Mass automated translation of DNA sequences and motif search on translated sequences

Overview

To find motifs of interest, for instance cyclotides, from transcriptomic data retrieved from DNA sequencing involves two major steps: (1) translating all DNA sequences into amino acid sequences based on all possible reading frames, and (2) motif search on all translated sequences for proteins matching target motif. The conventional method in doing the above tasks is time consuming as it typically involves processing individual sequences using web services, the steps outlined here utilizes automation and can drastically speed up this process especially for large amounts of sequences.

Pre-requisites and installation guide

These are the programs, with brief installation guide, required to perform the steps for DNA translation and motif search.

Python3 and Biopython

Download and install the latest version of Python from the official website (https://www.python.org/downloads/). Latest version at the time of writing is Python 3.9.5.

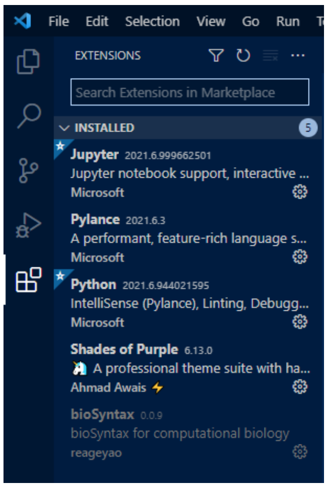
Biopython is a freely available library written for Python which includes many tools for biological computation. To install biopython, simply execute the following command in the command line (after installing Python):

```
pip install biopython
```

Visual Studio Code (vscode)

Visual Studio Code is a generic source code editor that works with Python. It is highly recommended due to its ease of use and high performance.

Download and install Visual Studio Code from the official website (https://code.visualstudio.com/). After installation, launch Visual Studio Code then search for and install the Python extension.


File Edit Selection View Go Run
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 5
Jupyter 2021.6.999662501
Jupyter notebook support, interactive ...
Microsoft
Pylance 2021.6.3
A performant, feature-rich language s...
Microsoft
Python 2021.6.944021595
IntelliSense (Pylance), Linting, Debugg...
Microsoft
Shades of Purple 6.13.0
A professional theme suite with ha...
Ahmad Awais
bioSyntax 0.0.9
bioSyntax for computational biology
reageyao

Finally, set the directory of Python in Visual Studio Code for proper integration. Go to View > Command Palette, type in and click on Python: Select Interpreter, select the directory of Python installation.

Perl and ScanProsite’s ps_scan
Perl is a feature-rich programming language. To run Perl programs, a Perl environment needs to be installed for the respective operating systems. For Windows it will be Strawberry Perl.

Download and install Strawberry Perl provided by the official site (https://strawberryperl.com/).

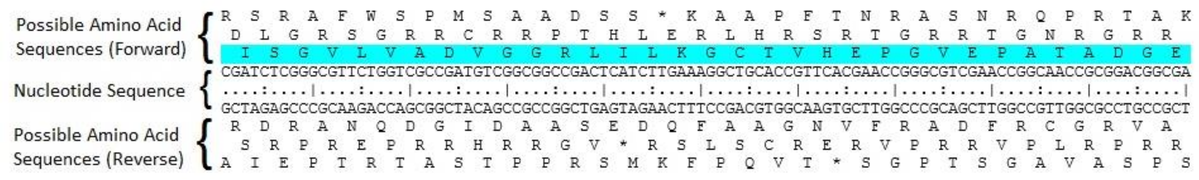
ps_scan is a Perl program used to scan one or several patterns, rules and/or profiles from PROSITE against one or several protein sequnces in Swiss-Prot or FASTA format. Download the package (https://ftp.expasy.org/databases/prosite/ps_scan/ps_scan_win32.zip) and extract it to a folder of convenience. Recommended directory: /Documents/ps_scan/

GitHub repository
All scripts described in this document are available in this GitHub repository:
https://github.com/Baboolal/bioinformatics-tools

Other scripts for different purposes are available too.

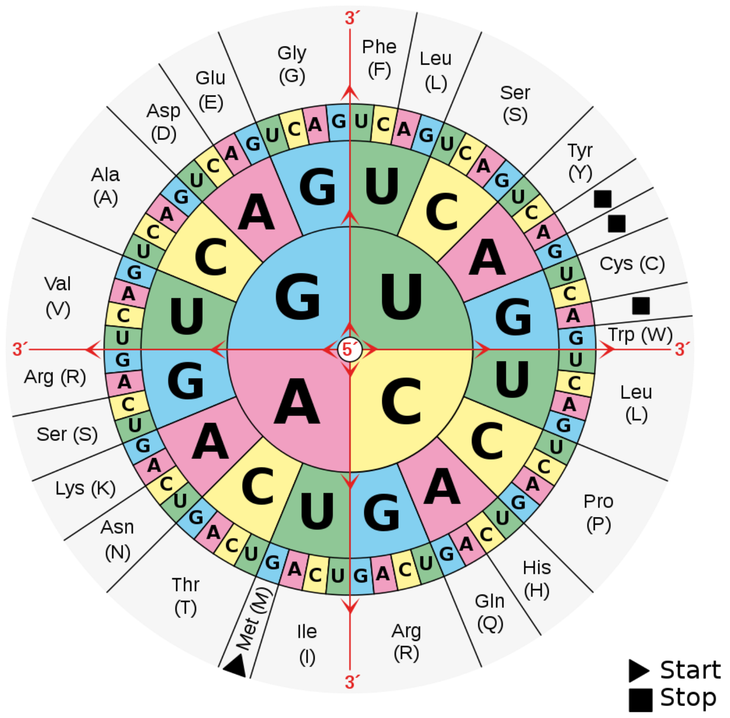
DNA Translation (dna2aa_mk3.py or dna2aa_mk3_allframes_mk2.py)

A Python script was written for automated translating a collection of DNA sequences in FASTA format into 6 collections of translated amino acid sequences in FASTA format, 1 collection for each reading frame. The 6 reading frames are 5’ to 3’ frames 1 to 3, and reverse complemented 3’ to 5’ frames 1 to 3, as shown in this figure:


Possible Amino Acid Sequences (Forward)
Nucleotide Sequence
Possible Amino Acid Sequences (Reverse)

```
R  S  R  A  F  W  S  P  M  S  A  A  D  S  S  *  K  A  A  P  F  T  N  R  A  S  N  R  Q  P  R  T  A  K
 D  L  G  R  S  G  R  R  C  R  R  P  T  H  L  E  R  L  H  R  S  R  T  G  R  R  T  G  N  R  G  R  R
  I  S  G  V  L  V  A  D  V  G  G  R  L  I  L  K  G  C  T  V  H  E  P  G  V  E  P  A  T  A  D  G  E
CGATCTCGGGCGTTCTGGTCGCCGATGTCGGCGGCCGACTCATCTTGAAAGGCTGCACCGTTCACGAACCGGGCGTCGAACCGGCAACCGCGGACGGCGA
....:....|....:....|....:....|....:....|....:....|....:....|....:....|....:....|....:....|....:....|
GCTAGAGCCCGCAAGACCAGCGGCTACAGCCGCCGGCTGAGTAGAACTTTCCGACGTGGCAAGTGCTTGGCCCGCAGCTTGGCCGTTGGCGCCTGCCGCT
 R  D  R  A  N  Q  D  G  I  D  A  A  S  E  D  Q  F  A  A  G  N  V  F  R  A  D  F  R  C  G  R  V  A
  S  R  P  R  E  P  R  R  H  R  R  G  V  *  R  S  L  S  C  R  E  R  V  P  R  R  V  P  L  R  P  R  R
A  I  E  P  T  R  T  A  S  T  P  P  R  S  M  K  F  P  Q  V  T  *  S  G  P  T  S  G  A  V  A  S  P  S
```

Codon chart for reference and verification:



Below is the Python script (dna2aa_mk3_allframes_mk2.py):

```python
# Translates DNA to AA mark 3, hopefully its faster and can translate 3 frames

# main feature: you get to visually pick the frame you want

from Bio import SeqIO
from Bio.Seq import Seq, reverse_complement, translate
from Bio.SeqRecord import SeqRecord

# storage of AA sequence in list (declaration)
aa_seq_data = []

# iterate and convert DNA to AA
for row in SeqIO.parse(r"HC-1A-Trinity.fa", "fasta"):
    frame_n_seq = [] # declare
    #print(row.id)
    frame_n_seq.append(translate(str(row.seq))) # frame 1
    frame_n_seq.append(translate(str(row.seq)[1:])) # frame 2
    frame_n_seq.append(translate(str(row.seq)[2:])) # frame 3
    frame_n_seq.append(str(translate(reverse_complement(row.seq)[2:]))) # frame -3
    frame_n_seq.append(str(translate(reverse_complement(row.seq)[1:]))) # frame -2
    frame_n_seq.append(str(translate(reverse_complement(row.seq)))) # frame -1

    # save data of all frames
    frames = [0, 1, 2, -1, -2, -3]

    for k in frames:
        frame_record = SeqRecord(
            Seq(frame_n_seq[k]),
            id = row.id,
            description = row.description,
        )
        aa_seq_data.append(frame_record)

# write all files here
SeqIO.write(aa_seq_data[0], "HC-1A-Trinity_aa_frame1.fa", "fasta")
SeqIO.write(aa_seq_data[1], "HC-1A-Trinity_aa_frame2.fa", "fasta")
SeqIO.write(aa_seq_data[2], "HC-1A-Trinity_aa_frame3.fa", "fasta")
SeqIO.write(aa_seq_data[3], "HC-1A-Trinity_aa_frame-1.fa", "fasta")
SeqIO.write(aa_seq_data[4], "HC-1A-Trinity_aa_frame-2.fa", "fasta")
SeqIO.write(aa_seq_data[5], "HC-1A-Trinity_aa_frame-3.fa", "fasta")
```

Copy and save it as a .py file (can do this in vscode). Recommended directory and file name: /Documents/python_scripts/dna2aa.py. Or you can just download this from GitHub: dna2aa_mk3_allframes_mk2.py

In this same directory, put the FASTA file of DNA sequences obtained from DNA sequencing.

Modify lines 12, 34-39 of the code such that it corresponds to the DNA FASTA file to be translated:

```python
for row in SeqIO.parse(r"<your_dna_file>.fa", "fasta"):
SeqIO.write(aa_seq_data[0], "<your_translated_file>_frame1.fa", "fasta")
SeqIO.write(aa_seq_data[1], "<your_translated_file>_frame2.fa", "fasta")
SeqIO.write(aa_seq_data[2], "<your_translated_file>_frame3.fa", "fasta")
SeqIO.write(aa_seq_data[3], "<your_translated_file>_frame-1.fa", "fasta")
SeqIO.write(aa_seq_data[4], "<your_translated_file>_frame-2.fa", "fasta")
SeqIO.write(aa_seq_data[5], "<your_translated_file>_frame-3.fa", "fasta")
```

Run the Python script by clicking Run ▷ at the top-right. After successful execution, the 6 FASTA files of translated will appear in the directory.

Motif Search

The next step after DNA translation will be to search for target motif(s) hits in the 6 translated amino acid sequence files.

Writing a motif pattern

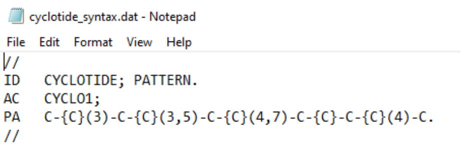
Before motif search can be performed using ps_scan, the motif of interest needs to be written into a PROSITE pattern format.

For example, the motif of interest is a Cyclotide motif: CxxxCxxx--Cxxxx---CxCxxxxC, where x is any amino acid excluding cysteine, hyphens indicate possible presence of x, in other words, this represents loop length variability.

The above motif can be converted into a PROSITE motif pattern as follows:
C-{C}(3)-C-{C}(3,5)-C-{C}(4,7)-C-{C}-C-{C}(4)-C.

Every C is a cysteine, every {C} is an amino acid that is not cysteine, digit(s) appended next to {C} indicates the length of {C}. For instance {C}(4,7) means 4 to 7 amino acids that are not cysteine. The period terminating the syntax is important.

Save the PROSITE motif pattern into a motif pattern file in this format:



```
//
ID   CYCLOTIDE; PATTERN.
AC   CYCL01;
PA   C-{C}(3)-C-{C}(3,5)-C-{C}(4,7)-C-{C}-C-{C}(4)-C.
//
```

Save this file as a .dat file into the ps_scan directory. Recommended directory: /Documents/ps_scan/cyclotide_syntax.dat. Or you can download it from GitHub: cyclotide_syntax.dat

Running the motif search

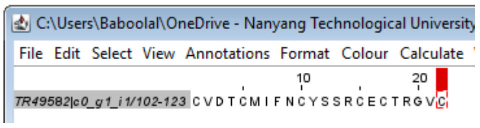
Copy the 6 translated FASTA files into the ps_scan directory then start Perl (command line). Navigate to the ps_scan directory by executing this line:

```
cd C:\Users\your_name\Documents\ps_scan
```

Execute motif search on your FASTA file (one at a time) with just one line:

```
ps_scan.pl -d cyclotide_syntax.dat translated_frame_1.fa -o fasta >result_frame_1.fa
```

Result file will be generated as result_frame_1.fa in this case. If there are results, it will be like this:


C:\Users\Baboolal\OneDrive - Nanyang Technological University
File Edit Select View Annotations Format Colour Calculate
10
20

```
TR49582|c0_g1_i1/102-123 CVDTCMIFNCYSSRCECTRGVC
```

Repeat this line for the remaining 5 translated files, renaming the input and output files correspondingly.

Collating results into Excel (optional) (get_full_sequence.py)

If you need your motif search results to be collated side-by-side with the corresponding full sequence, the following Python script may be used (get_full_sequence.py):

```
# retrieve full sequence of protein
# input: fasta file of short sequences
# output1: tab-delimited file of: id <tab> short seq <tab> full seq
# output2: fasta file of the full sequences

from Bio import SeqIO
from Bio.SeqRecord import SeqRecord

# read in the full sequences file into a dict (memory efficient)
full_record_dict = SeqIO.index("HC-1A-Trinity_aa_frame3.fa", "fasta")

# prep tab-delimited txt file to write
output_file = open("HC_cyclo_frame3_table.txt", "w")
fasta_file_write_entries = []

# real deal
for record in SeqIO.parse("HC_cyclo_frame3.fa", "fasta"):
    print(record.id, record.seq, full_record_dict[record.id.split("/")[0]].seq)
    output_file.write(str(record.id) + "\t" + str(record.seq) \
        + "\t" + str(full_record_dict[record.id.split("/")[0]].seq) + "\n")
    fasta_entry = SeqRecord(
        full_record_dict[record.id.split("/")[0]].seq,
        id = full_record_dict[record.id.split("/")[0]].id
    )
    fasta_file_write_entries.append(fasta_entry)

output_file.close()
SeqIO.write(fasta_file_write_entries, "HC_cyclo_frame3_full.fa", "fasta")
```

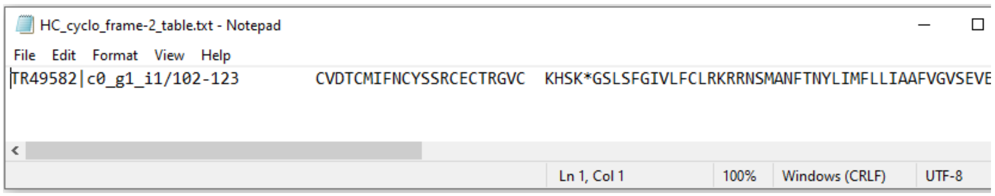
Copy the above script into vscode and save it into a .py file. Recommended directory: /Documents/python_scripts/get_full_sequence.py. Or you can download it from GitHub: get_full_sequence.py. The following files need to be in the same directory: motif search results FASTA file, protein sequences FASTA files. Make sure they are of the same reading frame, as shown in above code listing.

This script generates 2 files: a tab-delimited text file with target sequence and full sequence, and a FASTA file of your full sequence (may be ignored if not needed).

Modify lines 10 (input), 13 (output), 17 (input) and 28 (output) accordingly to your inputs and outputs:

```
full_record_dict = SeqIO.index("translated_frame3.fa", "fasta")
output_file = open("output1_frame3_table.txt", "w")
for record in SeqIO.parse("motif_search_results_frame3.fa", "fasta"):
SeqIO.write(fasta_file_write_entries, "output2_frame3_full.fa", "fasta")
```

The generated .txt file can be copy-pasted into Excel nicely:


HC_cyclo_frame-2_table.txt - Notepad
File Edit Format View Help

```
TR49582|c0_g1_i1/102-123        CVDTCMIFNCYSSRCECTRGVC   KHSK*GSLSFGIVLFCLRKRRNSMANFTNYLIMFLLIAAFVGVSEVE
```

Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

End of protocol. Protocol is property of Prof. James P Tam Lab.