Instructions to Run

Only one file needs to be run to be able to initiate the program, this is named "protein_analysis.py". To run it, you must type "./protein_analysis.py" in the command line. The program explains that by entering a protein family name and taxonomic tree group name, this query will be searched on NCBI protein database, from which relevant protein sequence FASTA files can be retrieved. Various analysis can them be carried out on these sequences.

As a general issue to note, when running the program to search on an NCBI database, if a search query you are sure should have results has flagged up as "not having any results", please re-enter the same query. Moreover, when entering text in response to a question outputted by the program, especially for one letter answers, if you mistype a character and backspace to edit the text, it is probable that the input may be affected, resulting in it not being identical to what you have entered. In this case, please re-enter the same text when asked again for it. Also, for any yes/no answers, any input other than for 'yes' is generally taken to be 'no'.

1. **Retrieving the query FASTA files**

Firstly, your protein family query is required. This may be as specific as desired, for example entering the enzyme "glucose-6-phosphatase" will result in a smaller subset of results of that exact protein. However, entering "kinases" is very broad – this is accepted and will be searched on NCBI protein database, but only the first 1000 results will be fetched by the program. This is to ensure that the analyses can work efficiently. It is advised that the protein family name is given in singular form (i.e. not plural) in order to limit the results achieved. The taxonomic group is asked for next, to which you can input any name in the taxonomic tree. To ensure that results and fairly limited, it is recommended to input a taxon from class onwards (i.e. class, order, family, or genus). Please be warned that entering a species may not result in more than one result, depending on how general the protein query is. Because of limit of 1000 protein sequences, it would be ideal if your query is not too broad. If it is quite generic (such as giving a phylum taxon), the sequences retrieved, alongside subsequent analyses, may not be representative of your query. Figure 1 shows the example of using "glucose-6-phosphatase" and the taxonomic class "aves". These two queries are not case sensitive and can have spaces between words. Some symbols are accepted, however most are not and will not return a correct result. In this case, the query must be re-entered until accepted.

All outputs from this program will be saved under one directory, which you can name yourself. Most files outputted will also start with this name, to be consistent. The prompt from the program indicates that any spaces in the specified name, as well as a few symbols indicated in Figure 1, will be changed to an underscore symbol. Moreover, you have the option to include all partial sequences too in your search or not. Choosing "Y" means all sequence form your search will be retrieved, and choosing "N" means only the complete sequences will be retrieved. The example in Figure 1 shows results when choosing no.

Finally, the output of the search on NCBI protein database is saved in the directory named by you. In this example, all outputs are saved in the directory "glucose-6-phosphatase_aves". A summary of the first 5 results are summarised and shown on the screen, so that you can identify if the result seems to be as desired. Any ambiguous protein name results have already been filtered out form the result. If 'yes' is chosen, a file containing all protein results FASTA header (starting with ">") followed by their corresponding FASTA sequences will be saved to /glucose-6-phosphatase_aves/glucose-6-

phosphatase_aves.pro.fa. If ‘no’ is chosen, you can re-enter your query until a desired output is achieved.

Furthermore, for all yes/no answers in this program, any uppercase or lowercase variant of “y” or “yes” can be used for yes, similarly as “n” or “no” for no.

```
By inputting a protein family and taxonomic group, fasta sequences of proteins found form NCBI protein database will be saved. Please be
 warned that if the query search contains more than 1000 protein results, only the first 1000 will be considered and your search will ta
ke longer.
What is the protein family of your query?
glucose-6-phosphatase
What is the taxonomic group of your organism query?
aves
What would you like to name the folder and subsequent outputs from this search? Please note that any of the following characters, as wel
l as a space, will be changed to '_' for readability: '\ / , ; :'
glucose-6-phosphatase_aves
Would you like to consider all protein sequences found from your search, including those that contain an incomplete version of of the am
ino acid sequence (partial)? Please input 'Y' for yes or 'N' for no.
N
Retrieving protein sequences of query from NCBI databases...
Your query has returned 61 protein sequences of the protein family glucose-6-phosphatase and taxonomic group aves. The first few outputs
 of this search are shown below:
>KAJ7421106.1 Glucose-6-phosphatase [Willisornis vidua]
>KAJ7396366.1 Glucose-6-phosphatase [Pitangus sulphuratus]
>KAI6072612.1 Glucose-6-phosphatase [Aix galericulata]
>KAI1230272.1 Glucose-6-phosphatase [Lamprotornis superbus]
>XP_040473657.1 glucose-6-phosphatase [Falco naumanni]
0
Is this your desired search? If true, please type 'Y' for yes or 'N' for no.
y
Great! The fasta sequences for your protein query are saved under ./glucose-6-phosphatase_aves/glucose-6-phosphatase_aves.pro.fa
```

Figure 1 Retrieving FASTA sequences for the query “glucose-6-phosphatase” and “aves”.

2. Plotting Protein Conservation

Figure 2 shows the interface for aligning the protein FASTA sequences retrieved and determining the conservation across them. This step will be initiated by typing “Y” after being prompted. In this example, the aligned sequences will be saved in /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_aligned.pro.fa which positions the amino acids that align, and fills gaps with “-“.

```
Would you like to determine and plot the level of conservation between the protein sequences of your query? Please input 'Y' for yes or
'N' for no.
Y
Aligning clustered query sequences...
Aligned fasta sequences of your protein query have been saved in ./glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_aligned.pro.fa
Generating plot of conservation of query sequence...
Your generated conservation plot has been saved in the path ./glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_conservation_plot.pn
g
Would you like to view the conservation plot? Please input 'Y' for yes or 'N' for no.
Y
Plot is opening in separate window...
```

Figure 2 Plotting conservation across protein and taxon query.

A corresponding conservation plot has also been saved to /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_conservation_plot.png which is made based on the aligned FASTA sequences. The program prompts you as to whether you would like to view the image. Returning “Y” will open the PNG plot, however it may take a short while. Figure 3 is the conservation plot for this example, where a higher similarity corresponds to more conservation in glucose-6-phosphatase across Aves species. In this case, it appears that amino acid positions of about 80 - 200, 210 - 292, 296 - 480 have remained the most conserved across Aves. Similarity scores have peaked at about 0.95, but mainly range between 0.45 - 0.9. In contrast, the first 90 amino acids seem to have not been conserved at all across species. This may indicate evolutionary divergence.

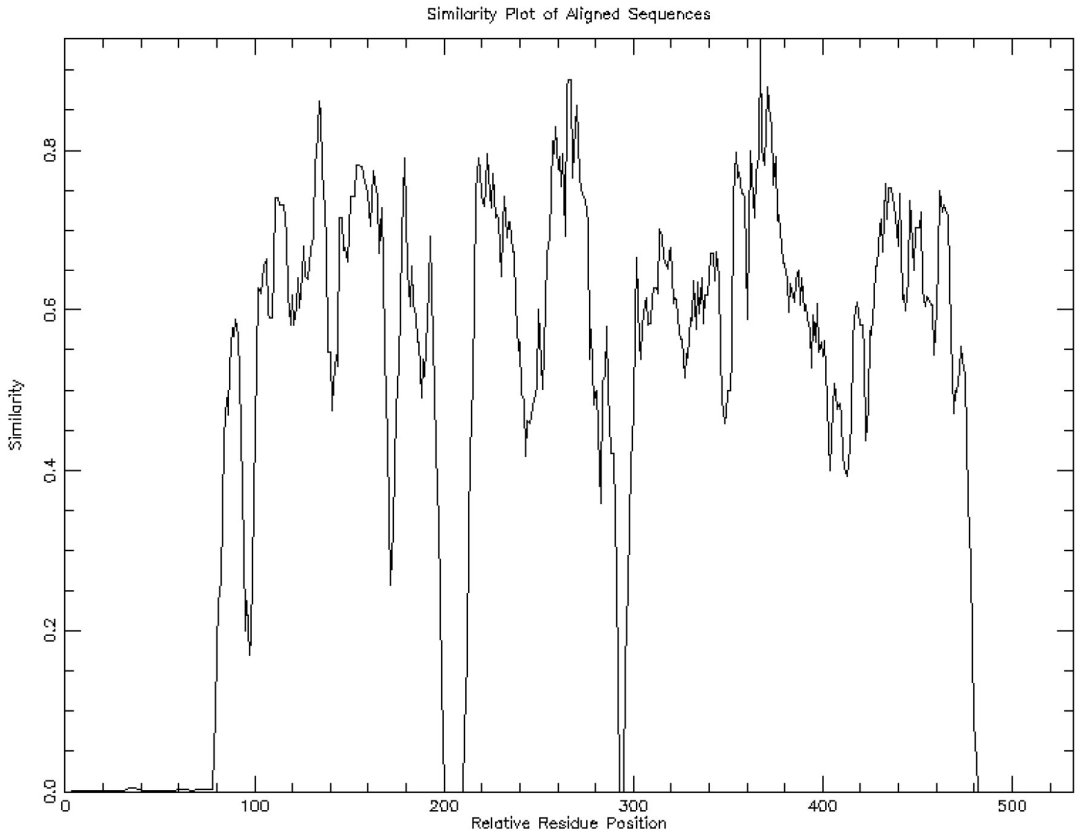

Figure 3 Plot of conservation of amino acid residues across the protein and taxon query.

3. Identifying Motifs from PROSITE

The program will prompt you to choose whether you would like to identify motifs from your query search. Returning "Y" will use the PROSITE database to identify any sequences corresponding to motifs are present in the FASTA sequences. A summary of the motifs found will be outputted to the screen, showing the name of the motif and the number of times it was found. In the example of glucose-6-phosphatase in Aves, the motif "amidation" was found 49 times. In order to get much more detail about the motifs found, the report showing the motifs per sequence in the search is found in /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_motifs.txt. In this file, each sequence is listed one at a time, and the "Hits found" per sequence refers to how many motifs were found per sequence. Glucose-6-phosphatase sequences across birds containing the same motif are likely more closely related to each other.

```
Would you like the identify the motifs from PROSITE in your protein query sequences? Please input 'Y' for yes or 'N' for no.
Y
Identifying motifs in the protein sequence query...
Summary of motifs found in protein sequence query below:
Found the motif AMIDATION 49 times.
The full output of motifs per protein sequence from your query can be found in ./glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_m
otifs.txt
```

Figure 4 Interface to identify motifs in the protein and taxon query sequences.

4. Running Protein BLAST

This section involves you picking a motif type found in your original protein and taxon search (identified in step 3). The FASTA sequences retrieved form your original query already saved will be searched to extract the sequences that contain the chosen motif. This subset of the sequences will be used to make a database for protein BLAST. As shown in Figure 5, the user is first prompted to return "Y" to initiate this step. In order to carry on, step 4 must have already been completed. If it has not, you will be prompted to do so now. If motifs have already been identified, then you can choose the motif to base the protein database on by entering the number corresponding to your chosen motif. In this case, only one motif type was identified, so there is only one option: inputting "1". Any other inputs will not be accepted, and you will have to re-enter a number until it matches one of the motifs shown in the list on the screen. The glucose-6-phosphatase proteins in Aves will be searched in the /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves.pro.fa file to extract the FASTA sequences containing that motif. A database will then be created on these sequences and saved in the directory /glucose-6-phosphatase_aves/blast_database/ with file names "AMIDATION" with various suffixes.

You will then get to choose what protein to use as the query in BLAST against this database. You are given the choice to either (1) enter your own accession number (already achieved from NCBI protein database), or to (2) input a different taxon for which NCBI protein database will be searched using the new taxon and same protein. If the same taxon is entered as in the original query, the results will not be biologically relevant, as BLAST will return very significant results, since the query and subject sequences are likely identical. If you are going to input your own accession number, be aware that the program will not disallow accession numbers of different proteins. For this analysis to be more biologically relevant, it is recommended to input an accession number of the same protein but of a different taxon. Please be aware that if you choose the second option, partial sequences are not filtered in the search, and only the first 10 results from NCBI will be retrieved and shown to the screen. Figure 5 shows the results for the new query of taxon, entering the species "homo sapiens". Only 3 results show up on the protein database in NCBI for "glucose-6-phosphatase" and "homo sapiens", so there are three accession numbers to choose from. You can then pick the accession number desired by entering the number associated with it for the list on the screen. In this case, AAA16222.1 is chosen, and this could similarly have bee entered during option (1) mentioned before.

```
Would you like to make a database of a certain motif found in your protein sequence query to subsequently run blastp on a separate query
 against? This involves running blastp across the same protein family, but with two different taxonomic groups as query (new) and subjec
t (original query: to make database). Please input 'Y' for yes or 'N' for no.
Y
Please pick the motif for which you would like to create a blast database. This database will be constructed using the proteins from you
r query search containing this motif.

1: AMIDATION
Input the number corresponding to your chosen motif.
1
Identifying motifs in the protein sequence query...
The database out of proteins from your query with your chosen motifs is saved in ./glucose-6-phosphatase_aves/blast_database/
Would you like to (1) enter your own query accession number for a taxon and glucose-6-phosphatase or would you like to (2) search up top
 results for a general taxon? Please enter '1' or '2' corresponding to the these options.
2
What is the taxon of your new query to blastp glucose-6-phosphatase againist your initial query of txid8782? Please note that only the f
irst 10 NCBI protein database results will be considered.
homo sapiens
Finding results for your new query on NCBI protein database...
1: P35575.2

2: AAA16222.1

3: CAA65638.1

Which species would you like to pick from the list to blast against your protein motif query database? Please input a number from the li
st of choices from above.
2
```

Figure 5 Picking the motif from those identified from your query. Proteins containing that motif are used to make a database that can be used for BLAST. This is followed by picking the new taxon for the same protein to use in BLAST.

Protein BLAST will then be run on your accession number query of the taxon you stated against the database made. In this example, the full BLAST output is saved in /glucose-6-phosphatase_aves/AMIDATION_homo_sapiens_blast.out. You will then be prompted to input the threshold for E-value for your BLAST results. If the threshold you choose is too small and all E-values

form your BLAST are higher than this value, you will be told the lowest E-value and then re-asked to enter the E-value threshold you desire. Then, the top 3 results (sorted on E-value) will be shown on the screen. A lower E-value is associated with a higher similarity between glucose-6-phosphatase in the query (*Homo sapiens*) and the Aves database. A higher bit score also corresponds to higher similarity between sequences. As seen in Figure 6, the top three results have an E-value of 0.0, meaning they are very similar sequences. The full results with E-value lower than the threshold you entered are also saved in /glucose-6-phosphatase_aves/AMIDATION_homo_sapiens_blast.csv in this example. This is a CSV file that contains the *Homo sapiens* query accession number, Aves accession number, % identity, alignment length, number of mismatches, number of gap openings, the *Homo sapiens* protein *query* sequence start and end position, Aves protein sequence start and end position, E-value, and bit score.

```
Running blastp of your new query against your intial protein motif query database...
What is your E-value threshold to indicate a significant match? An example of this may be 0.01 or 0.005
0.001
The top performing 3 results of the blast with your query are tabulated below (sorted by E-value):
   Query Accession Number aves Accesion Number  % Identity  ...  aves Sequence End  E-value  Bit Score
0              AAA16222.1       XP_010295685.1      69.553  ...                418      0.0      537.0
25             AAA16222.1       XP_027558020.1      69.188  ...                357      0.0      529.0
26             AAA16222.1       XP_031990407.1      69.468  ...                357      0.0      529.0

[3 rows x 12 columns]
The protein with the lowest E-value from your blast, and thus the most significant match to your query sequence of homo_sapiens has acce
ssion number XP_010295685.1
The full list of blast results with an E-value lower than 0.001 can be found in ./glucose-6-phosphatase_aves/AMIDATION_homo_sapiens_blas
t.out or in csv format in /glucose-6-phosphatase_aves/AMIDATION_homo_sapiens_blast.csv
Would you like to run blast again with a different motif or taxon query? Please input 'Y' for yes or 'N' for no.
no
```

Figure 6 Running protein BLAST of the new taxon protein sequence against the newly made database protein from the original query containing a certain chosen motif.

This process can be re-run with different queries, or by choosing different motifs. The BLAST results can be beneficial to understand which proteins have the highest similarity in sequence, and identify whether this is in large part due to the conserved motif domain. Further analysis of interest after BLAST includes determining the phylogenetic tree connecting the original taxon query sequences to the protein family in the new taxonomic group.

5. Predicting Transmembrane Segment Regions

The prompt will ask you whether you would like to identify the transmembrane segment regions, as seen in Figure 7. This is done through supplying the aligned FASTA sequences from step 2. The generated plot of the query of glucose-6-phosphatase and Aves in this case, is saved in /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_tmap.png. The raw prediction data is saved in /glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_tmap.res. This shows the number of hits corresponding to predicted transmembrane segments.

```
Would you like to predict whether there are transmembrane segments in the proteins of your query? Please input 'Y' for yes or 'N' for no
.
Y
Your transmembrane segment plot is saved in ./glucose-6-phosphatase_aves/glucose-6-phosphatase_aves_tmap.png
Opening this plot in a separate window...
```

Figure 7 Interface to predict transmembrane segments of the original query.

The plot is then opened to a separate window and may take a short while. Figure 8 shows the predicted regions of transmembrane segments for glucose-6-phosphatase and Aves aligned FASTA sequences. The solid line indicates the region predicted to be a transmembrane region, while the dashed line indicates the prediction to be at the end of the transmembrane region, meaning either sitting extracellularly or intracellularly. The y-axis corresponds to likelihood; therefore, a high likelihood in the solid line means that there is a high chance that that section is a transmembrane segment. The number of peaks in the solid line correspond to the number of transmembrane

segments. In this case, it appears that 8 segments are predicted to span the membrane. Glucose-6-phosphatase in reality contains 12 transmembrane segments, so it is not predicted completely accurately. However, this gives an indication to which regions in the sequence encode the protein that spans the membrane. If your query results in an empty plot, this means that your query likely does not have any membrane-spanning segments.

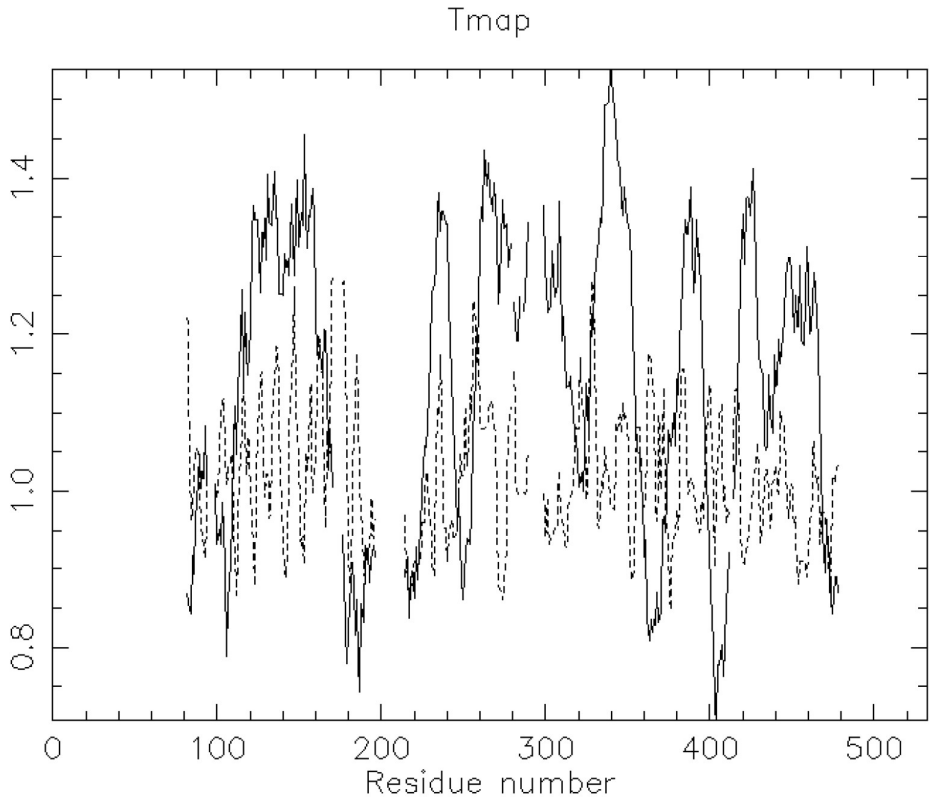

Figure 8 Prediction of transmembrane segments of protein and taxon query aligned FASTA sequences.

Maintenance manual

Each sections in the code is split with line of "#######". Figure 9 summarises the steps occurring in the program. Throughout the code, it is common to have while True loops that are only broken once the user has entered a valid input. They will be prompted repeatedly until so. Moreover, commands run through subprocess.call typically have stderr=subprocess.DEVNULL or stdout=subprocess.DEVNULL in order to prevent and errors or output of the command respectively to output to the user's screen.

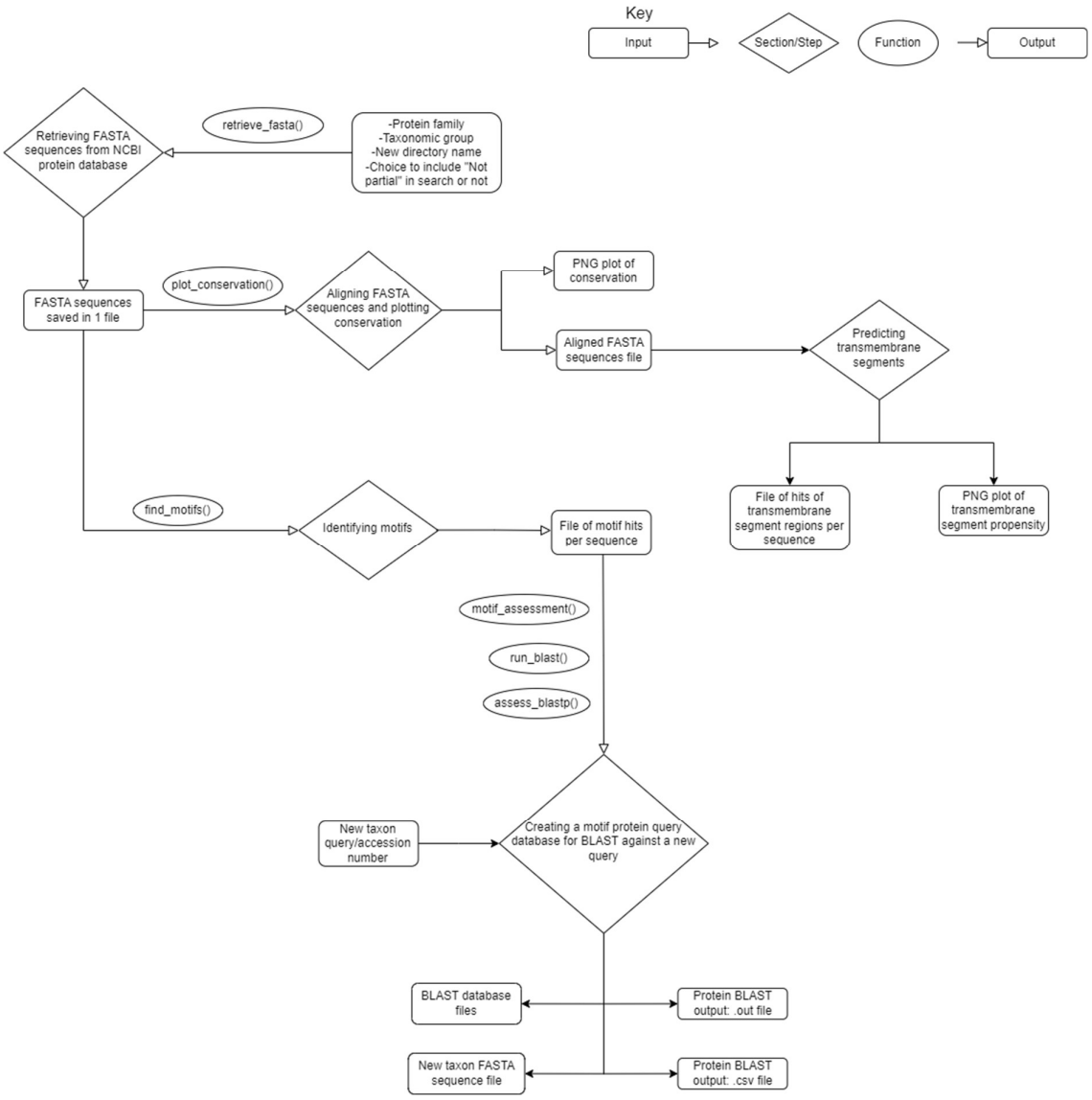

Figure 9 Flowchart of the inputs, outputs, functions, and main steps in the program.

1. Retrieving the query FASTA files

Firstly, the user inputs the protein family and taxonomic group to be queried on NCBI. Invalid characters or names are either replaced with a space, or detected followed by requesting the suer to re-enter a valid name, or will proceed to the search in which they will return no results. If the taxonomic group inputted is valid on NCBI taxonomy database, it will be converted to the taxonomic ID, in the example of Aves, it will change to “txid8782”. These protein_family and taxnonmic_group variables are mentioned throughout the code to ensure that each query will have its own version of output. The user will then pick the name for the folder for all outputs, and invalid characters will be replaced with an underscore. An error trap is in place to detect whether this file already exists; if it does, the user has the choice to save this query’s outputs in that same folder, or to create another with a different name. Each user input for “yes” or “no” allows any variation of case, meaning it is not case sensitive. If the user’s inputs are valid, the function retrieve_fasta() is called, that intakes an argument

that refers to the search query for the esearch command. Once the NCBI FASTA sequence results are fetched and saved to a file with suffix .pro.fa, the first 5 results are shown to the screen: if the user decides not to keep these results, the entire folder created will be deleted. This is repeated until a desired output is achieved, in which the folder and .pro.fa file is kept.

2. Plotting Protein Conservation

From now onwards, each section will only occur if the user has chosen a valid input of case insensitive "Y" or "YES". Moreover, from here onwards, the FASTA .pro.fa file must exist already, however this part of the code will not run until the user has decided to keep the results from step 1, which includes this FASTA file.

Secondly, the FASTA sequence file will be used to create a new file that contains the aligned FASTA sequences as output. For this to occur, the plot_conservation() function is called, that requires the arguments relating to the new folder name and the path for the new aligned sequence file. This function carries out the alignment and also generated a PNG file of the plot of conservation across sequences. This file is also saved in the same folder.

3. Identifying Motifs from PROSITE

If the user wishes to identify motifs in the sequences, the function find_motifs() is called, taking the arguments for a list of the sequences (split by ">" from the .pro.fa FASTA sequences file), and arguments "find" and "motif_chosen" which are defaulted to None. These latter two variable are not assigned during this step, so remain as None. The list of sequences is looped, and each sequence is saved (overwrites) to a temporary file. This temporary file is used as input to find motifs using patmatmotifs. A temporary file is required as patmatmotifs only reads one FASTA sequence per file. Each output is then appended to the end of a file with suffix motifs.txt. The temporary file is deleted and the new motifs.txt file is saved to a list variable "motif_sequences", separating each sequence. The separator "\nnew sequence" is used instead of a pre-existing character, to ensure that they are separated correctly. Then since find == None, each line from this list is looped through, each time extracting the section that contains the motif name. If a motif name is not present, an error trap is in place to continue to the next iteration. The count per motif is found using .get() and these are saved to a dictionary. If this "motifs" dictionary is not empty, found by len(motifs), the dictionary is looped through to print each motif and corresponding count.

4. Running Protein BLAST

This section makes use of four functions, summarised below:

- **find_motifs():** As described above, this function finds the motifs from the sequences. In this case, the variables "find" and "motif_chosen" are assigned when calling this function. Since the argument "find" is now equal to True, and thus not None, this function will return a non-redundant list (set) of accession numbers corresponding to a chosen motif. find_motifs() ensures that if this function has already been run, meaning a motif.txt has already been outputted, if will first be deleted before rerunning. This is because the output of patmatmotifs appends to the end of the file, and thus would double the results each time and adversely affects future analysis.

- **motif_assessment():** Intakes the chosen motif name and runs find_motifs() to extract the accession names for that motif. This function outputs a new subfolder with files forming the database for proteins containing this chosen motif. If the folder already exists, there is an error trap in place to use this same folder for outputs.
- **run_blast()**: Defines a subfunction carry_on() which is called later in the function if certain conditions are met. This function deals with using esearch and efetch to retrieve the new taxon query FASTA sequence and saves it to file also with suffix .pro.fa. Similar to step 1, there are conditionals and error traps in place to ensure that the user's input for the NCBI search is valid, and if there are no results, the user can re-enter the query. Any temporary files created and deleted at the end of the function. BLAST is then run for the new taxon query with the database created in motif_assessment(). The output is saved to the same main folder with suffix blast.out, and the name of the new taxon query is returned.
- **assess_blastp()**: Assesses the results of BLAST. Takes the chosen motif name and new taxon query as arguments and reads the BLAST output as a pandas data frame. Columns are assigned to the data frame and reads input form the user for an E-value threshold. There are conditionals and error traps in place to ensure that the a valid format of the threshold is given. If at least one E-value from the BLAST output file blast.out is lower than the threshold, this data frame will be sorted to contain rows with E-values lower than the threshold, and sorted in ascending order of E-value. This data frame is saved as a CSV file with suffix blast.csv.

The main code that calls all these functions first checks whether step 3 has been run, i.e. whether the motifs.txt file has already been created or not. If it hasn't, then the user has the choice to run it: if they agree, then step 4 will proceed, but if not then they cannot run step 4 and the while True loop is broken. This is controlled by conditional statements. If proceeding, the user is then asked to input a number corresponding to a motif found from the query sequences. If they enter an invalid input or a number that it not on the list of motifs, they are re-asked until they enter a valid input. After the functions above are called, the user has the choice to repeat this process or not. If they wish not to, then the while True loop is broken.

5. Predicting Transmembrane Segment Regions

This section will only run if the user has completed steps 1 and 2. Aligned FASTA sequences are required to run the tmap command from bash, to predict the transmembrane segment regions. If the user decides to proceed, this section outputs two files, one with a suffix of tmap.res that contains text referring to the predicted transmembrane sequences, and one with a suffix of tmap_plot.png which is a PNG file of a plot. They are both saved in the same general folder. The plot will automatically be opened to the user.