BT-3172: Special Topics in Bioinformatics: Practical computing for bioinformatics
Lab 7: Introduction to the Unix shell for biological applications.

W.M. Ayesha Sanahari | s13722 | 2017s16470

In this practical, you will learn how to use basic Unix shell commands to solve simple biological questions.

Create 2 folders for the two questions and name them in "Your_index_question_no" format. Use the Unix shell/command line to implement your commands. Follow the question specific instructions and save the necessary Python and shell script files in their folders. Also, make sure all the outputs specified in the questions are in the folders. Finally, compress the folders and upload them to the LMS. **You must write the Unix shell commands and Python scripts in the space below each question for evaluation.**

1) Processing multiple FASTA files using the Unix shell.
   In this problem, you will be working with APETALA2/ETHYLENE RESPONSIVE FACTOR (AP2/ERF) family transcription factors (Zhouli, et al., 2019) which contain the AP2/ERF DNA-binding domain.
   .

   I. Write a Python script to retrieve the GenBank records for the following accessions: "AAK43967.1","AED90870.1","NP_567720.1", and "AAK59861.1", and save their amino acid sequences in **separate** FASTA files. The FASTA files can be named by their respective accession number. You can use the Biopython module when writing the script. Save the script as "Your_index_multi_fasta.py".

```
from Bio import Entrez

seq = ["AAK43967.1", "AED90870.1", "NP_567720.1", "AAK59861.1"]
Entrez.email = "A.N.Other@example.com"
for sequences in seq:
   handle = Entrez.efetch(db="Protein", id=sequences, rettype="fasta", retmode ="text")
   file = str(sequences) + ".fasta"
   with open(file, 'w') as newfile:
      newfile.writelines(handle)
```

   II. Use the grep command to search for the "WGKWVAEIR" amino acid sequence fragment from the AP2/ERF domain in above retrieved sequences and output the FASTA headers of the sequences which contain the domain fragment in a separate file called "AP2_basic_headers.txt". Write the headers in the space below.

```
>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]

#!/bin/bash

files=$(ls *.fasta)
for i in $files
        do
```

```
        pattern=$(grep -B 20 "WGKWVAEIR" $i)
        echo "$pattern" >> pattern_contain.txt
    done
header=$(grep -e ">" pattern_contain.txt)
echo "$header" >> AP2_basic_headers.txt
```

III. Now, modify the above search term to include a REGEX expression to search for "WGKWV/AAEIR" amino acid fragment in the sequences and output the FASTA headers of the sequences which contain the domain fragment in a separate file called "AP2_advanced_headers.txt". Write the headers in the space below.

>AAK43967.1 putative AP2 domain-containing protein [Arabidopsis thaliana]
>AED90870.1 DRE-binding protein 2A [Arabidopsis thaliana]

```
#!/bin/bash

files=$(ls *.fasta)
for i in $files
    do
        pattern1=$(grep -E -r -l "WGKWV|AAEIR" $i)
        echo "$pattern" >> pattern_contain1.txt
    done
header=$(grep -e ">" pattern_contain1.txt)
echo "$header" >> AP2_advanced_headers.txt
```

IV. Write a shell command to concatenate the FASTA files downloaded in question (I) and count the number of FASTA files in the concatenated output. Write the count below.
Hint: Use the cat, grep, pipe and wc commands appropriately.

```
#!/bin/bash
num=$(cat *.fasta | grep -o '>' | wc -l )
echo "total number of files: $num"
```

total number of files: 4

V. Write a shell command to concatenate the FASTA files downloaded in question (I) into a single FASTA file called "combined.fasta".

cat *.fasta >> combined.fasta

2) Using the pipe command to write a simple bioinformatics pipeline. You can use Biopython package when writing the Python scripts. Because you are writing a bioinformatics pipeline, Python scripts must be properly commented and an introduction should be given for each script.

I. First, write a Python script (cds_seq_retrieve.py) to retrieve the GenBank record for an accession number (with version) of a protein coding DNA sequence or a reverse-transcribed mRNA complement given by the user and save the sequence in FASTA format (cds_seq.fasta). The script must prompt the user to input the accession number (with version).

```
#!/usr/bin/python3

"""
Author: Ayesha Sanahari
Date: 22/Jan/2021
Retrieve the GenBank record for an accession number (with version)
of a protein coding DNA sequence or a reverse-transcribed mRNA complement
given by the user and save the sequence in FASTA format (cds_seq.fasta).

Input: Accession number (with version)
Output: cds_seq.fasta file
"""
from Bio import Entrez

#getting Accesion number
AcNum = input("enter the accession number with version")
#retrive fasta file
Entrez.email = "Name@example.org"
handle = Entrez.efetch(db="Nucleotide", id=AcNum, rettype="fasta", retmode="text")

#write sequence into a new fasta file
with open("cds_seq.fasta",'w') as newfile :
    newfile.writelines(handle)
```

II. Then, write a Python script (transcribe.py) to transcribe the above sequence in the FASTA file and save the transcribed mRNA sequence in another FASTA file (mRNA_seq.fasta). The FASTA header must contain the added word “transcribed” at the end. The program should read the “cds_seq.fasta” file.

```
#!/usr/bin/python3
"""
Author: Ayesha Sanahari
Date: 22/Jan/2021
Transcribe the given sequences in the FASTA file and save the
transcribed mRNA sequence in another FASTA file.
The FASTA header must contain the added word “transcribed” at the end.

Input: “cds_seq.fasta” file.
Output: mRNA_seq.fasta file
"""
from Bio import SeqIO

for record in SeqIO.parse( "cds_seq.fasta" , 'fasta'):
  seq=record.seq
  seq=seq.transcribe()
  title= record.description
  header=str(title)+"_transcribed"
```

```
# write transcribed sequence into a new fasta file
newfile = open( "mRNA_seq.fasta" , 'w')
newfile.write(">"+str(header)+"\n"+str(seq))

newfile.close()
```

III. Then, write a Python script (translate.py) to translate the above sequence in the FASTA file and save the translated amino acid sequence in another FASTA file (aa_seq.FASTA). The FASTA header must contain the added word "translated" at the end. The script must read the "mRNA_seq.fasta" file.
Hint: for this example, it is not needed to start the amino acid sequence with methionine. Simply translate the mRNA sequence using translate() function in Biopython.

```
#!/usr/bin/python3
"""
Author: Ayesha Sanahari
Date: 22/Jan/2021
Translate the given sequence in the FASTA file and save the translated amino acid
sequence in another FASTA file.
The FASTA header must contain the added word "translated" at the end.

Input: "mRNA_seq.fasta" file.
Output: aa_seq..fasta file
"""

from Bio import SeqIO
for record in SeqIO.parse("mRNA_seq.fasta", "fasta"):
  seq=record.seq
  # To perform biopython translate function perfectly, added Ns until the final sequence is a
multiple of 3
  seq = seq + (len(seq) % 3 - 1) * 'N'
  aa=seq.translate()
  title=str(record.description)
  # remove the transcribed word at the end of header
  header=str(title[:-12])+"_translated"

with open("aa_seq.fasta", 'w') as newfile:
  newfile.write(">"+str(header)+"\n"+str(aa))
```

IV. Finally, write a Python script (aa_seq_analyze.py) to analyze the aa_seq.fasta file and calculate the length, molecular weight, alanine percentage, and glycine percentage of the sequence. Save the calculated parameters in a new text file called "aa_stats.txt". The script must read the "aa_seq.fasta" file as the input.
Hint: You can use Biopython for above calculations. Find out the specific sub module for protein sequence analysis.

```
#!/usr/bin/python3
"""
Author: Ayesha Sanahari
Date: 22/Jan/2021
Analyze the aa_seq.fasta file and calculate the length, molecular weight, alanine percentage,
and glycine percentage of the sequence. Save the calculated parameters in a new text file
called "aa_stats.txt".

Input: aa_seq.fasta file
Output: "aa_stats.txt" file
"""

from Bio import SeqIO
from Bio.SeqUtils.ProtParam import ProteinAnalysis

for seq_record in SeqIO.parse("aa_seq.fasta", "fasta"):
  myseq=str(seq_record.seq)

  #remove unwanted letters for make protein unambigous else it gives a error
  protein=myseq.replace('*','')
  protein=protein.replace('X','')
new = ProteinAnalysis(str(protein))
header=seq_record.id
weight = new.molecular_weight()
newfile=open("aa_stats.txt" , 'w')
newfile.write(str(header)+"\n"+"length of amino acid :"+str(len(seq_record))+"\n"
    +"molecular weight of amino acid :"+str(weight)+"\n"+"alanine percentage
:"+str(new.get_amino_acids_percent()['A'])+"\n"
    +"glycine percentage :"+str(new.get_amino_acids_percent()['G']))
newfile.close()
```

V. Now, using a shell script, build a simple pipeline to combine the above 4 scripts in the given order. Further, using the same shell script, create two folders: intermediate_files and output. Move the "cds_seq.fasta", "mRNA_seq.fasta", and "aa_seq.fasta " into the intermediate_files folder and the final output file: "aa_stats.txt" into the output folder. Save the shell script as "your_index_bi_pipeline.sh". Use the "NM_000188.3" accession as the input to the pipeline, which is for human hexokinase 1 gene. Write the amino acid statistics you calculated below, which would be in the output folder.

```
#!/bin/bash

python3 cds_seq_retrieve.py
python3 transcribe.py
python3 translate.py
python3 aa_seq_analyze.py

mkdir -p ./intermediate_files
```

```
mkdir -p ./output

mv cds_seq.fasta ./intermediate_files
mv mRNA_seq.fasta ./intermediate_files
mv aa_seq.fasta ./intermediate_files
mv aa_stats.txt ./output/aa_stats.txt
```

amino acid statistics:

```
NM_000188.3
length of amino acid :1201
molecular weight of amino acid :128469.92550000083
alanine percentage :0.06538796861377506
glycine percentage :0.06800348735832606
```

References

- Xie, Zhouli, et al. "AP2/ERF transcription factor regulatory networks in hormone and abiotic stress responses in Arabidopsis." *Frontiers in plant science* 10 (2019): 228.