BPSM ICA2 Help Manual

B217754-2022

Git repository details

Repository address: https://github.com/B217754-2022/ICA2.git

CCRYPT password for B217754-2022.ICA2.tar.gz.cpt: functioningtokens

User guide (P.Q.P for Dummies)

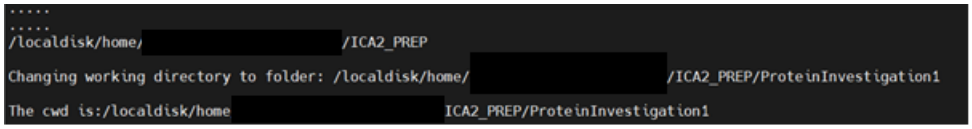
Instructions to use program, how to interpret output

Contents

Starting P.Q.P.

When running the program, you will be greeted with a choice to run the Protein Query Programme (P.Q.P.) by pressing **Y**. Though this choice may seem nonsensical at this stage, the programme eventually loops around to this point, so if investigations were concluded, pressing **N** closes the script.

The script will make a folder for future files from the current investigation to be kept in, and inform you of the location of that folder on your system.



N.B.: The parts of P.Q.P. which open and display files, charts and graphs can take a long time to open said documents, so please have patience if you have chosen to open any documents.

Enter your Query

When prompted, please **enter the name of the protein group under investigation, press return and enter the taxonomic group** too ('pyruvate dehydrogenase' and 'Ascomycete fungi' are offered here as examples). Text entry for the query terms is mandatory, and the script will search any characters entered here against NCBI's Protein database, so a typo would still be run (though it would likely return no results and P.Q.P. would inform you of this).

You will be prompted to confirm your query search terms, if you spot a typo here, typing the option C will allow a new query to be typed in. If the query terms seem correct, press P to **P**roceed.

```
.....
Please type the protein family of interest. An example is pyruvate dehydrogenase
pyruvate dehydrogenase
Please type the taxnomic group of interest. An example is Ascomycete fungi
^VAscomycete fungi
The query terms you entered are:
pyruvate dehydrogenase Ascomycete fungi.

Continuing...

Your query is pyruvate dehydrogenase Ascomycete fungi, enter P to Proceed or C to Change your query
P
```

Sequence retrieval

P.Q.P. will now retrieve results from NCBI Protein (NCBIp) for your query, the interface may state ERRORs or WARNINGs as it contacts NCBI servers. Occasionally, the query will fail because servers are non-responsive, in which case you should re-run the script.

The interface will scroll slowly, stating:

"There are x entries on NCBI Protein for [your protein] in [your taxonomic group]"

Then, shortly afterwards:

"x unique species in total for this search"

[The list of species which appears in the results of your query]

The list of species is accompanied by a value showing the number of entries the corresponding species name on that line had on NCBIp with your query. This may inform investigation if one species is over-/under-represented in the results. The list of species is saved as file **speciesList.txt**, which you are prompted to open by pressing **Y**, if you choose. There is also an option to **C**hange your query terms and restart the process if you are not satisfied with the results on offer. Otherwise, press **N** to carry on.

```
Counting the number of entries on NCBI Protein for your query: pyruvate dehydrogenase Ascomycete fungi
There are 232 entries on NCBI Protein for pyruvate dehydrogenase in Ascomycete fungi
Moving onto species retrieval
.....
.....
.....
.....
.....
The following list shows the species with a recorded pyruvate dehydrogenase protein.
    The numbers in the first column represent the number of entries a species has in NCBI Protein with your query
    (130 unique species in total for this search):
.....
      1 Plectosphaerella cucumerina
      1 Plectosphaerella plurivora
      1 Polyplosphaeria fusca
      1 Saccharomycopsis javanensis
      2 Saitoella complicata NRRL Y-17804
      1 Sclerotinia borealis F-4128
      3 Spathaspora sp. JA1
      2 Sporothrix brasiliensis 5110
      1 Sporothrix insectorum RCEF 264
      2 Sporothrix schenckii 1099-18
      1 Starmera amethionina
      3 Talaromyces pinophilus
      1 Terfezia boudieri ATCC MYA-4762
      1 Thozetella sp. PMI_491
      1 Tolypocladium paradoxum
      2 Trematosphaeria pertusa
      2 Trichoderma citrinoviride
      2 Trichoderma lentiforme
      1 Trichoderma longibrachiatum ATCC 18648
      1 white Piedmont truffle
      1 Wickerhamiella sorbophila
      1 Wickerhamomyces alni
      1 Wilcoxina mikolae CBS 423.85
      2 Xylona heveae TC161
      2 Yamadazyma tenuis ATCC 10573

The list of species has also been saved as file speciesList.txt.
       Would you like to view it or change your query? (Y/N/C)
```

GenBank generation and download

The NCBIp entries will be downloaded in GenBank format in one long file (**queryGBs.txt**) and into separate files for each entry too (numbered in order that the results appear in the large **queryGBs.txt** file). This format contains the core information held in the database entry, and the first results' Genbank information is shown on screen as a demonstrative example of this.

```
Writing Gb file 227
Writing Gb file 228
Writing Gb file 229
Writing Gb file 230
Writing Gb file 231
Writing Gb file 232
Writing Gb file 233
Genbanks files generated for later.

Making separate files of each sequence in Fasta format for use later. The full list of fastas can be found in the file queryseqs.fasta
.....
.....
.....
.....
.....
Please enter the desired minimum and maximum length of sequences
30 650
```

```
Retrieving Genbank reports of the query results, please wait.
A file with the query results in GenBank format has been saved as queryGBs.txt for your viewing.
.....
.....
.....
the first entry in GBs is

 LOCUS       KAI5905415              597 aa            linear   PLN 05-AUG-2022
DEFINITION  Pyruvate dehydrogenase [Candida parapsilosis].
ACCESSION   KAI5905415
VERSION     KAI5905415.1
DBLINK      BioProject: PRJNA610714
            BioSample: SAMN14309842
DBSOURCE    accession JADCQT010000062.1
KEYWORDS    .
SOURCE      Candida parapsilosis
  ORGANISM  Candida parapsilosis
            Eukaryota; Fungi; Dikarya; Ascomycota; Saccharomycotina;
            Saccharomycetes; Saccharomycetales; Debaryomycetaceae;
            Candida/Lodderomyces clade; Candida.
REFERENCE   1  (residues 1 to 597)
  AUTHORS   Yamin,D., Wan Juhari,W.K., Hanis Zainal Abidin,N., Mat-Sharani,S.
            and Harun,A.
  TITLE     Draft Genome Sequence of Clinical Isolate USM039K of the Pathogenic
            Yeast Candida parapsilosis
  JOURNAL   Microbiol Resour Announc, e0084122 (2022) In press
   PUBMED   36301110
  REMARK    Publication Status: Available-Online prior to print
REFERENCE   2  (residues 1 to 597)
  AUTHORS   Marzouq Yamin,D.H., Wan Juhari,W.K., Zainal Abidin,N.W.H., Harun,A.
            and Mat-Sharani,S.
  TITLE     Direct Submission
  JOURNAL   Submitted (15-OCT-2020) Department of Medical Microbiology and
            Parasitology, USM-Health Campus, Jalan Raja Perempuan Zainab 2,
            Kota Bharu, Kelantan 16150, Malaysia
```

FASTA generation and selection

The NCBIp sequences are then downloaded in FASTA format in one long file (**queryseqs.fasta**), but the script will prompt you to choose a minimum and maximum sequence length to filter out any short read sequences which may have been obvious from the species list. The values entered here should be **two numerical values separated by a space**, the script will correct you if none are entered or the minimum and maximum values are entered in the wrong order. The list of length-filtered FASTA sequences are saved to **filteredQuerySeqs.fasta** and similarly to the GenBank format section, the FASTAs will be saved to their own files, numbered in the same way too. You will be offered to view **filteredQuerySeqs.fasta** if you choose with **Y**. Pressing **N** moves on to sequence clustering.

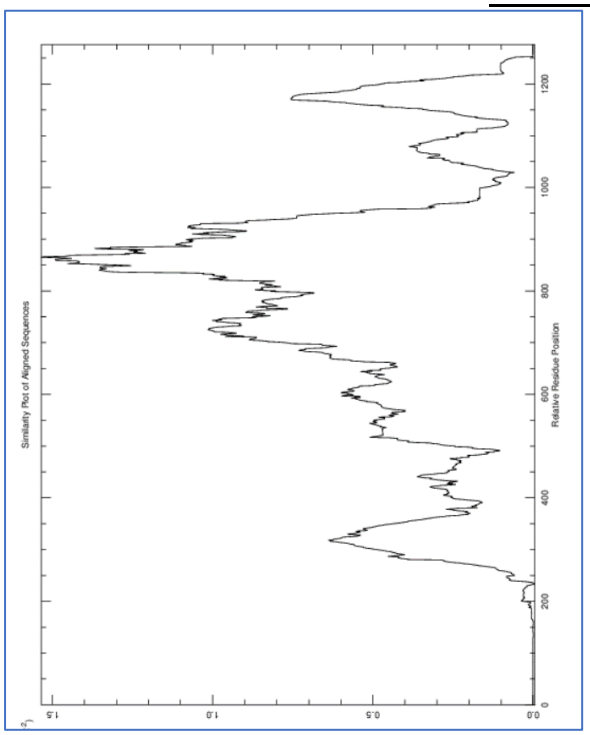
Sequence clustering

P.Q.P. uses Clustal Omega to cluster sequences into a multiple sequence alignment using seeded guide trees. The FASTA sequences made in the last step will be used to make this alignment in the **prettyQueryAlignment.txt** file, which you are prompted to open with **Y**. Otherwise, move on with **N**.

```
2576 EJP62222.1     GDSNSK-------------------R---------------E------AIYEPFDFGQFWK-----GVNWKFVKGRTTNQ.------DNAAV--------
2577 EFX04630.1     QLR------------------L---GTGTPPPIEVTICEA-----QQ--HIIIRVSDQGGGVSQDMLPYLWSFS--KGEHSQ-RLLENLQKVPRMSA-T
2578 EEH11658.1     HGA----------------------EKEAFPVTKVIVAEG-----RE--DITIKISDEGGGIPRSSIPLVWTYMYTTVDQTP-----------NLD---
2579 EEH03746.1     NDPI---------------------AGLKQKLLDWGVTSE------------------------EEL-KAIDKDARKYV.D------------------
2580 EEH03442.1     KQKY-----GVET-----EVINL---RSIKPLDVETIIKSVKKTGHLMAVESGFPMFGVGSEILALSMEYAFDYLQAPAVRVT-----GA----------
2581 XP_007819596.1 GA-GTE-------------------E---------------P------KSYPPFDFGAFQE-----GVSPGFVRERTIIQ.------DNAAV--------
2582 EAW11285.1     ---------------------------RERLIEWGIITE----------------------DEA-KAMDKDVHGIInQ------------------
2583 KIH87759.1     TTPSKR-------------------P-----------------------------------VRLGRAFHASYTTTR.------GNAAV--------
2584 KIE00886.1     GA-GTE-------------------E---------------P------TSYPPFDFGAFRE-----GVSPGFVKERTIIQ.------DNAAV--------
2585 KID85608.1     GA-GTE-------------------E---------------P------TSYPPFDFGSFRE-----GVSPGFVKERTIIQ.------DNAAV--------
2586 KID70379.1     GA-GTE-------------------E---------------P------TSYPPFDFGSFRE-----GVSPGFVKERTIVQ.------DNAAV--------
2587 KID69557.1     GT-GRE-------------------E---------------P------TSYPPFDFGSFRE-----GVSPGFVKERTIIQ.------DNAAV--------
2588 KIA75519.1     HGQKAV-------------------DKGSLEPG--------SDFG---VRGGQFDFGRLGK---KTSEPWGFAEGRTTVQ.------SNSAV--------
2589 EFZ00811.1     GA-GTE-------------------E---------------P------KSYPPFDFGAFQE-----GVSPGFVRERTIIQ.------DNAAV--------
2590 KGQ11384.1     A-------FSTDGPVLVDVVVAKD--ELAIPPQIKLEQAKG-----FSLYMLRAIISGRGDEVIELAKTNWFR-------------------------
2591 XP_008602215.1 GDSNSK-------------------R---------------E------AIYEPFDFGQFWK-----GVNWKFVKGRTTNQ.------DNAAV--------
2592 XP_008593908.1 NLEEMK-------------------MKLKITPETAWWKKEPQAEA---DYPAGFDFLERWN-----DFDIRFRQERAVIE.L-----DNVAV--------
2593 XP_001399196.2 SR-PG-----------------------------------M----KSPGRFDFSRFWN-----GVDWRFVEGRTAVQ.------ENAAV--------
2594 XP_001272711.1 ---------------------------RERLIEWGIITE----------------------DEA-KAMDKDVHGIInQ------------------
2595 Consensus      xxxxxxxxxxxxxxxxx-xxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxx-xxxxxxxxxxxxxxxxxxxxxxxxxxxDxxxxxxxxxxxxxxxxxx
```

ANALYSIS: Sequence conservation analysis

P.Q.P. uses EMBOSS plotcon to quantify conservation of residues across the sequences under investigation with a parameter known as the window size. Information on the algorithm behind plotcon can be accessed here, but the size of the window determines sensitivity to conservation (with smaller windows producing more precise signals at the risk of higher noise). You will be prompted to specify a window size **up to 150 by typing a numerical value with digits**. The conservation file **conservationPlot.ps** can be opened here with **Y** or you can move on with **N**.



ANALYSIS: Motif identification using PROSITE

The interface will show lots of output stating that the previously-generated FASTA files are being scanned as the script checks for domains in the sequences against the PROSITE database. Results from this scan are added to a table, which can be viewed in full file format (**[your query terms]MotifResults.tsv**) by pressing **Y**. Otherwise, **N** will move on.

```
Scanning Fastas
Iterating through detected motif hits and adding them to a table
              Start  End Score Strand         MotifName
SeqName
ATY61333.1      254  257     4      +         AMIDATION
EAW18214.1      398  400     3      +  MICROBODIES_CTER
EEH03746.1      153  156     4      +         AMIDATION
EEH11658.1      161  164     4      +         AMIDATION
EGV65900.1       66   95    30      +            LIPOYL
...             ...  ...   ...    ...               ...
XP_040653805.1  157  165     9      +             PPM_1
XP_045284227.1  153  156     4      +         AMIDATION
XP_045292138.1  161  164     4      +         AMIDATION
XP_047786132.1   40   51    12      +  AA_TRNA_LIGASE_I
XP_047786132.1   59   88    30      +            LIPOYL
```

ANALYSIS: Residue property analysis (Wildcard Section)

The interface will show lots of output stating that amino acid properties of a protein sequence are being plotted, and will create an image corresponding to the residue properties of each entry from the previous GenBank file, saved as **GBs*graphs.ps** (where * is the number assigned to the entry by the order it appears in the GenBank file). You are prompted to specify **one such number**, and the file will be displayed. If you choose, you can view another file after with **Y** and specify the **next number**, or finish P.Q.P. with **N**.

Finishing P.Q.P. returns you to the starting prompt, at which point pressing **N** will exit the script gracefully. Thank you for using P.Q.P.!

BPSM ICA2 Help Manual

B217754-2022

Code walkthrough for competent Python3 code-writers

This script is primarily divided into functions which are called in a sequence to present functions to the user. As a point of self-evaluation, I had originally wanted to use a while loop to allow the user to simply 'continue' back to query input at the start of the programme, however I was unable to implement this as continue commands which restart while loops must be used as syntactically in the while loop, not called by a function. I highlight this point as an area for improvement in the future versions.

It is recommended that this script is viewed in an editor with line numbers for ease of navigation.

Line Number	Explanation, notes and evaluation of code
14	Importing the required modules for: access to the system directory, making dataframes, and enabling time suspensions.
17	The delay time function is defined with customisable parameters to extend or shorten the duration of the delay. This is re-used many times in the script with varying values for the length of delay.
23	The function containing most of the code for the script is defined from this point onwards. The function is called at line 434 within the greeting function.
25-31	Creates a uniquely numbered directory in the current directory. NOTE: this unfortunately results in the nesting of directories on each run, see later notes for complications.
42-17	The placeholder objects are created to hold the user's query. They are defined both within and outside of the function in case the user calls the function again.
48-49	Take the user input from prompts included in the command (asking for protein and taxon family).
52-57	The if loop evaluates the user's input with a comparison operator which checks whether anything was typed at all. If this condition is not satisfied, the query selection function is called again (line 55).
65-75	This function shows the user their query again and allows them to overwrite it if they wish. Inputs to the prompts are limited to Y or N by an if statement that calls the function on itself again if the conditions are not met.
83-88	These variables are strings which will be fed into the edirect searching commands mentioned in the comments. The first variable acts as a common stem for all of the EDirect search commands, whilst the other variables hold the specific search parameters such as the desired result formats.
91-97	The query variables are concatenated using f strings into a variable, which itself is used to search the NCBI protein database and store the count of results in a variable which can be printed to the user.
99-110	This function is used twice in the next block, taking the user's input as a decision to 'restart' the programme by calling back the functions used so far, essentially resetting the programme. The input is restricted to Y or N by if statements, so the function calls itself if the input does not satisfy this condition. NOTE: If the while loop was operational, I would not recall the functions, but instead use 'continue' to restart the PQP function anew. KNOWN BUG/ISSUE: If the user restarts at this point, some of the variables involved in query searching are not reset (reason unknown) RESOLUTION: simply follow options to quit the programme and run from the start.
112-121	This if statement evaluates the value describing the count of results retrieved from the block at lines 91-97. If the count of results is less than 0 or more than 1000, the script offers a choice to resume or enter a new query.
123-133	Similarly to lines 91-97, this block uses concatenation to assemble a full query string for EDirect, but this time saves the output to a file, and counts the number of lines in the output to enumerate the unique species in the results set. The list is returned to the screen after a message describing the format and a short delay using the 'ReadingTime' function.
140-165	This function contains a block of code which allows the user to open the recently created text file of entries per species. The '&' character in the command being passed to the OS allows the rest of the script to carry on running whilst the file reader opens, as this can take some time to load to the user's screen. The block also evaluates the user's response, including an option to restart the programme again. NOTE: As before, recalling these functions seems to be missing a line to overwrite the previously saved variables, which would be rectified with the use of a while loop.
168-172	The block uses concatenated variables containing strings to retrieve the query results in GenBank format into a single long string. The output is also saved to a variable for manipulation by Python. GenBank is a human-friendly format, with extensive information for the biologist-user to browse.

174-184	The GenBank-format results held in the variable are split into a list of results in a new variable using the .split() command and parameter of '//', which appears at the end of each GenBank entry. The first entry of this list is printed to the screen for the user to see an example of GenBank entries. A for loop is used to iterate through the items of the GBs list containing the results in GenBank format. For each item in the list (query result) the contents are written to a file created with a unique name generated by using a counter variable (set up outside of the loop).
199-205	This block of code retrieves the sequences using concatenated variables, saving the output of all the fastas to a single file and a variable for manipulation (similar to the GenBank outputs). The fasta sequences are then split into a list as individual items, this removes the '>' symbol from each sequence wo this is added back later when making individual files.
207-240	This function takes user input of two numerical values and uses this range to filter out sequences of a certain length when creating a new list of sequences from the first list. The filtered list of sequences is written to a new file NOTE: the sequence item in the list include the fasta headers for each sequence, meaning that the range specified will not be accurate to desired sequence length. Time constraints prevented this from being corrected, however the list object items could be split into key:value pairs in a dictionary, allowing the identification of the sequences more accurately. The filtered sequences are written to their own files in a similar fashion to the genbank entries, using a counter variable to name the files. A dictionary object would also facilitate the naming of the files more accurately here (using the sequence accession number as the key). There is an error trap here, arising if the conversion of the user's input to number results in an error (in the case where the input contains non-numerical characters). The error trap calls the function on itself to prompt the user for correct input.
242-250	This function gives the user the option to open the file containing the list of length-filtered sequences. Acceptable input is restricted to Y or N
261-267	Running clustal omega with 30 threads to process the data in parallel. The –auto and –force parameters prevents prompts from being printed and overwrite existing files with the same name. '-i' denotes input file, '-o' denotes output file name. Permissions for the created file are altered to allow the rest of the script to function. NOTE: time did not allow for testing of the appropriate command, so both the os.popen() and subprocess.call() methods of executing the two commands in the command line shell were included. Future streamlining of the code could remove the redundant commands here.
270	PQP uses the 'showalign' command to generate a nicely formatted sequence alignment for the user to access. '-sformat' denotes the desired format of the output written to the output file, '-width' is the number of characters per line to be written in fasta format in the output. '-auto -warning -die' prevents prompts from being printed but prints error messages to the screen.
273-285	This function gives the user the option to open the nicely-formatted alignment which was recently created. Again, user inputs are limited to Y or N and the '&' character in the OS command allows the script to keep running without waiting for the file viewer to open.
292-306	This function allows the user to specify the conservation window that they want to use in the plotcon command. An if statement ensures that the window is under 150, and an error trap ensures that the user has entered a numerical value using numbers
308-318	This function gives the user the option to open the newly generated plot from plotcon, the '&' is not required here as the graph image opens rapidly. User input is restricted to Y or N with and if statement.
321-335	This function calls the previous two functions to use the plotcon functionality. After the process is complete, the user can repeat the graph generation with a different window size if they wish. The plotcon command is passed to the OS, with parameters '-graph ps' and '-goutfile' to save the conservation graph as an image named 'conservationPlot.gs'.
348-355	A for loop iterates through every file in the OS current working directory, running patmatmotifs to find protein domains known to the PROSITE database. Results are written to one file per sequence, with the name generated by stripping the '.fasta' file extension and using the name of the file. The patmatmotifs command is passed to the OS with the parameters: '-sprotein1 Yes' to specify that the input is protein sequence data, '-auto' to stop prompts from appearing '-sformat1 fasta' to specify that the input protein sequences are in fasta format, '-rformat excel' returns the results as an tsv format file. The for loop uses 'continue' to loop through every file in the directory, but the if statement ensures that only fasta files are being fed to patmatmotifs. NOTE: It appears that EMBOSS: prosextract has been run and the required data for patmatmotifs is present at '/usr/share/EMBOSS/data/PROSITE/prosite.lines' https://emboss.sourceforge.net/apps/release/6.6/emboss/apps/prosextract.html

	EMBOSS:patmatmotifs https://emboss.sourceforge.net/apps/release/6.6/emboss/apps/patmatmotifs.html
358	Creates a pandas dataframe for patmatmotif results to be filled into, with corresponding column names.
360-366	The files generated by the code in lines 348-355 are iterated through using a for loop, read into a temporary pandas dataframe (if there are no values to write, '-' is filled in place in the dataframe), and concatenated to the end of the previously created dataframe. This forms the results table of motifs in the query results.
369-376	The column names for the results table are adjusted by using '.rename' and 'inplace=True' to overwrite the old names. The results are sorted based on the name of the associated sequence. NOTE: in the future, the sorting feature could be interactive for the user, allowing them to sort by any of the columns of the results table by using a function and indexing the column names for the user to choose by entering a single number as input. The results table is exported as a .tsv file, with tabs as the separators and headers (column names) preserved. The results table dataframe is also printed for the user to preview.
378-390	This function gives the user the option to view the recently created .tsv file. Input options are restricted to Y or N. The '&' sign is used in the OS command to progress the script without delay whilst the file viewer opens.
399-405	This for loop iterates through the previously-created GenBank format files as input for the 'pepinfo' command which returns information on the amino acid properties for the sequences. The parameters '-sformat gb' specifies that the input files are in GenBank format, '-graph ps' the out is an image, '-goutfile {outputName}graphs -outfile {outputName}pepinfo.ps' specify the names of the resulting output files.
407-428	This function gives the user the option to open the pepinfo file of their choice by specifying the numeric identifier of the sequence of interest. The if statement opens a file if the '!=' comparison operator to detect an input that is not N is satisfied. The user input is converted into an integer (and an error trap repeats the prompt if the input is not a number, specifying that the numeric identifier will be a number between 0 and the total count of results). User input of N ends the PQP programme.
430-445	This function is a greeting for the user, giving the option to run the programme or exit if they choose. This is presented at the end of the PQP programme as seen on line 428 in case the user wants to run another query without having to exit the programme. User input is restricted to Y or N with an if statement.