Bioinformatics
Group Assignment 2

Done by

Ana Sofia Teixeira (201906031)

Guilherme Duarte (201905583)

José Carvalho (202005827)

Curricular Unit of Bioinformatics

May 14, 2024

Department of Computer Science

1 Introduction

The goal of this assignment is to explore the phylogenetics evolution of hemoglobin across species, with relation to the hemoglobin gene found in human. We were challenged to explore and analyse different Python packages, namely *uniprot*, *biopython* and *matplotlib*; to retrieve sequence information given its identifier (Section 2); to execute the Basic Local Alignment Search Tool (BLAST) algorithm to find other species different from human with similar molecular sequences (Section 3); to perform Multiple Sequence Alignment (MSA) on the sequences found (Section 4); and to build and draw a phylogenetic tree (Section 5). Section 6 concludes our work.

2 Data Collection

For this task we developed the method `data_collection()`, where we retrieved the molecular sequence and additional information from a given identifier. One of the goals for this task was to explore the *uniprot* package. However, during implementation, we found out that this package is not updated and is not able to fetch data from the server. In fact:

```
>>> import uniprot
>>> uniprot.batch_uniprot_metadata('P68871')
Fetching metadata for 5 Uniprot IDs from http://uniprot.org ...
Error in fetching metadata
{}
```

This problem seems to be around since 2016 [1]. As a work around, we used the *requests* Python package to send a GET request to the UniProt server (*https://rest.uniprot.org/uniprotkb/*), using the given sequence identifier as a query. The server then returns a JSON which we are able to analyse and extract the organism's scientific name and molecular sequence. Finally, we store this information in the file `sequence.fasta`.

3 BLAST Analysis

The goal of this task is to obtain the sequences of the ten species other than human with the highest matching score with the one obtained in Task 1. It is implemented in the method `BLAST_analysis()`. We use the function `Bio.Blast.NCBIWWW.qblast` from *biopython* to perform the BLAST search, that communicates with the NCBI's QBLAST server (*https://blast.ncbi.nlm.nih.gov/Blast.cgi*). The call to the function takes several arguments:

```
NCBIWWW.qblast(program='blastp', database='nr',
               sequence=sequence, hitlist_size=100)
```

where the program `blastp` stands for protein blast and implies that the comparison will be between protein sequences; the database `nr` contains protein sequences where all the entries with absolutely identical sequences have been merged; the `sequence` is the one obtained previously; and the size of the hit list must be large enough to contain ten different species, since there may be multiple sequences from the same species. Due to the long execution time of this function (4 minutes on average), we decided to store the results in the file `blast_results.xml`. Then, we use the function `Bio.Blast.NCBIXML.read` to return the single BLAST from the query. Finally, we simply iterate over each alignment and store the ten sequences with the highest scores and their respective species in the file `sequences_to_analyse.fasta`.

4 Multiple Sequence Alignment

After combining our local FASTA file and the sequences generated from the BLAST analysis, we submit them to the EBI's Job Dispatcher tool (*https://www.ebi.ac.uk/Tools/services/rest/clustalo*) for alignment. We monitor the request, wait for its completion, and then retrieve it in the `clustal` format. The primary result is stored in the file `alignment.txt`, which contains all sequences aligned according to Clustal Omega's algorithm. The main difficulty of this process involves monitoring the status of the job, which might not be efficient; hence, the implementation must manage the various responses from the web and handle possible errors in job processing. Handling of responses and potential errors occurs throughout the `do_request()` function, responsible for all communication with the Clustal Omega API. All this code is present in the `multiple_sequence_alignment()` function.

5 Phylogenetic Tree

The code needed for this task is implemented in the function `phylogenetic_tree()`. We first need to construct a distance matrix from the alignment data obtained on the last task. We use the class `Bio.Phylo.TreeConstruction.DistanceCalculator` and, in particular, the `identity` model as standart parameter for this metric. Then, we build the distance matrix by calling the function `get_distance(alignment)`, where the parameter is the set of alignments retrieved from the MSA. After that, we instantiate the class `Bio.Phylo.TreeConstruction.DistanceTreeConstructor` and we use the Unweighted Pair Group Method Using Arithmetic Averages (UPGMA) algorithm to build the tree, by calling the function `upgma(distance_matrix)` from the class, that takes the distance matrix previously calculated as input. Finally, we run:

```
Phylo.draw(tree, branch_labels=lambda c: round(c.branch_length, 5))
```

with the `tree` obtained from the last step, and we set the labels of the branches as their distances. The obtained tree is depicted in Figure 1.

6 Conclusions

This assignment allowed us to explore different packages and tools and learn their utility. At the end, we learned how the phylogenetic tree reveals the evolutionary relationships between human hemoglobin and other species hemoglobin genes. It illustrates how this gene has diverged and evolved over time, allowing us to pinpoint common ancestry among species.

References

[1] *Error in fetching metadata · Issue 5 · boscoh/uniprot — github.com.* `https://github.com/boscoh/uniprot/issues/5`. [Accessed 13-05-2024].

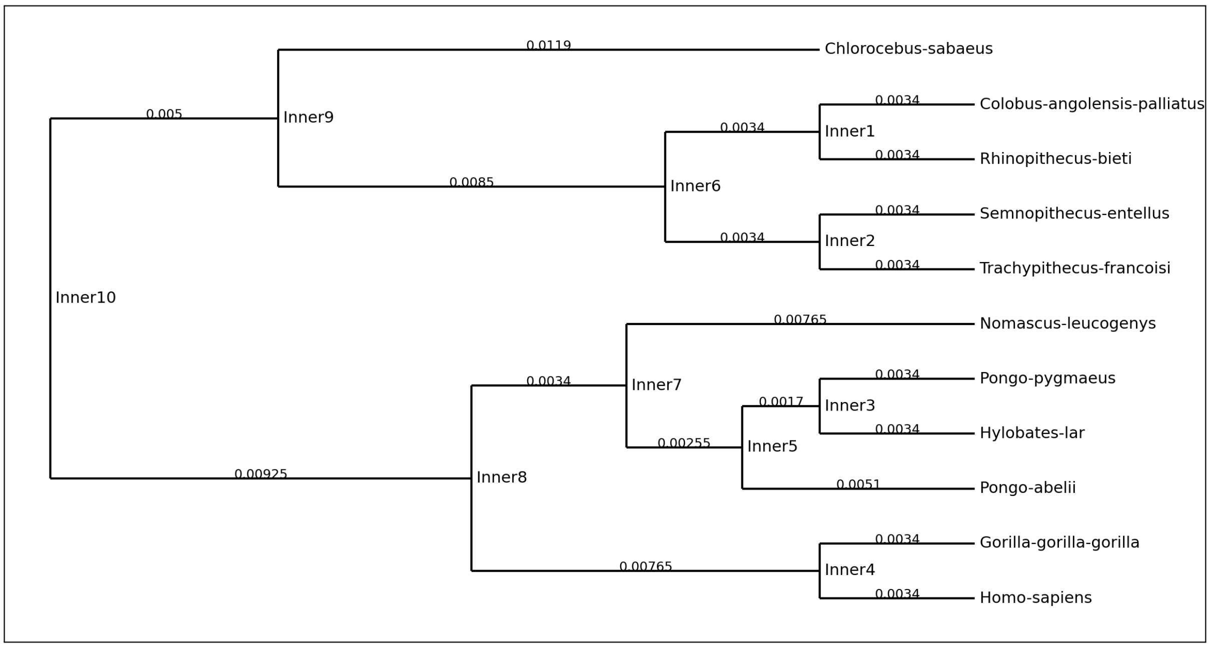

Figure 1: Phylogenetic tree for the hemoglobin subunit beta