BT-3172: Special Topics in Bioinformatics: Practical computing for bioinformatics
Lab 4: Use of Python modules, packages and advanced programming in bioinformatics.

W.M. Ayesha Sanahari | s13722 | 2017s16470

In this practical, you will learn how to import and use built-in or third-party Python modules and packages to solve biological questions. Moreover, you will implement the majority voting algorithm on *Arabidopsis thaliana* DREB2A containing protein network to predict protein candidates for stress tolerance.

For this practical, again, you will be working with several genes and proteins involved with the plant stress response pathways. Here, you will be expanding from ABA-independent (e.g., DREB proteins) to ABA-dependent pathway.

After using PyCharm to write your scripts, **copy the codes to the space below the questions**. Also, submit the Python files separately so they can be tested. Use the following format to name the script: YourIndexNo_PrimaryQuestion.py (submit two programs for the two questions)

1) Using Biopython and re modules. *(55 marks)*
   I. Search the NCBI RefSeq **Gene** database for *Arabidopsis thaliana* DREB2A gene. Write its gene ID below. Locate its genomic sequence record from the GenBank. Write its accession ID with the version number. What is the length of the gene sequence according to the GenBank record? Download the gene sequence in FASTA format and name the file as "ATdreb2a.fasta".

      gene ID – 830424
      Ac. No. - NC_003076.8
      length of the gene sequence - 1908 bp

   II. Use the Biopython module to perform the following tasks.
      Load the downloaded FASTA file as a sequence record object and print the following attributes of the record: sequence ID, description, sequence, and sequence length.

   III. Using the Biopython module, run the web-based nucleotide blast program on the ATDREB2A sequence. Refer to the Biopython manual for instructions. Save the blast output as "dreb2a_blast.xml".

   IV. Parsing the blast output file. Open the xml file saved during the previous question and parse through the blast hits. Define an E-value threshold of 0.05 and only select the hits that are below the threshold. Print the following attributes of each blast hit selected based on the above criteria: blast hit title, alignment length, E-value, score, hit/subject sequence, and hit sequence length.

   V. Now, modify the above code to identify the blast hits with the ABRE cis-acting ABA-dependent transcription factor binding element using Python regular expressions. First, define a search string to search for the ABRE element (ACGTG/TC). Then check each blast hit sequence for the presence of the ABRE element and print the detected sequence fragment (e.g., ACGTGC or ACGTTC) along with the sequence location of each finding.

Please note that the ABRE element can be found in multiple locations in the same sequence. Further, print the number of blast hits with ABRE element present in the sequence and write that number below.

number of blast hits with ABRE element: 31

```
"""
Author : Ayesha Sanahari
date : 15/Jan/2021
Use built-in or third-party Python modules and packages (Biopython and re module)
to solve biological questions

Input: ATdreb2a.fasta file containing DREB2A gene sequence
Output: blast result as "dreb2a_blast.xml"
        hits that are below the threshold E value 0.05
        number of blast hits with ABRE element
"""
from Bio import SeqIO
import re

# Load the downloaded FASTA file as a sequence record
for seq_record in SeqIO.parse("ATdreb2a.fasta", "fasta"):
    print("sequence ID :", seq_record.id)
    print("Description :", seq_record.description)
    print(repr(seq_record.seq))
    print("Seq length: ", len(seq_record))

from Bio.Blast import NCBIWWW

record = SeqIO.read("ATdreb2a.fasta", format="fasta")
result_handle = NCBIWWW.qblast("blastn", "nt", record.format("fasta"))

print('aye')

# Write the BLAST result into an xml file
with open("dreb2a_blast.xml", "w") as out_handle:
    out_handle.write(result_handle.read())

result_handle.close()
result_handle = open("dreb2a_blast.xml")

from Bio.Blast import NCBIXML
blast_record = NCBIXML.read(result_handle)

#Define a counter for count the number of blast hits with ABRE element
hit_count =0
E_VALUE_THRESH = 0.05
```

```python
# check for each blast hit sequence for the presence of the ABRE element
for alignment in blast_record.alignments:
    for hsp in alignment.hsps:
        if hsp.expect < E_VALUE_THRESH:

            print("\n ****Alignment****")
            print("blast hit title:", alignment.title)
            print("alignment length:", alignment.length)
            print("E value:", hsp.expect)
            print("score:", hsp.score)
            print("hit/subject sequence:", hsp.sbjct)
            print("hit sequence length:", len(hsp.sbjct))

            print(hsp.query[0:75] + "...")
            print(hsp.match[0:75] + "...")
            print(hsp.sbjct[0:75] + "...")

            # Finding the ABRE elements present in the sequence
            # print the detected sequence fragment (e.g., ACGTGC or ACGTTC) along with the sequence location of each finding.
            # (Please note that the ABRE element can be found in multiple locations in the same sequence) So used re.finditer
            pattern = re.compile("ACGT[GT]C")
            matches = re.finditer(pattern, hsp.sbjct)

            ay = "ABRE elements : Not present"
            for item in matches:
                hit_count += 1
                ay= "ABRE elements : " , item.group() , item.span()

            print(ay)
print("\n number of blast hits with ABRE element: ", hit_count)
```

2) Implementing the majority voting network-based candidate protein prediction algorithm. *(45 marks)*
    I. Search for the *Arabidopsis thaliana* DREB2A protein in the STRING protein-protein interaction database. Write its STRING ID below. Increase the maximum interactors to 500 and download the interactions in tabular format.

    STRING ID - 3702.AT5G05410.1

    II. Write the steps of an algorithm to predict the majority voting score of unknown proteins for a given function in a network. Assume that a list of known proteins annotated to the particular function is given as a text file. This should output/print the list of unknown proteins with the predicted majority voting score.

Predict the majority voting scores for all the unknown protein members for the stress tolerance biological process of the ATDREB2A network

Input:

- text file containing a list of known *Arabidopsis thaliana* proteins annotated to the particular function – "AT_stress_proteins.txt"
- unknown protein members for the stress tolerance biological process of the ATDREB2A network ("string_interactions_1.tsv" file in tabular format)
-

Output: list of unknown proteins with the predicted majority voting score

Read the file containing the list of known stress proteins
Extract the relevant information & make a list called stress_proteins
Read the tsv file which contains the protein-protein interactions of unknown proteins
Extract the relavant information & save it into a list called tsv_data
Convert the names of proteins into UPPERCASE
Create an interaction graph of the unknown proteins from the tsv_data

Seperate the proteins in the tsv_data whose functions aren't already know (which are not belong to stress_proteins) as unknown_proteins
For each of these unknown proteins, determine how many of their neighbors are known proteins
Assign them a score based on the number of known neighbours. (Predict majority voting scores)
Sort the unknown proteins in descending order based on the scores, with proteins with high scores at the top.
Write the ordered list to an output file.

III. Implement the above algorithm in Python to predict the majority voting scores for all the unknown protein members for the stress tolerance biological process of the ATDREB2A network you downloaded in question (I). A data file containing known *Arabidopsis thaliana* proteins for stress tolerance is provided. ("AT_stress_proteins.txt). Please make sure you complete the following tasks.

i. Print and write the degree of the ATDREB2A protein and the number of unknown proteins in the network for stress tolerance below.

No. of unknown proteins in the network for stress tolerance: 55
degree of the ATDREB2A protein: 149

ii. After predicting the majority voting scores, you should sort them in descending order based on the scores, with proteins with high scores at the top. Then, write the ordered list to an output file.
Hint: you can use OrderedDict submodule from the Collections Python package for sorting a dictionary based on values.

iii. Pay a special attention to the names of the proteins. During the counting step, you have to match the protein names from the network and the input list. Pay a special attention to the sentence case of the protein names.
iv. You can use Python set operators to perform set matching, difference and removal of duplicates from a list. Please refer to a Python tutorial for more information.

```
"""
Author : Ayesha Sanahari
date : 15/Jan/2020
Predict the majority voting scores for all the unknown protein members
for the stress tolerance biological process of the ATDREB2A network

Input:
• text file containing a list of known Arabidopsis thaliana proteins
annotated to the particular function – "AT_stress_proteins.txt"
• unknown protein members for the stress tolerance biological process of the
ATDREB2A network ("string_interactions_1.tsv" file in tabular format)

Output: list of unknown proteins with the predicted majority voting score
"""
import networkx as nx;

# Read the file containing the list of known stress proteins
stress_proteins=[]
with open('AT_stress_proteins.txt', 'r') as file:
   for line in file:
      if '\n' != line :
         stress_proteins_all = line.strip().split('\t')
         # Extract the relevant information & make a list called stress_proteins
         stress_proteins += stress_proteins_all[1:2]


# Read the tsv file which contains the protein-protein interactions of unknown proteins
tsv_data=[]
with open('string_interactions_1.tsv', 'r') as file:
   for line in file:
      if '\n' != line and 'node1' not in line:
         tsv_data_all = line.strip().split('\t')
         # Extract the relavant information & save it into a list called tsv_data
         tsv_data += tsv_data_all[0:2]

# Define an empty dictionary to store interactions
tsv_dict={}
# define the key of dictionary
num=1
#open/read file and store in a variable
with open('string_interactions_1.tsv', 'r') as file:
   for line in file:
      if '\n' != line and 'node1' not in line:
```

```
        interaction = line.strip().split('\t')
        tsv_dict[num]= interaction[0:2]
        num +=1


# Convert the names of proteins into  UPPERCASE
Capilalize_tsv_dict = {key: [x.upper() for x in tsv_dict[key]] for key in tsv_dict}


# Create an interaction graph of the unknown proteins from the tsv_dict
G = nx.Graph()
for key, values in Capilalize_tsv_dict.items():
  G.add_edge(values[0], values[1])

# Convert the names of proteins into  UPPERCASE
unique_tsv_data = set(x.upper() for x in tsv_data)
unique_stress_proteins = set(x.upper() for x in stress_proteins)


# Seperate the proteins in the tsv_data whose functions aren't already know (which are not
belong to stress_proteins ) as unknown_proteins
unknown_proteins = unique_tsv_data - unique_stress_proteins


# For each of these unknown proteins, determine how many of their neighbors are known
proteins
unknown_dict ={}
score =0
for protein in unknown_proteins:
  for neighbour in G.neighbors(protein):
    # Assign them a score based on the number of known neighbours. ( Predict majority
voting scores)
    if neighbour in unique_stress_proteins:
      score += 1
      unknown_dict[protein] = score


# Sort the unknown proteins in descending order based on the scores, with proteins with high
scores at the top.
from collections import OrderedDict
import operator
desending = sorted(unknown_dict.items(),key=operator.itemgetter(1), reverse=True)


# Write the ordered list to an output file.
ordered_list = str(desending)
with open('Ordered_list.txt', 'w') as output:
    output = output.write(ordered_list)

print("Descending ordered list:", ordered_list)
```

```
print("No. of unknown proteins in the network for stress tolerance: "
,len(unknown_proteins))
print("degree of the ATDREB2A protein: ", G.degree['DREB2A'])
```