Bioinformatics: Group Assignment 2
Phylogenetics of the Hemoglobin Gene

Pedro Carvalho - 201906291
Pedro Leite - 201906697

Tools and Packages
For this project we used the programming language Python3 (3.9.5), with the packages: biopython (1.79), uniprot, unipressed and matplotlib.

Exercise 1: Data Collection
In the first exercise we used the package "uniprot", to fetch the data from the input argument. The input argument is a single Uniprot Id, if it isn't, the function doesn't return anything and prints out an error message. From the data fetched, that is in dictionary form, we filter the "sequence" and its "value". Finally, we print out the "id" and the "sequence" in a new "sequence.fasta" file, to be analysed in the next exercise.

Exercise 2 : Blast Analysis
This function reads the file "sequence.fasta" containing the sequence from the Uniprot Id in the input argument, and performs a BLAST search against the "nr" (non-redundant) protein database using the "blastp" (protein-protein BLAST) algorithm, from the package "biopython". This search identifies proteins from a wide variety of species that align with the input sequence.

From the BLAST search, it parses the XML output, filtering the top 10 non-human hits based on alignment scores. The function collects the species name, e-value, hit identifier, and query sequence for these top 10 hits. It sorts these hits by e-value to ensure the best matches are at the top. The function then writes these top 10 sequences in a new file "sequences_to_analyse.fasta".

For the Uniprot ID "P68871", it gave 8 answers.

Exercise 3: Multiple Sequence Analysis
The MSA function, starts by reading the sequences from the file "sequences_to_analyse.fasta", and sends them to Clustal Omega through a POST request, specifying email for notifications and the output format ("clustal_num").

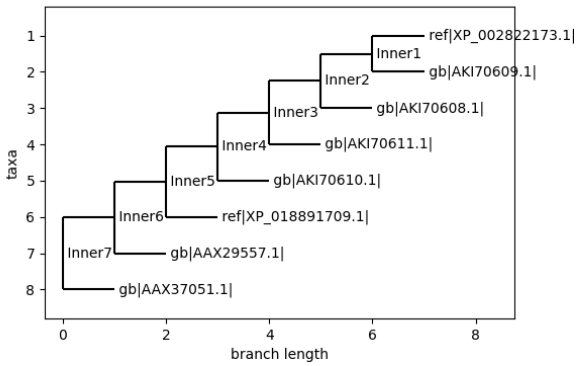
If the request is successful, the function gets a job ID and repeatedly checks the job's status, printing messages to indicate whether the alignment is running, pending, or finished. Once the job is complete, it retrieves the alignment results and saves them to

"alignment.txt." If errors occur, it displays appropriate messages. This approach allows for automatic multiple sequence alignment through a remote service.

The results that we got show an high level of conservation between all the sequences.

Exercise 4: Phylogenetic Tree

This function begins by reading “alignment.txt”, then using the “DistanceCalculator” with "identity" as the distance metric, it generates a distance matrix from the alignment. With this distance matrix, the “DistanceTreeConstructor” constructs a tree using the UPGMA algorithm. To visualize and save the tree, we used the package “matplotlib”.



Conclusion

The phylogenetic tree represents the evolutionary relationships among a set of sequences. The sequences gb|AAX29557.1| and gb|AAX37051.1| are the closest relatives, sharing a common ancestor with no additional branches between them. A larger group of sequences, including ref|XP_002822173.1|, gb|AKI70609.1|, gb|AKI70608.1|, gb|AKI70611.1|, gb|AKI70610.1|, and ref|XP_018891709.1|, form a closely related cluster, with slight variations indicated by the branching structure. The branches' lengths suggest differing degrees of evolutionary divergence, with shorter branches indicating closer genetic relationships. Overall, this tree shows a common evolutionary origin, with variations among sequences representing their evolutionary distances.