Project description

The aim of this software is to allow users to perform various tasks on genomic sequences obtained through FASTA files.
The architecture of the code allows it to be utilizable for different kinds of sequences, starting from DNA (nuclear, mitochondrial, short sequences, long genes) up to protein sequences, in order to analyze their main characteristics and behaviours.
The class “Parser” is a class that allows users to perform the parsing of different possible kinds of files available for genomic sequence storing and analysis.
This class has then a subclass called “Fasta”,which inherits from “Parser” all of its methods and attributes, that is specifically made to open and parse FASTA files, one of the most common formats to visualize nucleic acids and protein sequences.
This feature allows users to download sequences from databases and analyze them using our tools.
The output of the FASTA file parsing is returned in terms of a dictionary, or a Pandas dataframe, containing all the sequences that can be found in the file.
Then extensive analysis is enabled by a series of methods contained in the class “Analyzer”, most of which are static methods.
They are in fact utility methods, not connected to specific instances of that class, that can be useful and called on whichever genomic sequence desired by the user.
The methods contained in this class include:

- “GC_content” : performs the calculation of GC content of a given sequence in terms of percentages. The GC content is one of the most important characteristics of genomic sequences and is used to recognize their origin inside a genome and to identify sequences from different genomes.
- “reverse_comp” : computes the reverse complement of a given sequence
- “align” : performs the global alignment of two given sequences using the Needleman Wunsch algorithm. This is particularly important to understand similarities and differences between sequences and to study their evolutionary relationships and potential homology.
- “motif_analysis” : determines the distribution of a particular kind of sequence motif in the input sequence. This kind of analysis is particularly

useful to identify repeated motifs and is implemented using Pandas DataFrames
- “sequence_length” : determines the length of a given sequence
- “transcription” : simulates transcription of a DNA sequence to get the corresponding RNA sequence
- “extract subsequence” : extracts given subsequences from a genomic sequence if they are found in it.

Finally, to represent short sequence motifs we have implemented a class “Sequence_motif”, which can interact with the Analyzer class to enable high-level analysis of sequence motifs in genomic sequences, finding the length of motif, their position with respect to a longer subsequence and their function.