ICA2

GitHub Link: https://github.com/B248888/ICA2.git

Encryption key: 020175

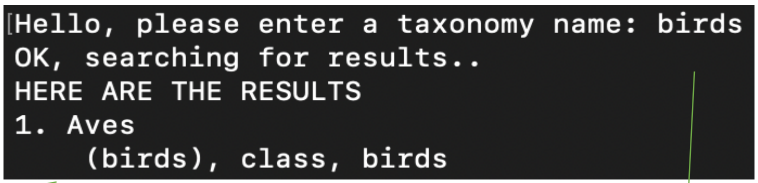
HELP MANUAL

For ordinary user:

The aim of this manual is to navigate you through the program and show possible outcomes you might get depending on your request. This program was designed to identify a family of protein sequences based on taxonomy group and the protein of your choice. The program will plot the level of conservation between protein sequences graph. It also will perform scanning of protein sequences from the PROSITE database and produce a CSV file and plot that will show counts of motifs for each sequence. Finally, the program will perform two additional analyses from EMBROSS. It will produce a grainier file that will contain information regarding the predicted secondary structure of protein sequences. Pepstats analysis will be done and certain parameters will be taken from pepstats files. You will be able to see the results in a separate CSV file. Below you will find more specific instructions with examples. Please run Begin3.py to begin the program.

1) Taxonomy group:

The program will firstly ask you, for taxonomy group name, to later get the tax ID for further analysis. Please note, you can exit the program, if you don't want to proceed with chosen taxonomy group.



```
[Hello, please enter a taxonomy name: birds
OK, searching for results..
HERE ARE THE RESULTS
1. Aves
    (birds), class, birds
```

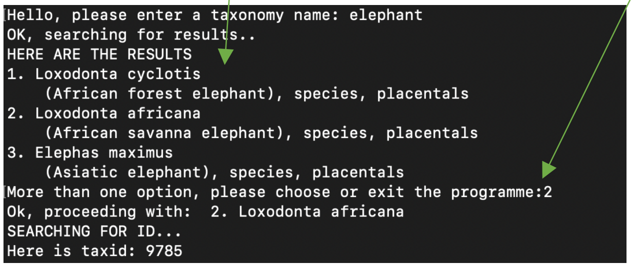
The matching results from the NCBI database will be displayed here.

Here you will put the name.

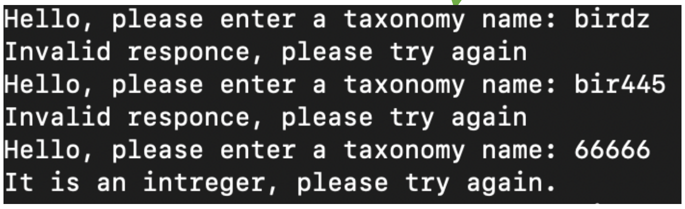
Please note in some cases there will be more then one option.

Please don't leave space between integer and :

Please try to put a valid response, the program will not tolerate, invalid responses, such as just numbers.



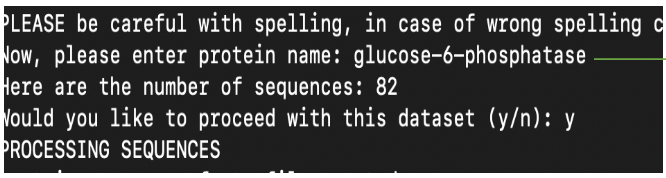
```
Hello, please enter a taxonomy name: elephant
OK, searching for results..
HERE ARE THE RESULTS
1. Loxodonta cyclotis
    (African forest elephant), species, placentals
2. Loxodonta africana
    (African savanna elephant), species, placentals
3. Elephas maximus
    (Asiatic elephant), species, placentals
More than one option, please choose or exit the programme:2
Ok, proceeding with:  2. Loxodonta africana
SEARCHING FOR ID...
Here is taxid: 9785
```



```
Hello, please enter a taxonomy name: birdz
Invalid responce, please try again
Hello, please enter a taxonomy name: bir445
Invalid responce, please try again
Hello, please enter a taxonomy name: 66666
It is an intreger, please try again.
```

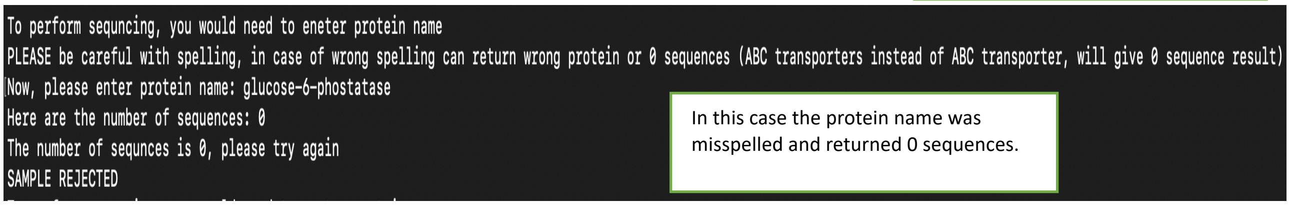
2) Protein Name:

Once you get taxonomy ID, the program will ask you for the protein name, please note that misspelling protein name, may return result of 0 or return a wrong protein name, with wrong amount of sequences. Please make sure you put the correct name. If you misspelled the name, please make sure to choose n, when ask to proceed. You will be able to change your request.



```
PLEASE be careful with spelling, in case of wrong spelling c
Now, please enter protein name: glucose-6-phosphatase
Here are the number of sequences: 82
Would you like to proceed with this dataset (y/n): y
PROCESSING SEQUENCES
```

In this example, the name of protein spelled correctly and the correct amount of sequence was displayed. In this case user chose to proceed with dataset by choosing: y.



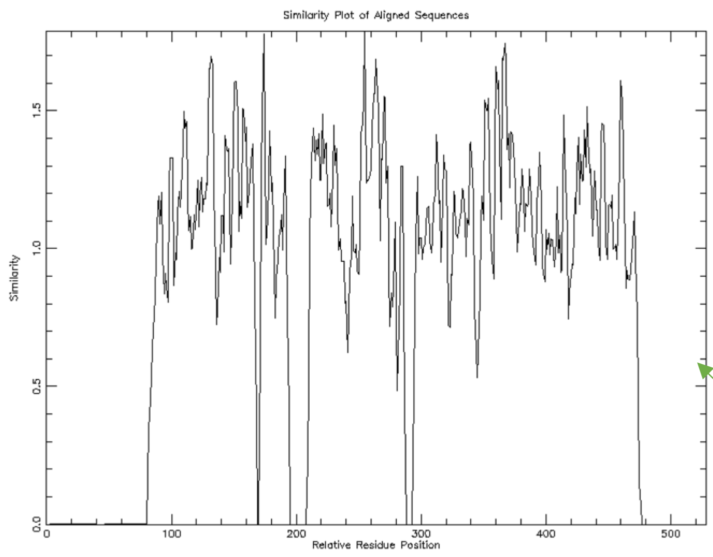
```
To perform sequncing, you would need to eneter protein name
PLEASE be careful with spelling, in case of wrong spelling can return wrong protein or 0 sequences (ABC transporters instead of ABC transporter, will give 0 sequence result)
Now, please enter protein name: glucose-6-phostatase
Here are the number of sequences: 0
The number of sequnces is 0, please try again
SAMPLE REJECTED
```

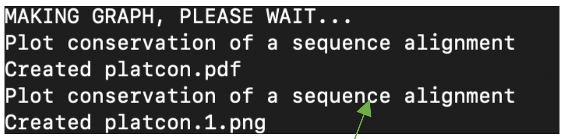
In this case the protein name was misspelled and returned 0 sequences.

Also please note that program has a set limit, it would not process more then 1000 sequence, automatically rejecting sample. All results will be stored in protein_sequences.fasta file.

3) Plotting the conservation between protein sequences:

You don't need to do anything to plot graph, it will automatically plot and will be shown on your screen. PDF (platcon.pdf) and PNG (platcon.1.png) versions will be saved in the directory, so you can download if needed. The graph will look like this:





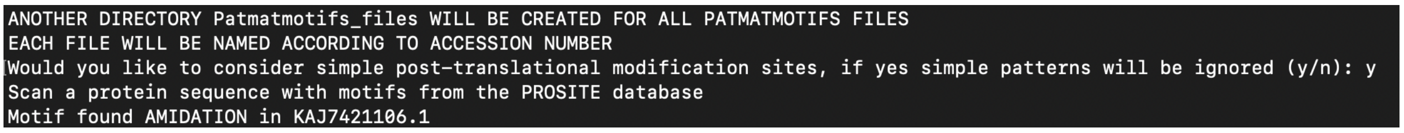
```
MAKING GRAPH, PLEASE WAIT...
Plot conservation of a sequence alignment
Created platcon.pdf
Plot conservation of a sequence alignment
Created platcon.1.png
```

The program will display this message.

Please note plot can take up to 30s to get displayed on a screen

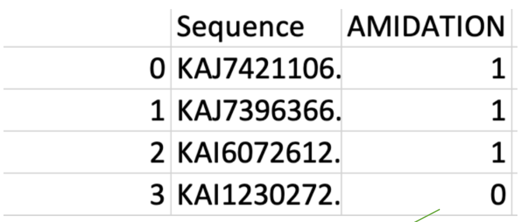
4) Scanning protein sequences with motifs PROSITE:

You would need to specify whether you would like to proceed with simple post-transitional modification sites (y or n response). The analysis will be done, according to your response. Another directory Patmatmotifs_files will be created automatically, it will have all the patmatmotifs files for each sequence. The files will be named according to accession number. Additionally, a file with all the information from each patamtmotif file will be created : patmotifs_all.

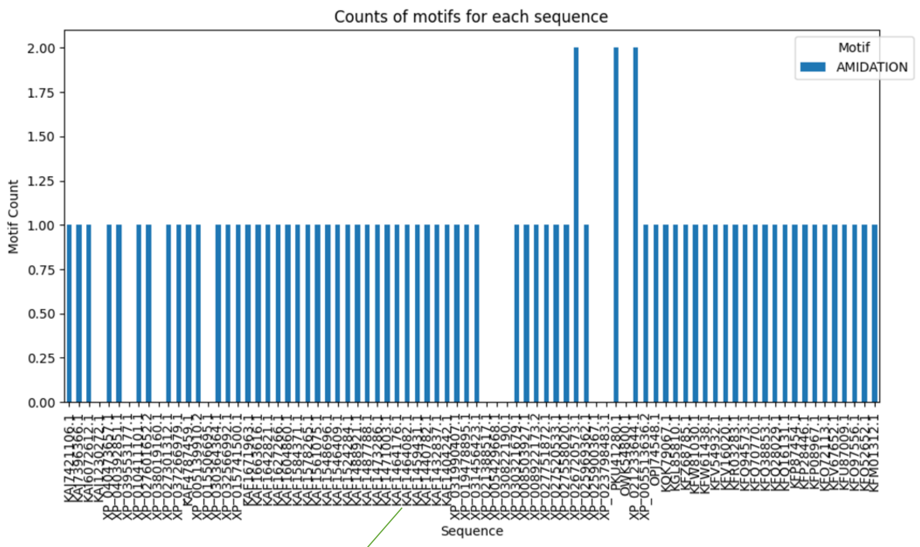


```
ANOTHER DIRECTORY Patmatmotifs_files WILL BE CREATED FOR ALL PATMATMOTIFS FILES
EACH FILE WILL BE NAMED ACCORDING TO ACCESSION NUMBER
Would you like to consider simple post-translational modification sites, if yes simple patterns will be ignored (y/n): y
Scan a protein sequence with motifs from the PROSITE database
Motif found AMIDATION in KAJ7421106.1
```

You will be able to see which motif was found in certain sequence. Additionally, csv file (motif_data.csv) and plot (bar_plot.png) will be created for motif counts for each sequence, the example below is for glucose-6-phosphatase in birds:



	Sequence	AMIDATION
0	KAJ7421106.	1
1	KAJ7396366.	1
2	KAI6072612.	1
3	KAI1230272.	0



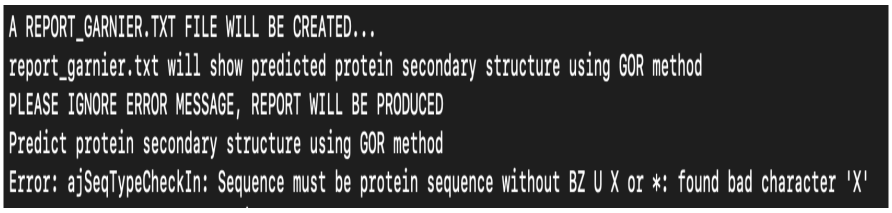
In csv, you will be able to see all the sequences. In the other column, you will see the name of the motif and count of motif in sequence (in this particular example, there is only one motif. With other example, there can be more motifs found in one sequence and some motifs can be present more then twice in one sequence)

Depending on your query, plot can look different too. If there are more than one motif found, they will be displayed with different colors.

5) Garnier analysis:

Is one of the analyses that is done automatically by program, to produce a file (report_garnier) to demonstrate information about predicted secondary structure of protein sequences. The file will be automatically saved in the directory. It might help you with predictions if you analyzing secondary structure of the protein.

Here is the message you will see in terminal. Please ignore error message.



```
A REPORT_GARNIER.TXT FILE WILL BE CREATED...
report_garnier.txt will show predicted protein secondary structure using GOR method
PLEASE IGNORE ERROR MESSAGE, REPORT WILL BE PRODUCED
Predict protein secondary structure using GOR method
Error: ajSeqTypeCheckIn: Sequence must be protein sequence without BZ U X or *: found bad character 'X'
```

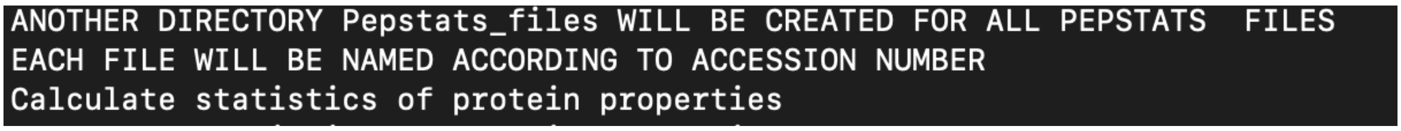
```
#=========================================
#
# Sequence: KAJ7421106.1     from: 1   to: 358
# HitCount: 109
#
# DCH = 0, DCS = 0
#
#  Please cite:
#  Garnier, Osguthorpe and Robson (1978) J. Mol. Biol. 120:97-120
#
#
#=========================================

          .   10    .   20    .   30    .   40    .   50
      MEANMNLLHDLGIWATHWLQQRFQGSQDWFLFISYAADLRNAFFVLFPIW
helix HHHHHHHHHHH  HH                    HHHHHHH       H
sheet                              EEEE         EEEEE   
turns             T   T  TTTTTT  TT                   T 
 coil            C   C CC      CC      CC            C  
          .   60    .   70    .   80    .   90    .  100
```

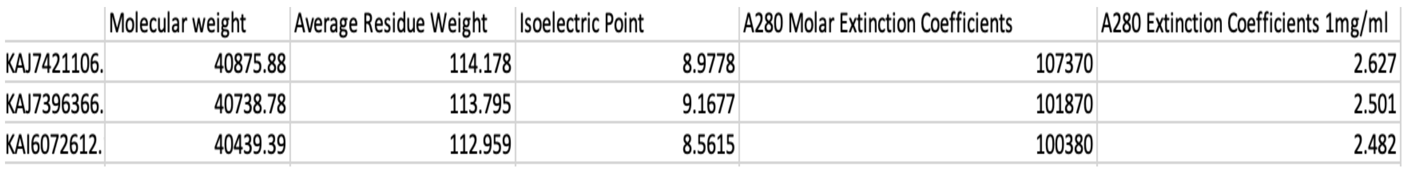
Here is how the report would look like for each sequence analyzed.

6) Pepstats analysis:

This analysis will be automatically done by the program. The csv file (pepstats_data.csv) will be produced. You can download the file and see general statistics of protein properties, such as: molecular weight, average residue weight, isoelectric point, A280 Molar Extinction Coefficients (reduced) and A280 Extinction Coefficients 1mg/ml (reduced). All pepstats files will be stored in Pepstats_files directory and a file with all of the information from pepstat analysis for each sequence will be also stored in current directory (ICA2_programme). Below you can see, the message from terminal and example of csv files (for glucose-6-phosphatase in birds).



```
ANOTHER DIRECTORY Pepstats_files WILL BE CREATED FOR ALL PEPSTATS  FILES
EACH FILE WILL BE NAMED ACCORDING TO ACCESSION NUMBER
Calculate statistics of protein properties
```



	Molecular weight	Average Residue Weight	Isoelectric Point	A280 Molar Extinction Coefficients	A280 Extinction Coefficients 1mg/ml
KAJ7421106.	40875.88	114.178	8.9778	107370	2.627
KAJ7396366.	40738.78	113.795	9.1677	101870	2.501
KAI6072612.	40439.39	112.959	8.5615	100380	2.482

After this stage the program will automatically end, all the files will be saved in ICA2_directory.

For user familiar with python3:

This part of the manual designed for users that have experience with python3. In this section, you will find an overview of the structure of the script. The script contains 9 different functions, that this manual will go through. The script uses multiple conditions (if, elif, else), it also uses lot's of for loops, and regular expressions to retrieve needed information from certain files.

1) Beginning:

Several modulus are imported first at the beginning of the program:

- Import os
- Import shutil
- Import re
- Import time
- Import pandas as pd
- Import matplotlib.pyplot as plt

Now the new directory will be produced, the script first checks if directory already exists, if os.path.exists('ICA2_programme'). If it does it removes it and creates a new one by using shutil.rmtree('ICA2_programme'). The new directory is created: os.mkdir('ICA2_programme') and then changing to current directory by using os.chdir('ICA2_progarmme'). After that script enters first function. (time module was imported, so whenever message is displayed user have some time to read it before script proceeds).

2) get_Taxonome():

This first function was designed to get the taxonomy group name from the user.

Function begins with infinite While True loop. So not proceeding without user input which will be stored in taxonome variable.

A few simple error traps are set up, so if user puts invalid response or leaves a blank space, the loop does not proceed:

1) if taxonome.isdidgit(), the command (if all values in string are digits), won't take it as input and proceed further.
2) if not taxonome.strip(), the command makes sure if user provides blank spaces or just accidently press enter with no input, condition becomes true, if block will executed, stopping user from proceeding further.

If user input is fine, proceeds with responce= subprocess.getoutput(responce= subprocess.getoutput('esearch -db taxonomy -query ' + taxoname + ' | efetch')

Another set of error traps set up for response=

1) if not responce that will check if the string returns empty (as whitespace), meaning nothing was processed, hence the response was invalid.
2) if FAILURE/ERROR/WARNING in responce, a few error traps to make sure if responce returns any of these messages, user will need to repeat query or put another input
3) else: in case when the query processed successfully and there is an output such as: Avens for birds query, user can proceed to next question.

3)get_Taxid(result):

This function, uses result from other one to see whether there is 1 or more options to choose from for user. So, splitting result into list:
result_list= result_list.splitlines() and then converts splitted lines into integers, goes through it and divides by 2, to get the result number:
result_number = int(len(result_list) / 2)

Enters while True loop

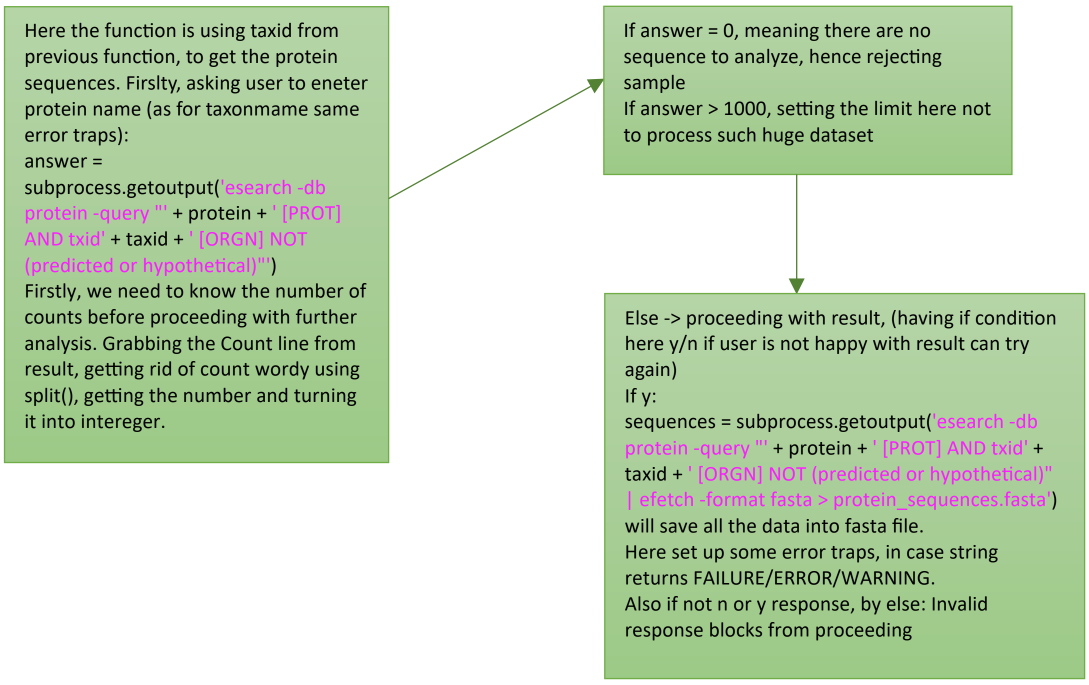
If result_number =1 user can proceed with the result of exit program (if 'y' will grab [0] from result_list which will be the name of the group

If result_number >1:
Making sure it's within the list range:
question2.isdigit() and 1 <= int(question2) <= result_number and then getting the correct line from list based on provided response:
tax_chosen= (int(question2) - 1) * 2
tax_result= result_list[tax_chosen]

3) **get_Taxid2(result2):**

Based on the result from previous function, proceeds the chosen line with taxonomy group name getting the tax ID:
taxid = subprocess.getoutput('esearch -db taxonomy -query "' + result2 + '" | efetch -format taxid')

4) **get_Protein(taxid):**



Here the function is using taxid from previous function, to get the protein sequences. Firslty, asking user to eneter protein name (as for taxonmame same error traps):
answer = subprocess.getoutput('esearch -db protein -query "' + protein + ' [PROT] AND txid' + taxid + ' [ORGN] NOT (predicted or hypothetical)"')
Firstly, we need to know the number of counts before proceeding with further analysis. Grabbing the Count line from result, getting rid of count wordy using split(), getting the number and turning it into intereger.

If answer = 0, meaning there are no sequence to analyze, hence rejecting sample
If answer > 1000, setting the limit here not to process such huge dataset

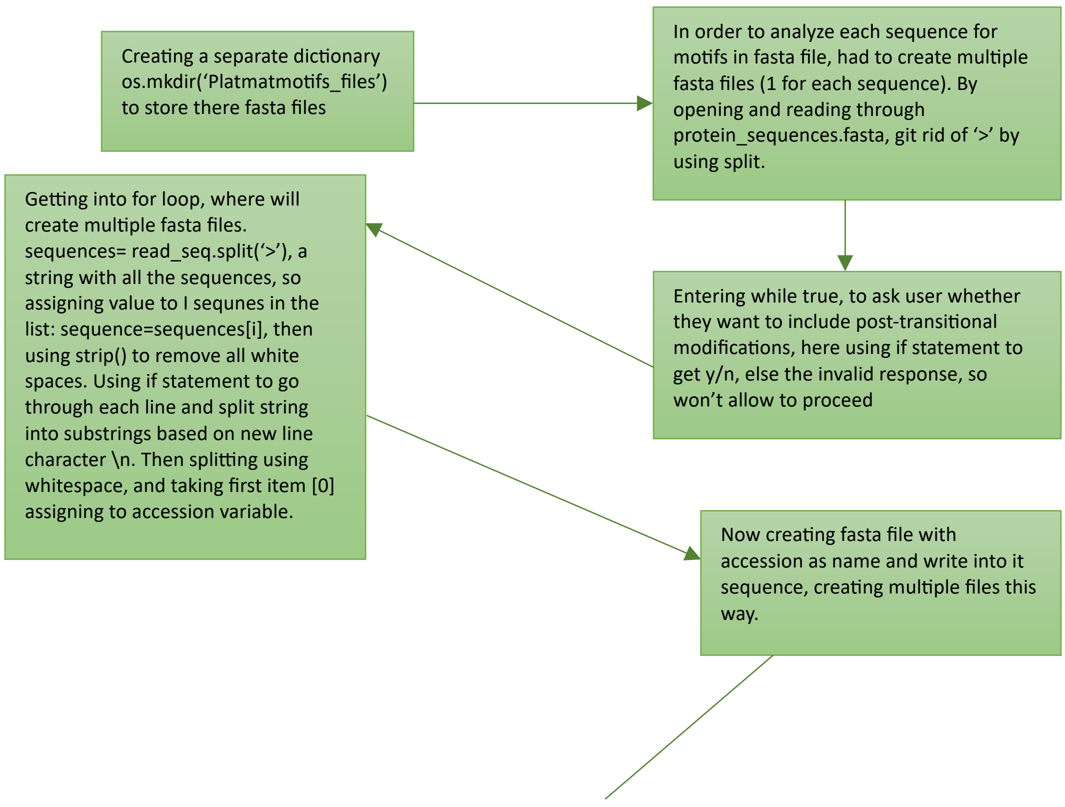
Else -> proceeding with result, (having if condition here y/n if user is not happy with result can try again)
If y:
sequences = subprocess.getoutput('esearch -db protein -query "' + protein + ' [PROT] AND txid' + taxid + ' [ORGN] NOT (predicted or hypothetical)" | efetch -format fasta > protein_sequences.fasta')
will save all the data into fasta file.
Here set up some error traps, in case string returns FAILURE/ERROR/WARNING.
Also if not n or y response, by else: Invalid response blocks from proceeding

5) **get_Plot():**

This function just processes fasta file to get the plot by using clustalo and platcon (clustalo will create alignmnet.msf (making sequence alignment) file that is used by platcon later to get the plot (conservation plot, as png file). This is achieved by using subprocess.call (running it in shell, shell=True). Additionally getting pdf file (also using platcon). Using eog to display the plot on the screen.

6) **get_Motifs():**

This function is designed to scan protein sequences from fastsa file with motifs from PROSITE database using platmatmotifs:



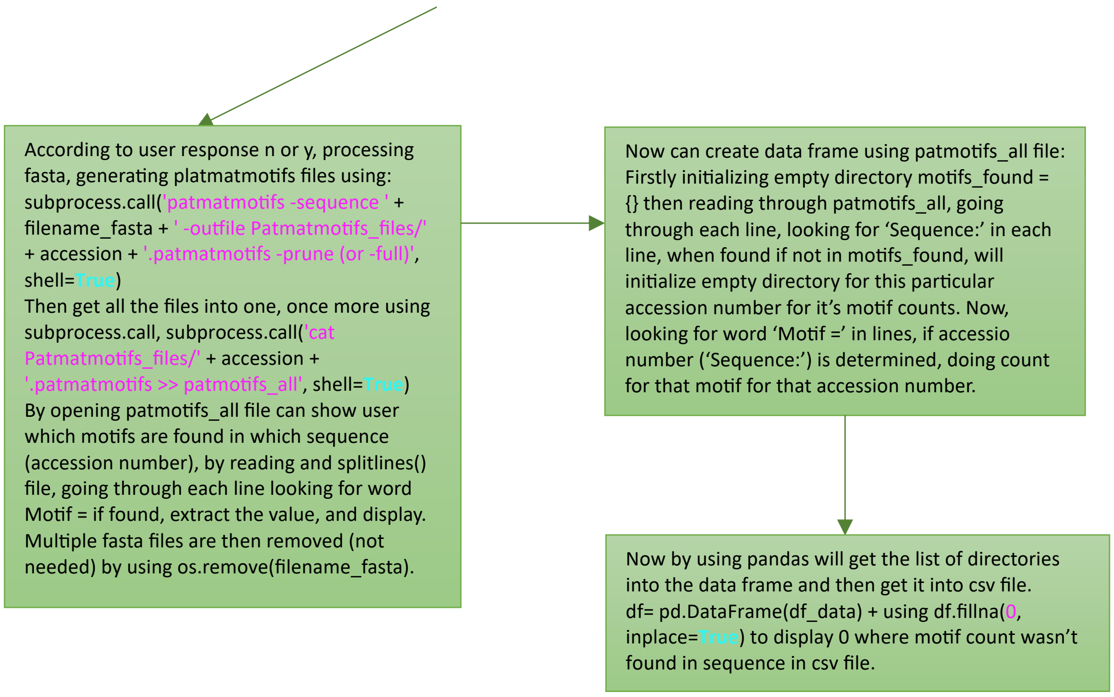
Creating a separate dictionary os.mkdir('Platmatmotifs_files') to store there fasta files

In order to analyze each sequence for motifs in fasta file, had to create multiple fasta files (1 for each sequence). By opening and reading through protein_sequences.fasta, git rid of '>' by using split.

Entering while true, to ask user whether they want to include post-transitional modifications, here using if statement to get y/n, else the invalid response, so won't allow to proceed

Getting into for loop, where will create multiple fasta files. sequences= read_seq.split('>'), a string with all the sequences, so assigning value to I sequnes in the list: sequence=sequences[i], then using strip() to remove all white spaces. Using if statement to go through each line and split string into substrings based on new line character \n. Then splitting using whitespace, and taking first item [0] assigning to accession variable.

Now creating fasta file with accession as name and write into it sequence, creating multiple files this way.



According to user response n or y, processing fasta, generating platmatmotifs files using: subprocess.call('patmatmotifs -sequence ' + filename_fasta + ' -outfile Patmatmotifs_files/' + accession + '.patmatmotifs -prune (or -full)', shell=True)
Then get all the files into one, once more using subprocess.call, subprocess.call('cat Patmatmotifs_files/' + accession + '.patmatmotifs >> patmotifs_all', shell=True)
By opening patmotifs_all file can show user which motifs are found in which sequence (accession number), by reading and splitlines() file, going through each line looking for word Motif = if found, extract the value, and display. Multiple fasta files are then removed (not needed) by using os.remove(filename_fasta).

Now can create data frame using patmotifs_all file: Firstly initializing empty directory motifs_found = {} then reading through patmotifs_all, going through each line, looking for 'Sequence:' in each line, when found if not in motifs_found, will initialize empty directory for this particular accession number for it's motif counts. Now, looking for word 'Motif =' in lines, if accessio number ('Sequence:') is determined, doing count for that motif for that accession number.

Now by using pandas will get the list of directories into the data frame and then get it into csv file. df= pd.DataFrame(df_data) + using df.fillna(0, inplace=True) to display 0 where motif count wasn't found in sequence in csv file.

7) get_Barplot():

This function plots a plot by reading csv file using matplotlib.pyplot. It makes sure that 0 counts are not included into the plot. Then sets parameters to plot bar, with certain figure sizes. Assigns x-axis to Sequences (accession numbers) and y-axis to motif counts. Adds legend to the plot, in this case legend will represent motifs, legend will be placed in right corner of bounding box (e.g Motif ADMIRATION), bbox_to_anchor, will specify the location of the bounding box in the plot (must be nit very small, but not large). Depending on the amount of motifs the plot will change accordingly.

8) get_Garnier():

This function will perform garnier analysis and generate file report.garnier for user to look at the results, using subprocess.call:
subprocess.call('garnier -sequence protein_sequences.fasta -outfile report.garnier', shell= True)

9) get_analysis():

This is the last function in this script. It will perform pepstats analysis. This function is very similar to the get_Motif(), and also generates new directory Pepstats_files and generate multiple fasta files for pepstats analysis. Will produce multiple pepstats files and store them in directory and one pepstat_all file with all the information.

The difference in this function, is to how it gets information form the file. This time, just looking in the line for the word and getting the position of the value, wouldn't work as pepstats files, have certain structure, so in this case the best approach is to use regular expressions, to make sure to catch needed information. In this case 5 parameters from pepstat_all were taken: Molecular weight, Residues, Average Residue Weight, Isoelectric point and Charge. As with previous function an empty directory was created. Pepstats_all file is open and read through, this time to get he accession number can use line.startswirh() as each accession number will be in the line that starts with PEPSTATS, so easy to get it's position and get the accession number for each sequence (initializes empty directory). Then a while loop is created to look through lines until it reaches 'Improbabaility of expression in inclusion bodies' (interested in parameters listed before that). Now using re.match to match the lines with parameters needed using \S+ to get nonwhitespace characters and \d+ to get digit characters. To do re.match for every single line in the file assigning key_value_match with re.match(r'(.+?)\s*=\s*(\S+)', line). To look for character in line(? Stops it from capturing when it clashes the following part of pattern), to match = sign and get nonwhitespace character (value). Going like this through each line and then put values in a directory. Finally, creating a data frame from list of directories using pandas as in step 3.

The program will stop after this, printing 'THANK YOU, BYE' statement to user.