**D.P.B.Hendalage**

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

**Lab 7: Introduction to the Unix shell for biological applications.**

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.

.

* 1. 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  
sequences = [**"AAK43967.1"**,**"AED90870.1"**,**"NP\_567720.1"**,**"AAK59861.1"**]  
Entrez.email = **"Your.Name.Here@example.org"  
for** id **in** sequences:  
 handle = Entrez.efetch(db=**"protein"**, id=id, rettype=**"fasta"**, retmode=**"text"**)  
 file1 = str(id) + **".fasta"  
 with** open(file1, **'w'**) **as** file:  
 file.writelines(handle)  
handle.close()

* 1. 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.fasta

AED90870.1.fasta

#!/bin/bash

files=$(ls \*.fasta)

for file in $files

do

seq=$(grep -r -l "WGKWVAEIR" $file)

echo "$seq" >> AP2\_basic\_headers.txt

done

* 1. 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.fasta

AED90870.1.fasta

#!/bin/bash

files=$(ls \*.fasta)

for i in $files

do

pat=$(grep -E -r -l "WGKWV|AAEIR" $i)

echo "$pat" >> des\_contain.txt

done

header=$(grep -e ">" des\_contain.txt)

echo "$header" >> AP2\_advanced\_headers.txt

* 1. 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

* 1. Write a shell command to concatenate the FASTA files downloaded in question (I) into a single FASTA file called “combined.fasta”.

#!/bin/bash

cat \*.fasta >> combined.fasta

1. 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.
   1. 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).

*# import the packages***from** Bio **import** Entrez  
  
Entrez.email = **"Your.Name.Here@example.org"***# to get the accession number with version*id = input(**"Enter the acccession number(with version): "**)  
*# retrieve the fasta file from database*handle = Entrez.efetch(db=**"Nucleotide"**, id=id, rettype=**"fasta"**, retmode=**"text"**)  
*# write file as cds\_seq.fasta***with** open(**'cds\_seq.fasta'**, **'w'**) **as** file:  
 file.writelines(handle)  
handle.close()

* 1. 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  
# import packages***from** Bio **import** SeqIO  
*#get the file***for** record **in** SeqIO.parse(**'cds\_seq.fasta'** , **'fasta'**):  
 *# take the header part* accession = record.id  
 *# take the sequence part* seq = record.seq  
 *# transcribe the sequence* sequence=seq.transcribe()  
 *# add the 'transcribed' part to the end of the header* header=str(accession)+**'\_transcribed'***# write the mRNA\_seq.fasta file*file = open( **'mRNA\_seq.fasta'** , **'w'**)  
file.write(**'>'**+str(header)+**'\n'**+str(sequence))  
file.close()

* 1. 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  
# import packages***from** Bio **import** SeqIO  
*# take the file***for** record **in** SeqIO.parse(**"mRNA\_seq.fasta"**, **"fasta"**):  
 sequence=record.seq  
 seq = sequence + (len(sequence)%3-1)\***'N'** amino\_acids=seq.translate()  
 header=str(record.id)  
 header\_parts=header.split(**"\_"**)  
 header1=str(header\_parts[0])+str(header\_parts[1])+**"\_translated\n"**file = open( **"aa\_seq.fasta"** , **'w'**)  
file.write(**">"**+str(header1)+str(amino\_acids))  
file.close()

* 1. 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  
# import the packages***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* prot = myseq.replace(**'\*'**, **''**)  
 prot = prot.replace(**'X'**, **''**)  
new = ProteinAnalysis(str(prot))  
header = seq\_record.id  
weight = new.molecular\_weight()  
file = open(**"aa\_stats.txt"**, **'w'**)  
file.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'**]))  
file.close()

* 1. 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

NM000188.3\_translated

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.