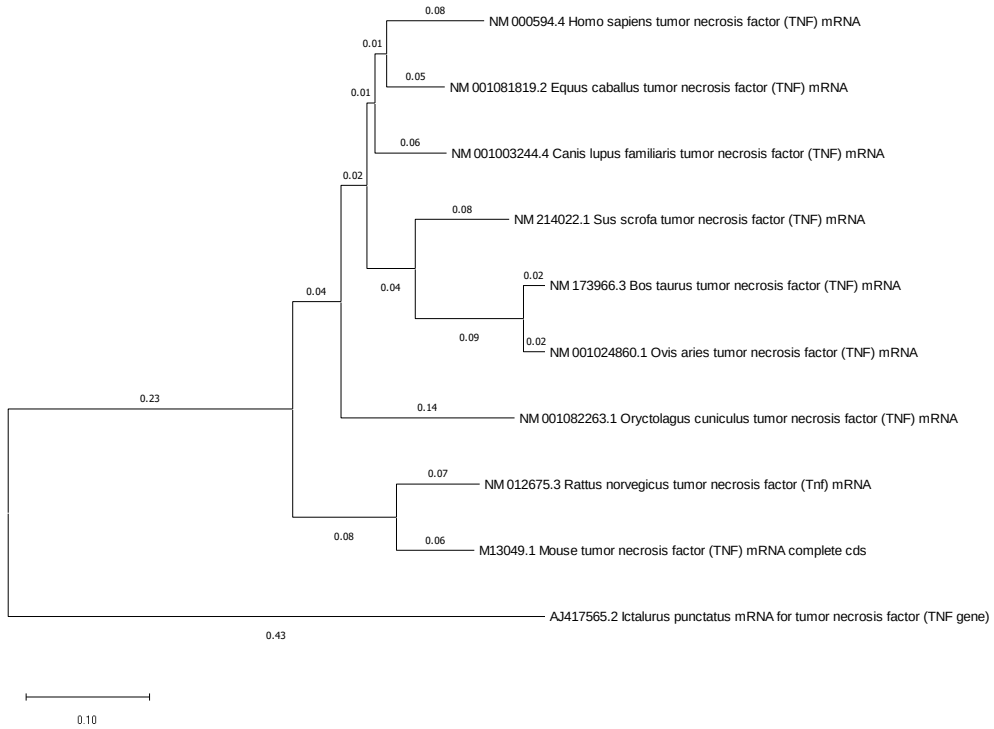

Figure 5: NJ Tree from MEGA

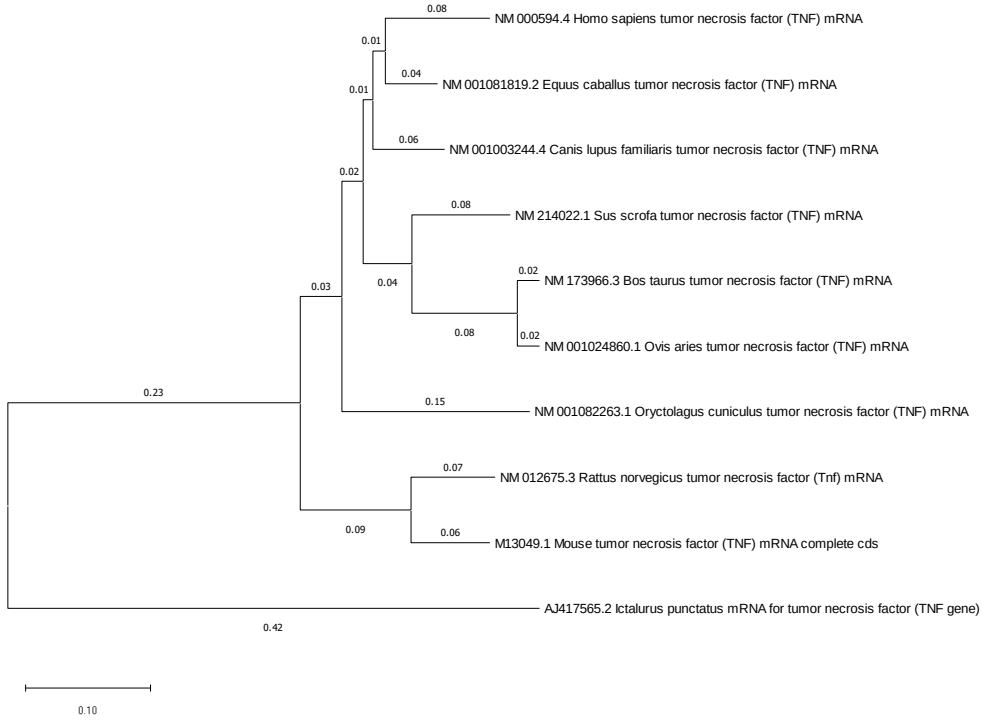

Figure 6: ME Tree from MEGA

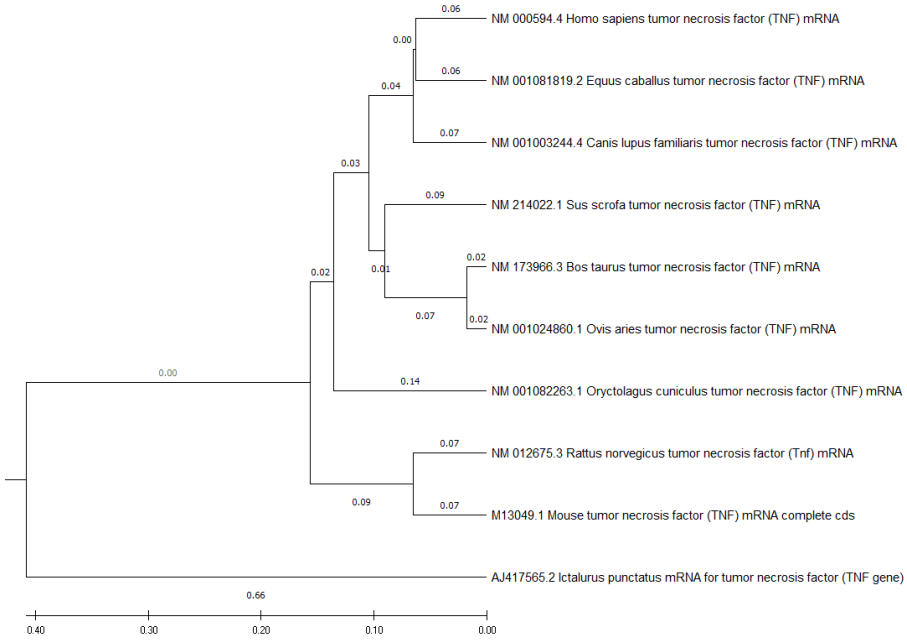

Figure 7: UPGMA Tree from MEGA

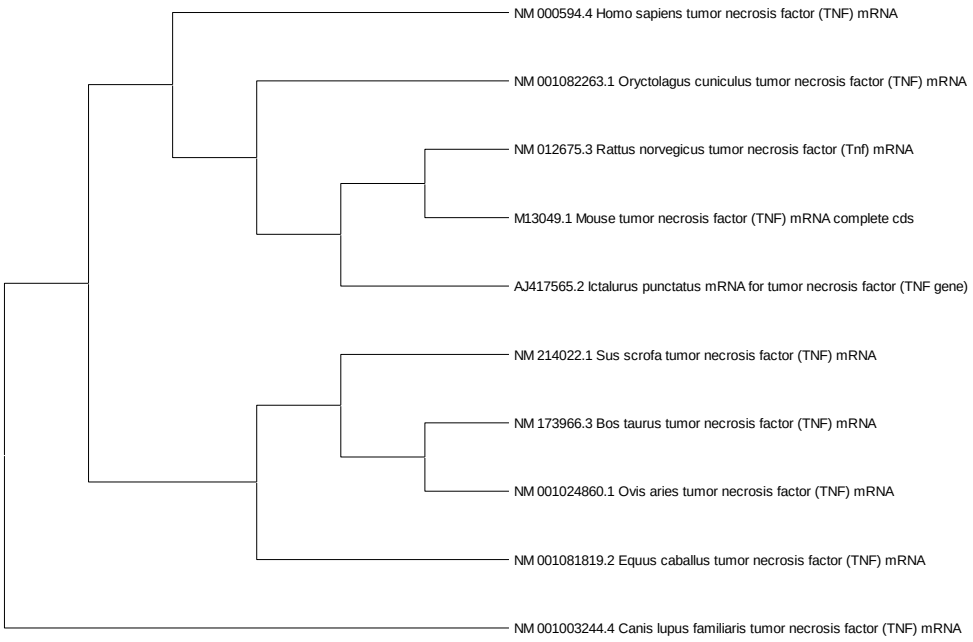

Figure 8: MP Tree from MEGA

Please recall the following accession number assignments:

NM_000594.4	Humans
NM_214022.1	Wild boar
NM_001082263.1	European rabbit
NM_012675.3	Brown rat
M13049.1	Mouse
AJ417565.2	Channel catfish
NM_173966.3	Cattle
NM_001024860.1	Sheep
NM_001003244.4	Wolf
NM_001081819.2	Horse

To begin, only three of the eight trees are exactly the same: NJ tree from MEGA, ME tree from MEGA, and UPGMA tree from MEGA, (figures 5, 6 and 7, respectively). However, these trees do share some notable characteristics with the remaining six trees. The one trait that is shared between all eight trees is that sheep and cattle TNF sequences are grouped and branch off from the same node. Beyond this one trait, there are no other characteristics that are shared across all the trees recorded. However, in figures 2, 3, 4, 5, 6, 7, and 8, the sheep and cattle group is most closely related to wild boar, who shares a common ancestor with the sheep and cattle group. Additionally, while not exact between all trees, another notable characteristic is the fairly

consistent relation between wolf and horse TNF. In all eight trees, horse and wolf TNF are either grouped and share the same node (figures 1, 2), share a common ancestor (figures 3, 4, 5, 6, 7) or are separated by two internal nodes (figure 8). Additionally, in seven of the eight trees (that is, excluding figure 1), brown rat and mouse TNF are grouped together and branch off of the same node, however the common ancestors of this group do differ in the trees in figures 2-8. In figures 2, 3, 4, 5, 6 and 7, European rabbit is separated by two internal nodes descending from the common ancestor of the brown rat and mouse group, while in figure 8, the brown rat and mouse group is two internal nodes descending from the European rabbit leaf. When looking at all eight trees, there is no consistent trend observed for the placement of humans, however in figures 5, 6, and 7, human TNF is grouped with horse with a common ancestor of wolf TNF.

When comparing the two trees in figures 1 and 2 that were generated via Python, there are some similarities that can be noted. First is the clustering of wolf and horse in both trees, with that cluster sharing a common ancestor with channel catfish. Also, both trees cluster sheep and cattle, and this clustere shares a common ancestor with the wild boar. However, beyond those two points, the trees differ in placement of the remaining organisms. For example in the NJ tree in figure 1, humans appear to be the most distantly related species, while in the UPGMA tree in figure 2, humans seem to be more closely related to the European rabbit, wild boar, sheep and cattle.

Next, to investigate the differences in software in making trees using the same algorithm, the NJ trees, which are pictured in figures 1, 3, and 5, will be compared. To begin, as stated previously, while differing in orientation, all three NJ tree cluster horse, and wolf TNF closely. In the case of figure 1, which shows the tree resulting from the python tree building implementation, horse and wolf share a common node and have a common ancestor to the channel catfish. In figure 3, whose NJ tree was built using MUSCLE, wolf is shown to be more closely related to human, with a common ancestor to horse. In figure 5, whose NJ tree was built using MEGA, humans and horse are shown to be closely related, and share a common ancestor to wolf. Additionally, between the NJ trees and also mentioned previously, all three NJ trees group sheep and cattle as closely related, sharing a common ancestor with wild boar. However, notice in figure 1 that mouse and brown rat are not grouped together, but in the other NJ trees in figures 3 and 5, they are. Interestingly, figure 1 shows mouse and brown rat to share a common ancestor, but also shows channel catfish, horse, and wolf to descend from a common ancestor shared by mouse but not brown rat. But given that point, in all three trees, European rabbit shares a common ancestor with mouse and brown rat in figures 1 and 3, but figure 5 shows European rabbit to be a descendant of the internal node that branches off to the mouse and brown rat group. Lastly, between the three trees, there is no consistency in where human is placed. Figure 1 is most striking in this point, since it shows human to be the least related to the remaining nine species. However, recall that the ordering of species in an NJ is not necessarily indicative of ancestral relatedness due to the fact that NJ trees are not rooted. So when investigating the topology of the tree, that must be taken into account, which as stated before in the Methods section makes NJ trees less accurate in that respect.

Related to the NJ tree is the ME tree (recall that the NJ algorithm is based on ME principle). Comparing the NJ trees (figures 1, 3, 5) to the ME tree (figure 6) reveals similar trends as observed and noted above. For instance, in all four trees, sheep and cattle are grouped and share a common ancestor with wild boar, and in figures 3, 5, and 6, mouse and brown rat are grouped together. In figures 5 and 6, European rabbit is placed as a descendant from the common ancestor that includes a branch to the mouse and brown rat group. Also notable in figures 5 and 6 is the placement of Channel catfish, which is shown to be the most distantly related species from the remaining nine species. In contrast, figures 1 depicts Channel catfish sharing the same common ancestor as the horse and wolf group, and figure 3 showing Channel catfish to be branching off a common ancestor that leads to the brown rat and mouse group.

Next, the UPGMA trees produced via python implementation and MEGA (figures 2 and 7, respectively) will be compared. As mentioned above, sheep and cattle are grouped together with wild boar branching off the common ancestor of the group. Also mentioned above is the grouping of mouse and brown rat together, with European rabbit as a descendant from a common ancestor of mouse and brown rat. Additionally, also as mentioned above, horse and wolf are depicted as closely related, sharing the same node in figure 2 and sharing a common ancestor in figure 7. Beyond these similarities, figures 2 and 7 have humans placed in completely different parts of the tree; figure 2 shows humans sharing a common ancestor with cattle, sheep and wild boar, while figure 7 shows humans grouped with horse and sharing a common ancestor with wolf.

`The two trees remaining, the ML tree (figure 4) and the MP tree (figure 8) share multiple characteristics with the other six trees as stated (i.e. the grouping of sheep and cattle). However, it is notable to observe that figure 8 is the only tree that shows horse branching off of a common ancestor with wild boar, sheep, and cattle. Figures 4 and 8 also differ in their placement of Channel catfish, with figure 4 depicting Channel catfish as the most distantly related from the nine other species, and figure 8 depicting Channel catfish as sharing a common ancestor with the brown rat and mouse group.

An important note when comparing these trees, namely the Python implementation trees in figures 1 and 2 is that NJ trees are not rooted, so even though some of the organisms are clustered the same as the UPGMA Python tree, the location of the other organisms in relation to each other does not necessarily indicate how similar or dissimilar they are to each other ancestrally. Therefore, the UPGMA tree, which is rooted and indicates a common ancestor of the ten organisms is going to paint a more accurate picture of ancestral relationships of the ten organisms in relation to each other.

The next aspect of the trees to compare are the edge distances, mainly focusing on those produced via the python implementation since that was the focus of the project. In the python implementation, the edges are not put directly on the tree but are available to access using the ".get_terminals()" and the ".get_nonterminals()" functions. The terminal (leaf) edge distances for the trees shown in figures 1 and 2 follow:

Name	Terminal branch length
M13049.1	0.13
NM_001081819.2	0.07
NM_001003244.4	0.05
AJ417565.2	0.27
NM_012675.3	0.13
NM_001082263.1	0.13
NM_214022.1	0.08
NM_001024860.1	0.02
NM_173966.3	0.02
NM_000594.4	0.10

Table 1a: terminal (leaf) branch lengths of NJ tree (figure 1) via Python implementation

Name	Terminal branch length
NM_000594.4	0.11
NM_214022.1	0.09
NM_001024860.1	0.02
NM_173966.3	0.02
NM_001082263.1	0.13
M13049.1	0.15
NM_012675.3	0.15
NM_001081819.2	0.06
NM_001003244.4	0.06
AJ417565.2	0.21

Table 2a: terminal (leaf) branch lengths of UPGMA tree (figure 2) via Python implementation

Notice that between the NJ tree and the UPGMA tree, the branch lengths are fairly similar, with values only being a couple of hundredths different. Also note in tables 1a and 2a, that the Channel catfish (AJ417565.2) has the longest branch distance, suggesting its TNF sequence is the most dissimilar to the other 9 species. The following tables compare the internal edge distances between the trees shown in figures 1 and 2:

Name	Non-terminal branch length
Inner 7	0.01
Inner 5	0.06
Inner 3	0.10
Inner 2	0.11
Inner 1	0.09
Inner 6	0.04
Inner 4	0.08

Figure 1b: non-terminal (internal) branch lengths of NJ tree (figure 1) svia Python implementation

Name	Non-terminal branch length
Inner 7	0.13
Inner 5	0.06
Inner 4	0.01
Inner 3	0.02
Inner 1	0.07
Inner 6	0.03
Inner 8	0.07
Inner 2	0.15

Figure 2b: non-terminal (internal) branch lengths of UPGMA tree (figure 2) via Python implementation

Note that since the NJ tree and UPGMA tree are constructed through different algorithms, the number of internal nodes differ (please refer to figures 1 and 2 above to observe internal node placement). It can be observed in tables 1b and 2b (and 1a and 2a) that the edge weight values are very small, suggesting that all the species have relatively similar TNF sequences and very little divergence has occurred.

Additionally, looking at figures 4-7, MEGA tree-building (with the exception of the MP tree in figure 8) places the edge distances on each tree constructed. It can be observed in those figures that the branch lengths do not exceed the distance value 0.66 (figure 7), again with the longest branch distances being attributed to AJ417565.2.

The NJ tree constructed from MUSCLE resulted in the following branch lengths:

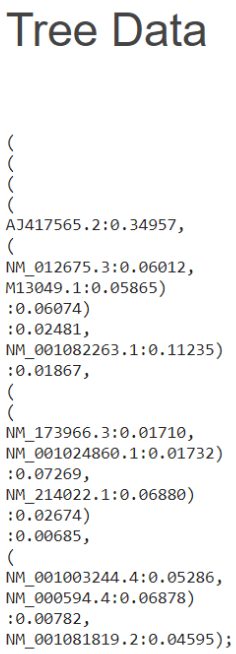

Figure 9: MUSCLE software NJ tree (figure 3) branch lengths

Notice in figure 9 that again, the longest branch length is attributed to AJ417565.2, again suggesting its dissimilarity with the other nine species' TNF sequences.

Since all of the branch length values for all trees (exception: figure 8) are very small values within hundredths of each other and no outliers, it can be said with confidence that all of the 10 species and their TNF sequences are closely related, and the main difference between the methods lies in the tree orientations, not the branch lengths. Also, note that the lack of edge distances in the MP tree in figure 8 makes it rather ambiguous, but it was still included as a comparison of the topology of the eight trees and the grouping of the 10 organisms using different tree-building algorithms. However, an important note regarding branch lengths is the fact that UPGMA trees assume a constant molecular clock while NJ trees do not, so the branch lengths of the NJ tree are liable to be more accurate.

Overall, while some similarities are obvious between all the trees and between the trees constructed using the same method, there are still notable differences between them, which makes phylogenetics such a complex and inexact field, even when based on sequencing data.

Conclusions

As can be seen from the above figures, due to the variation in the trees produced, it is not possible to draw absolute conclusions about which tree is the correct one. Even though three of the above trees are exactly the same even generated from different methods, many methods are used in conjunction to construct phylogenetic trees to best show the most likely and accurate

evolutionary picture, and as can be seen above, they all do not produce the same results. However, it can be said with a certain degree of confidence that characteristics shared between all eight trees (i.e. the grouping of sheep and cattle) and traits shared between seven of the eight trees (i.e. the grouping of mouse and brown rat) are likely correct. Additionally, as observed in figures 4-7 and in tables 1a, 2a, 1b and 2b, there were no stark differences in branch lengths, again suggesting the fact that all 10 species TNF sequences are closely related, even though trees in figures 4-7 were produced from four different tree-building algorithms.

Additionally, recall that TNF is considered an orthologous gene. A lack of divergence would be expected in its sequence between the 10 selected species, which was indeed observed since the TNF gene and its associated protein and its role within the 10 organisms has not altered significantly. Additionally, recall that UPGMA trees are susceptible to long branch attraction and assume a constant evolutionary clock. This makes NJ trees more evolutionary accurate, and this is related to branch distance, with NJ trees being more accurate. Also recall above in the Results section, there was no significant or outlier branch distance in any of the trees regardless of the method used. This would be expected since TNF is an orthologous gene between the species and its sequence and thus function has changed very little between the species.

As a whole, phylogenetics helps us to make sense of the world around us, to connect molecular pieces together into a larger puzzle that gives us an idea of how organisms have adapted and evolved over time. With DNA sequencing becoming faster and cheaper by the year, this provides accurate data on which to build trees. But again, sequencing data does not make phylogenetic trees infallible, and it is not 100% possible to guarantee that a given phylogenetic tree is completely accurate in its groupings or distance calculations. It is possible this will never be known with absolute certainty, since these organisms as we know them now have been evolving for millions of years, having predecessors that might not be discovered yet. Therefore, as stated above, these facts must be kept in mind when interpreting these trees, and a certain level of uncertainty must always be accounted for.

Overall, the Python implementation of phylogenetic tree construction was successful, and this project reiterated the importance and practical and functional use of the Biopython package, which has a plethora of modules and functions for many different bioinformatics tasks besides phylogenetic tree construction, all of which can be accessed on biopython.org. And through using Biopython, I was exposed to different programming functions I was not previously familiar with, notably the ".draw" function which prints an image to a new window. All in all, this project enhanced my understanding of phylogenetics and its history and uses, introduced me to Biopython and its implementation, and increased my programming and problem solving skills in a practical and relevant way.

Ultimately, the tree construction method of choice is going to vary depending on the data being investigated, the size of the data, and the desired results. Again, it is important that multiple methods be used in conjunction because even as I observed in this project, trees built using the same algorithm under different tools still produced different trees. Therefore, the pros and cons must be weighed when choosing a method and the results must be interpreted scrupulously.

References:

Introduction and Background:

[1a] Baum, David. (2008). Reading a Phylogenetic Tree: The Meaning of Monophyletic Groups. *Nature Education*. https://www.nature.com/scitable/topicpage/reading-a-phylogenetic-tree-the-meaning-of-41956/

[2a] Mackenzie, Ruairi J. (2023). RNA-Seq: Basics, Applications and Protocol. *Technology Networks.* https://www.technologynetworks.com/genomics/articles/rna-seq-basics-applications-and-protocol-299461#:~:text=First%2C%20RNA%20is%20extracted%20from,sequenced%20using%20next%2Dgeneration%20sequencing

[3a] Pace, Norman R. et al. (2012). Phylogeny and beyond: Scientific, historical, and conceptual significance of the first tree of life. *PNAS, 109*(4), 1011-1018. https://doi.org/10.1073/pnas.1109716109

[4a] *Phylogenies and the History of Life*. (n.d.). Lumen Learning. https://courses.lumenlearning.com/suny-wmopen-biology2/chapter/phylogenies-and-the-history-of-life/#:~:text=Many%20phylogenetic%20trees%20have%20been,for%20more%20than%20a%20century

[5a] Qie, Lixin. (2023). *3B: PAM* [PowerPoint slides]. Johns Hopkins University, retrieved from https://canvas.jhu.edu/.

Methods and Algorithms:

[1b] *Calculation of trees from alignments*. (n.d.). Jalview Help. https://www.jalview.org/help/html/calculations/tree.html

[2b] Dr. Samanthi. (2019). Difference Between Maximum Parsimony and Maximum Likelihood. *Difference Between*. https://www.differencebetween.com/difference-between-maximum-parsimony-and-maximum-likelihood/#Maximum%20Likelihood

[3b] *First Time User.* (n.d.). MEGAX-Help. https://www.megasoftware.net/web_help_11/index.htm#t=First_Time_User.htm

[4b] Guindon, Stéphane et al. (2010). New Algorithms and Methods to Estimate Maximum-Likelihood Phylogenies: Assessing the Performance of PhyML 3.0. *Systematic Biology, 59*(3), 307-321. https://doi.org/10.1093/sysbio/syq010

[5b] Hall, Barry G. (2013). Building phylogenetic trees from molecular data with MEGA. *Molecular Biology and Evolution, 30*(5), 1229-35. 10.1093/molbev/mst012

[6b] Hong, Yan et al. (2020). ENJ algorithm can construct triple phylogenetic trees. *Molecular Therapy Nucleic Acids,* 23, 286-293. https://doi.org/10.1016/j.omtn.2020.11.004

[7b] Kannan, Lavanya & Ward C Wheeler. (2012). Maximum Parsimony on Phylogenetic networks. *Algorithms for Molecular Biology, 7*(9). https://doi.org/10.1186/1748-7188-7-9

[8b] McCarthy, Charley G.P. & David A. Fitzpatrick. (2017). *Chapter Six - Multiple Approaches to Phylogenomic Reconstruction of the Fungal Kingdom* in *Advances in Genetics*, vo.l. 100, pp. 211-266. Academic Press. https://doi.org/10.1016/bs.adgen.2017.09.006

[9b] *Molecular Evolutionary Genetics Analysis.* (n.d.). MEGA. https://www.megasoftware.net/

[10b] Munjal G. et al. (2018). *Phylogenetics Algorithms and Applications* in *Ambient Communications and Computer Systems,* pp. 187-94. 10.1007/978-981-13-5934-7_17

[11b] Qie, Lixin. (2023). *9D: More on UPGMA, NJ* [PowerPoint slides]. Johns Hopkins University, retrieved from https://canvas.jhu.edu/.

[12b] Rains, Molly. (n.d.). Study. https://study.com/learn/lesson/maximum-parsimony-evolution-analysis-model-phylogeny.html

[13b] Rzhetsky, A & M Nei. (1993). Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution, 10*(5), 1073-95. 10.1093/oxfordjournals.molbev.a040056

[14b] Sharma, Ravit. (2019). UPGMA Method: Designing a Phylogenetic Tree. *Medium*. https://medium.com/@sharma.ravit/upgma-method-designing-a-phylogenetic-tree-9a708de18419

[15b] Waterhouse, A.M. et al. (2009) Jalview Version 2 - a multiple sequence alignment editor and analysis workbench. *Bioinformatics, 25*(9), 1189-1191. 10.1093/bioinformatics/btp033

[16b] *What Is the Genetic Algorithm?* (n.d.). MathWorks. https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

Implementation:

[1c] *Bio.Phylo.TreeConstruction module*. (n.d.). Biopython. https://biopython.org/docs/dev/api/Bio.Phylo.TreeConstruction.html

[2c] *Bio.Phylo.TreeConstruction.DistanceCalculator*. (n.d.). Homolog. https://homolog.us/Biopython/Bio.Phylo.TreeConstruction.DistanceCalculator.html

[3c] *Bio.Phylo.TreeConstruction.DistanceTreeConstructor*. (n.d.). Homolog. https://homolog.us/Biopython/Bio.Phylo.TreeConstruction.DistanceTreeConstructor.html#content

[4c] *Four point condition*. (n.d.). University of Chicago. https://people.cs.uchicago.edu/~ridg/digbio08/talkaddree.pdf

[5c] Gupta, Rishika. (2021). Phylogenetic Trees: Implement in Python. *Medium*. https://medium.com/geekculture/phylogenetic-trees-implement-in-python-3f9df96c0c32

[6c] *MUSCLE.* (n.d.). EMBL-EBI. https://www.ebi.ac.uk/Tools/msa/muscle/

[7c] *Permutation and Combination in Python*. (2023). GeeksforGeeks. https://www.geeksforgeeks.org/permutation-and-combination-in-python/

[8c] *Phylo - Working with Phylogenetic Trees*. (n.d.). Biopython. https://biopython.org/wiki/Phylo

[9c] Poudel, Mohit. (2022). Beginner's guide to Phylogenetic Tree construction using BioPython. *Medium*. https://medium.com/@poudelmohit59/beginners-guide-to-phylogenetic-tree-construction-using-biopython-5accbd8345a2

[10c] *Python math.comb() Method*. (n.d.). W3 Schools. https://www.w3schools.com/python/ref_math_comb.asp

[11c] Qie, Lixin. (2023). *9B: Distance Matrix Based Approach* [PowerPoint slides]. Johns Hopkins University, retrieved from https://canvas.jhu.edu/.

[12c] Sharma, Natasha. (2019). Importance of Distance Metrics in Machine Learning Modelling. *Towards Data Science.* https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d

[13c] *The module for multiple sequence alignments, AlignIO*. (n.d.). Biopython. https://biopython.org/wiki/AlignIO

[14c] *TNF Gene - Tumor Necrosis Factor*. (2023). GeneCards. https://www.genecards.org/cgi-bin/carddisp.pl?gene=TNF

Other:

[1d] National Center for Biotechnology Information (NCBI)[Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – [cited 2023 Dec 06]. Available from: https://www.ncbi.nlm.nih.gov/

[2d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_000594.4, Homo sapiens tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_000594.4

[3d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_214022.1, Sus scrofa

tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_214022.1

[4d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_001082263.1, Oryctolagus cuniculus tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_001082263.1

[5d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_012675.3, Rattus norvegicus tumor necrosis factor (Tnf), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_012675.3

[6d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. M13049.1, Mouse tumor necrosis factor (TNF) mRNA, complete cds; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/M13049.1

[7d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. AJ417565.2, Ictalurus punctatus mRNA for tumor necrosis factor (TNF gene); [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/AJ417565.2

[8d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_173966.3, Bos taurus tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_173966.3

[9d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_001024860.1, Ovis aries tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_001024860.1

[10d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_001003244.4, Canis lupus familiaris tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_001003244.4

[11d] Nucleotide [Internet]. Bethesda (MD): National Library of Medicine (US), National Center for Biotechnology Information; [1988] – . Accession No. NM_001081819.2, Equus caballus tumor necrosis factor (TNF), mRNA; [cited 2023 12 07]. Available from: https://www.ncbi.nlm.nih.gov/nuccore/NM_001081819.2