Help manual for non-coders:

Programme Name: ProteoQuest

Introduction: Welcome to ProteoQuest, an interactive tool for exploring protein sequences within the NCBI protein database. This user-friendly programme guides you through the following steps:

1. Specify the taxonomic group name.
2. Choose the protein group of interest.
3. Decide on a partial or non-partial search. After refining your search term, ProteoQuest will search the NCBI protein database, determine and plot sequence conservation, scan for motifs using the PROSITE database, and calculate protein statistics.

Software Prerequisites:

Before using ProteoQuest, ensure the following software is installed on your computer:

clustalo: Used for multiple sequence alignment.

- Install on Linux using: `sudo apt-get install clustalo`
- Install on macOS using: `brew install clustalo`
- Install on Windows using the installer from the Clustal Omega website: Clustal Omega Downloads

patmatmotifs: Used for scanning protein sequences with motifs.

- Install on Linux using the EMBOSS package: `sudo apt-get install emboss`
- Install on macOS using: `brew install emboss`
- Install on Windows by downloading and installing the EMBOSS package from: EMBOSS Downloads

pepstats: Used for calculating protein statistics.

- Install on Linux using the EMBOSS package: `sudo apt-get install emboss`
- Install on macOS using: `brew install emboss`
- Install on Windows by downloading and installing the EMBOSS package from: EMBOSS Downloads

Ensure these tools are accessible from the command line before running ProteoQuest.

Step 1: User Specifies Input

- Input: User-defined taxonomic group, protein type, and partial/non-partial search.
- Process: Gather input, refine user input, and save relevant FASTA sequence files.
- The user should follow the following steps:
  - Specify your taxonomic group: e.g., birds
  - Specify your protein name: e.g., glucose-6-phosphatase
  - Specify partial or non-partial search: NOT PARTIAL
- Your search term will be in the format of: `birds[ORGN] AND glucose-6-phosphatase[PROT] NOT PARTIAL` ; you can refine your search term if you wish, by typing the whole search term **in the correct format**.
- If the programme returns error, it is likely due to a typo, using/not using the plural form of the organism or the protein and due to no result found on NCBI. To double check, search on the NCBI protein database here.
- Output: FASTA sequence file (`{search_term}.fasta`)

Step 2: Determine and Plot Conservation

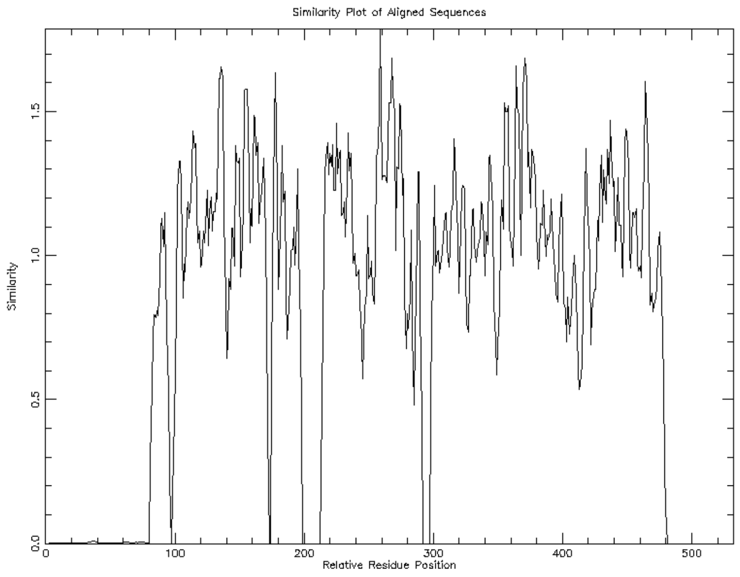
- Input: FASTA sequence file
- Process: Limit the number of sequences by setting maximum and minimum sequence lengths, determine level of conservation using clustalo, and plot conservation levels.
- The user will be told how many sequences there are, returned by the user's search term. The user will also be told the maximum and minimum sequence lengths found in those sequences so they can decide whether they'd like to reduce this.
- Output: Clustalo report file (`{search_term}.msf`), Plotcon png and pdf files. (`{search_term}.1.png & {search_term}.pdf`)

An example of a clustalo output file:

```
>KAJ7421106.1 Glucose-6-phosphatase [Willisornis vidua]
------------------------------------------------------------
--------------------MEANMNLLHDLGI----WATHWLQQRFQGSQDWFLFISY
AADLRNAFFVLFPIWFHFSEAVGIRLIWVAVIGDWLNLVFKWILFGERPY-----WWVHD
TDYYSNNSAPEIQQFP-----------------LTCETGPGSPSGHAMGAAGVYYVMVVA
LLSAAGGKKQSRTLTYWVLWTVLWMGFWAVQVCVCLSRV-FIAA--HFP---------HQ
VIAGVFSGIAVAKTFQHVRCIYHASLRRYLGITFFLFSFTLGFYLLLRVLGVDLLWTLEK
AQRWCSHPEWVHIDTTPFASLLRNLGILFGLGLALNSHMYQKSFRGKQSQQL-PFRLGCV
AASLLILHIFDAFKPPSHMQLLFYFLSFCKSAAVPLATVSLIPYCLSHLLATQDKKSA--
---------------------------------------------------
>KAJ7396366.1 Glucose-6-phosphatase [Pitangus sulphuratus]
------------------------------------------------------------
--------------------MEANMNLLHDLGI----QTTHWLQQRFQGSQDWFLFISY
```

```
AADLRNAFFVLFPIWFHFSEAVGIRLIWVAVIGDWLNLVFKWILFGERPY-----WWVHD
TDYYNNKSAPEIQQFP-----------------LTCETGPGSPSGHAMGAAGVYYIMVVA
LLSAAGGKKQSRTFRYWVLWTVLWMGFWAVQVCVCLSRV-FIAA--HFP---------HQ
VIAGVFSGMAVAKTFQHIRCIYHASLGRYTGITLFLFSFTIGFYLLLRVLGVDLLWTLEK
AQRWCSRPEWVHIDTTPFASLLRNLGILFGLGLAVNSHMYQKSCRGKQGQQL-PFRLGCV
AASLLILHIFDAFKPPSHMQLLFYALSFCKSAAVPLTTVSLIPYCLSQLLATQDKKAA--
------------------------------------------------------------
```

An example of a plotcon output file:



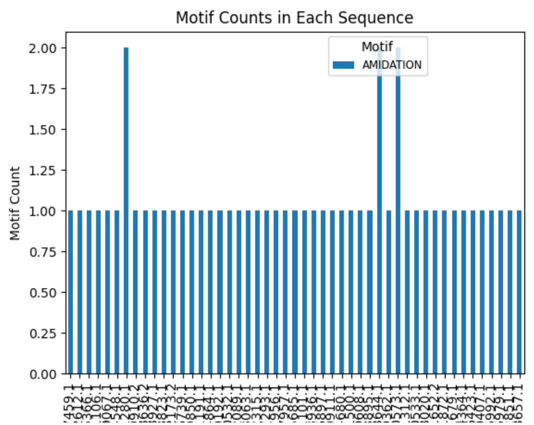
Step 3: Scan with Motifs from PROSITE

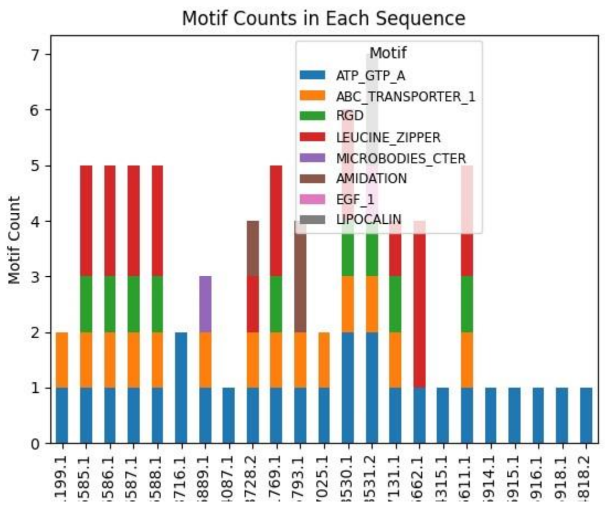
- Input: Protein sequences from the FASTA file.
- Process: Extract sequences, scan with motifs using patmatmotifs, and plot motif counts in each sequence.
    - PROSITE is a database of protein families and domains with annotated patterns, it provides conserved motifs, functional insights, and cross-references to aid in protein analysis. It is usually used for predicting protein functions and guiding experimental studies.
- Output: Report files for each sequence (/sequences_{search_term}/patmatmotifs_{search_term}/{search_term}.fasta.patmatmotifs), summary file with motif counts ({search_term}.csv), bar plot ({search_term}.png).

An example of a summary file with motif counts:

	ATP_GTP_	ABC_TRAN	RGD	LEUCINE_	MICROBO	AMIDATIO	EGF_1	LIPOCALIN
AAA9119	1	1	0	0	0	0	0	0
AAB3658	1	1	1	2	0	0	0	0
AAB3658	1	1	1	2	0	0	0	0
AAB3658	1	1	1	2	0	0	0	0
AAB3658	1	1	1	2	0	0	0	0
AAC4871	2	0	0	0	0	0	0	0

Bar plot examples:





Step 4: Calculate Protein Statistics

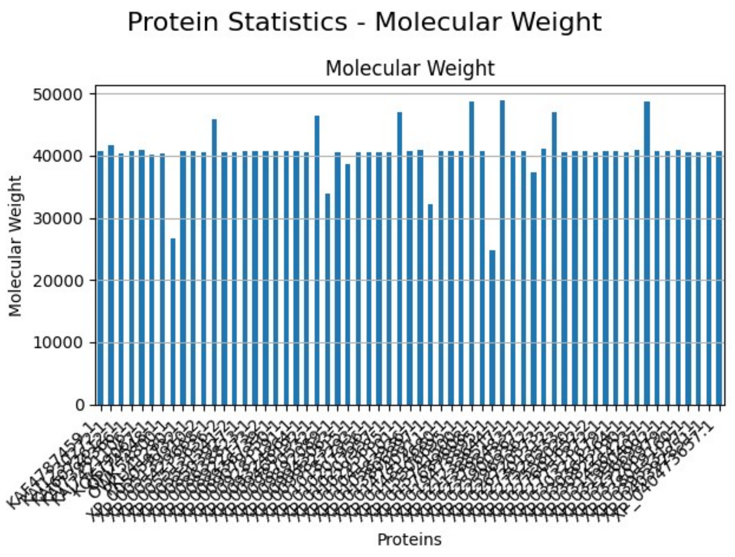
- Input: Individual protein sequences in FASTA format.

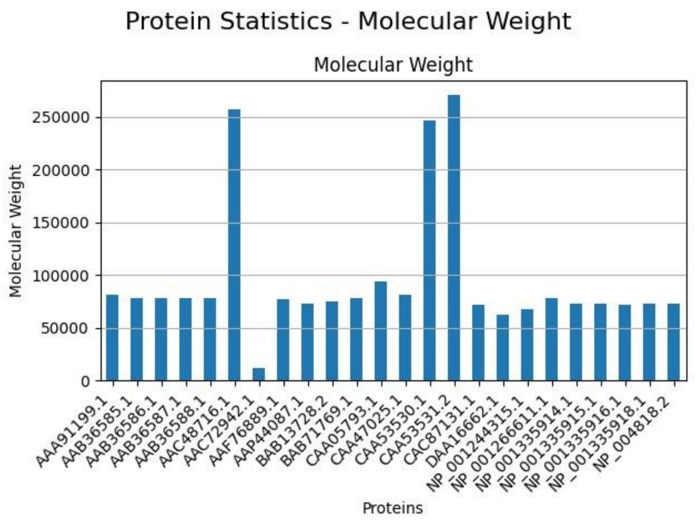
- Process: Run pepstats to calculate protein properties, extract statistics from reports into a CSV file, and plot key properties in each sequence:
    - Molecular Weight
    - Charge
    - Isoeletric Point
    - A280 Molar Extinction (Reduced)
- Output: Pepstats report for each sequence ((/sequences_{search_term}/pepstats_{search_term}/{search_term}.fasta.pepstats), summary file ({search_term}_stats.csv), bar plots for molecular weight, charge, isoelectric point, and molar extinction. ({search_term}_stats.png)

Example summary file:

	Molecular	Number of	Average Re	Charge	Isoelectric	A280 Mola	A280 Mola	A280 Extin	A280 Extin
KAF478745	40667.51	358	113.596	12.5	8.7466	107370	107745	2.64	2.649
KAI123027	41753.77	370	112.848	9.5	8.1176	107370	107870	2.572	2.583
KAI607261	40439.39	358	112.959	13.5	8.5615	100380	100880	2.482	2.495

Example bar plot:





To run the programme, execute it on the command line by typing python ProteoQuest.py

Feel free to use CTRL + C at any point to abort the programme. Enjoy your exploration with ProteoQuest!

<u>Maintenance manual (for competent Python3 code-writer):</u>

Introduction:

This programme is used for searching protein sequences within the NCBI protein database. It will start by asking the user to specify their search terms step by step:

1) The taxonomic group name
2) The protein group
3) Partial or not partial search

The user will be given an opportunity to refine the whole search term before any protein sequence is analysed. The programme will then proceed to search the NCBI protein database for the specified search terms.

Then the programme will proceed to determine and plot the level of conservation between protein sequences.

Then, it will scan the protein sequences of interest with motifs from the PROSITE database and plot any motifs found and their counts.

Finally, it will calculate protein statistics and plot them in a bar plot.

To run the programme, execute it on the commandline by typing python ProteoQuest.py

You can use CTRL + C to abort the programme at any point.

<u>STEP 1: USER SPECIFIES AN INPUT (THE PROTEIN FAMILY, THE TAXONOMIC GROUP AND PARTIAL/NON-PARTIAL PROTEIN SEARCH), USE ESEARCH AND / OR EFETCH TO OBTAIN FASTA SEQUENCE FILES.</u>

INPUT: user input collected by using input(), saved as variables:

protein_type - the protein type

taxonomic_group - the taxonomic group

partial_protein - whether the search is partial

search_term - a combination of all of the above, in the format of:

Organism[ORGN] Protein[PROT] NOT PARTIAL

PROCESS STEP 1_1: GET USER INITIAL INPUT OF TAXONOMY

1. Define a function (get_input) to gather input from the user: the taxonomic group. To do this, there should be a quality check (QC) function pre-defined:
    a. A function that checks the quality of the user input: quality_check_user_input, this is how it works:
        i. If the user input is empty, then return False (QC fails)
        ii. If the user input returns 0 result on NCBI's taxonomy database using esearch, then return False (QC fails)
        iii. If the user input returns 'FAILURE', 'WARNING' or 'ERROR' on NCBI's taxonomy search, return False. (QC fails)
        iv. Otherwise, return True (QC passes)
2. Check the user input, if it passes QC, i.e. QC returns True, then the function returns the user input as a variable; Otherwise, it tells the user to specify the taxonomic group again.
3. Use the function get_input and save its output user_input as a global variable.

PROCESS STEP 1_2 REFINE USER INPUT

1. We first need to get the scientific name(s) of the taxonomic group that the user defined, and save it/them with their index in a dictionary (so that the user can choose from it if there are more than 1). To do this:
    a. We search for the taxonomic group within the NCBI taxonomy database to get the result: e.g., 1. Aves \n\t (birds), class, birds
        i. If esearch returns 'FAILURE', 'WARNING' or 'ERROR' on NCBI, print to the screen to inform the user that their input is invalid, abort the programme and they need to start again.
    b. We search for the taxonomic group and get its scientific name by 1) search for it in the Document Summary (XML file) format, and extract it's Scientific Name by using '-element ScientificName'; we save it as a list and count how many elements there are in the list, index them. Then save it as a dictionary, where the index is the key and the scientific name is the value.
    c. We save the dictionary containing the indexes of the number of results and the scientific names (so that the user can choose which one to search for in the next step); the number of results, the scientific name and initial esearch result using the user input as global variables.
2. Once the taxonomic group is properly defined, we will do one more QC on the taxonomic group, and refine the taxonomy search term:
    a. QC: if the results dictionary is empty, then this fails, and will prompt the user to go back to STEP1_1.
    b. If there is 1 item in the dictionary, we get confirmation from the user to proceed (get_confirmation function is externally defined).
        i. Proceed with the search term if True
        ii. User allowed to redefine search term if False (going back to STEP 1_1); ask the user again if they'd like to proceed with their refined search term:
            1. Update and proceed with the refined search term if True,
            2. Exit the programme otherwise.
    c. If there is more than 1 item in the dictionary, then the user must choose 1 from the dictionary by keying its index or they're stuck in a never ending while loop. Once a choice is made, we get confirmation from the user:
        i. Update and proceed with the refined search term if True,
        ii. Exit the programme otherwise.
3. Get the search term properly defined by:
    a. Asking the user to input a specific protein they're searching for
    b. Asking the user whether they'd like to search for partially matched results
        i. If the protein is partial, we add PARTIAL at the end of the search term
        ii. If the protein is not partial we add NOT PARTIAL at the end of the search term.
        iii. If no input is given, we don't add PARTIAL or NOT PARTIAL at the end of the search term.
4. Search for the protein using the refined search term, get the number of results found, and conduct one final QC:
    a. If there are more than 1000 sequences found or 0 sequence found:
        i. Inform the user how many sequences were found and prompt the user to enter the sequence again. The user is stuck inside this while loop unless a sensible answer is given.
5. Ask if the user would like to start fresh in a new directory
    a. If True, create new directory and change directory to it
    b. If not, proceed in the current working directory.
6. Get the relevant fasta sequences and save as a fasta file.

OUTPUT:

New directory (or not) containing fasta files

STEP 2: DETERMINING AND PLOTTING THE LEVEL OF CONSERVATION BETWEEN THE PROTEIN SEQUENCES

INPUT:
Fasta sequence info from the new directory created in STEP 1

PROCESS STEP 2_1 LIMIT PROTEIN SEQUENCE NUMBERS FROM FASTA FILE

1. Get the number of sequences and inform the user
2. Limit the number of sequences used for the conservation analysis by giving the user an option to define a max and min sequence length.
    a. Error trap: if the user defines an inappropriate min and max sequence length, they are prompted to try again; if they didn't enter anything, they will exit the programme.
    b. Once a min and max sequence length has been defined, inform the user of the trimmed sequence length and ask the user for confirmation
        i. If the user would like to retrim, they have the option to do so again
        ii. Otherwise, we proceed with the original min and max lengths of the sequences

PROCESS STEP 2_2 PLOTTING THE LEVEL OF CONSERVATION BETWEEN THE PROTEIN SEQUENCES

1. Determine the level of conservation between the protein sequences, and establish the degree of similarity within the sequence set chosen using clustalo. Save the clustalo file output.
2. Use plotcon to plot the sequences, save it as an output file and send it to the screen.

OUTPUT:

- Clustalo report file
- Plotcon file

STEP 3: SCAN THE PROTEIN SEQUENCES WITH MOTIFS FROM THE PROSITE DATABASE TO DETERMINE KNOWN MOTIFS ASSOCIATED WITH THE SEQUENCES SCANNED, USING PATMATMOTIFS

INPUT:
The protein sequences extracted from the fasta sequence file containing multiple sequences.

PROCESS STEP 3_1 EXTRACTING SEQUENCES FROM THE FASTA FILE TO A FOLDER CONTAINING ALL SEQUENCES AS FASTA FILES

1. Extract the sequences from the fasta sequence file to a folder containing all sequence as fasta files and change directory to that folder.
2. Scan through each sequence in the folder by a for loop and write a new file every time a new sequence is read. The new file will contain a header starting with '>', followed by the actual sequence.

PROCESS STEP 3_2 SCANNING THE PROTEIN SEQUENCE WITH MOTIFS FROM THE PROSITE DATABASE

1. Make a new folder to collect results from this step and change directory to it.
2. Using patmatmotifs to search for the motif within the protein sequence(s) of interest and print out the names of the protein sequences where the motifs are found, and if found print out the total number of matches for the motif within that sequence. The user will be asked if they'd like to include simple post-translational modifications or not in patmatmotifs.
    a. Save all report files for patmatmotifs, there should be one report per sequence.
    b. Move all report files into the new folder created and change directory to it.
    c. We open an empty dictionary to collect our data as a nested dictionary in the format of {sequence_name : {motif_name : count}}
    d. Save a summary file containing the nested dictionary as a pandas dataframe.
    e. Plot a bar plot to visualise the motif counts in each sequence.

OUTPUT:

- Report file for each sequence
- A summary file containing the motif counts in each sequence
- A bar plot to visualise the motif counts in each sequence

STEP 4: Using pepstats to calculate statistics of protein properties
INPUT: individual protein sequence in fasta format

PROCESS STEP 4_1 RUN PEPSTATS TO CALCULATE THE STATISTICS OF THE PROTEIN PROPERTIES

1. Make a new folder to collect pepstats reports

2. Run pepstats for each protein sequence
3. Move all pepstats report into the new folder

PROCESS STEP 4_2 EXTRACT THE STATISTICS FROM THE REPORTS INTO A CSV FILE

1. We specify the regex pattern for each protein statistics property we'd like to extract.
2. We open an empty dictionary to collect our data as a nested dictionary in the format of {sequence_name : {protein_stat_name : stat_data}}
3. Save a summary file containing the nested dictionary as a pandas dataframe.

PROCESS STEP 4_3 PLOT THE STATISTICS IN BAR PLOTS

Plot a bar plot to visualise key properties in each sequence:

1. Molecular Weight
2. Charge
3. Isoeletric Point
4. A280 Molar Extinction (Reduced)

OUTPUT:

- Pepstats report for each sequence
- A summary file containing the motif counts in each sequence
- A bar plot to visualise the motif counts in each sequence