**S13674**

**D.P.B.Hendalage**

**BT-3172: Special Topics in Bioinformatics: Practical computing for bioinformatics**

**Lab 4: Use of Python modules, packages and advanced programming in bioinformatics.**

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)
   1. 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”.

GeneID:[830424](https://www.ncbi.nlm.nih.gov/gene/830424)

Accession ID:[NC\_003076](https://www.ncbi.nlm.nih.gov/nuccore/NC_003076).8

Length: 1908 bp

* 1. 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.

*#Question01 part01***from** Bio **import** SeqIO  
**for** seq\_record **in** SeqIO.parse(**"ATdreb2a.fasta.fasta"**, **"fasta"**):  
 print(seq\_record.id)  
 print(seq\_record.description)  
 print(seq\_record.seq)  
 print(len(seq\_record))

* 1. 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”.

**from** Bio.Blast **import** NCBIWWW  
**from** Bio **import** SeqIO  
record = SeqIO.read(**"ATdreb2a.fasta.fasta"**, format=**"fasta"**)  
result\_handle = NCBIWWW.qblast(**"blastn"**, **"nt"**, record.seq)  
  
**with** open(**"dreb2a\_blast.xml"**, **"w"**) **as** out\_handle:  
 out\_handle.write(result\_handle.read())  
 result\_handle.close()  
result\_handle = open(**"dreb2a\_blast.xml"**)

* 1. 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.

**from** Bio.Blast **import** NCBIWWW  
**from** Bio **import** SeqIO  
record = SeqIO.read(**"ATdreb2a.fasta.fasta"**, format=**"fasta"**)  
result\_handle = NCBIWWW.qblast(**"blastn"**, **"nt"**, record.seq)  
**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)  
E\_VALUE\_THRESH = 0.05  
**for** alignment **in** blast\_record.alignments:  
 **for** hsp **in** alignment.hsps:  
 **if** hsp.expect < E\_VALUE\_THRESH:  
 print(**"\*\*\*\*Alignment\*\*\*\*"**)  
 print(**"sequence:"**, alignment.hit\_def)  
 print(**"length:"**, alignment.length)  
 print(**"e value:"**, hsp.expect)  
 print(**"Score"**, hsp.score)  
 print(**"Hit sequence:"**, hsp.sbjct)  
 print(**"Hit sequence length:"**, hsp.align\_length)

* 1. 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.

**import** re  
**from** Bio.Blast **import** NCBIXML  
blast\_record = NCBIXML.read(result\_handle)  
E\_VALUE\_THRESH = 0.05  
pattern=**"ACGT[GT]C"**counter=0  
**for** alignment **in** blast\_record.alignments:  
 **for** hsp **in** alignment.hsps:  
 **if** hsp.expect < E\_VALUE\_THRESH:  
 string=hsp.sbjct  
 mo=re.finditer(pattern,string)  
 **for** item **in** mo:  
 counter += 1  
 print(**"\*\*\*\*Alignment\*\*\*\*"**)  
 print(**"sequence:"**, alignment.hit\_def)  
 print(**"length:"**, alignment.length)  
 print(**"e value:"**, hsp.expect)  
 print(**"Score"**, hsp.score)  
 print(**"Hit sequence:"**, hsp.sbjct)  
 print(item.group())  
 print(item.span())  
 print(**"Hit sequence length:"**, hsp.align\_length)  
print(**"the number of blast hits with ABRE element present in the sequence"**,counter)

1. Implementing the majority voting network-based candidate protein prediction algorithm. (45 marks)
   1. 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.

[AT5G05410.1](https://string-db.org/network/3702.AT5G05410.1)

[3702.AT5G05410.1](https://string-db.org/network/3702.AT5G05410.1)

* 1. 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.

Read the text file and store the protein name as a known protein list

Read the tsv file and make a list and graph of interactions using network

Find the unknown proteins(make a list) using interaction list and known protein list

To all items in unknown list find the neighbors in the known protein list and count it

Make a dictionary with key is unknown proteins and count is assign to value

Order the dictionary from the values

Dictionary is converted to list

* 1. 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.
     1. Print and write the degree of the ATDREB2A protein and the number of unknown proteins in the network for stress tolerance below.
     2. 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.*

* + 1. 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.
    2. 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:Bhagya Hendalage  
Date:16/01/2021  
input:protein\_interactions\_pr4.tsv and AT\_stress\_proteins.txt  
output: degree of DREB2A  
 the number of unknown proteins in the network for stress tolerance  
 the ordered list of predicted the majority voting scores with unknown protein name  
to predict the majority voting score of unknown proteins for a given function in a network  
a list of known proteins annotated to the particular function  
output the list of unknown proteins with the predicted majority voting score.  
'''***import** networkx **as** nx  
proteins=[]  
known\_prot=[]  
  
*# Create an empty graph*g = nx.Graph()  
  
*# read interactions and store it in variable***with** open (**'protein\_interactions\_pr4.tsv'** ,**'r'** ) **as** file:  
 **for** lines **in** file:  
 **if '#' not in** lines:  
 lines=lines.strip().split(**'\t'**)  
 proteins.append(lines[0].upper())  
 proteins.append(lines[1].upper())  
 g.add\_edge(lines[0].upper(), lines[1].upper())  
  
print(**'The degree of the ATDREB2A protein :'**,g.degree(**'DREB2A'**))  
*#print(g.degree)  
#print(proteins)  
  
# Read the proteins and store it in variable***with** open (**'AT\_stress\_proteins.txt'** ,**'r'** ) **as** file:  
 **for** lines **in** file:  
 lines=lines.strip().split(**'\t'**)  
 known\_prot.append(lines[1].upper())  
*#print(known\_prot)  
  
# Find the unknown proteins*unkown\_prot=[]  
**for** protein **in** proteins:  
 *# using set, find the proteins which are in the network but not know the function* unkown\_prot= list(set(proteins) - set(known\_prot))  
*#print(unkown\_prot)  
  
#create a dictionary  
# key is function unknown proteins value is how many neighbors they have amoung the function known proteins*prot\_scoring = {}  
key = **''***# find neighbors of unknown proteins***for** un\_p **in** unkown\_prot:  
 key = un\_p  
 neighbors\_list = (list(nx.all\_neighbors(g, un\_p)))  
 *# get proteins which are function is known* neighbors = list(set(neighbors\_list) & set(known\_prot))  
 *# count the number of items in list, asign as the value of the dictionary* prot\_scoring[key] = len(neighbors)  
print(**'The number of unknown proteins in the network for stress tolerance :'**,len(prot\_scoring))  
*#print(scoring)***import** operator  
*# sort the dictionary key according to the descending order of the value*sorted\_dict =dict(sorted(prot\_scoring.items(),key=operator.itemgetter(1),reverse=**True**))  
*# convert the dictionary to a list*print(**'The ordered list of predicted the majority voting scores with unknown protein name :'**)  
print(list(sorted\_dict.items()))