Welcome To the Help Manual for "*Script for ICA2: A Python3 workflow*"

INDEX (click to jump to section):

2. USER MANUAL FOR "ORDINARY USER" (NON-CODER)

Welcome to the help manual for this Programme.

This section of the help manual is directed towards an ordinary user.
(Refer the maintenance manual if you are a competent python3 code writer)
The User Manual will focus mainly on:

- Scope of the programme (functionality)
- What the inputs and outputs are (examples will be provided of a test set),
- How to run the programme (inputs required),
- How *not* to run the programme (invalid or wrong inputs).

Description of the programme:

The code has 4 main sections:

1. Input Section
2. 1st Processing Section
3. 2nd Processing Section
4. Wildcard of biological importance

What you can do with this programme:

- **Input** a **taxonomic group** and a **protein** of your choice to query with.
- Generate a **.fasta** file which will contain all the ***Not Partial*** sequences in the protein database of NCBI for a certain taxonomic group and protein which are input by the user.
- Determine and plot the **level of conservation** of the protein sequences in the .fasta file after performing multiple sequence alignment using **Clustal Omega** (clustalo).
- Search for **motifs** in the generated protein sequences that are present in the **PROSITE database**, list them and count them using **patmatmotif** from EMBOSS suite.
- **Back-translate** the protein sequences to get the nucleic acid sequences that the proteins in our sequences most likely came from using **backtranseq** from EMBOSS suite.

This document will give you an idea of what inputs and outputs of each section of the code looks like and will try to aid the user in generating the results they want, in the way they want, with the help of a test set.

In this test set the input are as follows:
Taxonomic group: **aves**
Protein: **glucose 6 phosphatase**

Let's look at each section of the code sequentially.

A list of the inputs and outputs can be (optionally) viewed when the program is initiated. Choose y or yes to view them OR n or A list of the inputs and outputs can be (optionally) viewed when the program is initiated. Choose **y** or **yes** to view them OR **n** or **no** to not view them.

Input Section:

How the code works sequentially (what you should do):

- The user can input a taxonomic group of their choice and a protein of their choice.

```
Welcome to the input section of the code where we generate the dataset that will be used in the processing stages that follow

We require the protein to be searched for and the taxonomic group to be searched in as the 2 inputs

What is the taxonomic group that you want to query in
aves
```

- A bit of information is displayed about the taxonomic group of closest match in the NCBI taxonomy database to the input taxonomic group.

```
 This is the taxonomic group you have chosen and a few details about it.

TaxId   ScientificName  GenbankCommonName       Division

8782    Aves    birds   Vertebrates
```

- The user is given the choice to proceed or terminate the program.
  Input **y** or **yes** to proceed.

```
WARNING! Choosing n=no for the next choice will terminate the program

Shall we proceed to input the protein (y/n)?    (y=yes and n=no)
y

You have chosen to proceed
```

- The user is prompted to input the protein name. Input protein name is displayed.

```
What is the protein that you want to query with
glucose 6 phosphatase

The protein that you have chosen to query with is  glucose_6_phosphatase
```

- The number of sequences in the dataset is determined. If number of sequences are less than two, a new dataset has to be generated. In our test set, we had 61.

```
Before we proceed to view the list of species that will be obtained for the inputs we first need to check what the total number of sequences obtained is

This is crucial because if we exceed 1000 sequences it will be very time consuming. So the maximum allowable sequences limit is set to 1000

We also cannot do alignment with just 1 or 0 sequences, so in these scenarios we need to generate new dataset

The number of sequences obtained was:  61
```

- The user is given the option to proceed and view the list of species in the 61 sequences of our test set OR terminate the program. (**y** or **yes** to proceed)

- For the test set, this is the ordered list of species in decreasing order of frequency which our input dataset protein sequences come from. (list not shown fully)

```
This is the ordered list of species which contain the input protein sorted in descending order of their frequency

     2 Lonchura striata domestica
     2 Colius striatus
     2 Calypte anna
     1 Willisornis vidua
     1 Turdus rufiventris
     1 Tauraco erythrolophus
     1 Sturnus vulgaris
     1 Strigops habroptila
     1 Pygoscelis adeliae
     1 Pseudopodoces humilis
     1 Pitangus sulphuratus
     1 Pipra filicauda
     1 Phasianus colchicus
```

- A .fasta file called **'glucose_6_phosphatase_aves.txt'** has been generated which contains all our sequences.
- The user is given a choice to continue with the current one or to generate a new dataset which will take us back to the start of the input stage to generate a new dataset. Choose **y** or **yes** to proceed with our test set.

```
Shall we proceed to the processing stages with the current dataset (y/n)?       (y=yes and n=no)
yes
You have chosen to proceed with the current dataset
```

What ***Not*** to do in the input section of the code:

- For input **taxon group**, anything other than alphabets (i.e. letters) is not accepted as a valid input. It is not case sensitive. Spaces are not an issue.
- For input **protein name**, combinations of letters and numbers is permitted as seen in our test set. Input containing only digits is not permitted. Spaces are not an issue.
- For all **choices** that are given to the user, all options are not case sensitive but inputs other than **y, yes, n or no** are invalid.
- Dataset can be generated as many times as needed but can proceed to the processing stages with only one of the datasets.

1st Processing Section *(clustalo & plotcon):*

How the code works sequentially (what you should do):

- Some information is displayed regarding what is going to be done to the user. The software that will be used is also described. Alignment **begins** using **clustalo**.

```
### PROCESSING SECTION OF THE CODE: Level of conservation between the protein sequences ###

Welcome to the Processing stages where we will have some interesting outputs and inferences

We will first determine and plot the level of conservation of the dataset protein sequences
We will be using Clustal Omega (aka clustalo) to both cluster and perform multiple sequence alignment on the dataset protein sequences

Here we go ... Initiating alignment ...
```

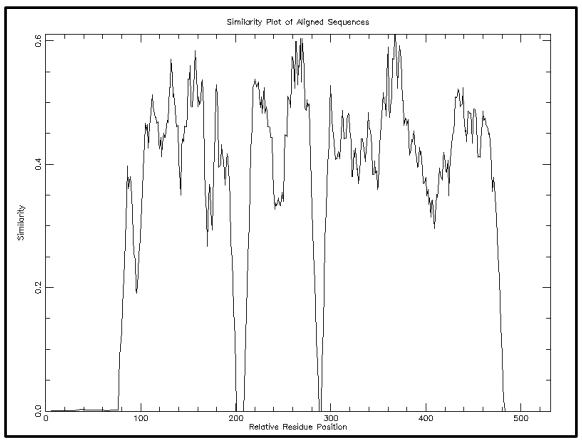
- Alignment is completed. A single **.aln** file containing aligned sequences is created which will be used in the steps that follow.

```
Progressive alignment progress: 100 % (60 out of 60)
Progressive alignment progress done. CPU time: 52.39u 0.15s 00:00:52.54 Elapsed: 00:00:02
Alignment written to glucose_6_phosphatase_aves_alignment_fasta.aln

 The output file is 'glucose_6_phosphatase_aves_alignment_fasta.aln'
```

- Now the level of conservation will be plotted using **plotcon**. It will generate a **.png** file of the similarity plot of aligned sequences. Conservation indicates that a sequence has been maintained by natural selection.

- The user is given the **choice** to plot the level of conservation and to choose a **winsize** (window size) of their liking for the plot. A large window (e.g. 100) gives a nice, smooth curve, and very low 'similarity score' units, whereas a small window (e.g. 4) gives a very spikey, noisy plot with 'similarity score' units of a round 1.00. Choosing a value of **10** for our test set gives us the image below.



Sections with higher y value on the plot indicate higher conservation of proteins for our test set dataset protein sequences at those specific residue positions of the aligned sequences.

- **<u>V. IMP</u>:** Make sure to close the plot that pops up otherwise the programme wont proceed.

<u>What ***Not*** to do in the 1st Processing section of the code:</u>

- For all **choices** that are given to the user, all options are not case sensitive but inputs other than **y, yes, n or no** are invalid.
- Unless you **do not** want to plot the level of conservation do not choose the **n** or **no** after the alignment step. It will skip to the next processing stage without giving any images to you.
- For **winsize** input, **only integers** are accepted. Any other input is invalid and the programme will not proceed till it gets an integer input. Do not enter a value of **0**.
- The plot displayed has to be closed for the programme to proceed. Do not forget to do this.

2nd Processing Section *(patmatmotif):*

How the code works sequentially (what you should do):

- Using the fasta file that was generated in the input section, an EMBOSS program known as patmatmotif is used to search if motifs in the PROSITE database are found in our dataset protein sequences.

```
The fasta file we will be using to scan is  glucose_6_phosphatase_aves.fasta

Fasta File copied successfully to be used for searching
```

- User is asked if they want to proceed with searching for motifs. Choose **y** or **yes** to proceed OR **n** or **no** to terminate the program. Choose **y** or **yes**.

- Patmatmotf is run on all of our 61 sequences and the file containing all the results is made.

```
 Time to start the search ... Hold on to your seat ... This is fast and furious ...

Scan a protein sequence with motifs from the PROSITE database
Scan a protein sequence with motifs from the PROSITE database
Scan a protein sequence with motifs from the PROSITE database
```

```
All patmatmotif results are combined into glucose_6_phosphatase_aves_combined_patmatmotifs_results.txt
```

- The motifs found in our sequences is counted and an ordered list of the motifs in decreasing order of frequency is output to the user.

```
These are the motifs that were present in our database:

    52  AMIDATION
```

What ***Not*** to do in the 2nd Processing section of the code:

- For all **choices** that are given to the user, all options are not case sensitive but inputs other than **y, yes, n or no** are invalid. Make sure to choose **y** or **yes** to search for motifs.

Wildcard *(backtranseq):*

How the code works sequentially (what you should do):

- Takes the generated .fasta file and runs backtranseq on it. It reads the sequences and writes the nucleic acid sequence it is most likely to have come from. No inputs/chances of error for this as there is no user input involved.

```
 When we run backtranseq on our dataset we get nucleic acid sequences that the proteins in our sequences most likely came from

Back-translate a protein sequence to a nucleotide sequence

 Process Completed

 The file containing the result is stored in glucose_6_phosphatase_aves_nucleotide.txt', if you want to view it
```

3. MAINTENANCE MANUAL FOR Competent Python3 Code-writer:

Welcome to the Maintenance Manual for this Programme.
This manual is meant for a user who knows how Python3 works and is proficient in coding programmes/scripts.
(Refer the user manual if you are an non-coder user for a simpler description of the programme and what it does)

The Maintenance manual will mainly focus on:

1. What each section of the code does (workflows attached at the end),
2. How each section of the code links to the next,
3. Possible Improvements that can be made for the script to run better.

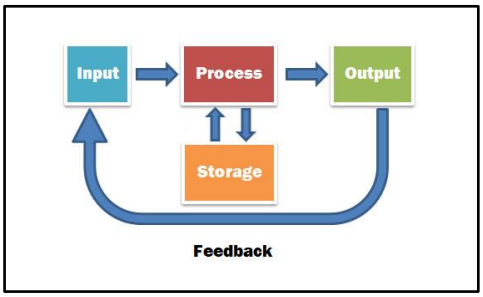
The code has 4 main sections:

1. Input Section
2. 1st Processing Section
3. 2nd Processing Section
4. Wildcard of biological importance

Let's look at how the code executes in detail:

A list of the inputs and outputs can be (optionally) viewed when the program is initiated. Choose **y** or **yes** to view them OR **n** or **no** to not view them.

Refer workflow images at the end for more information about the flow of information in the programme i.e. what are the inputs, outputs and processes of the programme. Just like the image shown below.



***Input Section*:**

- The user is given the option to input the taxonomic group and the protein name, both of which are stored as **variables**, **used repeatedly** in this section to generate various outputs such as the short description of the taxonomic group from the NCBI taxonomy database and to display the inputs to the user.

- For every **input** in the programme the method **.lower()** is applied so as to negate the effect of case sensitivity in various commands. The **.replace()** is also used to replace all **spaces** in the input with **"_"** which is another method of **input sanitisation**.
- Both **inputs** are obtained using **separate functions** which are multi-pronged. They sanitise the input and also check if the inputs are in the right format so that there are no errors in the execution of the program. **while True** loops are very helpful in these functions to ensure iteration continues till the right input is obtained.
- Number of sequences obtained is first **checked** and if inadequate (i.e. 0 or 1) then new dataset generation is initiated. If there are more than 1000 sequences, then the limit is set to 1000. Multiple sequence alignment (1st Processing stage) needs at least 2 sequences.
- **Choice** to generate new datasets are given **multiple times** to ensure that the user gets the right dataset they want to use to proceed to the processing stages.
- If you choose not to view the list of species for the dataset, programme will terminate. After viewing the list, a **.fasta file** containing all the sequences is **generated** which is what will be **used for** all the **processing steps** that are to follow.
- Ask the user one more time if they are satisfied with the dataset and ready to move on to the processing stages of the programme.

Possible Improvements:

- More **error trapping** could be done. **Better error reporting** is also a functionality which will greatly improve the usability of the code.
- Option to **toggle** the **time.sleep()** lines of the code **on and off** so that it can be executed quickly, especially if you are running the script multiple times.
- Choices where the code **terminates** could be **replaced** with an option to return to the top of the input section.

1st Processing Section *(clustalo & plotcon):*

- After the user has decided to proceed with the dataset generated, all the files are moved to a **new working directory** called **'/{protein_sequence}_{taxo_group}_files'**.
- All processing to follow will store the outputs here. This was done so the user can share the folder easily if they wish to and anyone receiving the data can immediately identify what the inputs used for the dataset are.
- Using the **.fasta** file, **clustalo** performs a multiple sequence alignment of the protein sequences and generates a **.aln** file which contains the aligned sequences.
- ***Working principle:***
  Clustalo (in our code specifically) first calculates **k-tuple distances** to build the **distance matrix** which is further used to build the **guide tree**. Alignment is done in the order specified by the guide tree. We use the **--iter=1** parameter to iterate once to create a new guide tree on which we can apply **kimura corrections** to generate a better alignment for our dataset of protein sequences.

- **--full** and **--use-kimura** are the two main parameters which will help us in gaining the best idea about what the level of conservation is between our protein sequences which we can visualise using the **plotcon** program. They take longer but the output is more accurate for our use case. As we have limited the number of sequences to 1000 in the input section, using **–threads=100** should get the alignment done quickly even if there are 1000 sequences in our dataset.
- A function is run which allows us to plot the current dataset with different **winsizes** so we can get a good idea of how the plot changes with respect to the winsize input.

Possible Improvements:

- Edit the code so that the output "I hope you liked these plots only executes if any plot is created".
- Adjust the **–threads parameter** to a different number with better understanding of what processes of clustalo can benefit from multi-threading.

***2nd Processing Section** (patmatmotif):*

- Created a new working directory which will store the outputs of running this section of the code as a lot of files which have no benefit to the user are created and to **differentiate it** from the last processing stage.
- The .fasta file is stored into a **list type variable** in which each element is a single sequence from the **.fasta file**. We read each element of the list variable using a **for loop** and write each sequence into its own file which is stored separately in a directory which stores all the individual .fasta files.
- **Patmatmotif** is run on each of the fasta sequences using **subprocess.run()** and the output of each run is stored into a folder which contains all the .**patmatmotif** files.
- The extensions are important as the **files** are **manipulated** using their **extensions**.
- The outputs of these files are **combined** to one file for easy manipulation.
- A variable which contains a **grep command** to extract the motifs and list them with their count is run using the **os.system(variable)** command and the output is displayed to the user.

Possible Improvements:

- Delete the unnecessary files at the end of the section.
- Could have used a different program like **pullseq** to write the fasta sequences to different files. More efficient way to tackle this problem
- Alternate way to extract the **motifs** rather than using a bash command.

***Wildcard** (backtranseq):*

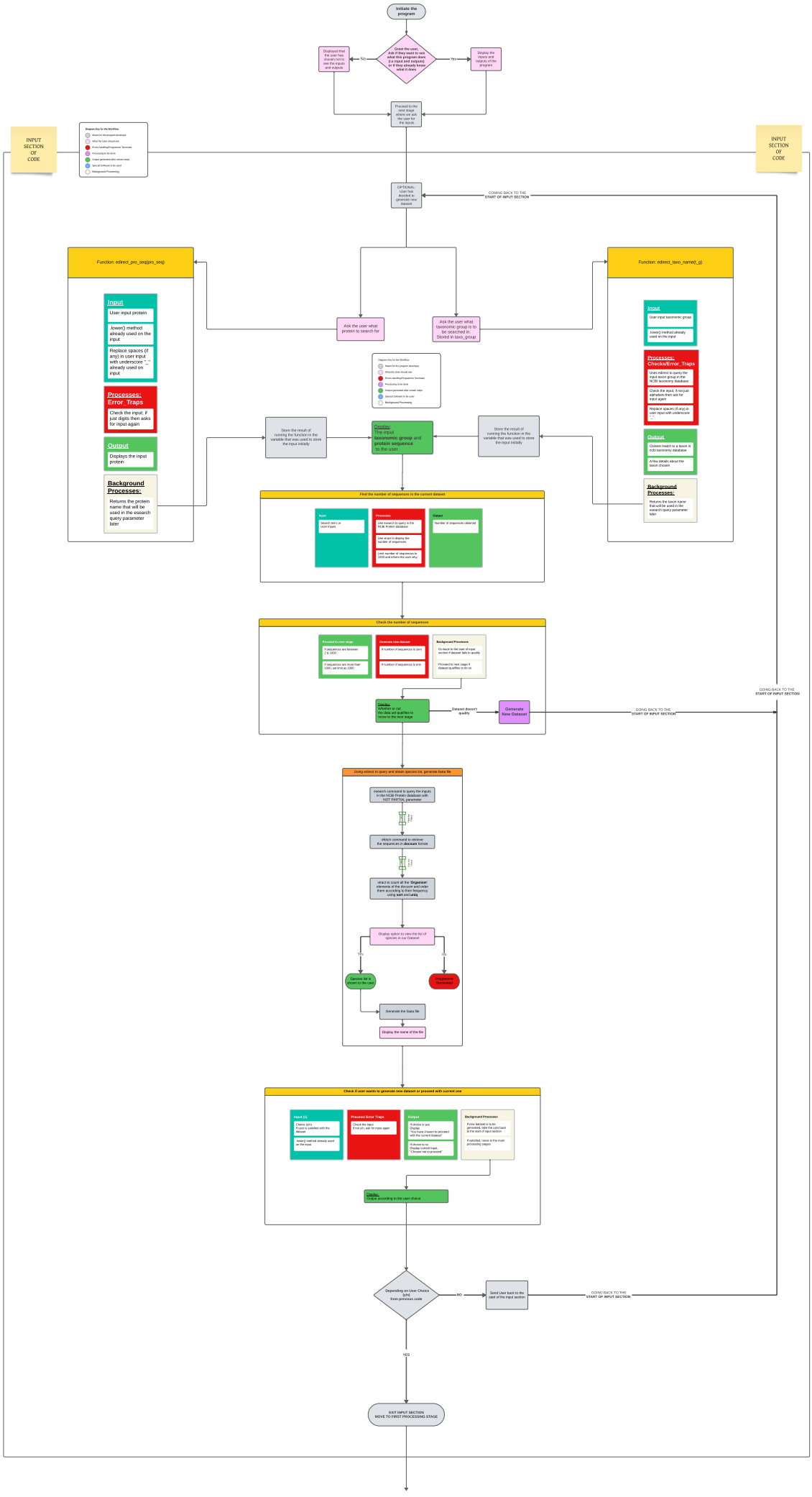
- It reads the **.fasta file** and writes the **nucleic acid sequences** that the proteins are most likely to **come from**. Can be viewed if the user wishes to do so. backtranseq uses a **codon usage table** which gives the frequency of usage of each codon for each amino acid. For each amino acid in the input sequence, the corresponding most frequently occurring codon is used in the nucleic acid sequence that is output.

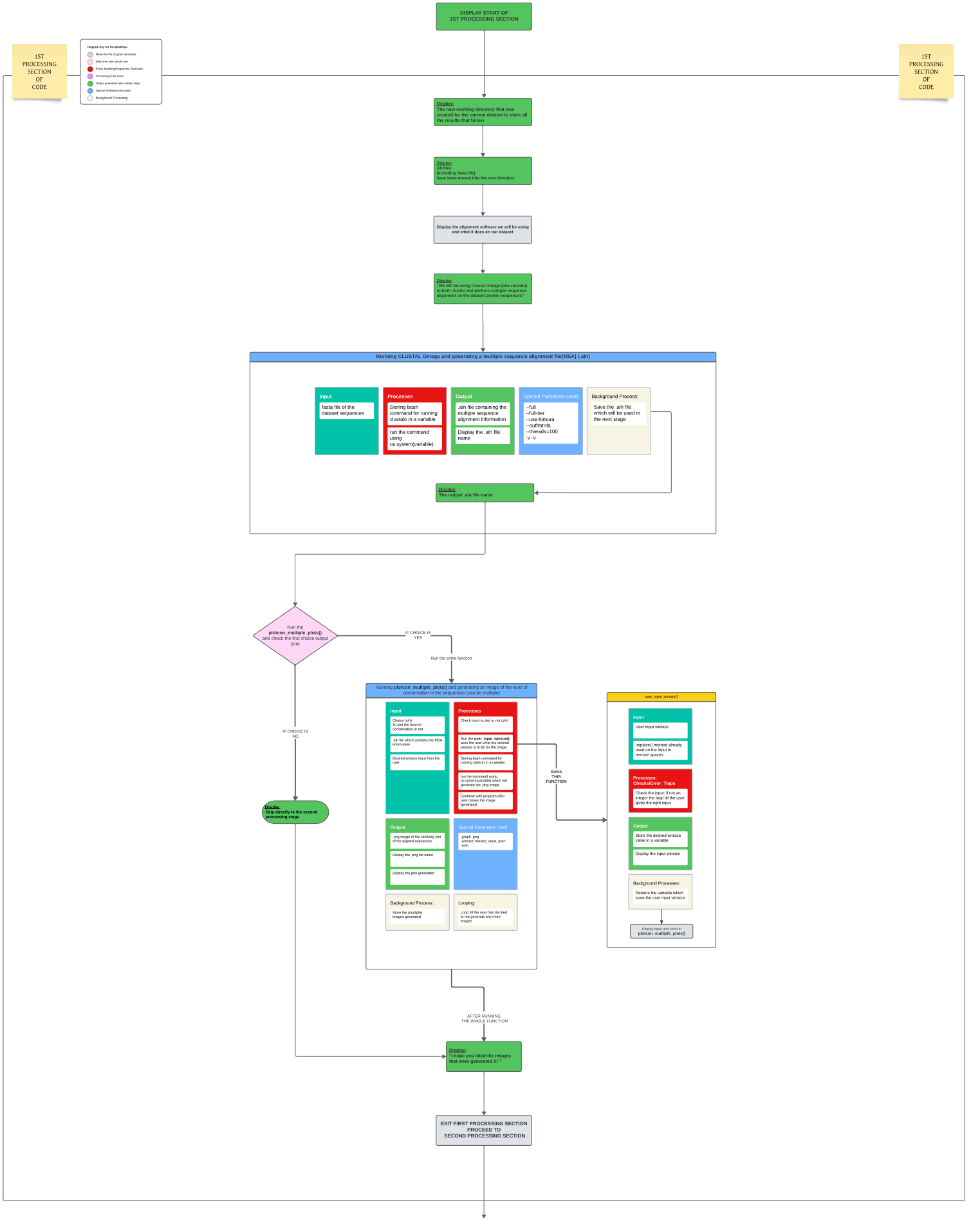
Useful Links as Supplementary Material:

1. EMBOSS Tutorial Comprehensive: https://emboss.sourceforge.net/docs/emboss_tutorial/emboss_tutorial.html
2. https://www.ncbi.nlm.nih.gov/books/NBK179288/
3. https://github.com/bcthomas/pullseq/blob/master/README
4. https://www.bioinformatics.nl/emboss-explorer/
5. http://www.clustal.org/omega/clustalo-api/
6. https://emboss.bioinformatics.nl/cgi-bin/emboss/help/plotcon
7. https://emboss.bioinformatics.nl/cgi-bin/emboss/help/patmatmotifs
8. https://emboss.bioinformatics.nl/cgi-bin/emboss/help/backtranseq

End of Help Manual Descriptive Sections
Please go through the Workflows to get a complete picture of the working
of the programme and how everything ties up together

Please see next 3 pages for the workflows of the different sections of the code.
Thank you for using this Help Manual, Hope it was helpful !


Initiate the program
Check the user, ask if they want to see what this program does (i.e input and outputs) or if they already know what it does
NO
Displayed that the user has chosen not to see the inputs and outputs
Yes
Display the inputs and outputs of the program
Proceed to the next stage where we ask the user for the input
INPUT SECTION OF CODE
Diagram Key for the Workflow
Input for the program developer
What the User should see
Points handing/Programme Terminate
Processing to be done
Output generated after certain steps
Special Software to be used
Background Processing
INPUT SECTION OF CODE
OPTIONAL: User has decided to generate new dataset
COMING BACK TO THE START OF INPUT SECTION
Function: edirect_pro_seq(pro_seq)
Input
User input protein
.lower() method already used on the input
Replace spaces (if any) in user input with underscore "_" already used on input
Processes: Error_Traps
Check the input; if just digits then asks for input again
Output
Displays the input protein
Background Processes:
Returns the protein name that will be used in the esearch query parameter later
Ask the user what protein to search for
Ask the user what taxonomic group is to be searched in. Stored in taxo_group
Function: edirect_taxo_name(t_g)
Input
User input taxonomic group
.lower() method already used on the input
Processes: Checks/Error_Traps
Uses edirect to query the input taxon group in the NCBI taxonomy database
Check the input; if not just digits then ask for input again
Replace spaces (if any) in user input with underscore "_"
Output
Closest match to a taxon in ncbi taxonomy database
A few details about the taxon chosen
Background Processes:
Returns the taxon name that will be used in the esearch query parameter later
Store the result of running the function in the variable that was used to store the input initially
Display: The input taxonomic group and protein sequence as the user
Store the result of running the function in the variable that was used to store the input initially
Find the number of sequences in the current dataset
Input
Search term i.e User inputs
Processes
Use esearch to query in the NCBI Protein database
Use xtract to display the number of sequences
Limit number of sequences to 1000 and inform the user why
Output
Number of sequences obtained
Check the number of sequences
Proceed to next stage
If sequences are between 2 & 1000
If sequences are more than 1000, set limit as 1000
Overcare core dataset
If number of sequences is zero
If number of sequences is one
Background Processes
Go back to the start of input section if dataset fails to qualify
Proceed to next stage if dataset qualifies to do so
Checker: Whether or not the data set qualifies to moves to the next stage
Dataset doesn't qualify
Generate New Dataset
GOING BACK TO THE START OF INPUT SECTION
GOING BACK TO THE START OF INPUT SECTION
Using edirect to query and obtain species list, generate fasta file
esearch command to query the inputs in the NCBI Protein database with -NOT INATIVE parameter
Output
efetch command to retrieve the sequences in docsum format
Output
xtract to count all the 'Organism' elements of the docsum and order them according to their frequency using sort and uniq
Display option to view the list of species in our Dataset
Yes
No
Species list is shown to the User
Programme Terminated
Generate the fasta file
Display the name of the file
Check if user wants to generate new dataset or proceed with current one
Input (i)
If below 1(int)
If user is satisfied with the dataset
.lower() method already used on the input
Processes Error Traps
Check the input
If not int, ask for input again
Output
If choice is yes
Display:
"You have chosen to proceed with the current dataset"
If choice is no
Display current input
"Chosen not to proceed"
Background Processes
If new dataset is to be generated, take the user back to the start of input section
If satisfied, move to the main processing stages
Display: Output according to the user choice
Depending on User Choice (y/n) from previous code
NO
Send User back to the start of the input section
GOING BACK TO THE START OF INPUT SECTION
YES
EXIT INPUT SECTION
MOVE TO FIRST PROCESSING STAGE


1ST PROCESSING SECTION OF CODE
Diagram Key for the Workflow
Meant for the program developer
What the User should see
Errors handling/Programme Terminate
Processing to be done
Output generated after certain steps
Special Software to be used
Background Processing
DISPLAY START OF 1ST PROCESSING SECTION
Display:
The new working directory that was created for the current dataset to store all the results that follow
Display:
All files
(including fasta file)
have been moved into the new directory
Display the alignment software we will be using and what it does on our dataset
Display:
"We will be using Clustal Omega (aka clustalo) to both cluster and perform multiple sequence alignment on the dataset protein sequences"
Running CLUSTAL Omega and generating a multiple sequence alignment file[MSA] (.aln)
Input
fasta file of the dataset sequences
Processes
Storing bash command for running clustalo in a variable
run the command using os.system(variable)
Output
.aln file containing the multiple sequence alignment information
Display the .aln file name
Special Paramters Used
--full
--full-iter
--use-kimura
--outfmt=fa
--threads=100
-v -v
Background Process:
Save the .aln file which will be used in the next stage
Display:
The output .aln file name
Run the plotcon_multiple_plots() and check the first choice output (y/n)
IF CHOICE IS YES
Run the entire function
IF CHOICE IS N/O
Display:
Skip directly to the second processing stage
Running plotcon_multiple_plots() and generating an image of the level of conservation in the sequences (can be multiple)
Input
Choice (y/n)
To plot the level of conservation or not
.aln file which contains the MSA information
Desired winsize input from the user
Processes
Check input to plot or not (y/n)
Run the user_input_winsize(), asks the user what the desired winsize is to be for the image
Storing bash command for running plotcon in a variable
run the command using os.system(variable) which will generate the .png image
Continue with program after user closes the image generated
Output
.png image of the similarity plot of the aligned sequences
Display the .png file name
Display the plot generated
Special Paramters Used
-graph .png
-winsize 'winsize_input_user'
-auto
Background Process:
Store the (multiple) images generated
Looping:
Loop till the user has decided to not generate any more images
RUNS THIS FUNCTION
user_input_winsize()
Input
User input winsize
.replace() method already used on the input to remove spaces
Processes: Checks/Error_Traps
Check the input, if not an integer the loop till the user gives the right input
Output
Store the desired winsize value in a variable
Display the input winsize
Background Processes:
Returns the variable which store the user input winsize
Display input and send to plotcon_multiple_plots()
AFTER RUNNING THE WHOLE FUNCTION
Display:
"I hope you liked the images that were generated !!! "
EXIT FIRST PROCESSING SECTION
PROCEED TO
SECOND PROCESSING SECTION
1ST PROCESSING SECTION OF CODE

2ND PROCESSING SECTION OF CODE

DISPLAY START OF 2ND PROCESSING SECTION

Display:
"Welcome to the Motif searching section of this Programme"

Display:
"We will be using patmatmotifs to search for motifs in our sequences"

Choice
(y/n)
If the user wants to search for motifs or terminate the program

NO
I.E
Do not want to proceed

Display:
"Thankyou for using this programme"

YES
I.E
Proceed to search for motifs

Display:
The new working directory that was created for the current dataset to store all the results that follow

Display:
The fasta file we are going to use and "Fasta File copied successfully to be used for searching"

Making individual fasta files from the fasta file

Input
fasta file of the dataset sequences

Processes
1.Create a folder to store the individual fasta seqs
2.Storing the sequences in a list variable separated by the ">"
3.Loop through the list variable and write each individual sequence to its own file in the fasta dir

Output
Display where the fasta files have been stored

Background Process:
While looping the list variable, a variable stores the path of the current sequence being looped by it's name in the fasta dir
fasta file is created by writing files to the path stored in the variable for each iteration of the loop

Display:
" Time to start the search ... Hold on to your seat ... This is fast and furious ... "

Running patmatmotif on each individual fasta file & Combining all .patmatmotif files to one file

Input
fasta files of the each individual sequence

Processes
1. Create a folder to store the individual patmatmotif files
2.Loop through the fasta dir, checking if the file ends with .fasta
3. Run patmatmotif on each of the .fasta files and store them in the patmatmotif dir
4. Write the output of running patmatmotif on each file to a single file which will store all the outputs

Output
Display where the .patmatmotif files have been stored
Display the name of the final combined output file which has the outputs of all the .patmatmotif files

Background Process:
While looping the list variable, a variable stores the path of the current .patmatmotif file being looped by it's name in the patmatmotif dir
.txt file is created which contains the combined outputs

Display:
"All patmatmotif results are combined into {combined_output_filename}"

Process:
run the following bash command to get the list of motifs present
"grep 'Motif' {combined_output_filename} | cut -d'=' -f2 | sort | uniq -c | sort -nr > {output_file_name}"

Display:
"Motifs present have been successfully extracted and saved"

Display:
These are the motifs that were present in our database

2ND PROCESSING SECTION OF CODE