Practical 1
12 Feb 2024
Questions

Instructions
1. Upload your answers (along with codes) in moodle with naming "roll no._p1".
2. Do not copy codes, it will be checked for plagiarism

1. Download the EMBOSS package (http://emboss.sourceforge.net/download/) and copy to your Windows system. In case of **Linux** use the command: sudo apt-get install jemboss. For Mac users, use the online tool links given in the instructions doc.

2. Using EMBOSS, find the complementary strand for the sequence:
`CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG`

3. Write a program to find the complementary strand for the sequence given in Q2.

4. Using EMBOSS, find the protein sequence for
`GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAG`
`CGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCATGAGCCTGAATTTGGT`
`CCATGCAGAAGAAAACTGGATGGG`
(ii) Identify the reading frame equivalent to the following sequence.
`PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV`

5. Write a program to translate the given DNA sequence (refer Q4) to protein sequence.

6. Write a code to search for the following strings 'AAG', 'GTC', 'GAG, 'ACTA', and 'ATAT' in the DNA sequence provided in Q4. The program should print the total number of matches for the each of the given strings and the start positions of the matches

Example: Enter the string: ACTA
Total match:1
Position of match: 88

7. Familiarize with other applications in EMBOSS. For example, melting temperature, bending, curvature etc.

8. Write a program to compute the average base stacking energy for the sequence given in Q2 (AA: -4; AT: -7; AC: -5; AG: -11; TA: -7; TT: -2; TC: -3; TG: -4; CA: -9; CT: -5; CC: -6; CG: -7; GA: -9; GT: -6; GC: -4; GG: -11).

9. Compute the average melting temperature of the following sequences using Seq2Feature tool (https://www.iitm.ac.in/bioinfo/SBFE/index.html) and comment on the results (Enter one sequence at a time in fasta format)

(i) ATATATATAT ii) GCGCGCGCGC

10. Calculate the AT and GC content of the sequence AAATGGCCCTAA using Seq2Feature tool

Deadline: 18-02-2024

BT 3040: BIOINFORMATICS

Assignment 1

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) Download the EMBOSS package (http://emboss.sourceforge.net/download/) and copy to your Windows system. In case of Linux use the command: sudo apt-get install jemboss. For Mac users, use the online tool links given in the instructions doc.

Installed Emboss following the instructions, on Windows 11 PC. Created Shortcuts in Desktop

Q2) Using EMBOSS, find the complementary strand for the sequence: CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG

Used Reverse SQ to find the complementary strand. We can get the non-reversed output by deselecting the respective box under 'Advance Options'

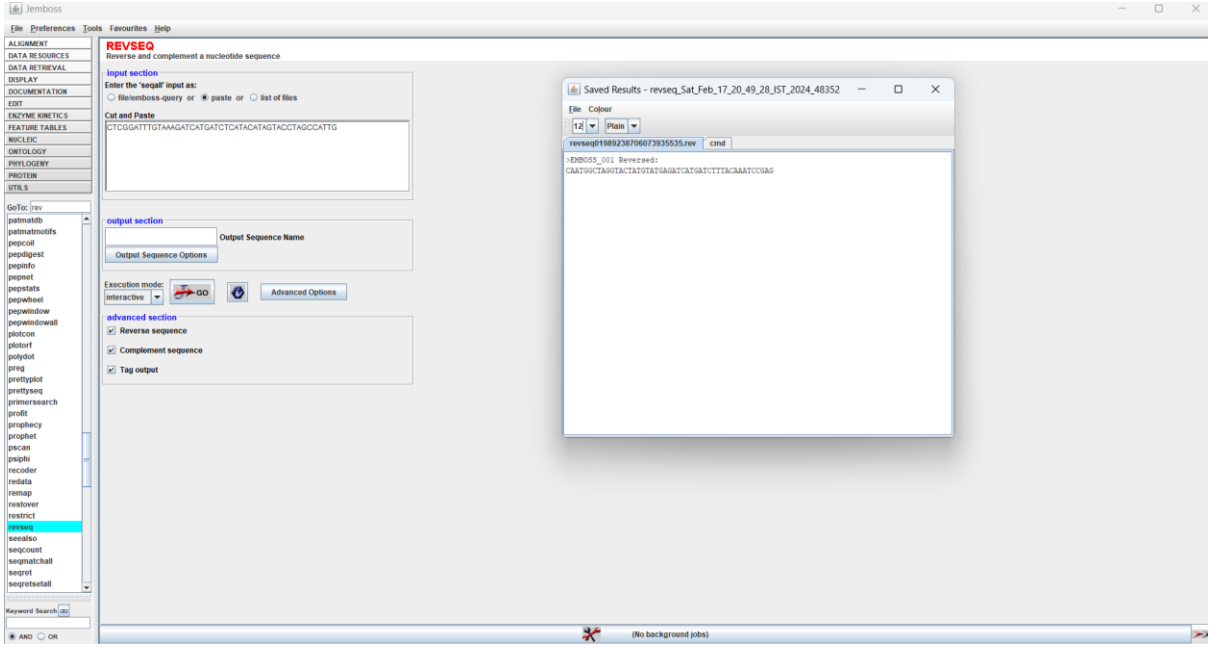
Outputs received:

>EMBOSS_001

GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC

>EMBOSS_001 Reversed:

CAATGGCTAGGTACTATGTATGAGATCATGATCTTTACAAATCCGAG



Q3) Write a program to find the complementary strand for the sequence given in Q2.

```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q3

bp_dict={'A':'T', 'C':'G', 'G':'C' , 'T':'A'}
original_sequence=list(input().upper())
count=0
for i in original_sequence:
    original_sequence[count]=bp_dict[f'{i}']
    count=count+1
final_sequence=''.join(map(str, original_sequence))
final_sequence_reversed=final_sequence[::-1]
print(f"Complementary Sequence: {final_sequence}")
print(f"Complementary Sequence (Reversed): {final_sequence_reversed}") #Prints the reversed sequence
```

✓ 1.6s Python

```
Complementary Sequence: GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC
Complementary Sequence (Reversed): CAATGGCTAGGTACTATGTATGAGATCATGATCTTTACAAATCCGAG
```

Q4) (i)Using EMBOSS, find the protein sequence for GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATCGACAAGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG

(ii) Identify the reading frame equivalent to the following sequence. PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV

(i) Protein sequence Using Emboss. In Advance options select 'All Six' to get all the frames

>_1

DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG

>_2

TL*TVKKSMQCARSRRGSRASSAPWALPWTTLLCPSTSMSLNLVHAEENWMG

>_3

HCEQ*KSPCNAQGAEEEAGQAALHGLSHGLLSSAHRQA*A*IWSMQKKTGWX

>_4

PIQFSSAWTKFRLMLVDGQRRVVHGRAHGALLALLPLLLLAHCMDFFTVHNV

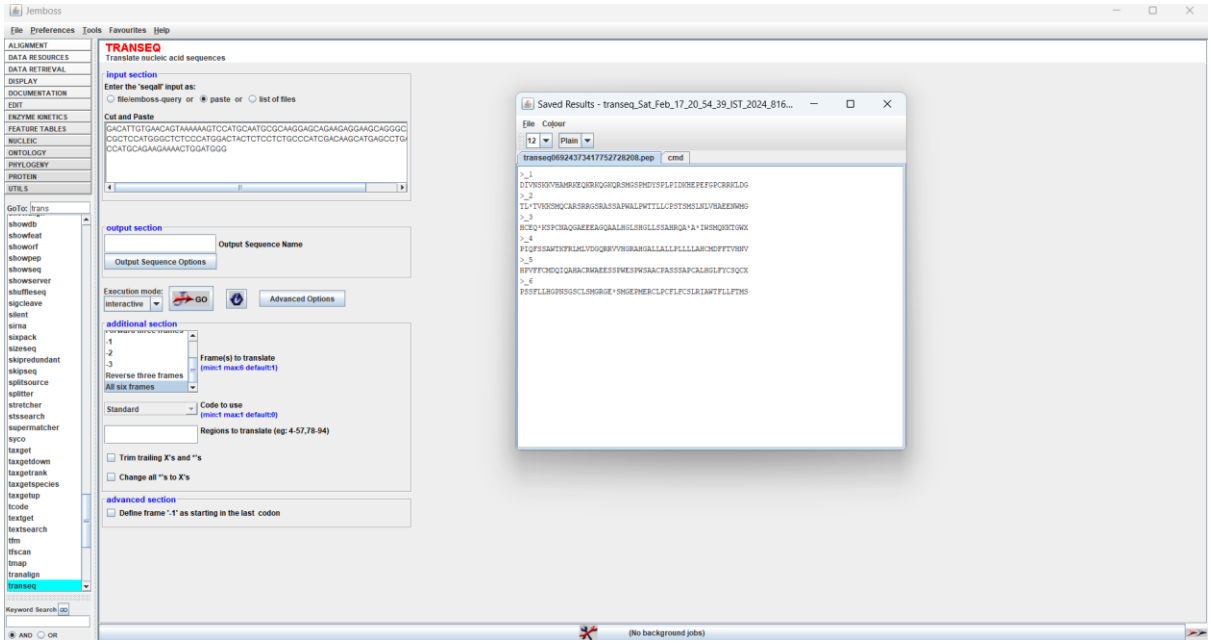
>_5

HPVFFCMDQIQAHACRWAEESSPWESPWSAACPASSSAPCALHGLFYCSQCX

>_6

PSSFLLHGPNSGSCLSMGRGE*SMGEPMERCLPCFLFCSLRIAWTFLLFTMS

(ii) The reading frame equivalent for the given sequence was found to be the **fourth sequence**. Highlighted in yellow above.



Q5) Write a program to translate the given DNA sequence (refer Q4) to protein sequence.

```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q5


#Codon table in the form of dictonary
codon_to_aa = {
    "TTT": "F", "TTC": "F", "TTA": "L", "TTG": "L",
    "CTT": "L", "CTC": "L", "CTA": "L", "CTG": "L",
    "ATT": "I", "ATC": "I", "ATA": "I", "ATG": "M",
    "GTT": "V", "GTC": "V", "GTA": "V", "GTG": "V",
    "TCT": "S", "TCC": "S", "TCA": "S", "TCG": "S",
    "CCT": "P", "CCC": "P", "CCA": "P", "CCG": "P",
    "ACT": "T", "ACC": "T", "ACA": "T", "ACG": "T",
    "GCT": "A", "GCC": "A", "GCA": "A", "GCG": "A",
    "TAT": "Y", "TAC": "Y", "TAA": "*", "TAG": "*",
    "CAT": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
    "AAT": "N", "AAC": "N", "AAA": "K", "AAG": "K",
    "GAT": "D", "GAC": "D", "GAA": "E", "GAG": "E",
    "TGT": "C", "TGC": "C", "TGA": "*", "TGG": "W",
    "CGT": "R", "CGC": "R", "CGA": "R", "CGG": "R",
    "AGT": "S", "AGC": "S", "AGA": "R", "AGG": "R",
    "GGT": "G", "GGC": "G", "GGA": "G", "GGG": "G"
}
#Fromatting the input
inputed_sequence=input().upper()
inputed_sequence=inputed_sequence.replace(" ","")
codons=[inputed_sequence[i:i+3] for i in range(0, len(inputed_sequence), 3)]
final_peptide=""

for i in range(len(codons)):
    codon_inputed=codons[i]
    if (len(codon_inputed)<4):
        final_peptide=final_peptide+codon_to_aa[codon_inputed]
    else:break
print(final_peptide)
```

✓ 2.8s Python

```
DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG
```

Q6) Write a code to search for the following strings 'AAG', 'GTC', 'GAG, 'ACTA', and 'ATAT' in the DNA sequence provided in Q4. The program should print the total number of matches for the each of the given strings and the start positions of the matches.

```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q6
main_peptide=input("Enter the main sequence:")
main_peptide=inputed_sequence.replace(" ","")
seq_to_be_searched=input("Enter the sequence to be searched:")
seq_to_be_searched=seq_to_be_searched.replace(" ","")
print(f"Enter the string: {seq_to_be_searched}")

splitted_list=[main_peptide[i:i+len(seq_to_be_searched)] for i in range(0,len(main_peptide)-len(seq_to_be_searched))]

index_list=[]
for i in splitted_list:
    if i == seq_to_be_searched:
        index_list.append(splitted_list.index(i))
        splitted_list[splitted_list.index(i)]="#"

index_string=' '.join(map(str,index_list))
print(f"Total match: {len(index_list)}")
print(f"Position of match : {index_string}")
```

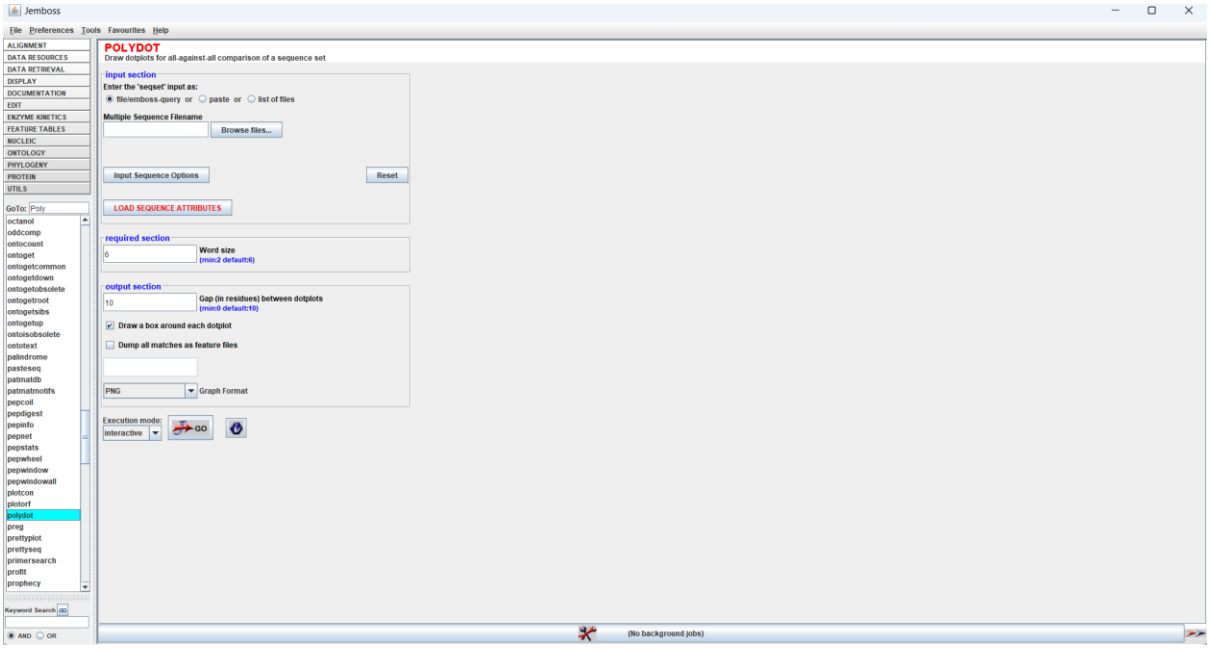
✓ 15.7s Python

```
Enter the string: ACTA
Total match: 1
Position of match : 88
```

Q7) Familiarize with other applications in EMBOSS. For example, melting temperature, bending, curvature etc.

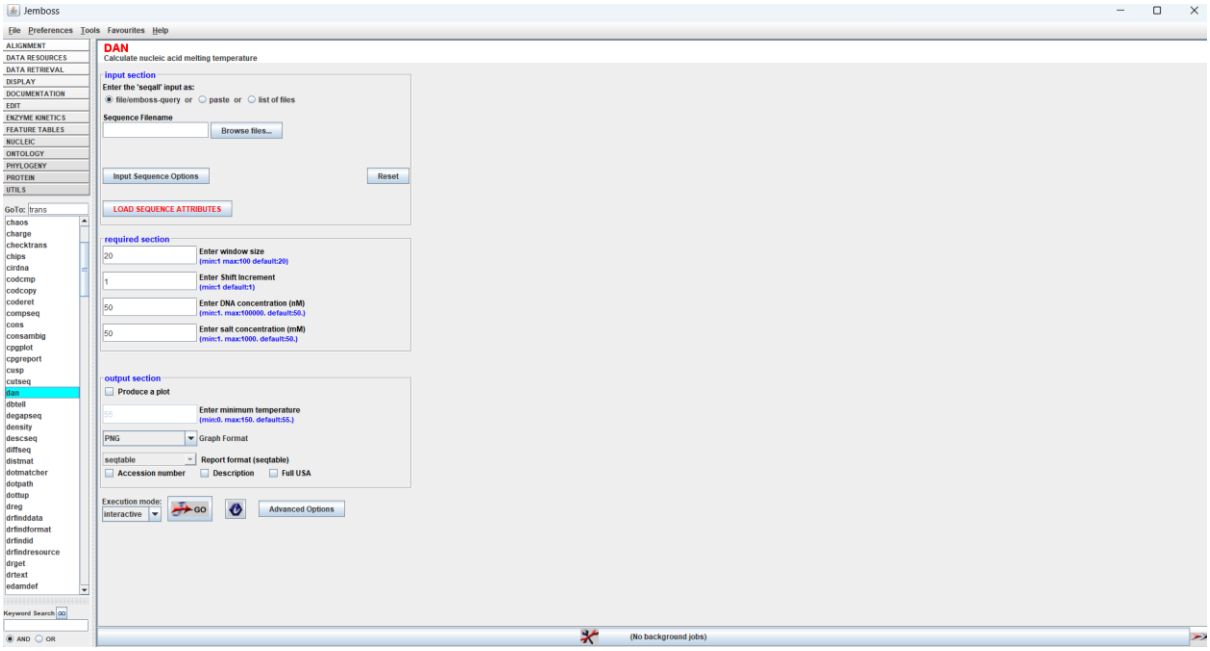
a) POLYDOT

Draws dot plot for all-against-all comparison of sequence set



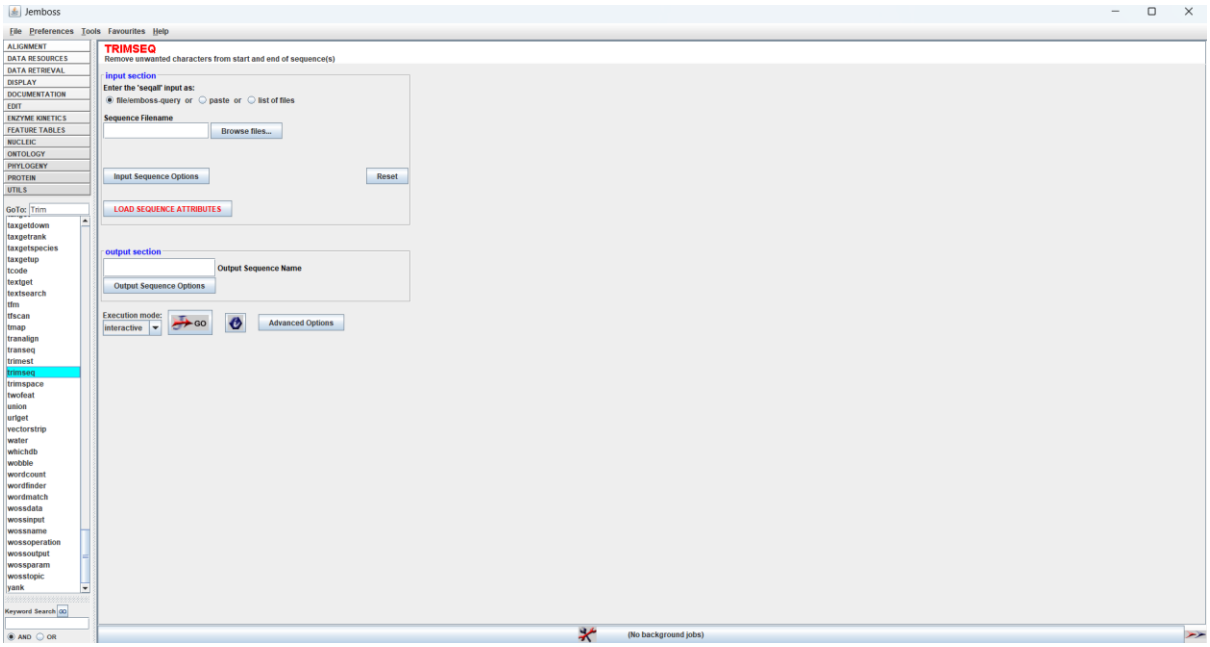
b) DAN

Calculates nucleic acid melting temperature



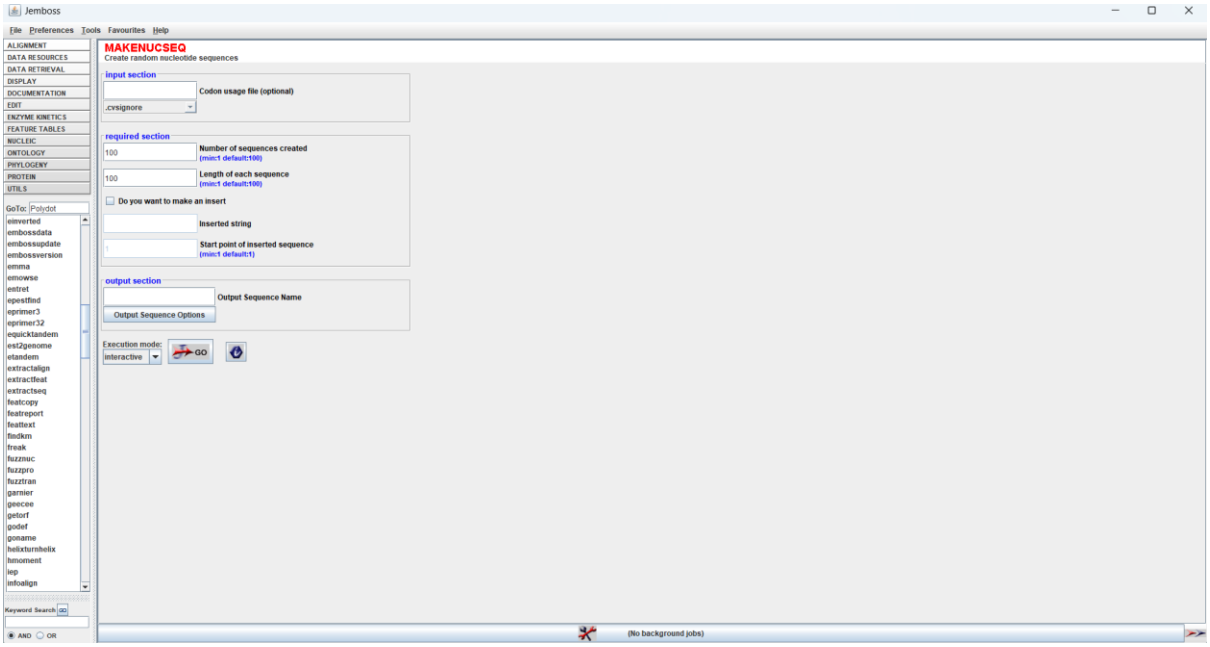
c) TRIMSEQ

Removes unwanted characters from start to end.



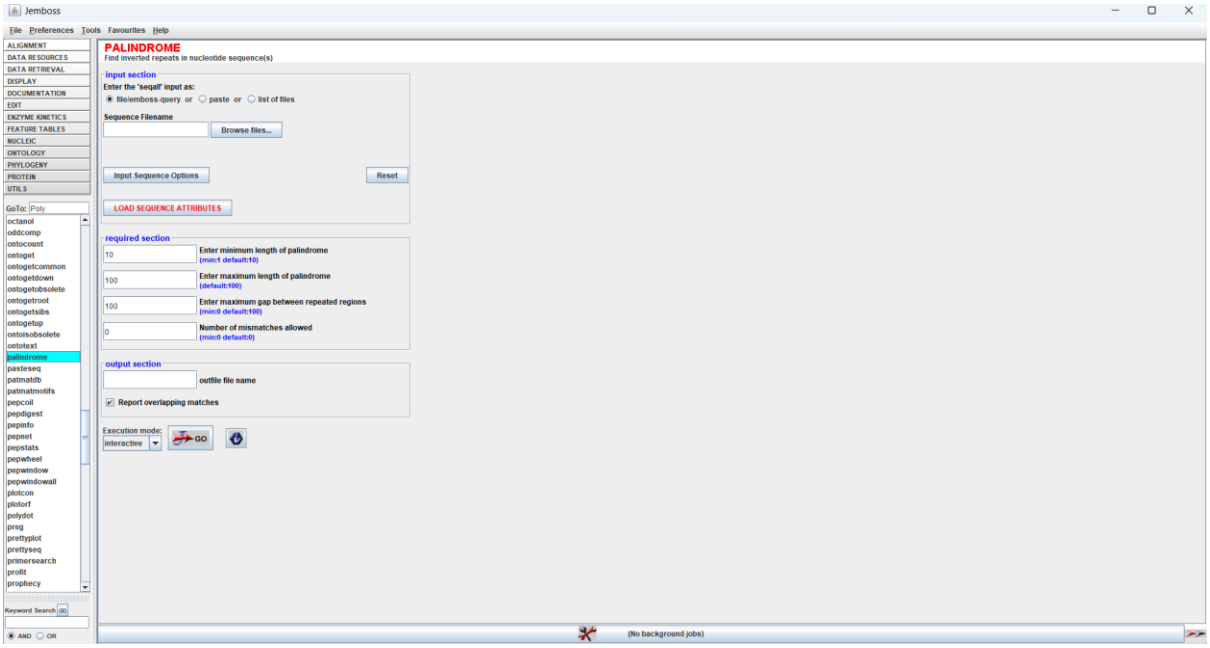
d) MAKEUCESQ

Creates random nucleotide sequences.



e) PALINDROME

Finds Palindromes, inverted repeats in the nucleotide sequence(s)



Q8) Write a program to compute the average base stacking energy for the sequence given in Q2 (AA: - 4; AT: -7; AC: -5; AG: -11; TA: -7; TT: -2; TC: -3; TG: -4; CA: -9; CT: -5; CC: -6; CG: -7; GA: - 9; GT: -6; GC: -4; GG: -11).

```python
#Atharva Mandar Phatak | BE21B009 | BT30340 Assignment 1 |
#Q8

#Creating a dicotnary to store the given values
energy_levels = {
    'AA': -4, 'AT': -7, 'AC': -5, 'AG': -11,
    'TA': -7, 'TT': -2, 'TC': -3, 'TG': -4,
    'CA': -9, 'CT': -5, 'CC': -6, 'CG': -7,
    'GA': -9, 'GT': -6, 'GC': -4, 'GG': -11}

inputed_base_stacking= input()
stacks=[inputed_base_stacking[i:i+2] for i in range(0, len(inputed_base_stacking)-1,1)]
stacking_energy=0
for i in stacks:
    stacking_energy=stacking_energy+energy_levels[i]
avg_stacking_energy=(stacking_energy/len(stacks))
avg_stacking_energy=round(avg_stacking_energy,3)
print(avg_stacking_energy)
```

✓ 29.1s Python

```
-6.283
```

Q9) Compute the average melting temperature of the following sequences using Seq2Feature tool (https://www.iitm.ac.in/bioinfo/SBFE/index.html) and comment on the results (Enter one sequence at a time in fasta format)
(i) ATATATATAT ii) GCGCGCGCGC

i) Average Melting Temperature: 48.0022° C
ii) Average Melting Temperature: 107.867° C

(i)

Your input seq is:

ATATATATAT

Physicochemical Properties:

Properties	Scaleunit	Average value
Stacking energy	kcal/mol	1.8
Enthalpy	kcal/mol	6.04444
Entropy	cal/mol/K	16.6222
Flexibility_shift	kJ mol^-1 A^-2	2.53
Flexibility_slide	kJ mol^-1 A^-2	9.66333
Free energy	kcal/mol	0.655556
Melting Temperature	degree	48.0022
Mobility to bend towards major groove	mu	1.09778
Mobility to bend towards minor groove	mu	1.03333
Probability contacting nucleosome core	%	6.75556
Rise stiffness	kcal/mol angstroem	7.80778
Roll stiffness	kcal/mol degree	19.3333
Shift stiffness	kcal/mol angstroem	0.892222
Slide stiffness	kcal/mol angstroem	2.66111
Tilt stiffness	kcal/mol degree	28
Twist stiffness	kcal/mol degree	25.8889

ii)

Your input seq is:

GCGCGCGCGC

Physicochemical Properties:

Properties	Scaleunit	Average value
Stacking energy	kcal/mol	1.75556
Enthalpy	kcal/mol	11.0778
Entropy	cal/mol/K	27.5556
Flexibility_shift	kJ mol^-1 A^-2	6.49111
Flexibility_slide	kJ mol^-1 A^-2	4.19778
Free energy	kcal/mol	1.85889
Melting Temperature	degree	107.867
Mobility to bend towards major groove	mu	0.997778
Mobility to bend towards minor groove	mu	1.20556
Probability contacting nucleosome core	%	3.37778
Rise stiffness	kcal/mol angstroem	8.06333
Roll stiffness	kcal/mol degree	21.5556
Shift stiffness	kcal/mol angstroem	1.14667
Slide stiffness	kcal/mol angstroem	2.33889
Tilt stiffness	kcal/mol degree	31.5556
Twist stiffness	kcal/mol degree	20.1111

Thus i) sequence shows lower melting point as compared to the second sequence.
Reason : G-C pairs have a higher stacking energy than A-T pairs because they make three hydrogen bonds in water whereas AT only forms two hydrogen bonds resulting into GC pair have high melting point. Thus, the sequence with more GC content will have a higher melting point.

Q10) Calculate the AT and GC content of the sequence AAATGGCCCTAA using Seq2Feature too

AT Content = 5.3334 %

GC Content = 41.6667 %

Your input seq is:

AAATGGCCCTAA

Nucleotide Content:

	Nucleotide content in %
AT_content	58.333333
Adenine_content	41.666667
Cytosine_content	25.000000
GC_content	41.666667
Guanine_content	16.666667
Keto_GT_content	33.333333
Purine_AG_content	58.333333
Thymine_content	16.666667

Practical 2
19 February 2024

1. How many "Homo sapiens" sequences deposited in DDBJ? Compare with Genbank and EMBL
   http://www.ddbj.nig.ac.jp/
   http://www.ncbi.nlm.nih.gov/genbank/
   http://www.ebi.ac.uk/embl/

2. What is the GC-content of the AY330867?

3. Compare the contents in DDBJ, Genbank and EMBL

4. Get the papers about "discrimination of beta barrel membrane proteins".
   https://pubmed.ncbi.nlm.nih.gov/

5. Find the papers published by any author (E.g.) Kihara D.

6. How many related articles are listed in PUBMED for the paper, Cell 2008 Dec 26;135(7):1158-9?

7. List the papers published in the journal "Nature" for the year 2024. Check the list in SCOPUS and PUBMED

8. Find the h-index and number of citations for "Burkhard Rost".

9. Find the class of the enzyme EC 1.7.2.3 and its function
   http://www.brenda-enzymes.org/

10. Find the catalytic site residues in Asparagine synthetase
    Hint: Find the EC number and search in Catalytic site atlas
    https://www.ebi.ac.uk/thornton-srv/m-csa/

11. Find the scientific name, the taxonomy ID and the number of chromosomes for the following organisms. http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Root
    Organisms: Human, Cat, Dog, domestic guinea pig, and Thale cress

12. What are NCBI E-utilities? Give the syntax for fetching a record in FASTA format using E-utilities. http://www.ncbi.nlm.nih.gov/books/NBK25500/

13. List two databases under each of the following category.
    a. Protein properties, b. Small molecules (Structure related), c. Cancer gene databases

    Hint: Use Nucleic Acids Research (NAR) – 'database category list'
    https://www.oxfordjournals.org/our_journals/nar/database/c/

Deadline: 25 Feb 2024

BT 3040: BIOINFORMATICS

Assignment 2

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) How many "Homo sapiens" sequences deposited in DDBJ? Compare with Genbank and

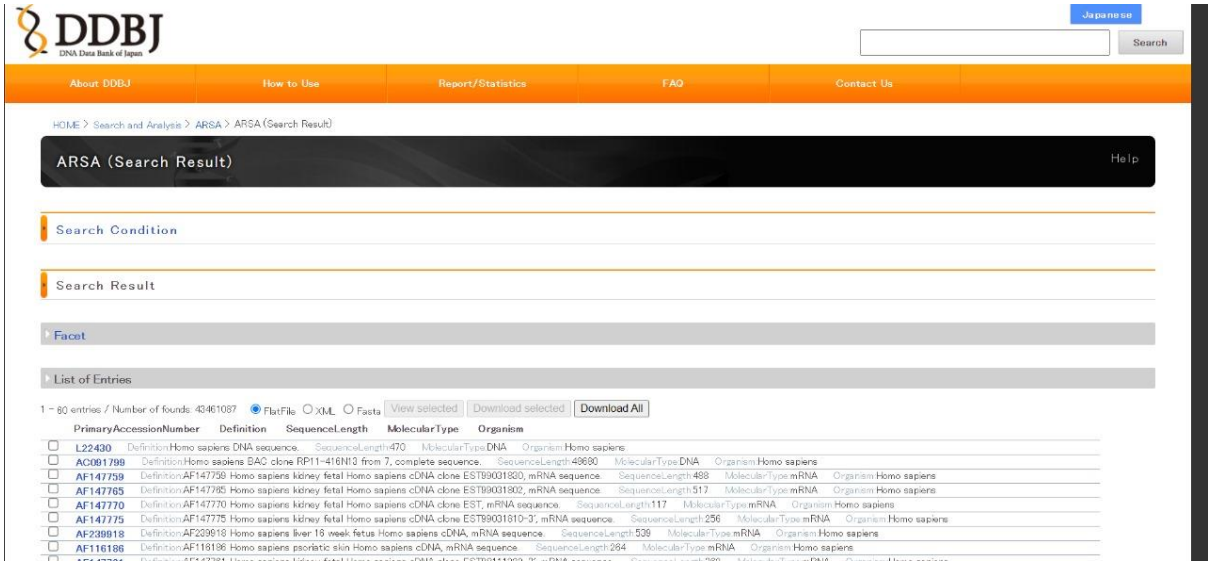
EMBL

http://www.ddbj.nig.ac.jp/

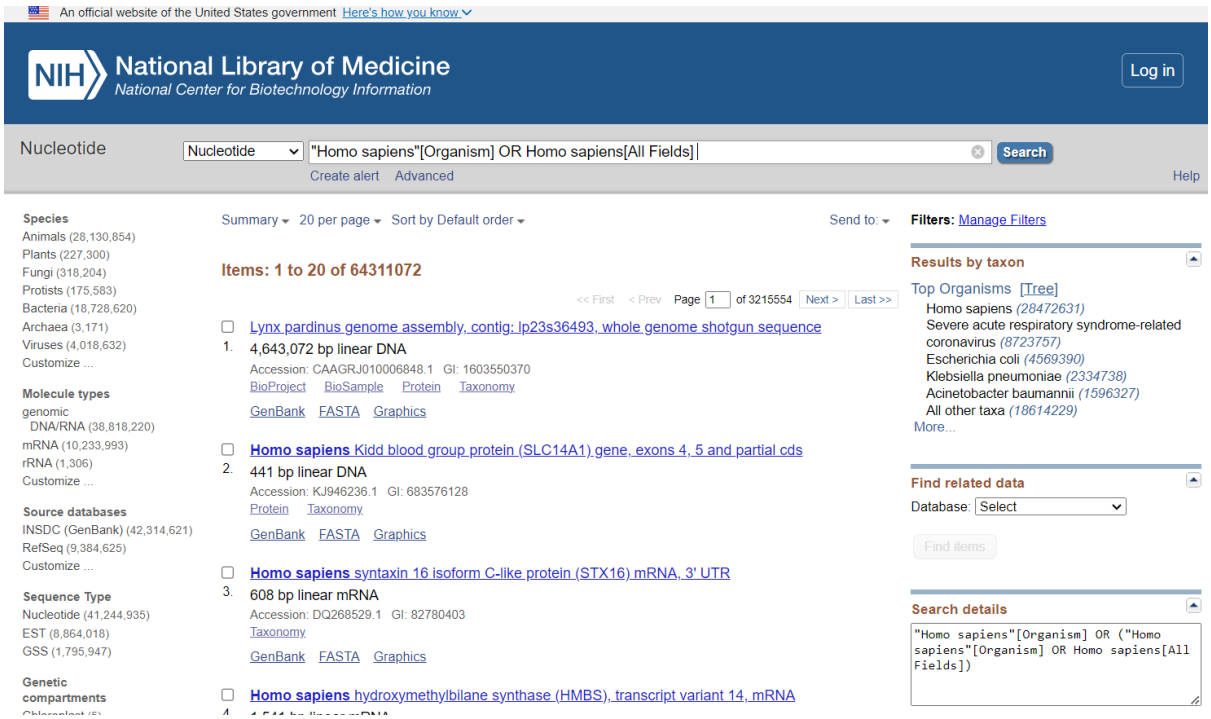
http://www.ncbi.nlm.nih.gov/genbank/

http://www.ebi.ac.uk/embl/

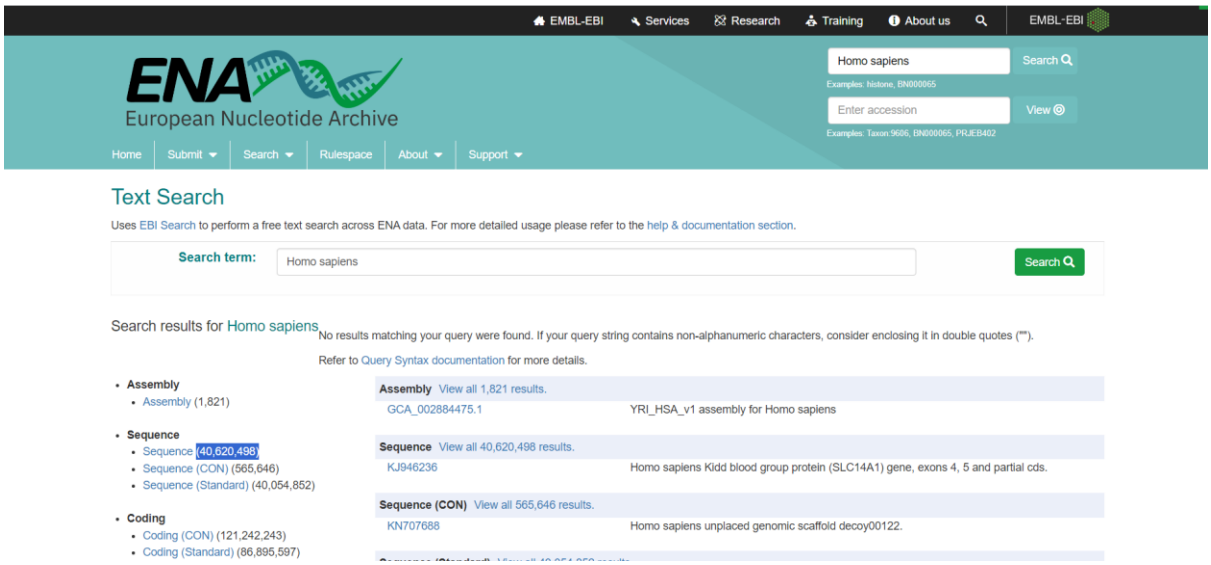
a) DDBJ: **43461087 sequences**



b) Gen bank: **64311072 sequences**



c) EMBL: **40620498**



Comparison (based on number of sequences) :

DDBJ	GenBank	EMBL
43461087	**64311072**	**40620498**

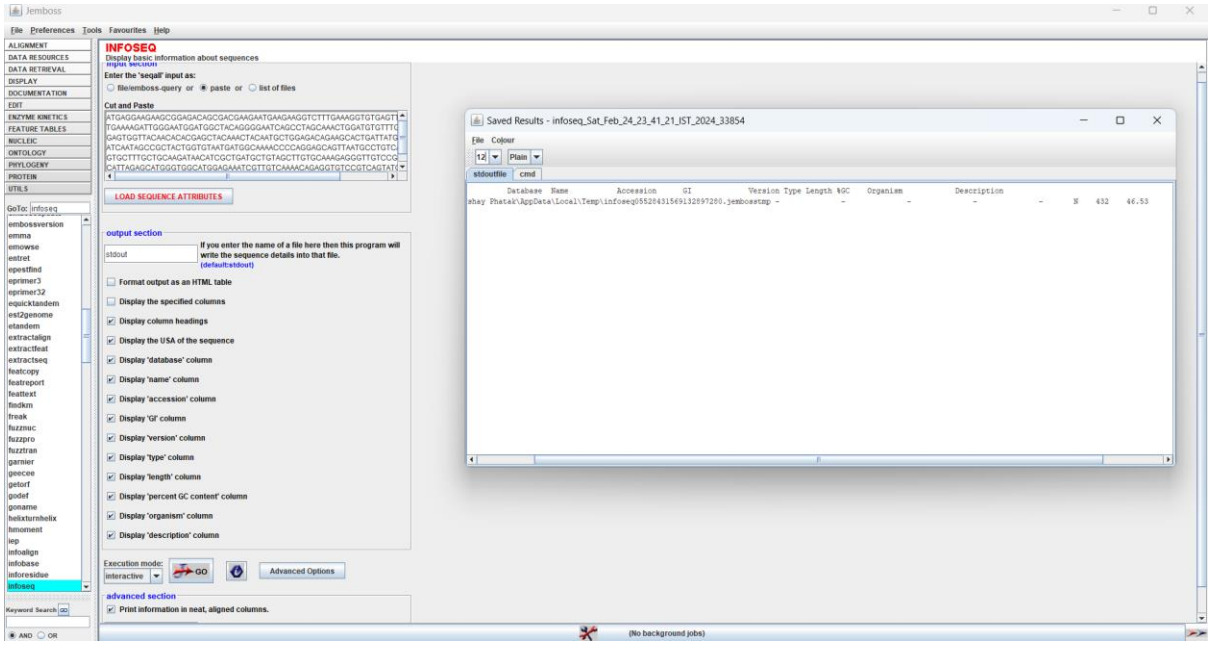
Thus GenBank has the highest number of deposited "Homo Sapiens" sequences

GenBank>DDBJ>EMBL

*Based on number of sequences

Q2) What is the GC-content of the AY330867?

GC content = **46.53%**



Q3 Compare the contents in DDBJ, Genbank and EMBL

DDBJ	GenBank	EMBL
I. LOCUS II. DEFINITION III. ACCESSION IV. VERSION V. DBLINK VI. KEYWORDS VII. SOURCE a. ORGANISM VIII. REFERENCE a. AUTHORS b. TITLE c. JOURNAL IX. COMMENT X. FEATURES XI. BASE COUNT XII. ORIGIN	I. LOCUS II. DEFINITION III. ACCESSION IV. VERSION V. KEYWORDS VI. SOURCE a) ORGANISM VII. REFERENCE a) AUTHORS b) TITLE c) JOURNAL	I. Submit entries II. Search options – i. Text search ii. Advanced search iii. Sequence search iv. Xref search v. Sequence versions III. Rulespace IV. General info a. EMBL ENA b. Research c. Industry d. Training

The contents of EMBL will be nearly identical, although the main sources of nucleoid sequences are direct submissions from individual researchers, genome sequencing projects and patent application. Another notable distinction between the databases is that GeneBank doesn't show base count while DDBJ, EMBL show it.

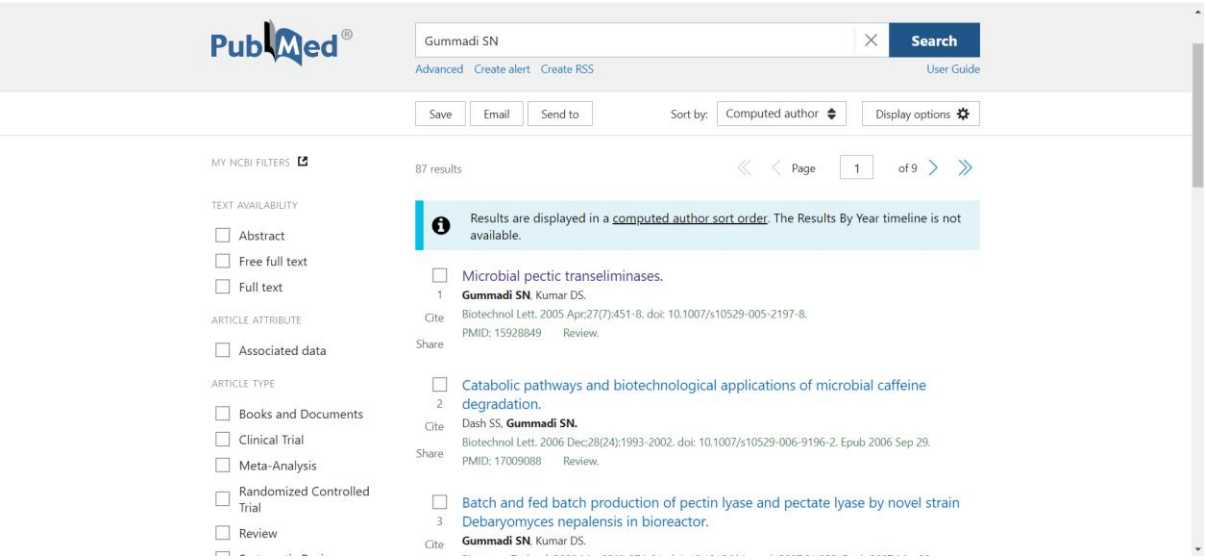
Q4) Get the papers about "discrimination of beta barrel membrane proteins".

https://pubmed.ncbi.nlm.nih.gov/

These two articles (out of 53) were found by **citation matching**

a) TMBETADISC-RBF: Discrimination of beta-barrel membrane proteins using RBF networks and PSSM profiles.
   Ou YY, et al. Comput Biol Chem. 2008. PMID: 18434251
b) Current developments on beta-barrel membrane proteins: sequence and structure analysis, discrimination and prediction.
   Gromiha MM, et al. Curr Protein Pept Sci. 2007. PMID: 18220845 Review.

Q5) The papers published by Dr. Sathyanarayana N Gummadi Sir.



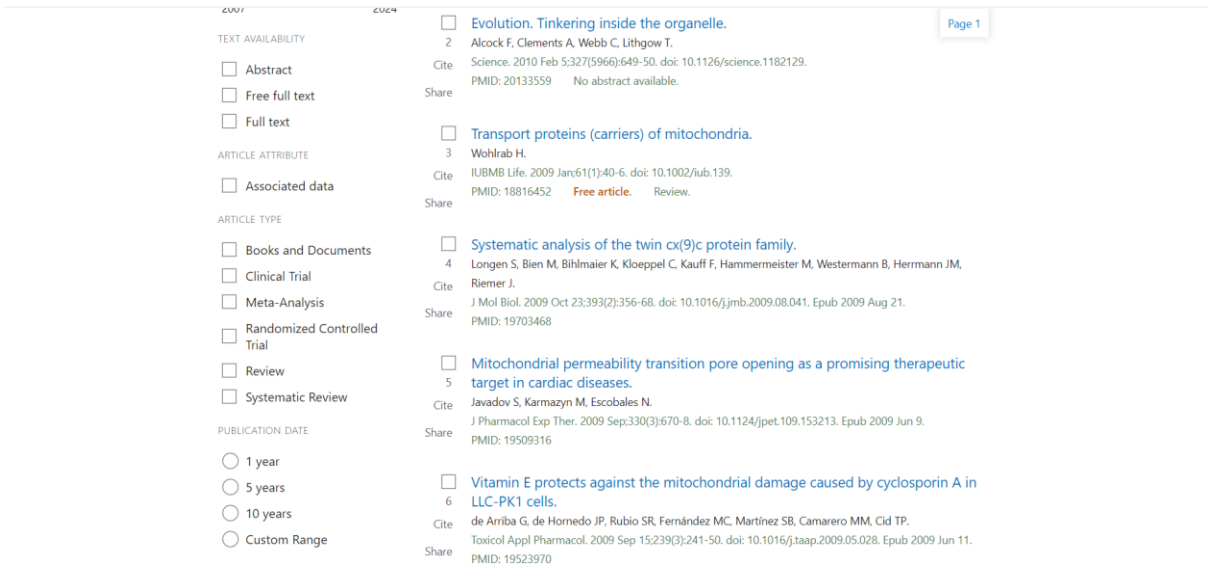
Q6) How many related articles are listed in PUBMED for the paper, Cell 2008 Dec 26;135(7):1158-9?

Following paper correspond to the given code.

https://pubmed.ncbi.nlm.nih.gov/19109882/

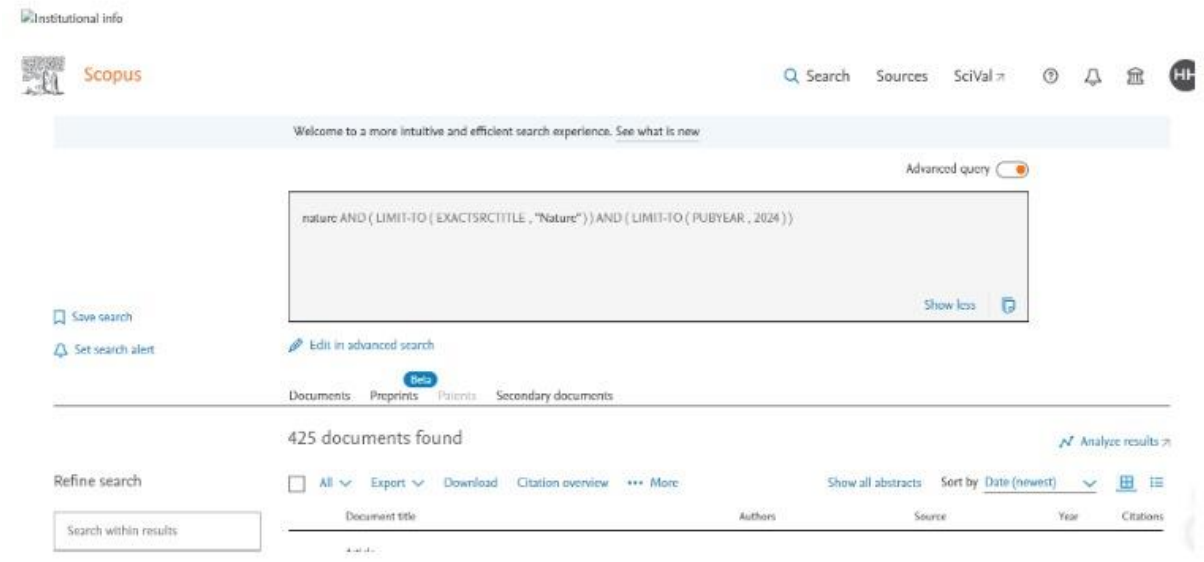
Authors: Kenichiro Imai, M Michael Gromiha, Paul Horton

On scrolling down to the 'Similar Articles' section, total of **27** articles re found. The screenshot shows the **top 5** similar articles.

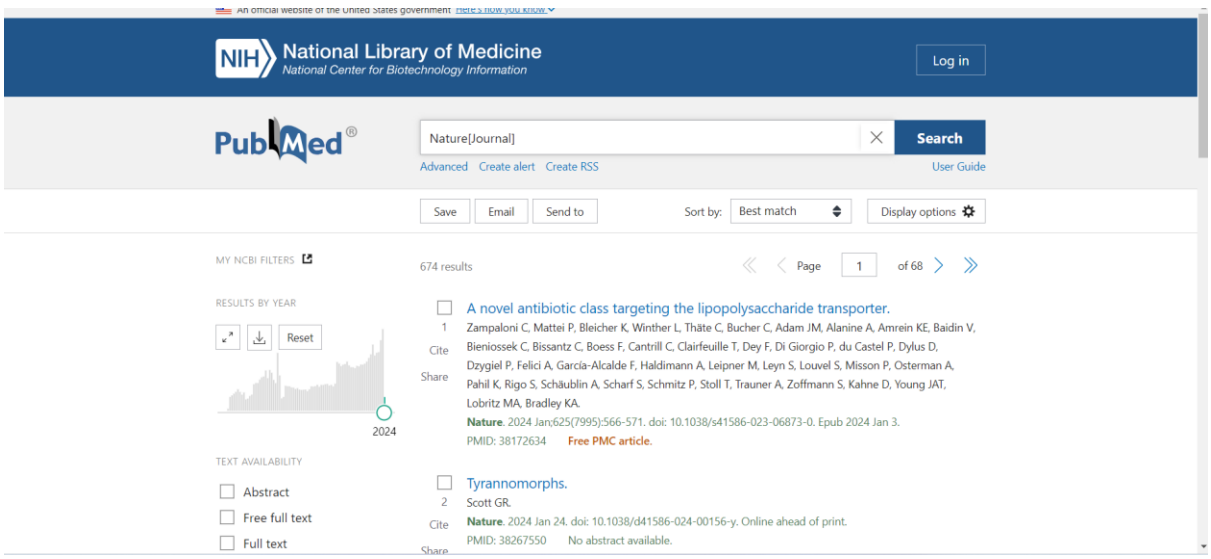


Q7) List the papers published in the journal "Nature" for the year 2024. Check the list in SCOPUS and PUBMED

Total of articles have been published in the year 2024 till now in "Nature".
a) Scopus: 425 papers

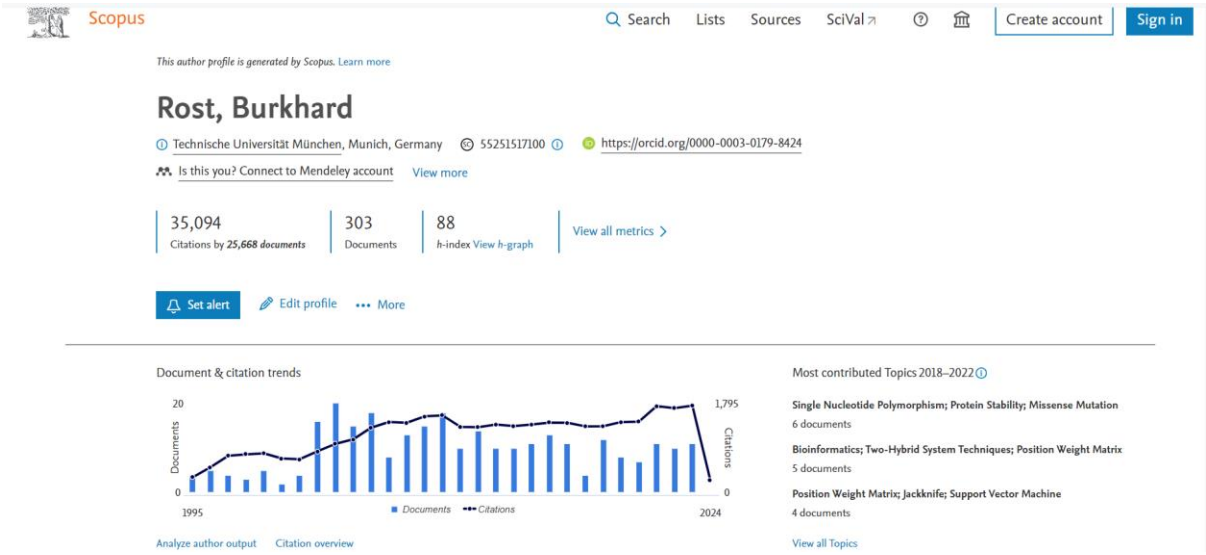


b) PubMed: 674 papers



Q8) Find the h-index and number of citations for “Burkhard Rost”.

The given screenshot shows the profile of “Burkhard Rost” on Scopus. “Burkhard Rost” has a h-index of **88.**



Q9) Find the class of the enzyme EC 1.7.2.3 and its function http://www.brenda-enzymes.org/

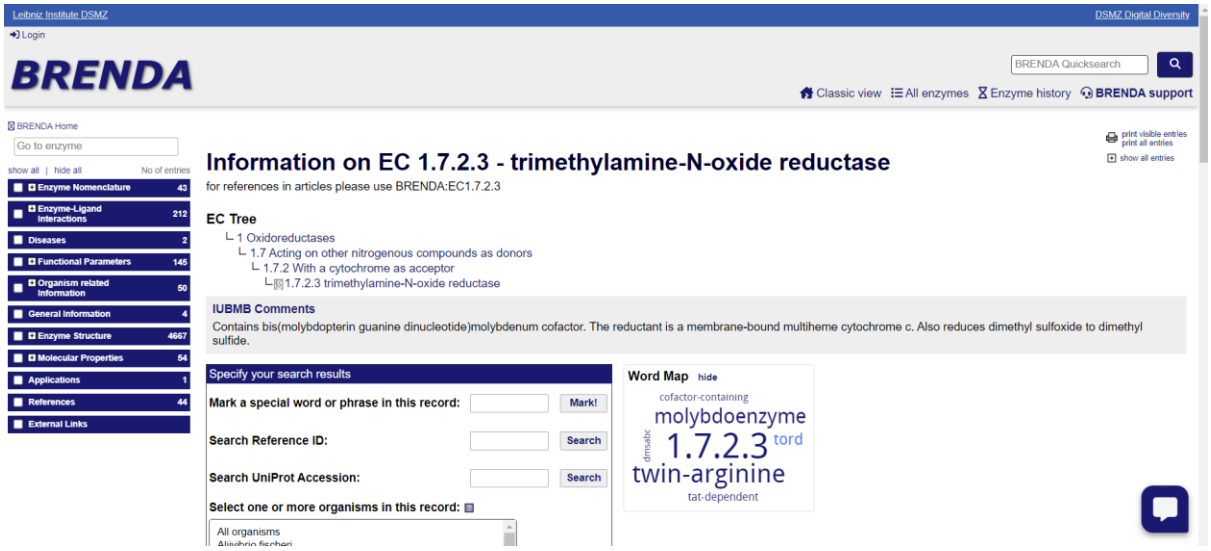
1: Oxidoreductases

1.7: Acting on other nitrogenous compounds as donors

1.7.2: With a cytochrome as acceptor

1.7.2.3: trimethylamine-N-oxide reductase

Function: It is the enzyme TMAO reductase, which reduces the cytochrome TorC . Also reduces dimethyl sulfoxide to dimethyl sulfide.



Q10) Find the catalytic site residues in Asparagine synthetase
Hint: Find the EC number and search in Catalytic site atlas
https://www.ebi.ac.uk/thornton-srv/m-csa/

The EC number of Asparagine Synthetase is **6.3.5.4**

The Catalytic Site Residues in Asparagine Synthetase as given in Catalytic Site atlas are.

Home | Browse | Search | Statistics | Download / API | Documentation | About | Contact Us | Log in

Kinetic analysis has shown the rate of catalysis at each site to be independent of one another, and so the stoichiometry between the sites must be maintained by the catalytic efficiency of the two sites.

Catalytic Residues Roles

UniProt	PDB* (1ct9)		
Cys2 (N-term)	Ala1A (N-term)	Acts as a general acid/base to activate the cysteine nucleophile.	proton acceptor, proton donor
Leu51 (main-C)	Leu50A (main-C)	Helps stabilise the reactive intermediates formed.	hydrogen bond acceptor, electrostatic stabiliser
Thr322, Arg325	Thr321A, Arg324A	Bind and stabilise the phosphate groups of the ATP and reactive intermediates formed.	hydrogen bond donor, electrostatic stabiliser
Cys2	Ala1A	Acts as a catalytic nucleophile in the glutaminase domain reaction.	covalently attached, hydrogen bond acceptor, nucleofuge, nucleophile, proton acceptor, proton donor
Gly76 (main-N), Asn75	Gly75A (main-N), Asn74A	Forms the oxyanion hole that stabilises the reactive intermediates and transition states formed.	hydrogen bond donor, electrostatic stabiliser

*PDB label guide - RESx(y)B(C) - RES: Residue Name; x: Residue ID in PDB file; y: Residue ID in PDB sequence if different from PDB file; B: PDB Chain; C: Biological Assembly Chain if different from PDB. If label is "Not Found" it means this residue is not found in the reference PDB.

Chemical Components

proton transfer, bimolecular nucleophilic addition, proton relay, enzyme-substrate complex formation, overall reactant used, intermediate formation, unimolecular elimination by the conjugate base, enzyme-substrate complex cleavage, deamination, intermediate collapse, overall product formed, native state of enzyme regenerated

Q11) Find the scientific name, the taxonomy ID and the number of chromosomes for the following organisms.
http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?mode=Root
Organisms: Human, Cat, Dog, domestic guinea pig, and Thale cress

Sr. No	Common Name	Scientific Name	Taxonomy ID	No. of Chromosomes (Total i.e 2n=)
1.	Human	*Homo sapiens* Linnaeus, 1758	9606	46
2.	Cat	*Felis catus* Linnaeus, 1758	9685	38
3.	Dog	*Canis lupus familiaris* Linnaeus, 1758	9615	78
4.	Domestic Guinea Pig	*Cavia porcellus*	10141	64
5.	Thale Cress	*Arabidopsis thaliana* (L.) Heynh., 1842	3702	10

Q12) What are NCBI E-utilities? Give the syntax for fetching a record in FASTA format using E-utilities. http://www.ncbi.nlm.nih.gov/books/NBK25500

The Entrez Programming Utilities (E-utilities) are a set of nine server-side programs that provide a stable interface into the Entrez query and database system at the National Center for Biotechnology Information (NCBI). The E-utilities use a fixed URL syntax that translates a standard set of input parameters into the values necessary for various NCBI software components to search for and retrieve the requested data. The E-utilities are therefore the structured interface to the Entrez system, which currently includes 38 databases covering a variety of biomedical data, including nucleotide and protein sequences, gene records, three-dimensional molecular structures, and the biomedical literature

Base URL

https://eutils.ncbi.nlm.nih.gov/entrez/eutils/einfo.fcgi

Syntax for fetching a record in FASTA format:

Downloading Full Records Go to:

Basic Downloading

```
efetch.fcgi?db=<database>&id=<uid_list>&rettype=<retrieval_type>
&retmode=<retrieval_mode>
```

Input: List of UIDs (&id); Entrez database (&db); Retrieval type (&rettype); Retrieval mode (&retmode)

Output: Formatted data records as specified

Example: Download nuccore GIs 34577062 and 24475906 in FASTA format

https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=34577062,24475906&rettype=fasta&retmode=text

Q13) List two databases under each of the following category. a. Protein properties, b. Small molecules (Structure related), c. Cancer gene databases Hint: Use Nucleic Acids Research (NAR) – 'database category list' https://www.oxfordjournals.org/our_journals/nar/database/c/

Two databases under the given category are as follows:

1.	**Protein Properties**	AAindex
		TOPPR
2.	**Small molecule (Structure related)**	BitterDB
		SuperToxic
3.	**Cancer Gene Database**	ArrayMap
		UMD-BRCA1/BRCA2 databases

Practical 3
22 Feb 2024

Questions

1. Find the amino acid sequence of human mitochondrial β barrel membrane protein VDAC1 and its function? How many transmembrane segments are present in the protein?

2. Obtain the sequences of "transcription factors" with 50% sequence identity in FASTA format. List the count of sequences and count of clusters.
http://www.uniprot.org/uniprot/

3. How many protein sequences from Homo sapiens are obtained at identity cutoff of 100%, 90% and 50% sequence identity?

4. In UniProt, how many mouse (*Mus musculus*) protein sequences are manually annotated? And how many of these manually annotated protein sequences are associated with PDB (3D structures)?
Hint: Use "Advanced" search

5. Map first 10 UniProt IDs of above manually curated mouse protein sequences with 3D structures to STRING database. How many STRING IDs are mapped?
Hint: Use "Retrieve/ID mapping", upload Uniprot IDs as text file

6. Using UniProt Statistics data, answer the following
a) What do you infer from the distribution of sequence length in UniProt?
b) The shortest and longest sequence in UniProtKB
c) Amino acid composition in percent for the complete database

Deadline: 3 Mar 2024

BT 3040: BIOINFORMATICS

Assignment 3

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) Find the amino acid sequence of human mitochondrial β barrel membrane protein VDAC1 and its function? How many transmembrane segments are present in the protein?

A1) Sequence of human mitochondrial beta barallel membrane protein VDAC

```
SQ   SEQUENCE   283 AA;  30773 MW;  89BA3378B04020D5 CRC64;
     MAVPPTYADL GKSARDVFTK GYGFGLIKLD LKTKSENGLE FTSSGSANTE TTKVTGSLET
     KYRWTEYGLT FTEKWNTDNT LGTEITVEDQ LARGLKLTFD SSFSPNTGKK NAKIKTGYKR
     EHINLGCDMD FDIAGPSIRG ALVLGYEGWL AGYQMNFETA KSRVTQSNFA VGYKTDEFQL
     HTNVNDGTEF GGSIYQKVNK KLETAVNLAW TAGNSNTRFG IAAKYQIDPD ACFSAKVNNS
     SLIGLGYTQT LKPGIKLTLS ALLDGKNVNA GGHKLGLGLE FQA
```

A2) Function:

Function

Forms a channel through the mitochondrial outer membrane and also the plasma membrane. The channel at the outer mitochondrial membrane allows diffusion of small hydrophilic molecules; in the plasma membrane it is involved in cell volume regulation and apoptosis. It adopts an open conformation at low or zero membrane potential and a closed conformation at potentials above 30-40 mV. The open state has a weak anion selectivity whereas the closed state is cation-selective (PubMed:**11845315**, PubMed:**18755977**, PubMed:**20230784**, PubMed:**8420959**).

Binds various signaling molecules, including the sphingolipid ceramide, the phospholipid phosphatidylcholine, and the sterol cholesterol (PubMed:**31015432**).

In depolarized mitochondria, acts downstream of PRKN and PINK1 to promote mitophagy or prevent apoptosis; polyubiquitination by PRKN promotes mitophagy, while monoubiquitination by PRKN decreases mitochondrial calcium influx which ultimately inhibits apoptosis (PubMed:**32047033**).

May participate in the formation of the permeability transition pore complex (PTPC) responsible for the release of mitochondrial products that triggers apoptosis (PubMed:**15033708**, PubMed:**25296756**).

May mediate ATP export from cells (PubMed:**30061676**). 9 Publications

B) Number of transmembrane sequence present in the protein = **19**

Transmembrane	26-35	Beta stranded	2 Publications	BLAST Add
Transmembrane	39-47	Beta stranded	2 Publications	BLAST Add
Transmembrane	54-64	Beta stranded	2 Publications	BLAST Add
Transmembrane	69-76	Beta stranded	2 Publications	BLAST Add
Transmembrane	80-89	Beta stranded	2 Publications	BLAST Add
Transmembrane	95-104	Beta stranded	2 Publications	BLAST Add
Transmembrane	111-120	Beta stranded	2 Publications	BLAST Add
Transmembrane	123-130	Beta stranded	2 Publications	BLAST Add
Transmembrane	137-145	Beta stranded	2 Publications	BLAST Add
Transmembrane	150-158	Beta stranded	2 Publications	BLAST Add
Transmembrane	163-175	Beta stranded	2 Publications	BLAST Add
Transmembrane	178-185	Beta stranded	2 Publications	BLAST Add
Transmembrane	189-198	Beta stranded	2 Publications	BLAST Add
Transmembrane	202-211	Beta stranded	2 Publications	BLAST Add
Transmembrane	218-227	Beta stranded	2 Publications	BLAST Add
Transmembrane	231-238	Beta stranded	2 Publications	BLAST Add
Transmembrane	242-251	Beta stranded	2 Publications	BLAST Add
Transmembrane	254-263	Beta stranded	2 Publications	BLAST Add
Transmembrane	273-282	Beta stranded	2 Publications	BLAST Add

Q2) Obtain the sequences of "transcription factors" with 50% sequence identity in FASTA format. List the count of sequences and count of clusters.
http://www.uniprot.org/uniprot/

Cluster ID	Cluster name	Types	Size	Organisms	Length	Identity
				More organisms		
UniRef50_A0A0M4FLP9	Cluster: NAC transcription factors 38		3 members	Manihot esculenta (Cassava) Corchorus capsularis (Jute) Hevea brasiliensis	200	UniRef50
UniRef50_A0A0M4FLS3	Cluster: NAC transcription factors 88		1 member	Manihot esculenta (Cassava)	264	UniRef50
UniRef50_A0A9Q0QM67	Cluster: HOMEOBOX PROTEIN TRANSCRIPTION FACTORS		1 member	Salix koriyanagi	351	UniRef50
UniRef50_A0A9Q0W5E4	Cluster: HOMEOBOX PROTEIN TRANSCRIPTION FACTORS		1 member	Salix purpurea (Purple osier willow)	90	UniRef50
UniRef50_A0A0M4FET5	Cluster: NAC transcription factors 45		1 member	Manihot esculenta (Cassava)	82	UniRef50
UniRef50_A0A0M4FKY5	Cluster: NAC transcription factors 39		1 member	Manihot esculenta (Cassava)	129	UniRef50
UniRef50_A0A0M5JF80	Cluster: NAC transcription factors 42		1 member	Manihot esculenta (Cassava)	124	UniRef50
UniRef50_A0A2P2JCL1	Cluster: NAC transcription factors 13		1 member	Rhizophora mucronata (Asiatic mangrove)	289	UniRef50
UniRef50_A0A9Q0P6B9	Cluster: HOMEOBOX PROTEIN TRANSCRIPTION FACTORS		1 member	Salix koriyanagi	384	UniRef50
UniRef50_A0A9Q0VVN3	Cluster: HOMEOBOX PROTEIN TRANSCRIPTION FACTORS		1 member	Salix purpurea (Purple osier willow)	275	UniRef50
UniRef50_A0A0G2SJ85	Cluster: AP3/EREBP Transcription Factors (Fragment)		1 member	Salvia miltiorrhiza (Chinese sage)	192	UniRef50
UniRef50_A0A5B6ZXL9	Cluster: Putative NAC transcription factors 48		1 member	Davidia involucrata (Dove tree)	187	UniRef50
UniRef50_A0A0G2SJB9	Cluster: AP11/EREBP Transcription Factors (Fragment)		1 member	Salvia miltiorrhiza (Chinese sage)	226	UniRef50
UniRef50_A0A0M3R857	Cluster: NAC transcription factors 43 (Fragment)		4 members	Manihot esculenta (Cassava)	367	UniRef50

The sequences' FASTA file is in the folder named 'Q2_ transcription_factors'

Q3) How many protein sequences from Homo sapiens are obtained at identity cutoff of 100%, 90% and 50% sequence identity?

Clusters

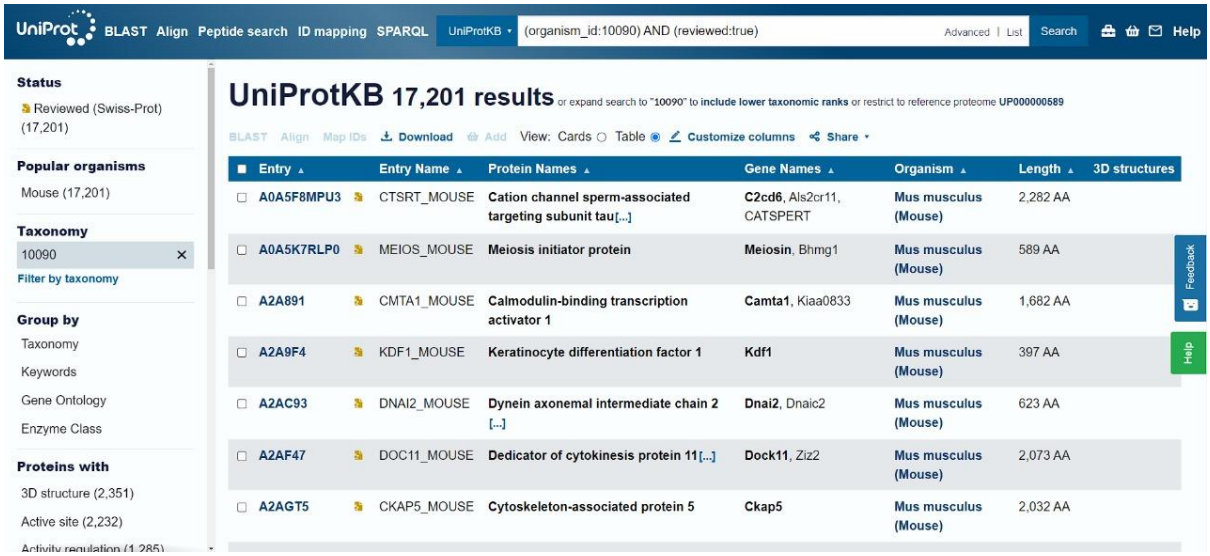
100% (236,474)

90% (106,848)

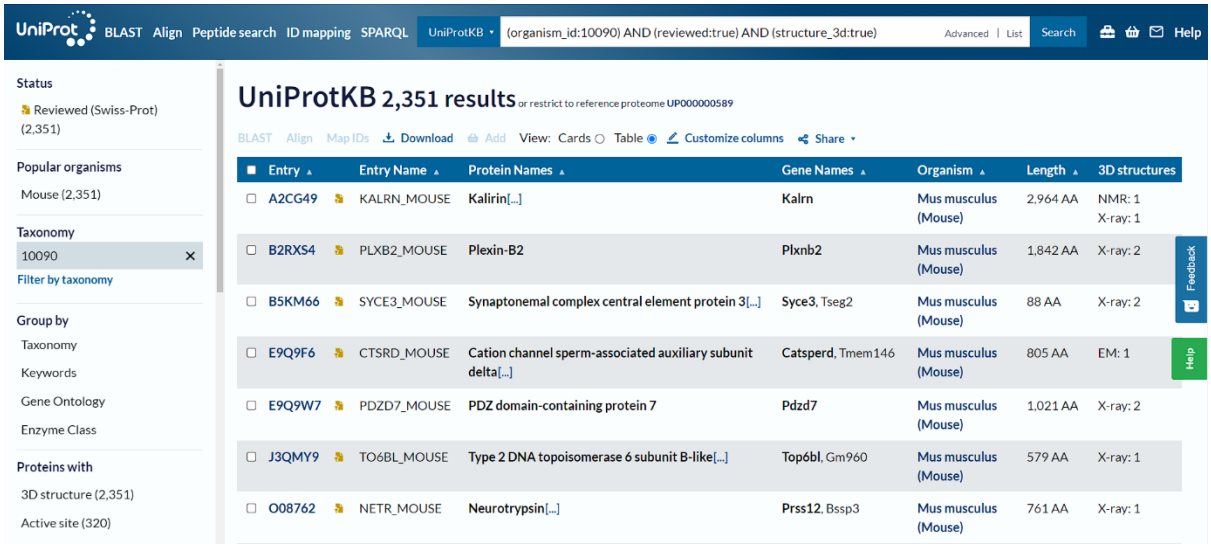
50% (54,154)

Q4) In UniProt, how many mouse (Mus musculus) protein sequences are manually annotated? And how many of these manually annotated protein sequences are associated with PDB (3D structures)?

a) Manually annotated sequences are **17807** of (Mus musculus) protein, under the Swiss-Prot Database.



b) There are **2351** manually annotated protein sequences are associated with PDB (3D structures)



Q5) Map first 10 UniProt IDs of above manually curated mouse protein sequences with 3D structures to STRING database. How many STRING IDs are mapped?

ID mapping 10 results found for UniProtKB_AC-ID → STRING

Overview | Input Parameters | API Request

Download View: Cards Table Resubmit

10 IDs were mapped to 10 results

From	To
Q91ZJ0	10090.ENSMUSP00000114895
P02762	10090.ENSMUSP00000079442
P11589	10090.ENSMUSP00000095655
Q920Q2	10090.ENSMUSP00000027251
Q64288	10090.ENSMUSP00000095882
P08882	10090.ENSMUSP00000015585
Q99P87	10090.ENSMUSP00000133024
Q91YN5	10090.ENSMUSP00000106983
Q99JT9	10090.ENSMUSP00000020957
Q8VHC3	10090.ENSMUSP00000092041

Feedback

Help

10 STRING IDs are mapped.

Q6) Using UniProt Statistics data, answer the following

a) What do you infer from the distribution of sequence length in UniProt?

b) The shortest and longest sequence in UniProtKB

c) Amino acid composition in percent for the complete database

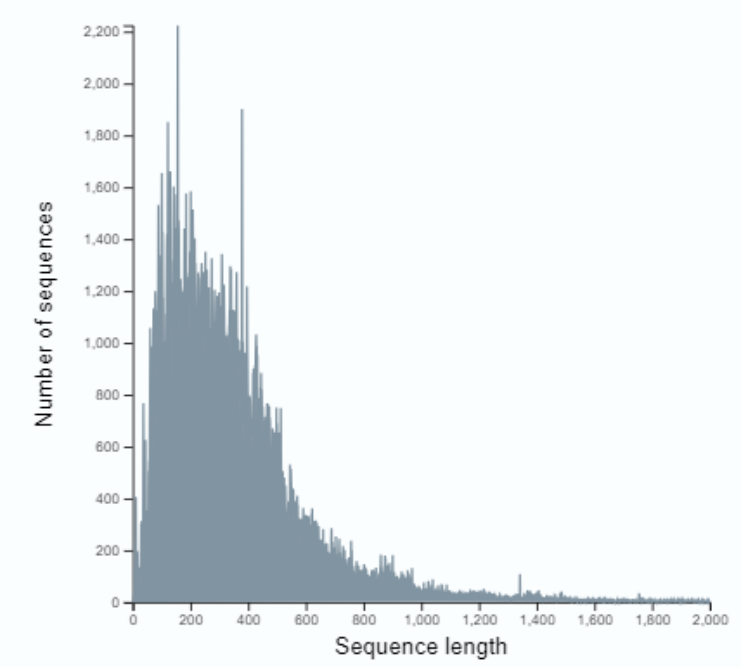
The following link has UniProt Statistics as of Feb 2024:
https://www.uniprot.org/uniprotkb/statistics#statistics-for-some-line-type

UniProtKB/TrEMBL : https://www.ebi.ac.uk/uniprot/TrEMBLstats

UniProtKB/Swiss-Prot: https://web.expasy.org/docs/relnotes/relstat.html

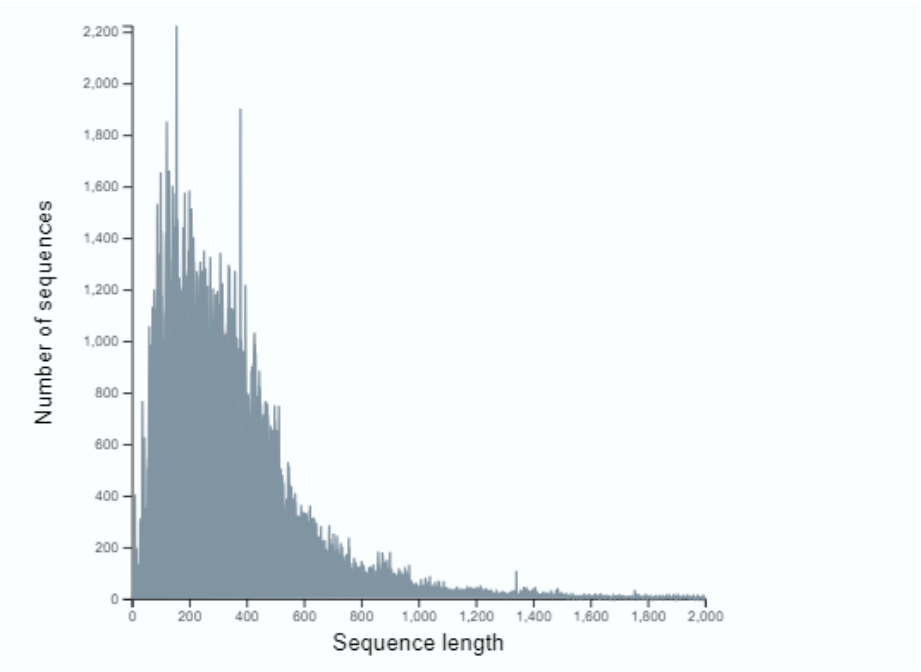
a) The sequence distribution chart plots the Number of sequences vs Sequence Length. We see a peak in the graph at range ~351 (TrEMBL) and ~361 (Swiss-Prot). The distribution is right-skewed, indicating more amino acids lie in the range of 0-600. We can also infer that it indicates the number of amino acids in the canonical sequence displayed by default in the entry's Sequence section.

a1) Reviewed (Swiss-Prot)



The average sequence length in UniProtKB/Swiss-Prot is 361 amino acids.

a2) Unreviewed (TrEMBL)



```
The average sequence length in UniProtKB/TrEMBL is   351 amino acids.
```

b)

b1) The shortest and longest sequence (in TrEMBL) are as follows:

```
The shortest sequence is A0A0G2JLF7_HUMAN:     7 amino acids.
The longest sequence is  A0A5A9P0L4_9TELE: 45354 amino acids.
```

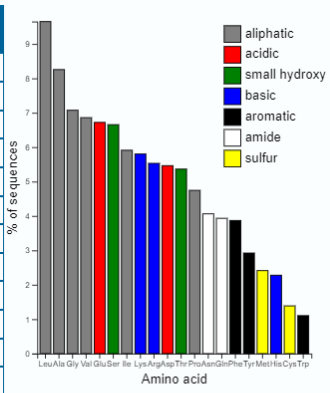
b2) The shortest and longest sequence (in SwissProt) are as follows:

```
The shortest sequence is   GWA_SEPOF (P83570):     2 amino acids.

The longest sequence is  TITIN_MOUSE (A2ASS6): 35213 amino acids.
```

c) The amino acid composition is as follows:

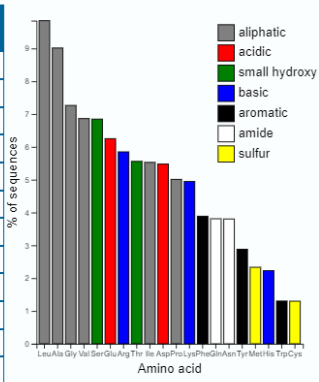
c1) Reviewed (Swiss-Prot)

Amino acid	Count	Percent	Entries with amino acid	Average count per reviewed entry
Leu	19,932,861	9.65%	567,649	34.92
Ala	17,051,081	8.26%	566,853	29.87
Gly	14,609,708	7.07%	567,747	25.59
Val	14,163,491	6.86%	567,085	24.81
Glu	13,880,281	6.72%	562,180	24.32
Ser	13,743,324	6.65%	567,138	24.08
Ile	12,207,933	5.91%	565,003	21.39
Lys	11,983,288	5.80%	563,811	20.99
Arg	11,420,988	5.53%	564,631	20.01
Asp	11,282,060	5.46%	561,315	19.76
Thr	11,076,320	5.36%	565,017	19.40
Pro	9,799,572	4.74%	561,695	17.17
Asn	8,391,346	4.06%	560,439	14.70
Gln	8,121,397	3.93%	557,812	14.23
Phe	7,989,951	3.87%	559,855	14.00
Tyr	6,036,657	2.92%	550,456	10.58
Met	4,983,743	2.41%	564,767	8.73
His	4,706,093	2.28%	539,008	8.24
Cys	2,864,637	1.39%	473,267	5.02
Trp	2,279,505	1.10%	454,465	3.99
AMINO_ACID_X	8,041	<0.01%	2,273	0.01
AMINO_ACID_U	329	<0.01%	254	0.00
AMINO_ACID_B	276	<0.01%	113	0.00
AMINO_ACID_Z	249	<0.01%	87	0.00
AMINO_ACID_O	29	<0.01%	29	0.00



c2) Unreviewed (TrEMBL)

Amino acid	Count	Percent	Entries with amino acid	Average count per unreviewed entry
Leu	8,646,856,405	9.84%	249,303,314	34.62
Ala	7,915,577,778	9.01%	248,721,380	31.69
Gly	6,378,270,739	7.26%	248,495,438	25.54
Val	6,027,240,933	6.86%	248,735,049	24.13
Ser	6,012,325,446	6.85%	248,832,843	24.07
Glu	5,489,092,503	6.25%	247,424,325	21.98
Arg	5,135,951,097	5.85%	247,778,023	20.56
Thr	4,881,727,427	5.56%	248,187,256	19.55
Ile	4,854,708,881	5.53%	247,736,715	19.44
Asp	4,809,963,037	5.48%	247,082,650	19.26
Pro	4,399,981,178	5.01%	246,327,937	17.62
Lys	4,342,592,769	4.94%	243,345,982	17.39
Phe	3,413,266,470	3.89%	245,203,498	13.67
Gln	3,345,659,092	3.81%	244,843,857	13.40
Asn	3,339,513,148	3.80%	243,271,710	13.37
Tyr	2,529,285,941	2.88%	240,531,652	10.13
Met	2,047,944,763	2.33%	247,749,460	8.20
His	1,956,609,024	2.23%	235,782,076	7.83
Trp	1,143,655,671	1.30%	215,592,125	4.58
Cys	1,139,776,125	1.30%	207,129,597	4.56
AMINO_ACID_X	22,129,989	0.03%	2,929,800	0.09
AMINO_ACID_U	20,769	<0.01%	19,804	0.00
AMINO_ACID_B	19,992	<0.01%	17,869	0.00
AMINO_ACID_Z	7,598	<0.01%	7,192	0.00
AMINO_ACID_O	311	<0.01%	305	0.00



Practical 4
26 Feb 2024

Questions

1. a. Write a program to construct a dot plot for the alignment of human and chicken hemoglobin β chain. Identify the segments, which are same in both sequences.

   b. Construct the dot plot manually for the residues 1-20 and verify with the plot obtained using program

2. Calculate the score for the following alignments **using code**:

   ```
   AATCTATA
   AAG--ATA
   ```

   Assume that the match score is 1, mismatch score is 0, origination penalty is -2, and length penalty is -1.

3. Verify the Q2 **manually**

4. Using the Needleman and Wunsch dynamic programming method, construct the partial alignment score table and align the following two sequences (**using code**):

   ACAGTCGAACG and ACCGTCCG

   use the scoring parameters: match score: +2; mismatch score: -1 and gap penalty: -2

5. Verify Q4 **manually**

6. Using the Smith-Waterman method, construct the partial alignment scoring table and align the following two sequences (**using code**):

   ACGTATCGCGTATA and GATGCGTATCG

   scoring parameters: match score: +2; mismatch score: -1 and gap penalty: -2

Deadline: 10 Mar 2024

BT 3040: BIOINFORMATICS

Assignment 4

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

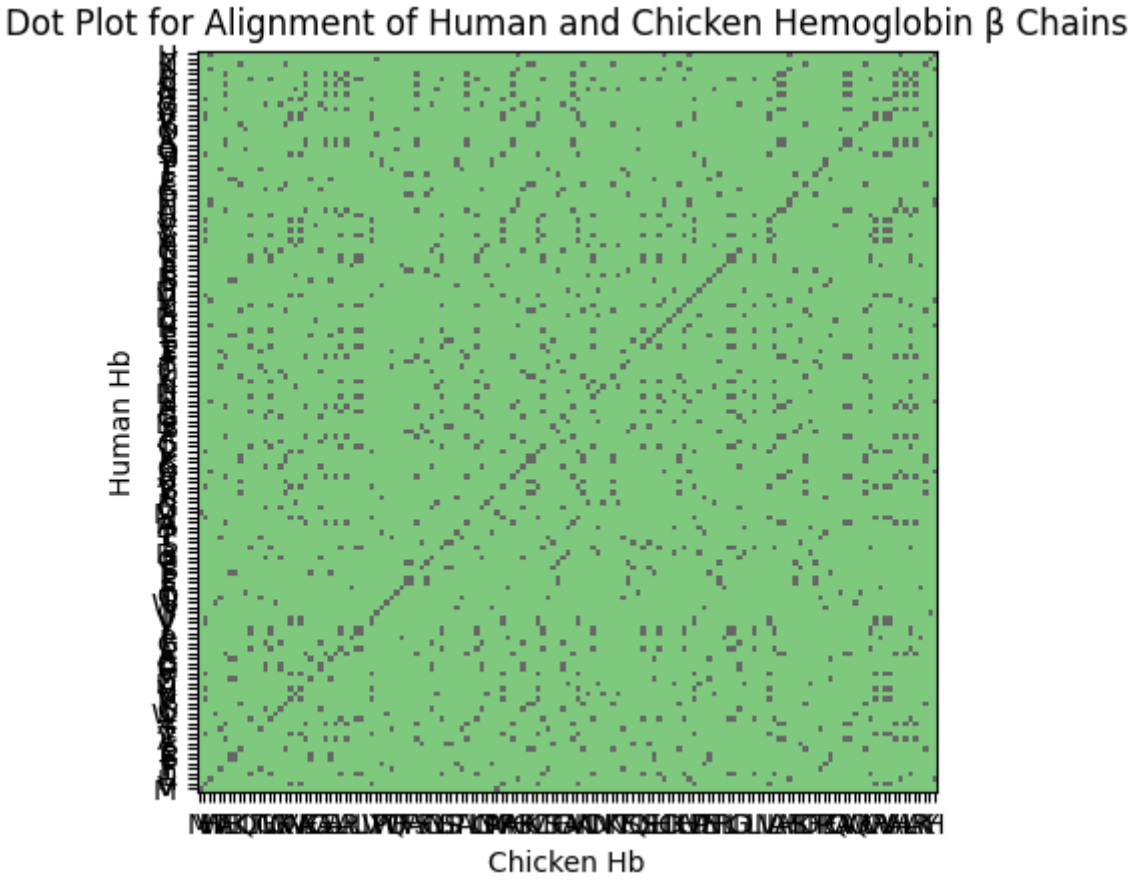
Q1) Write a program to construct a dot plot for the alignment of human and chicken hemoglobin β chain. Identify the segments, which are same in both sequences.

a)

```
# Atharva Mandar Phatak | BE21B009 | BT3040 Assignment 4
#Q1

import matplotlib.pyplot as plt

def dot_plot(seq1, seq2):
    len1 = len(seq1)
    len2 = len(seq2)

    dot_plot = [[0] * len2 for _ in range(len1)]

    # Fill the dot plot
    for i in range(len1):
        for j in range(len2):
            if seq1[i] == seq2[j]:
                dot_plot[i][j] = 1

    # Create the plot
    plt.imshow(dot_plot,cmap='Accent', interpolation='nearest')
    plt.xlabel('Chicken Hb')
    plt.ylabel('Human Hb')
    plt.title('Dot Plot for Alignment of Human and Chicken Hemoglobin β
Chains')
    plt.xticks(range(len2), seq2)
    plt.yticks(range(len1), seq1)
    plt.gca().invert_yaxis()
    plt.show()

def similar_seq(seq1, seq2):
    posn_vals= []
    for i in range(len(seq1)):
        if seq1[i] == seq2[i]:
            posn_vals.append(i+1)


    return posn_vals

#Given Sequences
human_hb =
"MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLA
HLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH"
```

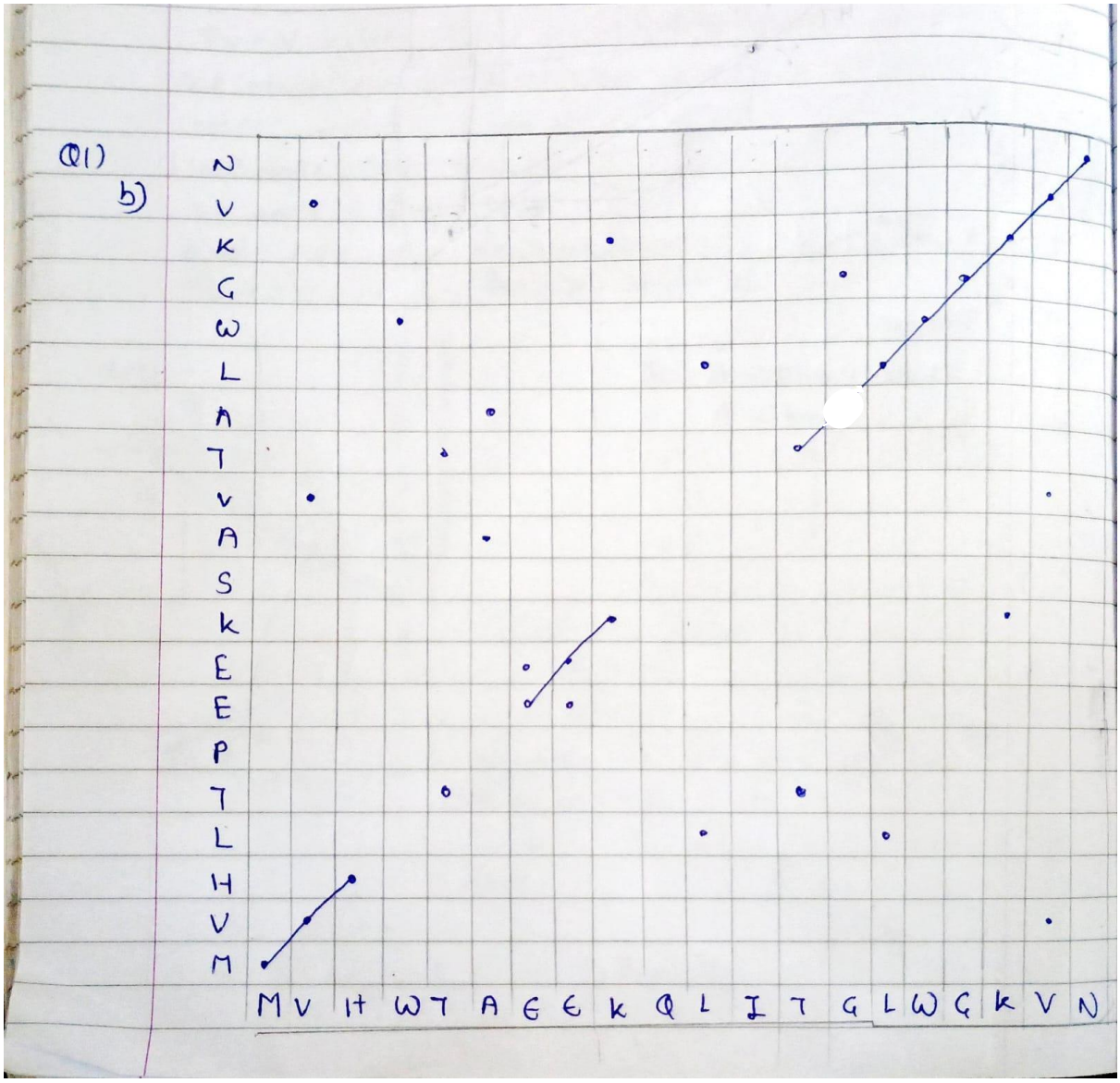
```
chicken_hb =
"MVHWTAEEKQLITGLWGKVNVAECGAEALARLLIVYPWTQRFFASFGNLSSPTAILGNPMVRAHGKKVLTSFGDAVK
NLDNIKNTFSQLSELHCDKLHVDPENFRLLGDILIIVLAAHFSKDFTPECQAAWQKLVRVVAHALARKYH"



dot_plot(human_hb, chicken_hb)
segments_indx = similar_seq(human_hb, chicken_hb)
#Align Output Sequence
def align_seq_output(segments):
    segs=segments
    seq=[]
    counter=0
    templst=[]

    while(counter<(len(segs)-1)):

        if counter==0:
            templst.append(segments[counter])

        if segs[counter+1]==(segs[counter]+1):
            templst.append(segments[counter+1])


        else:

            seq.append(templst)
            templst=[]
            templst.append(segments[counter+1])


        counter=counter+1


    seq_len=[len(i) for i in seq ]

    alignstr=""
    for i in seq[seq_len.index(max(seq_len))]:
        alignstr=alignstr+str(human_hb[i-1])
    print("Maximum matching sequence:", alignstr)

align_seq_output(segments_indx)
```

Dot Plot (Output):



Maximum matching sequence: LSELHCDKLHVDPENFRLLG

b) Manual Dot Plot



Longest Matching Sequence: WGKVN…

Q2) Calculate the score for the following alignments using code: AATCTATAAAG—ATA. Assume that the match score is 1, mismatch score is 0, origination penalty is -2, and length penalty is -1

```
# Atharva Mandar Phatak | BE21B009 | BT3040 Assignment 4
#Q2

inp_seq1='AATCTATA'
inp_seq2='AAG--ATA'


#Finding the total score (excluding penalty)
def score_find(seq1,seq2):
    score=0
    for i in range(len(seq1)):
        if(seq1[i]==seq2[i]):
            score=score+1
        elif(seq1[i]!=seq2[i]):
            score=score+0
    return score

F#Finding the penalty, and later subtracting from the total score
def penalty_find(seq2):
    count_arr=[]
    counter=0

    for i in range(len(seq2)):

        if(seq2[i]=='-'):
            counter+=1
        else:
            count_arr.append(counter)
            counter=0

    tot_penalty_vals=[]
    for i in count_arr:
        if(i!=0):
            tot_penalty_vals.append(i*(-1)-2)
    return(sum(tot_penalty_vals))

final_score= score_find(inp_seq1,inp_seq2)+penalty_find(inp_seq2)
print("Final Score =",final_score)
```

Output:

```
Final Score = 1
```

Q3) Verify the Q2 manually

Q2)

AATCTATA

AAG--ATA

+1 +1 +0 $\underbrace{}_{-4}$ +1 +1 H

$= 5 - 4 = 1$

$\therefore \boxed{\text{score} = 1}$.

Q4) Using the Needleman and Wunsch dynamic programming method, construct the partial alignment score table and align the following two sequences (using code): ACAGTCGAACG and ACCGTCCG use the scoring parameters: match score: +2; mismatch score: -1 and gap penalty: -2

```
# Atharva Mandar Phatak | BE21B009 | BT3040 Assignment 4
#Q4

def needleman_wunsch(seq1, seq2, match_score=+2, mismatch_penalty=-1,
gap_penalty=-2):
    # Initialize the score matrix
    score_matrix = [[0] * (len(seq2) + 1) for _ in range(len(seq1) + 1)]

    # Initialize the traceback matrix to store directions
    traceback_matrix = [[0] * (len(seq2) + 1) for _ in range(len(seq1) + 1)]

    # Initialize the first row and column of the score matrix
    for i in range(1, len(seq1) + 1):
        score_matrix[i][0] = gap_penalty * i
        traceback_matrix[i][0] = 'U'  # 'U' indicates up

    for j in range(1, len(seq2) + 1):
        score_matrix[0][j] = gap_penalty * j
        traceback_matrix[0][j] = 'L'  # 'L' indicates left

    # Fill the score and traceback matrices
    for i in range(1, len(seq1) + 1):
        for j in range(1, len(seq2) + 1):
            match = score_matrix[i - 1][j - 1] + (match_score if seq1[i - 1]
== seq2[j - 1] else mismatch_penalty)
            delete = score_matrix[i - 1][j] + gap_penalty
            insert = score_matrix[i][j - 1] + gap_penalty
            score_matrix[i][j] = max(match, delete, insert)

            # Update traceback matrix
            if score_matrix[i][j] == match:
                traceback_matrix[i][j] = 'D'  # 'D' indicates diagonal
            elif score_matrix[i][j] == delete:
                traceback_matrix[i][j] = 'U'  # 'U' indicates up
            else:
                traceback_matrix[i][j] = 'L'  # 'L' indicates left

    # Traceback to find the alignment
    alignment_seq1 = ''
    alignment_seq2 = ''
    i, j = len(seq1), len(seq2)
```

```python
    while i > 0 or j > 0:
        if traceback_matrix[i][j] == 'D':
            alignment_seq1 = seq1[i - 1] + alignment_seq1
            alignment_seq2 = seq2[j - 1] + alignment_seq2
            i -= 1
            j -= 1
        elif traceback_matrix[i][j] == 'U':
            alignment_seq1 = seq1[i - 1] + alignment_seq1
            alignment_seq2 = '-' + alignment_seq2
            i -= 1
        else:
            alignment_seq1 = '-' + alignment_seq1
            alignment_seq2 = seq2[j - 1] + alignment_seq2
            j -= 1

    return score_matrix, alignment_seq1, alignment_seq2

# Example sequences
sequence1 = "ACAGTCGAACG"
sequence2 = "ACCGTCCG"

# Call the needleman_wunsch function
score_table, aligned_seq1, aligned_seq2 = needleman_wunsch(sequence1,
sequence2)

# Print the alignment and the partial alignment score table
print("Sequence 1:", aligned_seq1)
print("Sequence 2:", aligned_seq2)

print("\nPartial Alignment Score Table:")
for row in score_table:
    print(row)
print(" ")
print(f"Aligmnent Score={max(score_table[-1])}")
```

Output

```
Sequence 1: ACAGTCGAACG
Sequence 2: ACCGTC---CG

Partial Alignment Score Table:
[0, -2, -4, -6, -8, -10, -12, -14, -16]
[-2, 2, 0, -2, -4, -6, -8, -10, -12]
[-4, 0, 4, 2, 0, -2, -4, -6, -8]
[-6, -2, 2, 3, 1, -1, -3, -5, -7]
[-8, -4, 0, 1, 5, 3, 1, -1, -3]
[-10, -6, -2, -1, 3, 7, 5, 3, 1]
[-12, -8, -4, 0, 1, 5, 9, 7, 5]
[-14, -10, -6, -2, 2, 3, 7, 8, 9]
[-16, -12, -8, -4, 0, 1, 5, 6, 7]
[-18, -14, -10, -6, -2, -1, 3, 4, 5]
[-20, -16, -12, -8, -4, -3, 1, 5, 3]
[-22, -18, -14, -10, -6, -5, -1, 3, 7]

Aligmnent Score = 7
```

Q5) Verify Q4 manually

Q5)

		A	C	C	G	T	C	C	G
	0	-2	-4	-6	-8	-10	-12	-14	-16
A	-2	2	0	-2	-4	-6	-8	-10	-12
C	-4	0	4	2	0	2	-4	-6	-8
A	-6	2	2	3	1	-1	-3	-5	-7
G	-8	-4	0	1	5	3	1	-1	-3
T	-10	-6	-2	-1	3	7	5	3	1
C	-12	-8	-4	0	1	5	9	7	5
G	-14	-10	-6	-2	2	3	7	8	9
A	-16	-12	-8	-4	0	1	5	6	7
A	-18	-14	-10	-6	-2	-1	3	4	5
C	-20	-16	-12	-8	-4	-3	1	5	3
G	-22	-18	-14	-10	-6	-5	-1	3	7

score = 7

Sequence

A C A G T C G A A C G

A C C G T C - - - C G

Q6) Using the Smith-Waterman method, construct the partial alignment scoring table and align the following two sequences (using code): ACGTATCGCGTATA and GATGCGTATCG scoring parameters: match score: +2; mismatch score: -1 and gap penalty: -2

```
# Atharva Mandar Phatak | BE21B009 | BT3040 Assignment 4
#Q6
def smith_waterman(seq1, seq2, match=2, mismatch=-1, gap=-2):
    # Initialize the scoring matrix
    rows = len(seq1) + 1
    cols = len(seq2) + 1
    score_matrix = [[0] * cols for _ in range(rows)]

    # Fill the scoring matrix
    for i in range(1, rows):
        for j in range(1, cols):
            if seq1[i - 1] == seq2[j - 1]:
                diag_score = score_matrix[i - 1][j - 1] + match
            else:
                diag_score = score_matrix[i - 1][j - 1] + mismatch

            up_score = score_matrix[i - 1][j] + gap
            left_score = score_matrix[i][j - 1] + gap

            score_matrix[i][j] = max(0, diag_score, up_score, left_score)

    # Find the maximum score in the matrix
    max_score = 0
    max_i, max_j = 0, 0
    for i in range(rows):
        for j in range(cols):
            if score_matrix[i][j] > max_score:
                max_score = score_matrix[i][j]
                max_i, max_j = i, j

    # Traceback to find the alignment
    alignment_seq1 = ""
    alignment_seq2 = ""
    i, j = max_i, max_j
    while score_matrix[i][j] != 0:
        if score_matrix[i][j] == score_matrix[i - 1][j - 1] + (match if seq1[i
- 1] == seq2[j - 1] else mismatch):
            alignment_seq1 = seq1[i - 1] + alignment_seq1
            alignment_seq2 = seq2[j - 1] + alignment_seq2
            i -= 1
```

```
            j -= 1
        elif score_matrix[i][j] == score_matrix[i - 1][j] + gap:
            alignment_seq1 = seq1[i - 1] + alignment_seq1
            alignment_seq2 = '-' + alignment_seq2
            i -= 1
        else:
            alignment_seq1 = '-' + alignment_seq1
            alignment_seq2 = seq2[j - 1] + alignment_seq2
            j -= 1

    return alignment_seq1, alignment_seq2, max_score, score_matrix

def print_matrix(matrix):
    for row in matrix:
        print(row)

def main():
    seq1 = "ACGTATCGCGTATA"
    seq2 = "GATGCGTATCG"

    alignment_seq1, alignment_seq2, max_score, score_matrix =
smith_waterman(seq1, seq2)

    print("Alignment Sequence 1:", alignment_seq1)
    print("Alignment Sequence 2:", alignment_seq2)
    print("Alignment Score:", max_score)
    print("\nScoring Matrix:")
    print_matrix(score_matrix)

if __name__ == "__main__":
    main()
```

Output:

```
Alignment Sequence 1: CGTATCG
Alignment Sequence 2: CGTATCG
Alignment Score: 14

Scoring Matrix:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 2, 0, 0, 0, 0, 0, 2, 0, 0, 0]
[0, 0, 0, 1, 0, 2, 0, 0, 0, 1, 2, 0]
[0, 2, 0, 0, 3, 1, 4, 2, 0, 0, 0, 4]
[0, 0, 1, 2, 1, 2, 2, 6, 4, 2, 0, 2]
[0, 0, 2, 0, 1, 0, 1, 4, 8, 6, 4, 2]
[0, 0, 0, 4, 2, 0, 0, 3, 6, 10, 8, 6]
[0, 0, 0, 2, 3, 4, 2, 1, 4, 8, 12, 10]
[0, 2, 0, 0, 4, 2, 6, 4, 2, 6, 10, 14]
[0, 0, 1, 0, 2, 6, 4, 5, 3, 4, 8, 12]
[0, 2, 0, 0, 2, 4, 8, 6, 4, 2, 6, 10]
[0, 0, 1, 2, 0, 2, 6, 10, 8, 6, 4, 8]
[0, 0, 2, 0, 1, 0, 4, 8, 12, 10, 8, 6]
[0, 0, 0, 4, 2, 0, 2, 6, 10, 14, 12, 10]
[0, 0, 2, 2, 3, 1, 0, 4, 8, 12, 13, 11]
```

Q6)

		G	A	T	G	C	G	T	A	T	C	G
	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	2	0	0	0	0	0	2	0	0	0
C	0	0	0	1	0	2	0	0	0	1	2	0
G	0	2	0	0	3	1	4	2	0	0	0	4
T	0	0	1	2	1	2	2	6	4	2	0	2
A	0	0	2	0	1	0	1	4	8	6	4	2
T	0	0	0	4	2	0	0	3	6	10	8	6
C	0	0	0	2	3	4	2	1	4	8	12	10
G	0	2	0	0	4	2	6	4	2	6	10	14
C	0	0	1	0	2	6	4	5	3	4	8	12
G	0	2	0	0	2	4	8	6	4	2	6	10
T	0	0	1	2	0	2	6	10	8	6	4	8
A	0	0	2	0	1	0	4	8	12	10	8	6
T	0	0	0	4	2	0	2	6	10	14	12	10
A	0	0	2	2	3	1	0	4	8	12	13	11

$\boxed{\text{Score} = 14}$

Aligned Sequence

```
CGTATCG
||||||
CGTATCG
```

Practical 5
29 Feb 2024

Questions

1. Analyze the occurrence of similar proteins in “nr” and SWISS-PROT database for the sequence given below:
```
>1336093|Genbank|Outer membrane integral membrane protein|HrcC
MVEKRELRCRLLGALLMLCATLPAGAQTPADWKEQSYAYSADRTPLSTVLQDFADGHSVD
LHLGNVEDTEVTAKIRAENASAFLDRLALEHHFQWFVYNNTLYVSPQDEQSSERLEISPD
AAPDIKQALSGIGLLDPRFGWGELPDDGVVLVTGPPQYLELVKRFSEQREKKEDRRKVMT
FPLRYASVADRTIHYRDQTVVIPGVATMLNELMNGKRAAPASASGIDSTPGGPDTNSMMQ
NTQTLLSRLSSRNKTSNRAGGRDNEIEDVSGRISADVRNNALLIRDDDKRHDEYSQLIAK
IDVPQNLVEIDAVILDIDRTALNRLEANWQATLGGVTGGSSLMSGSGTLFVSDFKRFFAD
IQALEGEGTASIVANPSVLTLENQPAVIDFSQTAYITATGERVADIQPVTAGTSLQVTPR
AVGNEGHSSIQLMIDIEDGHVQTNGDGQATGVKRGTVSTQALISENRALVLGGFHVEESA
DRDRRIPLLGDIPWLGQLFSSKRHEISQRQRLFILTPRLIGDQTDPTRYVTADNRQQLSD
AMGRVERRHSSVNQHDVVENALRDLAEGQSPAGFQPQTSGTRLSEVCRSTPALLFESTRG
QWYSSSTNGVQLSVGVVRNTSSKPLRFDEANCASKRTLAVAVWPHSALAPGESAEVYLAM
DPSRVLHASRESLLNR
```

2. List the algorithm parameters used for the search (Q1).

3. What is the sequence identity of the query sequence (given in Q1) with AAK81929.1 ?

4. How far are hemoglobin (beta) sequences in humans and chicken similar?

5. Write a program to list all the matching pentapeptides (which occur in both the sequences) and their frequency of occurrence in given sequences.

6. Write a program to compute sequence identity, similarity, query coverage and gap percentage from the alignment of human and chicken hemoglobin sequences (refer Q4).

7. Obtain the multiple sequence alignment for TIM barrel proteins from different organisms (select 20 proteins, for example). Compare the results obtained with Clustal Omega, MAFFT, and MUSCLE. List 5 residue positions which are aligned differently in these three methods.

8. Blast the below sequence 'EPDMRTPIAHTMAW' against the PDB database. Analyze the results and discuss the significance of the results.

Deadline: 13 March 2024

BT 3040: BIOINFORMATICS

Assignment 5

Atharva Mandar Phatak | BE21B009

Indian Institute of Technology
Madras

Q1) Analyze the occurrence of similar proteins in "nr" and SWISS-PROT database for the sequence given below:

The following tables provide the analysis of occurrence of similar protiens.

nr		
	Min	Max
Max Score	926	1387
Total Score	1387	1387
Query Cover	90%	100%
e value	0	0
Percentage Identity	66.96	100%
Accession Length	617	676

SwissProt		
	Min	Max
Max Score	42	566
Total Score	42	566
Query Cover	23%	98%
e value	0	0.017
Percentage Identity	21.75	44.16
Accession Length	234	777

NIH National Library of Medicine
National Center for Biotechnology Information

Log in

BLAST ® » blastp suite » results for RID-Y03GGDXX016

Home Recent Results Saved Strategies Help

< Edit Search | Save Search | Search Summary

How to read this report? | BLAST Help Videos | Back to Traditional Results Page

Job Title	1336093|Genbank|Outer membrane integral membrane...
RID	Y03GGDXX016 Search expires on 03-01 11:54 am Download All
Program	BLASTP Citation
Database	nr See details
Query ID	lcl|Query_6735452
Description	1336093|Genbank|Outer membrane integral membrane pr ...
Molecule type	amino acid
Query Length	675
Other reports	Distance tree of results Multiple alignment MSA viewer

Filter Results

Organism only top 20 will appear exclude

Type common name, binomial, taxid or group name

+ Add organism

Percent Identity to | E value to | Query Coverage to

Filter Reset

Compare these results against the new Clustered nr database BLAST

Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments

Download Select columns Show 100

select all 100 sequences selected

GenPept Graphics Distance tree of results Multiple alignment MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per ident	Acc Len	Accession
type III secretion system outer membrane ring subunit SctC [Erwinia amylovora]	Erwinia amylovora	1380	1380	100%	0.0	99.85%	675	WP_004155366.1
type III secretion system outer membrane ring subunit SctC [Erwinia amylovora]	Erwinia amylovora	1379	1379	100%	0.0	99.70%	675	WP_160421624.1
type III secretion system outer membrane ring subunit SctC [Erwinia amylovora]	Erwinia amylovora	1379	1379	100%	0.0	99.70%	675	WP_160305178.1
type III secretion system outer membrane ring subunit SctC [Erwinia amylovora]	Erwinia amylovora	1364	1364	100%	0.0	98.02%	677	WP_004160438.1

NIH National Library of Medicine
National Center for Biotechnology Information

Log in

BLAST ® » blastp suite » results for RID-Y03N10MC016

Home Recent Results Saved Strategies Help

< Edit Search | Save Search | Search Summary

How to read this report? | BLAST Help Videos | Back to Traditional Results Page

Job Title	1336093|Genbank|Outer membrane integral membrane...
RID	Y03N10MC016 Search expires on 03-01 11:57 am Download All
Program	BLASTP Citation
Database	swissprot See details
Query ID	lcl|Query_7061270
Description	1336093|Genbank|Outer membrane integral membrane pr ...
Molecule type	amino acid
Query Length	675
Other reports	Distance tree of results Multiple alignment MSA viewer

Filter Results

Organism only top 20 will appear exclude

Type common name, binomial, taxid or group name

+ Add organism

Percent Identity to | E value to | Query Coverage to

Filter Reset

Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments

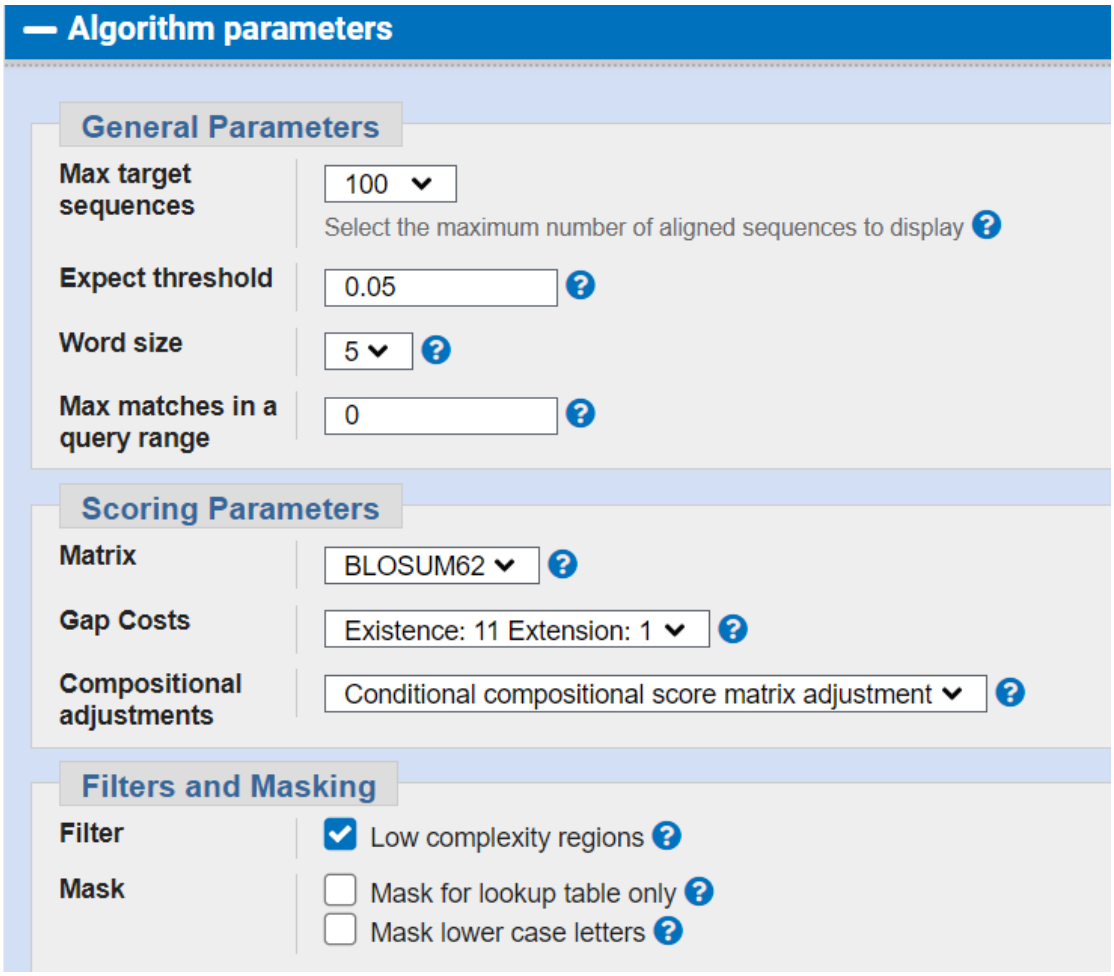
Download Select columns Show 100

select all 37 sequences selected

GenPept Graphics Distance tree of results Multiple alignment MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
RecName: Full=Type 3 secretion system secretin; Short=T3SS secretin; AltName: Full=Hypersensitivity response sec ...	Pseudomonas sy...	563	563	98%	0.0	44.16%	701	Q01723.2
RecName: Full=Type 3 secretion system secretin; Short=T3SS secretin; AltName: Full=YscC secretin; Flags: Precurs...	Yersinia enterocol ...	243	243	72%	1e-70	31.05%	607	Q01244.1
RecName: Full=Type 3 secretion system secretin; Short=T3SS secretin; AltName: Full=Type III secretion protein Ysc...	Yersinia pestis	239	239	75%	8e-69	30.34%	607	Q56974.1
RecName: Full=Type 3 secretion system secretin; Short=T3SS secretin; AltName: Full=Hypersensitivity response sec...	Ralstonia pseudo...	204	204	75%	3e-56	28.87%	568	Q57498.1

Q2) List the algorithm parameters used for the search (Q1)



- General parameters displayed:
    1. Max target sequences
    2. Expected threshold
    3. Word size
    4. Maximum matches in a query range
- Scoring Parameters
    1. Matrix
    2. Gap costs
    3. Compositional adjustments
- Filter and Masking
    1. Filter
    2. Mask

Q3) What is the sequence identity of the query sequence (given in Q1) with AAK81929.1

Job Title	AAB49179:HrcC [Erwinia amylovora]
RID	YYUTE6EF114 *Search expires on 03-13 04:35 am* Download All
Program	Blast 2 sequences Citation
Query ID	AAB49179.1 (amino acid)
Query Descr	HrcC [Erwinia amylovora]
Query Length	676
Subject ID	AAK81929.1 (amino acid)
Subject Descr	RscC [Pseudomonas fluorescens]
Subject Length	713
Other reports	Multiple alignment MSA viewer

Filter Results

Percent Identity		E value		Query Coverage	
	to		to		to

Filter Reset

Descriptions | Graphic Summary | Alignments | Dot Plot

Sequences producing significant alignments Download Select columns Show 100

select all *1 sequences selected* GenPept Graphics Multiple alignment MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
RscC [Pseudomonas fluorescens]	Pseudomonas fluorescens	530	530	96%	0.0	43.20%	713	AAK81929.1

- Max Score = 530
- Total Score = 530
- Query Cover = 96%
- E Value = 0
- **Per. Ident = 43.20%**
- Acc. Len = 713

Q4) How far are hemoglobin (beta) sequences in humans and chicken similar?

Job Title	P68871:RecName: Full=Hemoglobin subunit beta;...
RID	YZDJYV5W114 *Search expires on 03-13 09:56 am* Download All
Program	Blast 2 sequences Citation
Query ID	P68871.2 (amino acid)
Query Descr	RecName: Full=Hemoglobin subunit beta; AltName: Full=B ...
Query Length	147
Subject ID	P02112.2 (amino acid)
Subject Descr	RecName: Full=Hemoglobin subunit beta; AltName: Full=B ...
Subject Length	147
Other reports	Multiple alignment MSA viewer

Filter Results

Percent Identity		E value		Query Coverage	
	to		to		to

Filter Reset

Descriptions | Graphic Summary | Alignments | Dot Plot

Sequences producing significant alignments Download Select columns Show 100

select all *1 sequences selected* GenPept Graphics Multiple alignment MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
RecName: Full=Hemoglobin subunit beta; AltName: Full=Beta-globin; AltName: Full=Hemoglobin beta chain [Gallus gallus]	Gallus gallus	221	221	100%	1e-80	69.39%	147	P02112.2

When searched for Haemoglobin sequences in UniProt, we get Human and Chicken protein sequences which are analysed in blast and the Percentage Identity is 69.39%. (Where the query cover is 100%). The percentage Identity is **69.39%**

Q5) Write a program to list all the matching pentapeptides (which occur in both the sequences) and their frequency of occurrence in given sequences.

Code: https://colab.research.google.com/drive/1w2RDIZpFqqW5EFqo6AZ7fBpcZ-ZNdsNA?usp=sharing

Output:

```
The frequency of WTQRF in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of HVDPE in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of PWTQR in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of FRLLG in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of SELHC in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of TQRFF in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of LSELH in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of GKKVL in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of PENFR in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of LHVDP in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of LWGKV in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of AHGKK in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of YPWTQ in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of WGKVN in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of VYPWT in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of KLHVD in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of NFRLL in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of VDPEN in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of CDKLH in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of HCDKL in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of ELHCD in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of DPENF in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of HGKKV in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of ENFRL in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of GKVNV in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of LHCDK in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
The frequency of DKLHV in Human Hb Sequence is: 1 and in Chicken Hb Sequence is: 1
```

Q6) Write a program to compute sequence identity, similarity, query coverage and gap percentage from the alignment of human and chicken hemoglobin sequences (refer Q4).

```
seq_identity(human,chicken)
69.38775510204081

seq_similarlity(human,chicken,matrix)
82.31292517006803

gap_percentage(human,chicken)
0.0

query_coverage(human,chicken)
query_coverage for human is:  100.0
query_coverage for chicken is:  100.0
```

Sequence identity: 69.38775%

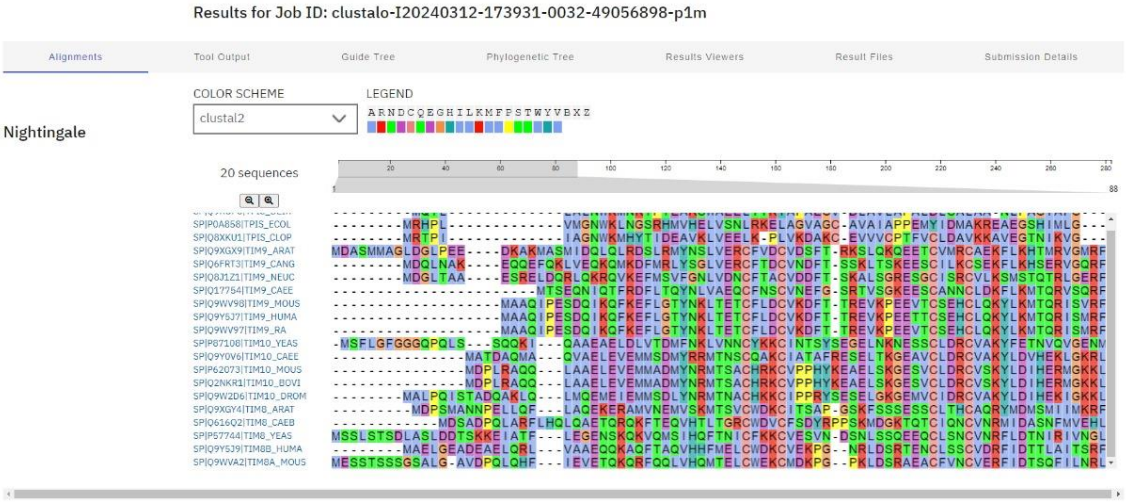
Sequence similarity: 82.3129%

Gap percentage: 0%

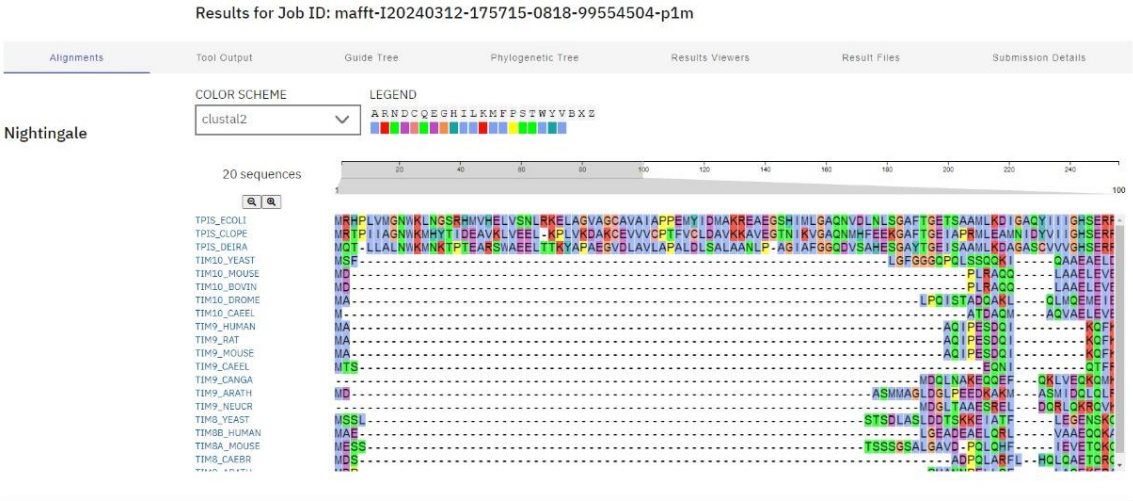
Query Coverage: 100%

Q7) Obtain the multiple sequence alignment for TIM barrel proteins from different organisms (select 20 proteins, for example). Compare the results obtained with Clustal Omega, MAFFT, and MUSCLE. List 5 residue positions which are aligned differently in these three methods.

a) Clustal



b) MAFFT

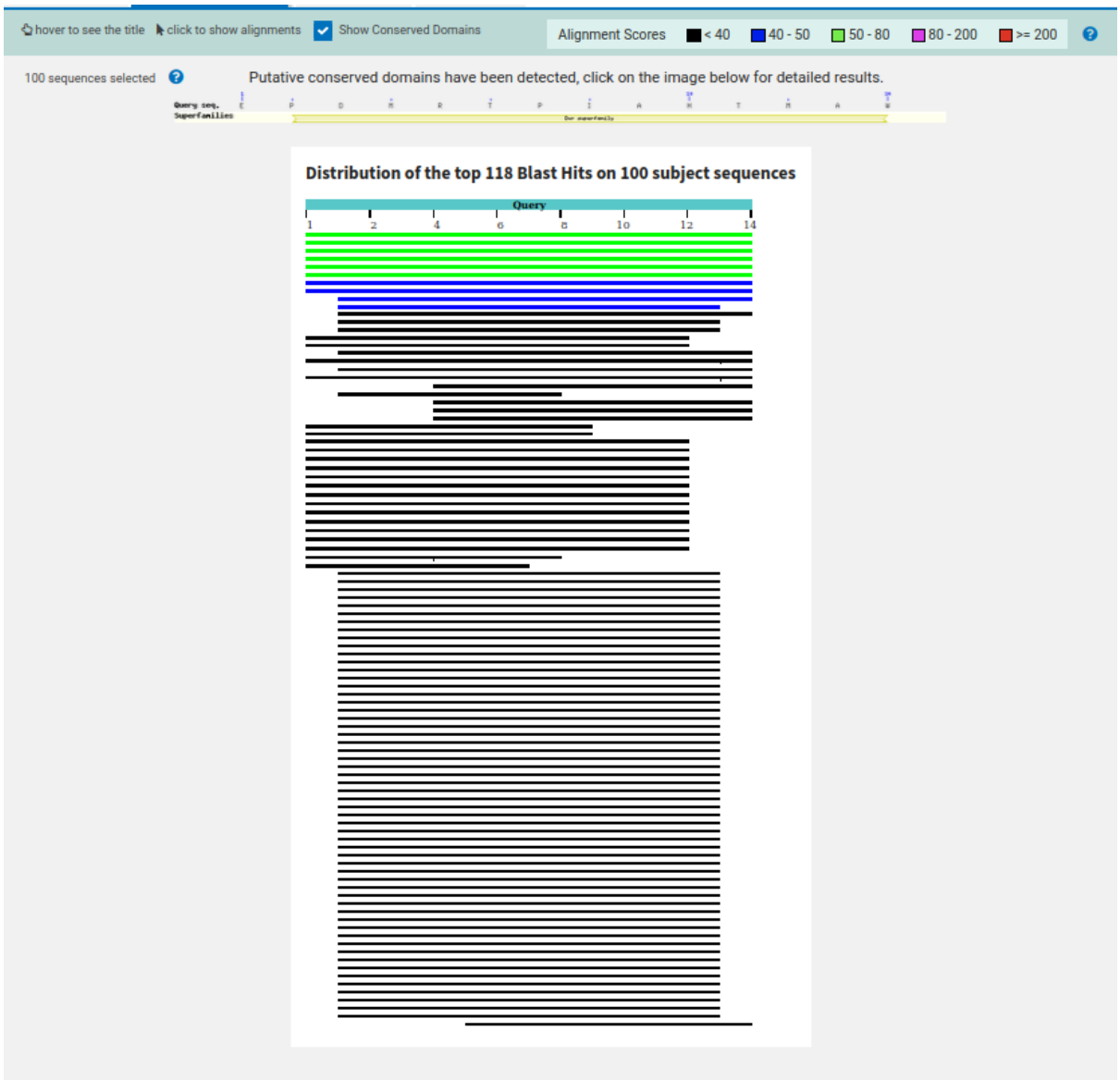


c) MUSCLE

Analysis:

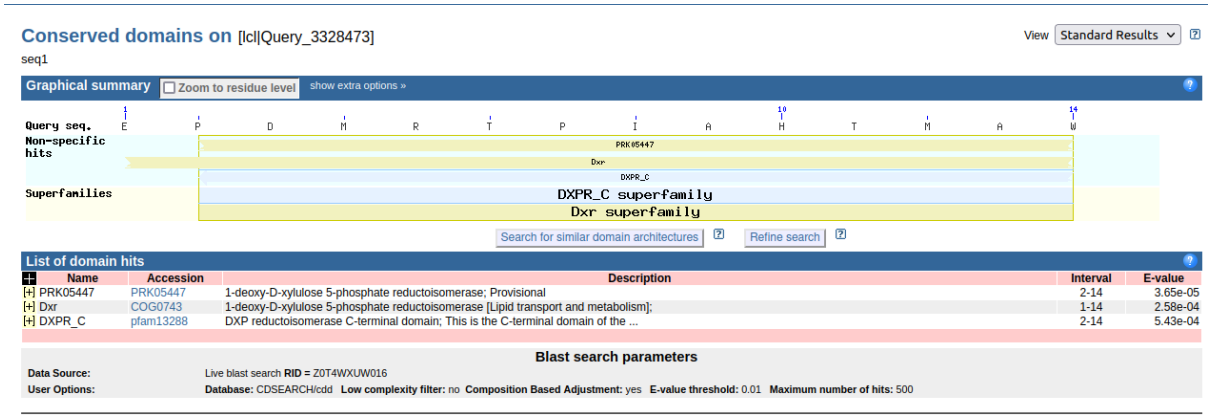
The lengths of the aligned sequences are 120, 117, and 116 for Clustal Omega, MAFFT, and MUSCLE, respectively. It's evident that the alignment of residues at positions 1, 2, 3, and 4 varies across these three methods. As we move further along the residue positions, the alignments become increasingly diverse.

Q8) Blast the below sequence 'EPDMRTPIAHTMAW' against the PDB database. Analyze the results and discuss the significance of the results.



Graphical representation of the BLAST sequence

PDB		
	Min	Max
Max Score	22.30%	53.20%
Total Score	22.30%	53.20%
Query Cover	42%	100%
e value	2.00E-10	21
Percentage Identity	57.14%	100%
Accession Length	201	1290



Thus there are three domain hits for the given sequence.

Domain Hits				
Name	Accession	Description	Interval	E-value
PRK05447	PRK05447	1-deoxy-D-xylulose 5-phosphate reductoisomerase; Provisional	2-14	0.00365%
Dxr	COG0743	1-deoxy-D-xylulose 5-phosphate reductoisomerase [Lipid transport and metabolism];	1-14	0.0258%
DXPR_C	pfam13288	DXP reductoisomerase C-terminal domain; This is the C-terminal domain of the ...	1-14	0.0543%

- Residue 2-14 is a domain named PRK05447 which is conserved across many organisms.
- BLAST results show this sequence is commonly found in - Escherichia coli, Klebsiella pneumoniae.
- 100 sequences, only 10% of the sequences have very minimal E value.

Practical 6
11 March 2024

Questions

1. Using AL2CO server (http://prodata.swmed.edu/al2co/al2co.php), obtain the positional conservation scores from multiple sequence alignment (MSA) of given set of protein sequences (set1 and set2) using the methods given below:

(i) Unweighted frequency and entropy-based measure
(ii) Unweighted frequency and variance-based measure
(iii) Unweighted frequency and sum of pairs measure
(iv) Weighted frequency and variance-based measure
(v) Normalize the scores obtained with (i)

Sequences:

Set 1
P69905, P01946, P01942, P01966, P01958, P01959, P01965, P06635, P60529, P80043 and P01980

Set 2
TPIS_HUMAN, TPIS_YEAST, TPIS_GRAGA, TPIS_TRYCR, TPIS_MAIZE, TPIS_MOUSE, TPIS_DROME, TPIS_RABIT, TPIS_CAEEL

2. Tabulate the topmost 10 residues with highest and lowest conservation scores (in both Set1 and Set 2) obtained with method (i).

3. Write a program to compute the conservation score from MSA using unweighted frequency, and entropy, variance and sum of pairs-based measures.

4. Using the program written in Q3 (unweighted frequency and entropy-based measure), compare the MSA from Clustal Omega, MAFFT, and MUSCLE. Identify the residues with (i) similar and (ii) different conservation scores among the three alignment methods.

5. Check the scores manually at positions 9, 11, 20, 22 and 30 (use MSA from Clustal Omega)

6. Obtain the conservation score of 1BTM, A-chain using Consurf server (https://consurf.tau.ac.il/)

Deadline: 17th March 2024

BT 3040: BIOINFORMATICS

Assignment 6

Atharva Mandar Phatak | BE21B009

Indian Institute of Technology
Madras

 # Combined data for Q1 & Q2 can be found in 'Excel_Assignment6_Q1&Q2'

Q1) Using AL2CO server (http://prodata.swmed.edu/al2co/al2co.php), obtain the positional conservation scores from multiple sequence alignment (MSA) of given set of protein sequences (set1 and set2) using the methods given below:

(i) Unweighted frequency and entropy-based measure

(ii) Unweighted frequency and variance-based measure

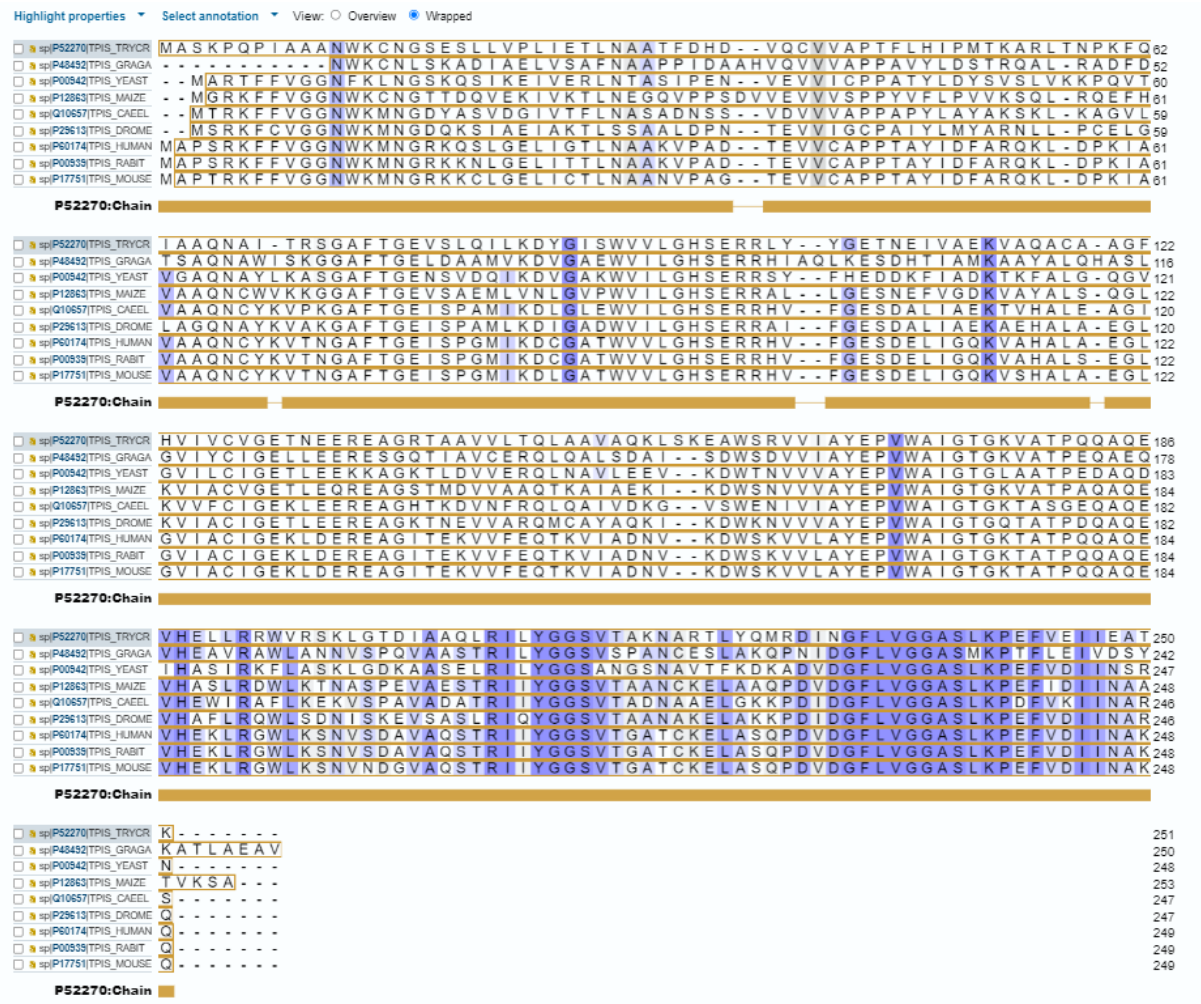
(iii) Unweighted frequency and sum of pairs measure

(iv) Weighted frequency and variance-based measure

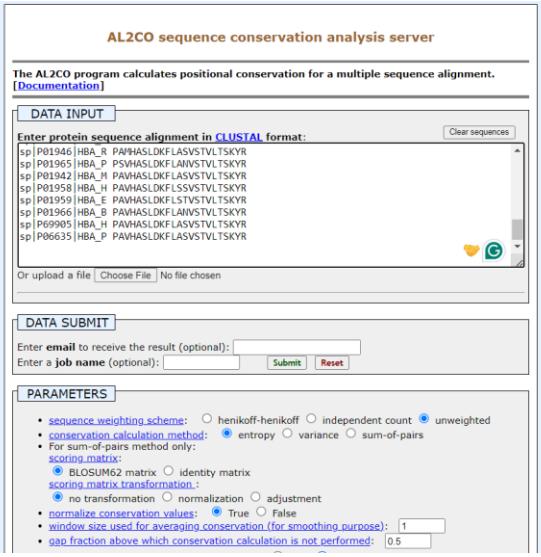
(v) Normalize the scores obtained with (i)

a. Set 1 Alignment

b. Set 2 Alignment



c. AL2CO interface



AL2CO alignment conservation server

RESULTS:

- The list of positional conservation values is here.
- The alignment with integer conservation indices is here.

INPUTS:

- Input alignment is here.
- Input pdb file: none.

 # Based on the question, we can tune the parameters and obtain the results. The outputs are mentioned in separate folders for Set 1 and Set 2

Q2) Tabulate the topmost 10 residues with highest and lowest conservation scores (in both Set1 and Set 2) obtained with method (i).

a. Set 1

Lowest Conservation Score Set 1		
Position	Residue	Condition 1
117	K	-1.846
113	T	-1.72
70	L	-1.673
73	S	-1.594
84	E	-1.414
133	G	-1.414
36	V	-1.367
35	V	-1.295
9	A	-1.169
6	K	-1.16

Highest Conservation Score Set 1		
Position	Residue	Condition 1
1	:	0
3	L	0
4	S	0
7	D	0
8	K	0
17	K	0
26	G	0
30	L	0
32	R	0
38	P	0

b. Set 2

Lowest Conservation Score Set 2		
Position	Residue	Condition 1
33	T	-1.889
64	Q	-1.831
147	A	-1.831
27	E	-1.735
23	V	-1.677
36	H	-1.677
49	L	-1.677
93	S	-1.677
145	R	-1.677
145	L	-1.677

Highest Conservation Score Set 2		
Position	Residue	Condition 1
12	N	0
14	K	0
16	N	0
43	V	0
68	Q	0
69	N	0
76	G	0
77	A	0
78	F	0
79	T	0

Q3) Write a program to compute the conservation score from MSA using unweighted frequency, and entropy, variance and sum of pairs-based measures.

```
#Atharva Mandar Phatak| BE21B009 | BT3040 Assignment 6 Q1|
import numpy as np

def calculate_frequencies(dat, aminoacids):
    totseq = len(dat)
    lenseq = len(dat[0])
    unweightedfreq = np.zeros((lenseq, 20))

    for i in range(lenseq):
        for j in range(totseq):
            if dat[j][i].isalpha():
                a = aminoacids.index(dat[j][i])
                unweightedfreq[i][a] += 1

    return unweightedfreq / totseq

def calculate_entropy_and_variance(unweightedfreq, aminoacidfreq):
    entropy = np.zeros(len(unweightedfreq))
    variance = np.zeros(len(unweightedfreq))

    for i, freq in enumerate(unweightedfreq):
        for j, f in enumerate(freq):
            if f != 0:
                entropy[i] += f * np.log(f)
            variance[i] += (f - aminoacidfreq[j]) ** 2

    entropy = [-e if e != 0 else 0 for e in entropy]
    variance = np.sqrt(variance)

    return entropy, variance

def calculate_sum_of_pairs(unweightedfreq, aminoacids, blosum62):
    sum_of_pairs = np.zeros(len(unweightedfreq))

    for i, freq in enumerate(unweightedfreq):
        for j in range(20):
            for k in range(20):
                tup = (aminoacids[j], aminoacids[k])
                if tup not in blosum62:
                    tup = (aminoacids[k], aminoacids[j])
                sum_of_pairs[i] += freq[j] * freq[k] * blosum62[tup]

    sum_of_pairs = np.sqrt(sum_of_pairs)
```

```
    return sum_of_pairs

def Con_score_MSA(dat):
    totseq = len(dat)
    lenseq = len(dat[0])
    aminoacids = 'ACDEFGHIKLMNPQRSTVWY'
    aminoacidfreq = np.zeros(20)
    blosum62 = {
    ('W', 'F'): 1, ('L', 'R'): -2, ('S', 'P'): -1, ('V', 'T'): 0,
    ('Q', 'Q'): 5, ('N', 'A'): -2, ('Z', 'Y'): -2, ('W', 'R'): -3,
    ('Q', 'A'): -1, ('S', 'D'): 0, ('H', 'H'): 8, ('S', 'H'): -1,
    ('H', 'D'): -1, ('L', 'N'): -3, ('W', 'A'): -3, ('Y', 'M'): -1,
    ('G', 'R'): -2, ('Y', 'I'): -1, ('Y', 'E'): -2, ('B', 'Y'): -3,
    ('Y', 'A'): -2, ('V', 'D'): -3, ('B', 'S'): 0, ('Y', 'Y'): 7,
    ('G', 'N'): 0, ('E', 'C'): -4, ('Y', 'Q'): -1, ('Z', 'Z'): 4,
    ('V', 'A'): 0, ('C', 'C'): 9, ('M', 'R'): -1, ('V', 'E'): -2,
    ('T', 'N'): 0, ('P', 'P'): 7, ('V', 'I'): 3, ('V', 'S'): -2,
    ('Z', 'P'): -1, ('V', 'M'): 1, ('T', 'F'): -2, ('V', 'Q'): -2,
    ('K', 'K'): 5, ('P', 'D'): -1, ('I', 'H'): -3, ('I', 'D'): -3,
    ('T', 'R'): -1, ('P', 'L'): -3, ('K', 'G'): -2, ('M', 'N'): -2,
    ('P', 'H'): -2, ('F', 'Q'): -3, ('Z', 'G'): -2, ('X', 'L'): -1,
    ('T', 'M'): -1, ('Z', 'C'): -3, ('X', 'H'): -1, ('D', 'R'): -2,
    ('B', 'W'): -4, ('X', 'D'): -1, ('Z', 'K'): 1, ('F', 'A'): -2,
    ('Z', 'W'): -3, ('F', 'E'): -3, ('D', 'N'): 1, ('B', 'K'): 0,
    ('X', 'X'): -1, ('F', 'I'): 0, ('B', 'G'): -1, ('X', 'T'): 0,
    ('F', 'M'): 0, ('B', 'C'): -3, ('Z', 'I'): -3, ('Z', 'V'): -2,
    ('S', 'S'): 4, ('L', 'Q'): -2, ('W', 'E'): -3, ('Q', 'R'): 1,
    ('N', 'N'): 6, ('W', 'M'): -1, ('Q', 'C'): -3, ('W', 'I'): -3,
    ('S', 'C'): -1, ('L', 'A'): -1, ('S', 'G'): 0, ('L', 'E'): -3,
    ('W', 'Q'): -2, ('H', 'G'): -2, ('S', 'K'): 0, ('Q', 'N'): 0,
    ('N', 'R'): 0, ('H', 'C'): -3, ('Y', 'N'): -2, ('G', 'Q'): -2,
    ('Y', 'F'): 3, ('C', 'A'): 0, ('V', 'L'): 1, ('G', 'E'): -2,
    ('G', 'A'): 0, ('K', 'R'): 2, ('E', 'D'): 2, ('Y', 'R'): -2,
    ('M', 'Q'): 0, ('T', 'I'): -1, ('C', 'D'): -3, ('V', 'F'): -1,
    ('T', 'A'): 0, ('T', 'P'): -1, ('B', 'P'): -2, ('T', 'E'): -1,
    ('V', 'N'): -3, ('P', 'G'): -2, ('M', 'A'): -1, ('K', 'H'): -1,
    ('V', 'R'): -3, ('P', 'C'): -3, ('M', 'E'): -2, ('K', 'L'): -2,
    ('V', 'V'): 4, ('M', 'I'): 1, ('T', 'Q'): -1, ('I', 'G'): -4,
    ('P', 'K'): -1, ('M', 'M'): 5, ('K', 'D'): -1, ('I', 'C'): -1,
    ('Z', 'D'): 1, ('F', 'R'): -3, ('X', 'K'): -1, ('Q', 'D'): 0,
    ('X', 'G'): -1, ('Z', 'L'): -3, ('X', 'C'): -2, ('Z', 'H'): 0,
    ('B', 'L'): -4, ('B', 'H'): 0, ('F', 'F'): 6, ('X', 'W'): -2,
    ('B', 'D'): 4, ('D', 'A'): -2, ('S', 'L'): -2, ('X', 'S'): 0,
    ('F', 'N'): -3, ('S', 'R'): -1, ('W', 'D'): -4, ('V', 'Y'): -1,
    ('W', 'L'): -2, ('H', 'R'): 0, ('W', 'H'): -2, ('H', 'N'): 1,
    ('W', 'T'): -2, ('T', 'T'): 5, ('S', 'F'): -2, ('W', 'P'): -4,
    ('L', 'D'): -4, ('B', 'I'): -3, ('L', 'H'): -3, ('S', 'N'): 1,
```

```
    ('B', 'T'): -1, ('L', 'L'): 4, ('Y', 'K'): -2, ('E', 'Q'): 2,
    ('Y', 'G'): -3, ('Z', 'S'): 0, ('Y', 'C'): -2, ('G', 'D'): -1,
    ('B', 'V'): -3, ('E', 'A'): -1, ('Y', 'W'): 2, ('E', 'E'): 5,
    ('Y', 'S'): -2, ('C', 'N'): -3, ('V', 'C'): -1, ('T', 'H'): -2,
    ('P', 'R'): -2, ('V', 'G'): -3, ('T', 'L'): -1, ('V', 'K'): -2,
    ('K', 'Q'): 1, ('R', 'A'): -1, ('I', 'R'): -3, ('T', 'D'): -1,
    ('P', 'F'): -4, ('I', 'N'): -3, ('K', 'I'): -3, ('M', 'D'): -3,
    ('V', 'W'): -3, ('W', 'W'): 11, ('M', 'H'): -2, ('P', 'N'): -2,
    ('K', 'A'): -1, ('M', 'L'): 2, ('K', 'E'): 1, ('Z', 'E'): 4,
    ('X', 'N'): -1, ('Z', 'A'): -1, ('Z', 'M'): -1, ('X', 'F'): -1,
    ('K', 'C'): -3, ('B', 'Q'): 0, ('X', 'B'): -1, ('B', 'M'): -3,
    ('F', 'C'): -2, ('Z', 'Q'): 3, ('X', 'Z'): -1, ('F', 'G'): -3,
    ('B', 'E'): 1, ('X', 'V'): -1, ('F', 'K'): -3, ('B', 'A'): -2,
    ('X', 'R'): -1, ('D', 'D'): 6, ('W', 'G'): -2, ('Z', 'F'): -3,
    ('S', 'Q'): 0, ('W', 'C'): -2, ('W', 'K'): -3, ('H', 'Q'): 0,
    ('L', 'C'): -1, ('W', 'N'): -4, ('S', 'A'): 1, ('L', 'G'): -4,
    ('W', 'S'): -3, ('S', 'E'): 0, ('H', 'E'): 0, ('S', 'I'): -2,
    ('H', 'A'): -2, ('S', 'M'): -1, ('Y', 'L'): -1, ('Y', 'H'): 2,
    ('Y', 'D'): -3, ('E', 'R'): 0, ('X', 'P'): -2, ('G', 'G'): 6,
    ('G', 'C'): -3, ('E', 'N'): 0, ('Y', 'T'): -2, ('Y', 'P'): -3,
    ('T', 'K'): -1, ('A', 'A'): 4, ('P', 'Q'): -1, ('T', 'C'): -1,
    ('V', 'H'): -3, ('T', 'G'): -2, ('I', 'Q'): -3, ('Z', 'T'): -1,
    ('C', 'R'): -3, ('V', 'P'): -2, ('P', 'E'): -1, ('M', 'C'): -1,
    ('K', 'N'): 0, ('I', 'I'): 4, ('P', 'A'): -1, ('M', 'G'): -3,
    ('T', 'S'): 1, ('I', 'E'): -3, ('P', 'M'): -2, ('M', 'K'): -1,
    ('I', 'A'): -1, ('P', 'I'): -3, ('R', 'R'): 5, ('X', 'M'): -1,
    ('L', 'I'): 2, ('X', 'I'): -1, ('Z', 'B'): 1, ('X', 'E'): -1,
    ('Z', 'N'): 0, ('X', 'A'): 0, ('B', 'R'): -1, ('B', 'N'): 3,
    ('F', 'D'): -3, ('X', 'Y'): -1, ('Z', 'R'): 0, ('F', 'H'): -1,
    ('B', 'F'): -3, ('F', 'L'): 0, ('X', 'Q'): -1, ('B', 'B'): 4
    }
    unweightedfreq = calculate_frequencies(dat, aminoacids)
    aminoacidfreq = np.sum(unweightedfreq, axis=0) / (totseq * lenseq)

    entropy, variance = calculate_entropy_and_variance(unweightedfreq, aminoacidfreq)
    sum_of_pairs = calculate_sum_of_pairs(unweightedfreq, aminoacids, blosum62)

    return entropy, variance, sum_of_pairs

if __name__ == "__main__":
    data = ['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHI-
KAH',
            'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',
            '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',
            'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQV-
KAH',
            '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQV-
KAH',
```

```
        'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSH-
GSAQVKGH',
        'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
        'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
        'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKDH']

    entropy, variance, sum_of_pairs = Con_score_MSA(data)
    print('Entropy:', entropy)
    print('Variance:', variance)
    print('Sum of pairs:', sum_of_pairs)
```

Output:

```
Entropy: [0.2316027135475243, 0.30463609734923813, 0, 0, 1.120950392673 5835, 1.15958881430 8626, 0, 0, 1.1685184057430877, 0.30463609734923813, 0.5859526183035508, 0.30463609734923813,
Variance: [0.72516596 0.90786165 0.99516507 0.99322544 0.61007929 0.64879275
 0.99599519 0.99253179 0.59799347 0.91073977 0.77140246 0.90605299
 0.66082115 0.64946791 0.91185277 0.65054671 0.99253179 0.70511735
 0.90454533 0.66125684 0.73757877 0.82546219 0.66216517 0.67900828
 0.74005327 0.99155987 0.73156132 0.90965294 0.90200844 0.99516507
 0.73907124 0.99861933 0.91084976 0.90810988 0.58634209 0.53951811
 0.83230564 0.99640998 0.83207994 0.99419573 0.99253179 0.99419573
 0.99737718 0.99461127 0.83316276 0.99461127 0.82259014 0.09005326
 0.9094602  0.77039538 0.82028832 0.77041163 0.99155987 0.99322544
 0.81641707 0.91130331 0.90838562 0.99253179 0.68429881 0.99461127]
Sum of pairs: [1.62623126 1.73443491 2.         2.         1.07180237 1.07565087
 2.44948974 2.23606798 1.36666331 2.15897129 1.89823755 2.1222941
 1.38169856 1.23315091 3.0505385  1.50481321 2.23606798 1.87193275
 2.23421922 1.55079292 2.09090909 1.6958871  1.49655251 1.5666989
 2.07702903 2.44948974 1.3514608  2.12424026 1.82951016 2.
 1.81590767 2.23606798 2.00206505 2.23421922 0.97912087 1.06406363
 2.26726617 2.64575131 1.79300754 2.23606798 2.23606798 2.23606798
 2.64575131 2.44948974 2.12618465 2.82842712 2.05904576 0.24052285
 2.27454473 1.67628081 1.65893524 1.9896426  2.44948974 2.
 1.52391406 2.04898685 1.95824175 2.23606798 1.35451495 2.82842712]
```

Q4) Using the program written in Q3 (unweighted frequency and entropy based measure), compare the MSA from Clustal Omega, MAFFT, and MUSCLE. Identify the residues with (i) similar and (ii) different conservation scores among the three alignment methods

```
import numpy as np

def Con_score_MSA(dat):
    totseq=len(dat)
    lenseq=len(dat[0])
    unweightedfreq=[[0 for i in range(20)] for j in range(lenseq)]
    aminoacids=['A','C','D','E','F','G','H','I','K','L','M','N','P','Q','R','S','T','V','W','Y']
    aminoacidfreq=[0 for i in range(20)]

    blosum62 = {
 ('W', 'F'): 1, ('L', 'R'): -2, ('S', 'P'): -1, ('V', 'T'): 0,
 ('Q', 'Q'): 5, ('N', 'A'): -2, ('Z', 'Y'): -2, ('W', 'R'): -3,
 ('Q', 'A'): -1, ('S', 'D'): 0, ('H', 'H'): 8, ('S', 'H'): -1,
 ('H', 'D'): -1, ('L', 'N'): -3, ('W', 'A'): -3, ('Y', 'M'): -1,
 ('G', 'R'): -2, ('Y', 'I'): -1, ('Y', 'E'): -2, ('B', 'Y'): -3,
 ('Y', 'A'): -2, ('V', 'D'): -3, ('B', 'S'): 0, ('Y', 'Y'): 7,
 ('G', 'N'): 0, ('E', 'C'): -4, ('Y', 'Q'): -1, ('Z', 'Z'): 4,
 ('V', 'A'): 0, ('C', 'C'): 9, ('M', 'R'): -1, ('V', 'E'): -2,
 ('T', 'N'): 0, ('P', 'P'): 7, ('V', 'I'): 3, ('V', 'S'): -2,
 ('Z', 'P'): -1, ('V', 'M'): 1, ('T', 'F'): -2, ('V', 'Q'): -2,
 ('K', 'K'): 5, ('P', 'D'): -1, ('I', 'H'): -3, ('I', 'D'): -3,
 ('T', 'R'): -1, ('P', 'L'): -3, ('K', 'G'): -2, ('M', 'N'): -2,
 ('P', 'H'): -2, ('F', 'Q'): -3, ('Z', 'G'): -2, ('X', 'L'): -1,
 ('T', 'M'): -1, ('Z', 'C'): -3, ('X', 'H'): -1, ('D', 'R'): -2,
 ('B', 'W'): -4, ('X', 'D'): -1, ('Z', 'K'): 1, ('F', 'A'): -2,
 ('Z', 'W'): -3, ('F', 'E'): -3, ('D', 'N'): 1, ('B', 'K'): 0,
 ('X', 'X'): -1, ('F', 'I'): 0, ('B', 'G'): -1, ('X', 'T'): 0,
 ('F', 'M'): 0, ('B', 'C'): -3, ('Z', 'I'): -3, ('Z', 'V'): -2,
 ('S', 'S'): 4, ('L', 'Q'): -2, ('W', 'E'): -3, ('Q', 'R'): 1,
 ('N', 'N'): 6, ('W', 'M'): -1, ('Q', 'C'): -3, ('W', 'I'): -3,
 ('S', 'C'): -1, ('L', 'A'): -1, ('S', 'G'): 0, ('L', 'E'): -3,
 ('W', 'Q'): -2, ('H', 'G'): -2, ('S', 'K'): 0, ('Q', 'N'): 0,
 ('N', 'R'): 0, ('H', 'C'): -3, ('Y', 'N'): -2, ('G', 'Q'): -2,
 ('Y', 'F'): 3, ('C', 'A'): 0, ('V', 'L'): 1, ('G', 'E'): -2,
 ('G', 'A'): 0, ('K', 'R'): 2, ('E', 'D'): 2, ('Y', 'R'): -2,
 ('M', 'Q'): 0, ('T', 'I'): -1, ('C', 'D'): -3, ('V', 'F'): -1,
 ('T', 'A'): 0, ('T', 'P'): -1, ('B', 'P'): -2, ('T', 'E'): -1,
 ('V', 'N'): -3, ('P', 'G'): -2, ('M', 'A'): -1, ('K', 'H'): -1,
 ('V', 'R'): -3, ('P', 'C'): -3, ('M', 'E'): -2, ('K', 'L'): -2,
 ('V', 'V'): 4, ('M', 'I'): 1, ('T', 'Q'): -1, ('I', 'G'): -4,
 ('P', 'K'): -1, ('M', 'M'): 5, ('K', 'D'): -1, ('I', 'C'): -1,
```

```
('Z', 'D'): 1, ('F', 'R'): -3, ('X', 'K'): -1, ('Q', 'D'): 0,
('X', 'G'): -1, ('Z', 'L'): -3, ('X', 'C'): -2, ('Z', 'H'): 0,
('B', 'L'): -4, ('B', 'H'): 0, ('F', 'F'): 6, ('X', 'W'): -2,
('B', 'D'): 4, ('D', 'A'): -2, ('S', 'L'): -2, ('X', 'S'): 0,
('F', 'N'): -3, ('S', 'R'): -1, ('W', 'D'): -4, ('V', 'Y'): -1,
('W', 'L'): -2, ('H', 'R'): 0, ('W', 'H'): -2, ('H', 'N'): 1,
('W', 'T'): -2, ('T', 'T'): 5, ('S', 'F'): -2, ('W', 'P'): -4,
('L', 'D'): -4, ('B', 'I'): -3, ('L', 'H'): -3, ('S', 'N'): 1,
('B', 'T'): -1, ('L', 'L'): 4, ('Y', 'K'): -2, ('E', 'Q'): 2,
('Y', 'G'): -3, ('Z', 'S'): 0, ('Y', 'C'): -2, ('G', 'D'): -1,
('B', 'V'): -3, ('E', 'A'): -1, ('Y', 'W'): 2, ('E', 'E'): 5,
('Y', 'S'): -2, ('C', 'N'): -3, ('V', 'C'): -1, ('T', 'H'): -2,
('P', 'R'): -2, ('V', 'G'): -3, ('T', 'L'): -1, ('V', 'K'): -2,
('K', 'Q'): 1, ('R', 'A'): -1, ('I', 'R'): -3, ('T', 'D'): -1,
('P', 'F'): -4, ('I', 'N'): -3, ('K', 'I'): -3, ('M', 'D'): -3,
('V', 'W'): -3, ('W', 'W'): 11, ('M', 'H'): -2, ('P', 'N'): -2,
('K', 'A'): -1, ('M', 'L'): 2, ('K', 'E'): 1, ('Z', 'E'): 4,
('X', 'N'): -1, ('Z', 'A'): -1, ('Z', 'M'): -1, ('X', 'F'): -1,
('K', 'C'): -3, ('B', 'Q'): 0, ('X', 'B'): -1, ('B', 'M'): -3,
('F', 'C'): -2, ('Z', 'Q'): 3, ('X', 'Z'): -1, ('F', 'G'): -3,
('B', 'E'): 1, ('X', 'V'): -1, ('F', 'K'): -3, ('B', 'A'): -2,
('X', 'R'): -1, ('D', 'D'): 6, ('W', 'G'): -2, ('Z', 'F'): -3,
('S', 'Q'): 0, ('W', 'C'): -2, ('W', 'K'): -3, ('H', 'Q'): 0,
('L', 'C'): -1, ('W', 'N'): -4, ('S', 'A'): 1, ('L', 'G'): -4,
('W', 'S'): -3, ('S', 'E'): 0, ('H', 'E'): 0, ('S', 'I'): -2,
('H', 'A'): -2, ('S', 'M'): -1, ('Y', 'L'): -1, ('Y', 'H'): 2,
('Y', 'D'): -3, ('E', 'R'): 0, ('X', 'P'): -2, ('G', 'G'): 6,
('G', 'C'): -3, ('E', 'N'): 0, ('Y', 'T'): -2, ('Y', 'P'): -3,
('T', 'K'): -1, ('A', 'A'): 4, ('P', 'Q'): -1, ('T', 'C'): -1,
('V', 'H'): -3, ('T', 'G'): -2, ('I', 'Q'): -3, ('Z', 'T'): -1,
('C', 'R'): -3, ('V', 'P'): -2, ('P', 'E'): -1, ('M', 'C'): -1,
('K', 'N'): 0, ('I', 'I'): 4, ('P', 'A'): -1, ('M', 'G'): -3,
('T', 'S'): 1, ('I', 'E'): -3, ('P', 'M'): -2, ('M', 'K'): -1,
('I', 'A'): -1, ('P', 'I'): -3, ('R', 'R'): 5, ('X', 'M'): -1,
('L', 'I'): 2, ('X', 'I'): -1, ('Z', 'B'): 1, ('X', 'E'): -1,
('Z', 'N'): 0, ('X', 'A'): 0, ('B', 'R'): -1, ('B', 'N'): 3,
('F', 'D'): -3, ('X', 'Y'): -1, ('Z', 'R'): 0, ('F', 'H'): -1,
('B', 'F'): -3, ('F', 'L'): 0, ('X', 'Q'): -1, ('B', 'B'): 4
}

    for i in range(lenseq):
        for j in range(totseq):
            if dat[j][i].isalpha()==1:
                a=aminoacids.index(str(dat[j][i]))
                unweightedfreq[i][a]+=1
    unweightedfreq=np.divide(unweightedfreq, totseq)
    for i in range(lenseq):
        for j in range(totseq):
```

```
            if dat[j][i].isalpha()==1:
                a=aminoacids.index(str(dat[j][i]))
                aminoacidfreq[a]+=1
    aminoacidfreq=np.divide(aminoacidfreq, totseq*lenseq)

    entropy=[0]*lenseq
    variance=[0]*lenseq
    sum_of_pairs=[0]*lenseq
    for i in range(lenseq):
        for j in range(20):
            if unweightedfreq[i][j]!=0:
                entropy[i]+=unweightedfreq[i][j]*np.log(unweightedfreq[i][j])
            variance[i]+=(unweightedfreq[i][j]-aminoacidfreq[j])**2
            for k in range(20):
                tup=(aminoacids[j],aminoacids[k])
                if tup in blosum62:
                    tup=(aminoacids[j],aminoacids[k])
                else:
                    tup=(aminoacids[k],aminoacids[j])
                sum_of_pairs[i]+=unweightedfreq[i][j]*unweightedfreq[i][k]*blosum62[tup]
    for i in range(lenseq):
        variance[i]=variance[i]**0.5
        sum_of_pairs[i]=sum_of_pairs[i]**0.5

        print('\nEntropy=\n', entropy)
        print('\nVariance=\n', variance)
        print('\nSum of pairs=\n', sum_of_pairs)
        return entropy, variance, sum_of_pairs

def compare_msa(a, b, c):
    similar=[]
    dissimilar=[]
    for i in range(len(a)):
        if a[i]==b[i]==c[i]:
            similar.append((i, a[i]))
        elif a[i]!=b[i]!=c[i]:
            dissimilar.append((i,a[i],b[i],c[i]))

        print('total number of similar scores=', len(similar))
        print('Similar residues through Clustal Omega, MAFFT and Muscle= (Posiiton, common conservation
score')
        print(similar)
        print('Total number of diff scores=',len(dissimilar))
        print('Different residues through Clustal Omega, MAFFT and MUSCLE are= (Position, same order
conservation scores')
        print(dissimilar)

def compare_results():
```

```
Clustal_Omega =['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTK-
TYFSHWPDVTPGSPHIKAH',
        'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',
        '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',
        'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQV-
KAH',
        '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQV-
KAH',
        'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSH-
GSAQVKGH',
        'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
        'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
        'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKDH']
MAFFT = ['MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
      'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKDH',
      'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
      'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
      'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
      'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',
      '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQV-
KAH',
      'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',
      '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',
      'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',
      '-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHIKAH']
MUSCLE = ['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPG-
SPHIKAH',
      'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',
      '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',
      'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',
      '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQV-
KAH',
      'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',
      'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
```

```
        'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
        'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
        'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKDH',]
    Clustal_Omega_entropy, Clustal_Omega_variance, Clustal_Omega_sum=Con_score_MSA(Clus-
tal_Omega)
    MAFFT_entropy, MAFFT_variance, MAFFT_sum=Con_score_MSA(MAFFT)
    MUSCLE_entropy, MUSCLE_variance, MUSCLE_sum=Con_score_MSA(MUSCLE)
    print('Compating entropy-based conservation scores...')
    compare_msa(Clustal_Omega_entropy, MAFFT_entropy, MUSCLE_entropy)
    print('comparing variance-based conservation scores...')
    compare_msa(Clustal_Omega_variance, MAFFT_variance, MUSCLE_variance)
    print('Comparing sum of all pairs-based conservation scores...')
    compare_msa(Clustal_Omega_sum, MAFFT_sum, MUSCLE_sum)
    return None
compare_results()
```

Output:*

```
Entropy=
 [-0.2316027135407 5243, -0.3046360973492 3813, 0.0, 0.0, -1.1209503926735835, -1.159588814308626, 0.0, 0.0, -1.1685184057430877, -0.30463609734923813, -0.5859526183035508, -0.304636097

Variance=
 [0.745228162225573, 0.779331955922865, 0.953987603305 7852, 0.9115633608815428, 0.26996556473829186, 0.3101859504132 2305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0

Sum of pairs=
 [1.626231256363483 5, 3.008264462809917, 4.0, 4.0, 1.1487603305785123, 1.1570247933884295, 6.0, 5.0, 1.867768595041322, 4.661157024793388, 3.603305785123967, 4.5041322314049586, 1.909

Entropy=
 [-0.2316027135407 5243, -0.3046360973492 3813, 0.0, 0.0, -1.1209503926735835, -1.159588814308626, 0.0, 0.0, -1.1685184057430877, -0.30463609734923813, -0.5859526183035508, -0.304636097

Variance=
 [0.745228162225573, 0.779331955922865, 0.953987603305 7852, 0.9115633608815428, 0.26996556473829186, 0.3101859504132 2305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0

Sum of pairs=
 [1.626231256363483 5, 3.008264462809917, 4.0, 4.0, 1.1487603305785123, 1.1570247933884295, 6.0, 5.0, 1.867768595041322, 4.661157024793388, 3.603305785123967, 4.5041322314049586, 1.909

Entropy=
 [-0.2316027135407 5243, -0.3046360973492 3813, 0.0, 0.0, -1.1209503926735835, -1.159588814308626, 0.0, 0.0, -1.1685184057430877, -0.30463609734923813, -0.5859526183035508, -0.304636097

Variance=
 [0.745228162225573, 0.779331955922865, 0.953987603305 7852, 0.9115633608815428, 0.26996556473829186, 0.3101859504132 2305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0
...
[(0, 1.626231256363483 5), (1, 3.008264462809917), (2, 4.0), (3, 4.0), (4, 1.1487603305785123), (5, 1.1570247933884295), (6, 6.0), (7, 5.0), (8, 1.867768595041322), (9, 4.6611570247933
Total number of diff scores= 0
Different residues through Clustal Omega, MAFFT and MUSCLE are= (Position, same order conservation scores
[]
```

**Only a small snippet is shown. The complete output is saved in text file named "Assignment6_Q4_Output"*

Q5). Check the scores manually at positions 9, 11, 20, 22 and 30 (use MSA from Clustal Omega)

BT3040

Atharva Mandar Phatak | BG21B008 | Assignment 6

Date: 17 March 2024

Q5) a) Set A

Position 9 : A S T T A S T T G T T

$F(A) = \frac{2}{11} = 0.182$

$F(S) = \frac{2}{11} = 0.182$

$F(T) = \frac{6}{11} = 0.545$

$F(G) = \frac{1}{11} = \cancel{0.182}\ 0.091$

$\therefore C^e(9) = 0.182 \ln(0.182) + 0.182 \ln(0.182) + 0.545 \ln(0.545) + 0.091 \ln(0.091)$

$= -1.169$

Position 11 : V V I I V I V V V V V

$F(V) = 8/11 = 0.727$ $\quad F(I) = 3/11 = 0.273$

$C^e(11) = 0.727 \ln(0.727) + 0.273 \ln(0.273)$

$= -0.5862$

Position 20 : K S G G G G G G G A A

$F(A) = 2/11 = 0.182$

$F(G) = 7/11 = 0.636$

$F(K) = 1/11 = 0.091$

$F(S) = 1/11 = 0.091$

$$C^e(20) = 0.182 \ln(0.182) + 0.636 \ln(0.636) + 0.091 \ln(0.091) + 0.091 \ln(0.091)$$

$$= -1.0341.$$

Position 22 : A A A G A G A A A A

$$F(A) = \frac{9}{11} = 0.818$$

$$F(G) = \frac{2}{11} = 0.182$$

$$C^e(22) = 0.818 \ln(0.818) + 0.182 \ln(0.182)$$

$$= -0.4744$$

Position 30 L L L L L L L L L L L

$F(L) = 11/11 = 1$

$C^e(30) = 1 \ln(1) = 0$

Set 2 :

Position 9 : A – V V V V V V V

$f(A) = \frac{1}{9}$

$f(V) = \frac{7}{9}$

$\therefore C'(9) = \cancel{0.111 + 0.778}$

$= \cancel{0.889}$

$= -0.439$

Position 11 : A – G G G G G G G

$f(A) = \frac{1}{9}$

$f(G) = \frac{7}{9}$

$C'(11) = \cancel{0.889} \; -0.439$

Position 20 : S A Q D A K Q K K

$f(S) = \frac{1}{9}$ $f(A) = f(K) = \frac{3}{9}$ $f(D) = \frac{1}{9}$

$f(A) = \frac{2}{9}$ $f(Q) = \frac{2}{9}$

$C'(20) = \cancel{0.889} \; -1.766.$

Pos

Position 22 LIIVVILLL

$F(V) = \frac{2}{9}$

$F(I) = \frac{3}{9}$

$F(L) = \frac{4}{9}$

$c'(22) = 0.999 - 1.06$

Position 30 : NNNNNNGNNN

$F(N) = \frac{8}{9}$

$F(S) = \frac{1}{9}$

$c'(30) = -0.349$

Q6) Obtain the conservation score of 1BTM, A-chain using Consurf server (https://consurf.tau.ac.il/

Grade		Residue Distribution for 1BTM Chain A	
9	Conserved	54	21.5%
8		18	7.2%
7		25	10%
6		32	12.7%
5	Average	37	14.7%
4		25	10%
3		34	13.5%
2		4	1.6%
1	Variable	22	8.8%
0	Uncertain	0	0%
	Total	251	amino acids in chain A.

Average grade for 251 residues grades 1-9: **5.6**

150 unique sequences in the MSA
sampled from 27,844 found (see Job Info). APD: **1.00**

Homologues, Alignment and Phylogeny

- 33874 homologues were collected from the UNIREF90 database using HMMER.
- Of these, 28566 homologues passed the thresholds (min/max similarity, coverage, etc), 27844 of them are CD-HIT unique.
- The calculations were conducted on 150 hits (query included), sampled from the unique hits. Click here if you wish to view the list of sequences which produced significant alignments, but were not chosen as hits.

- **Alignment details**
  - The average number of replacements between any two sequences in the alignment;
    A distance of 0.01 means that on average, the expected replacement for every 100 positions is 1.
  - Average pairwise distance : 0.998229
  - Lower bound : 0.18384
  - Upper bound : 1.87984
  - Residue variety per position in the MSA (The table is best viewed with an editor that respects Comma-Separated Values)
  - View MSA and phylogenetic tree using WASABI
  - Download Phylogenetic Tree (Newick format)
  - The best evolutionary model was selected to be: WAG. See details here

 # MSA Amino Acid Percentage, check excel file named 'Assignment6_Q6_msa_aa_variety_percentage'.

Practical 7
22 March 2024
Questions

1. Compute the amino acid composition of the following sequences. Provide the output as a table of amino acid percentage values for each sequence and comment on the results.

    1. RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAAL KQKSRHAA
    2. AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGL YVMFAVA
    3. AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEG AGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFAN GQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI

2. Assume the molecular weights of the 20 amino acid residues as given below. Compute the molecular weight of the three sequences given in question 1.

Ala: 85	Cys: 115	Asp: 130	Glu: 145	Phe: 160	Gly: 70	Trp: 200
His: 150	Ile: 125	Lys: 145	Leu: 125	Met: 143	Asn: 130	Tyr: 175
Pro: 110	Gln: 140	Arg: 170	Ser: 100	Thr: 115	Val: 110	

3. The amino acid composition of a standard set of Group A (first value) and Group B (second value) proteins are given below. Identify whether the given sequences in Question 1 belong to Group A or Group B and write your answer.

Ala: 8.47, 8.95	Asp: 5.97, 5.91	Cys: 1.39, 0.47	Glu: 6.32, 4.78	Thr: 5.79, 6.54
Phe: 3.91, 3.68	Gly: 7.82, 8.54	His: 2.26, 1.25	Ile: 5.71, 4.77	Val: 7.02, 6.76
Lys: 5.76, 4.93	Leu: 8.48, 8.78	Met: 2.21, 1.56	Asn: 4.54, 5.74	Trp: 1.44, 1.24
Pro: 4.63, 3.74	Gln: 3.82, 4.75	Arg: 4.93, 5.24	Ser: 5.94, 8.05	Tyr: 3.58, 4.13

4. Compute the residue pair preference for the three sequences given in question 1. The required output is a 20x20 table showing the pair preferences (a) $[N_{ij}*100/(N_i+N_j)]$, (b) $[N_{ij}*100/(N-1)]$ and (c) $[N_{ij}*100/(N_{i*}N_j)]$. List the top 10 preferred residues from each of the three pair-preferences.

5. Compute average hydrophobicity (H_{gm}), Helical contact area (Ca) and Total non-bonded energy (Et) for the sequences in Q1 and comment on the results. (Refer www.iitm.ac.in/bioinfo/fold_rate/prop_orig.html for the properties).

Deadline: 30th March 2024

BT 3040: BIOINFORMATICS

Assignment 7

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) Compute the amino acid composition of the following sequences. Provide the output as a table of amino acid percentage values for each sequence and comment on the results.

Calculated using EMBOSS

a) Sequence 1:
RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLA
AL KQKSRHAA

```
# Output from 'compseq'
#
# The Expected frequencies are calculated on the (false) assumption that every
# word has equal frequency.
#
# The input sequences are:
#


Word size 1
Total count          67

#
# Word     Obs Count  Obs Frequency        Exp Frequency        Obs/Exp Frequency
#
A          12                   0.1791045  0.0476190  3.7611940
C          1                    0.0149254  0.0476190  0.3134328
D          2                    0.0298507  0.0476190  0.6268657
E          2                    0.0298507  0.0476190  0.6268657
F          1                    0.0149254  0.0476190  0.3134328
G          4                    0.0597015  0.0476190  1.2537313
H          3                    0.0447761  0.0476190  0.9402985
I          1                    0.0149254  0.0476190  0.3134328
K          5                    0.0746269  0.0476190  1.5671642
L          6                    0.0895522  0.0476190  1.8805970
M          0                    0.0000000  0.0476190  0.0000000
N          1                    0.0149254  0.0476190  0.3134328
P          5                    0.0746269  0.0476190  1.5671642
Q          1                    0.0149254  0.0476190  0.3134328SS
R          4                    0.0597015  0.0476190  1.2537313
S          4                    0.0597015  0.0476190  1.2537313
T          7                    0.1044776  0.0476190  2.1940299
U          0                    0.0000000  0.0476190  0.0000000
V          1                    0.0149254  0.0476190  0.3134328
W          4                    0.0597015  0.0476190  1.2537313
Y          3                    0.0447761  0.0476190  0.9402985

Other      0                    0.0000000  0.0000000  10000000000.0000000
```

b) Sequence 2:
AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGL
GL YVMFAVA

```
#
# Output from 'compseq'
#
# The Expected frequencies are calculated on the (false) assumption that every
# word has equal frequency.
#
# The input sequences are:
#


Word size 1
Total count          68

#
# Word     Obs Count Obs Frequency        Exp Frequency        Obs/Exp Frequency
#
A          13                   0.1911765 0.0476190 4.0147059
C          0                    0.0000000 0.0476190 0.0000000
D          2                    0.0294118 0.0476190 0.6176471
E          1                    0.0147059 0.0476190 0.3088235
F          4                    0.0588235 0.0476190 1.2352941
G          10                   0.1470588 0.0476190 3.0882353
H          0                    0.0000000 0.0476190 0.0000000
I          8                    0.1176471 0.0476190 2.4705882
K          1                    0.0147059 0.0476190 0.3088235
L          9                    0.1323529 0.0476190 2.7794118
M          5                    0.0735294 0.0476190 1.5441176
N          0                    0.0000000 0.0476190 0.0000000
P          3                    0.0441176 0.0476190 0.9264706
Q          2                    0.0294118 0.0476190 0.6176471
R          2                    0.0294118 0.0476190 0.6176471
S          0                    0.0000000 0.0476190 0.0000000
T          1                    0.0147059 0.0476190 0.3088235
U          0                    0.0000000 0.0476190 0.0000000
V          6                    0.0882353 0.0476190 1.8529412
W          0                    0.0000000 0.0476190 0.0000000
Y          1                    0.0147059 0.0476190 0.3088235

Other      0                    0.0000000 0.0000000 10000000000.0000000
```

c) Sequence 3:
AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGD EG
AGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLK FAN GQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI

```
#
# Output from 'compseq'
#
# The Expected frequencies are calculated on the (false) assumption that every
# word has equal frequency.
#
# The input sequences are:
#


Word size 1
Total count          151

#
# Word     Obs Count  Obs Frequency        Exp Frequency        Obs/Exp Frequency
#
A          16                   0.1059603  0.0476190  2.2251656
C          0                    0.0000000  0.0476190  0.0000000
D          9                    0.0596026  0.0476190  1.2516556
E          6                    0.0397351  0.0476190  0.8344371
F          8                    0.0529801  0.0476190  1.1125828
G          23                   0.1523179  0.0476190  3.1986755
H          2                    0.0132450  0.0476190  0.2781457
I          8                    0.0529801  0.0476190  1.1125828
K          6                    0.0397351  0.0476190  0.8344371
L          9                    0.0596026  0.0476190  1.2516556
M          1                    0.0066225  0.0476190  0.1390728
N          5                    0.0331126  0.0476190  0.6953642
P          4                    0.0264901  0.0476190  0.5562914
Q          5                    0.0331126  0.0476190  0.6953642
R          5                    0.0331126  0.0476190  0.6953642
S          15                   0.0993377  0.0476190  2.0860927
T          7                    0.0463576  0.0476190  0.9735099
U          0                    0.0000000  0.0476190  0.0000000
V          8                    0.0529801  0.0476190  1.1125828
W          6                    0.0397351  0.0476190  0.8344371
Y          8                    0.0529801  0.0476190  1.1125828

Other      0                    0.0000000  0.0000000  10000000000.0000000
```

d) Combined Output

Sequence 1		
Residue	Frequency	Percentage Occurance
A	12	17.91045
C	1	1.49254
D	2	2.98507
E	2	2.98507
F	1	1.49254
G	4	5.97015
H	3	4.47761
I	1	1.49254
K	5	7.46269
L	6	8.95522
M	0	0
N	1	1.49254
P	5	7.46269
Q	1	1.49254
R	4	5.97015
S	4	5.97015
T	7	10.44776
U	0	0
V	1	1.49254
W	4	5.97015
Y	3	4.47761
Total		**100.00001**

Sequence 2		
Residue	Frequency	Percentage Occurance
A	13	19.11765
C	0	0
D	2	2.94118
E	1	1.47059
F	4	5.88235
G	10	14.70588
H	0	0
I	8	11.76471
K	1	1.47059
L	9	13.23529
M	5	7.35294
N	0	0
P	3	4.41176
Q	2	2.94118
R	2	2.94118
S	0	0
T	1	1.47059
U	0	0
V	6	8.82353
W	0	0
Y	1	1.47059
Total		**100.00001**

Sequence 3		
Residue	Frequency	Percentage Occurance
A	16	10.59603
C	0	0
D	9	5.96026
E	6	3.97351
F	8	5.29801
G	23	15.23179
H	2	1.3245
I	8	5.29801
K	6	3.97351
L	9	5.96026
M	1	0.66225
N	5	3.31126
P	4	2.64901
Q	5	3.31126
R	5	3.31126
S	15	9.93377
T	7	4.63576
U	0	0
V	8	5.29801
W	6	3.97351
Y	8	5.29801
Total		**99.99998**

Q2) Assume the molecular weights of the 20 amino acid residues as given below. Compute the molecular weight of the three sequences given in question 1

Code:

```
# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009
#Q2

def aa_molwt_calc(AA_input):
    Mol_Wt_dict={'A': 85,'C': 115, 'D': 130, 'E': 145, 'F': 160, 'G': 70, 'W': 200, 'H': 150, 'I': 125, 'K': 145,
'L': 125, 'M': 143, 'N': 130, 'Y': 175, 'P': 110, 'Q': 140, 'R': 170, 'S': 100, 'T': 115, 'V': 110}
    mol_wt_aa=0
    for i in AA_input:
        mol_wt_aa=mol_wt_aa+Mol_Wt_dict[i]
    return mol_wt_aa



print(f"Mol Wt for Sequence 1 is {aa_molwt_calc(seq1)} units")
print(f"Mol Wt for Sequence 2 is {aa_molwt_calc(seq2)} units")
print(f"Mol Wt for Sequence 3 is {aa_molwt_calc(seq3)} units")
```

Output:

```
Mol Wt for Sequence 1 is 8315 units
Mol Wt for Sequence 2 is 7735 units
Mol Wt for Sequence 3 is 18153 units
```

Q3) The amino acid composition of a standard set of Group A (first value) and Group B (second value) proteins are given below. Identify whether the given sequences in Question 1 belong to Group A or Group B and write your answer it.

```
# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009
#Q3
import numpy as np

group_A = {'A': 8.47, 'D': 5.97, 'C': 1.39, 'E': 6.32, 'T': 5.79, 'F': 3.91, 'G': 7.82, 'H': 2.26, 'I': 5.71, 'V': 7.02,
        'K': 5.76, 'L': 8.48, 'M': 2.21, 'N': 4.54, 'W': 1.44, 'P': 4.63, 'Q': 3.82, 'R': 4.93, 'S': 5.94, 'Y': 3.58}
group_B = {'A': 8.95, 'D': 5.91, 'C': 0.47, 'E': 4.78, 'T': 6.54, 'F': 3.68, 'G': 8.54, 'H': 1.25, 'I': 4.77, 'V': 6.76,
        'K': 4.93, 'L': 8.78, 'M': 1.56, 'N': 5.74, 'W': 1.24, 'P': 3.74, 'Q': 4.75, 'R': 5.24, 'S': 8.05, 'Y': 4.13}

strings =
["'RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQK
SRHAA'",
        "'AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGL
YVMFAVA'",
        "'AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEG
AGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSV
GVRAIHYSNAGLKQPNDGIESYSLFYKIPI'"]


def remse_calc(seq,seqnum):
    group_A_diff=0
    group_B_diff=0
    for _ in mwdict:
        group_A_diff=group_A_diff+np.abs((pandadict[_][seqnum]-group_A[_]))

        group_B_diff=group_B_diff+np.abs((pandadict[_][seqnum]-group_B[_]))

    if group_A_diff<group_B_diff:
        print(f"Sequence {seqnum +1} is in Group A")

    else:
        print(f"Sequence {seqnum+1} is in Group B")


remse_calc(strings[0],0)
remse_calc(strings[1],1)
remse_calc(strings[2],2)
```

Output:

```
Sequence 1 is in Group A
Sequence 2 is in Group A
Sequence 3 is in Group B
```

Q4) Compute the residue pair preference for the three sequences given in question 1. The required output is a 20x20 table showing the pair preferences (a) [Nij*100/(Ni+Nj)], (b) [Nij*100/(N-1)] and (c) [Nij*100/(Ni*Nj)]. List the top 10 preferred residues from each of the three pair-preferences.

```
import pandas as pd

def ResPairPref(seq):
    AA_all = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']

    Pref_1 = [[0 for _ in range(len(AA_all))] for _ in range(len(AA_all))]
    Pref_2 = [[0 for _ in range(len(AA_all))] for _ in range(len(AA_all))]
    Pref_3 = [[0 for _ in range(len(AA_all))] for _ in range(len(AA_all))]
    composition = [0] * len(AA_all)

    for aa in seq:
        ind = AA_all.index(aa)
        composition[ind] += 1

    for i in AA_all:
        for j in AA_all:
            pair = i + j
            for k in range(len(seq) - 1):
                if pair == seq[k:k + 2]:
                    Pref_1[AA_all.index(i)][AA_all.index(j)] += 1
                    Pref_2[AA_all.index(i)][AA_all.index(j)] += 1
                    Pref_3[AA_all.index(i)][AA_all.index(j)] += 1

            if composition[AA_all.index(i)] == 0 or composition[AA_all.index(j)] == 0:
                Pref_1[AA_all.index(i)][AA_all.index(j)] = 0
                Pref_2[AA_all.index(i)][AA_all.index(j)] = 0
                Pref_3[AA_all.index(i)][AA_all.index(j)] = 0
            else:
                Pref_1[AA_all.index(i)][AA_all.index(j)] = (Pref_1[AA_all.index(i)][AA_all.index(j)] * 100) / (
                        composition[AA_all.index(i)] + composition[AA_all.index(j)])
                Pref_2[AA_all.index(i)][AA_all.index(j)] = (Pref_2[AA_all.index(i)][AA_all.index(j)] * 100) / (
                        len(seq) - 1)
                Pref_3[AA_all.index(i)][AA_all.index(j)] = (Pref_3[AA_all.index(i)][AA_all.index(j)] * 100) / (
                        composition[AA_all.index(i)] * composition[AA_all.index(j)])

    print('\nPair-wise preference score for the given sequence:')
    print('\nFormula 1 : ')
    print(pd.DataFrame(Pref_1, columns=AA_all, dtype=float).to_string())

    print('\nPair-wise preference score for the given sequence:')
    print('\nFormula 2 : ')
    print(pd.DataFrame(Pref_2, columns=AA_all, dtype=float).to_string())
```

```
    print('\nPair-wise preference score for the given sequence:')
    print('\nFormula 3 : ')
    print(pd.DataFrame(Pref_3, columns=AA_all, dtype=float).to_string())

strings = [
    '''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQ
KSRHAA''',
    '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYV
MFAVA''',
    '''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGK
HSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVR
AIHYSNAGLKQPNDGIESYSLFYKIPI'''
]

for seq in strings:
    print("For Sequence", strings.index(seq))
    ResPairPref(seq)
```

Output:

(Complete output is given in notepad file: ‘Assignment7_Q4’

```
Pair-wise preference score for the given sequence:

Formula 1 :
           A    C    D         E    F         G         H    I         K         L   M    N         P         Q         R         S
0  12.500000  0.0  0.0  0.000000  0.0  6.250000  0.000000  0.0  0.000000  11.111111  0.0  0.0  0.000000  0.000000  0.000000  6.250000
1   0.000000  0.0  0.0  0.000000 50.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
2   0.000000  0.0  0.0 25.000000  0.0 16.666667  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
3   7.142857  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
4   0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0 50.0  0.000000  0.000000  0.000000  0.000000
5   0.000000 20.0  0.0  0.000000  0.0  0.000000  0.000000 20.0  0.000000  10.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
6  13.333333  0.0  0.0  0.000000  0.0  0.000000 16.666667  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
7   0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0 16.666667   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
8   5.882353  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0 10.000000 16.666667  0.000000 11.111111
9   5.555556  0.0 12.5  0.000000  0.0  0.000000 11.111111  0.0  9.090909   8.333333  0.0  0.0  0.000000  0.000000  0.000000  0.000000
10  0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
11  0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000 20.000000  0.000000
12  0.000000  0.0  0.0 14.285714  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000 11.111111
13  0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0 16.666667   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
14  6.250000  0.0  0.0  0.000000  0.0  0.000000 14.285714  0.0  0.000000   0.000000  0.0  0.0 11.111111  0.000000  0.000000  0.000000
15  0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0 11.111111  10.000000  0.0  0.0  0.000000  0.000000 12.500000  0.000000
16 10.526316  0.0  0.0  0.000000  0.0  9.090909  0.000000  0.0  8.333333   0.000000  0.0  0.0 16.666667  0.000000  9.090909  0.000000
17  0.000000  0.0  0.0  0.000000  0.0 20.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0  0.000000  0.000000  0.000000  0.000000
18  0.000000  0.0  0.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000   0.000000  0.0  0.0 11.111111  0.000000  0.000000 12.500000
19  6.666667  0.0 20.0  0.000000  0.0  0.000000  0.000000  0.0  0.000000  11.111111  0.0  0.0  0.000000  0.000000  0.000000  0.000000

Pair-wise preference score for the given sequence:
```

For sequence 1

1. Top 10 pairs: {'CF': 50.0, 'FN': 50.0, 'DE': 25.0, 'VG': 20.0, 'GC': 20.0, 'NR': 20.0, 'YD': 20.0, 'GI': 20.0, 'EY': 20.0, 'WT': 18.18}
2. Top 10 pairs: {'AT': 4.54545, 'AA': 4.54545, 'TP': 3.0303, 'WT':3.0303, 'AL': 3.0303, 'TA': 3.0303, 'HA':3.0303, 'RA': 1.51515, 'PT': 1.51515, 'TR':1.51515}
3. Top 10 pairs: {'CF': 100.0, 'FN': 100.0, 'VG': 25.0, 'GC': 25.0, 'NR': 25.0, 'DE': 25.0, 'GI': 25.0, 'PV': 20.0, 'IK':20.0, 'KQ': 20.0}

For sequence 2

1. Top 10 pairs: {'RT': 33.3333, 'TQ': 33.3333, 'VM': 27.2727, 'RQ': 25.0, 'GL': 21.0526, 'KF': 20.0, 'QP': 20.0, 'PD': 20.0, 'AA': 19.2308, 'IP':18.1818}
2. Top 10 pairs: {'AA': 7.46269, 'GL': 5.97015, 'AV': 4.47761, 'VM':4.47761, 'AI': 4.47761, 'IG': 4.47761, 'MG':2.98507, 'GA': 2.98507, 'GI': 2.98507, 'LG':2.98507}
3. Top 10 pairs: {'RT': 50.0, 'TQ': 50.0, 'KF': 25.0, 'RQ': 25.0, 'QP': 16.6667, 'PD':16.6667, 'YV': 16.6667, 'QF': 12.5, 'LE': 11.1111, 'LY':11.1111}

For sequence 3

1. Top 10 pairs: {'WD': 20.0, 'AG': 15.38, 'RL': 14.29, 'YW': 14.29, 'IE': 14.29, 'TY': 13.3333, 'LK': 13.3333, 'MT': 12.5, 'KH': 12.5, 'SL': 12.5}
2. Top 10 pairs: {'AG': 4.0, 'AA': 2.0, 'VG': 2.0, 'GA': 2.0, 'TG': 2.0, 'GL': 2.0, 'WD': 2.0, 'GD': 2.0, 'SL': 2.0, 'FA': 2.0}
3. Top 10 pairs: {'MT': 14.29, 'KH': 8.333, 'IH': 6.25, 'HY': 6.25, 'WD': 5.556, 'QP': 5.0, 'PN': 5.0, 'RL': 4.444, 'GM': 4.348, ‘YW': 4.167}

Q5) Compute average hydrophobicity (Hgm), Helical contact area (Ca) and Total non-bonded energy(Et) for the sequences in Q1 and comment on the results. (Refer www.iitm.ac.in/bioinfo/fold_rate/prop_orig.html for the properties)

```
# BT3040 Assignment 7| Atharva Mandar Phatak | BE21B009
#Q5


hgmdict = {'A': 13.85, 'C': 15.37, 'D': 11.61, 'E': 11.38, 'F': 13.93, 'G': 13.34, 'W': 15.48,
           'H': 13.82, 'I': 15.28, 'K': 11.58, 'L': 14.13, 'M': 13.86, 'N': 13.02, 'Y': 13.88,
           'P': 12.35, 'Q': 12.61, 'R': 13.10, 'S': 13.39, 'T': 12.70, 'V': 14.56}

cadict = {'A': 20, 'C': 25, 'D': 26, 'E': 33, 'F': 46, 'G': 13, 'W': 61,
          'H': 37, 'I': 39, 'K': 46, 'L': 35, 'M': 43, 'N': 28, 'Y': 46,
          'P': 22, 'Q': 36, 'R': 55, 'S': 20, 'T': 28, 'V': 33}

etdict = {'A': 1.9, 'C': 2.04, 'D': 1.52, 'E': 1.54, 'F': 1.86, 'G': 1.9, 'W': 1.87,
          'H': 1.76, 'I': 1.95, 'K': 1.37, 'L': 1.97, 'M': 1.96, 'N': 1.56, 'Y': 1.69,
          'P': 1.7, 'Q': 1.52, 'R': 1.48, 'S': 1.75, 'T': 1.77, 'V': 1.98}

strings = [
    '''RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQ
KSRHAA''',
    '''AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYV
MFAVA''',
    '''AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGK
HSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVR
AIHYSNAGLKQPNDGIESYSLFYKIPI'''
]

for seq in strings:
    hgm = sum(hgmdict[aa] for aa in seq) / len(seq)
    ca = sum(cadict[aa] for aa in seq)
    et = sum(etdict[aa] for aa in seq)

    print("For Sequence", strings.index(seq)+1)
    print("Hydrophobicity:", hgm)
    print("Helical Contact Area:", ca)
    print("Total Non-Bonded Energy:", et)
```

Output:

```
For Sequence 1
Hydrophobicity: 13.352537313432844
Helical Contact Area: 2156
Total Non-Bonded Energy: 117.74000000000005
For Sequence 2
Hydrophobicity: 13.77161764705882
Helical Contact Area: 2067
Total Non-Bonded Energy: 126.66000000000003
For Sequence 3
Hydrophobicity: 13.418675496688737
Helical Contact Area: 4616
Total Non-Bonded Energy: 267.7500000000001
```

Comment:

- The hydrophobicity levels across all mentioned sequences are comparable, mainly due to the prevalence of Alanine and other aliphatic amino acids within the polypeptide sequences.
- Sequence 3 exhibits a notably larger helix contact area compared to sequences 1 and 2. This difference may indicate that sequence 3 potentially adopts a right-handed alpha-helical structure, while the other peptides may favor a beta-sheet structure.
- Furthermore, the total non-bonded energy of sequence 3 surpasses that of sequences 1 and 2. This increase in energy is primarily attributed to significant values of "Et" for Glycine and Serine in sequence 3, resulting in a higher overall non-bonded energy content.

Practical 8
1 April 2024

Questions

1. Obtain the hydrophobicity profile for the sequences (Q1.fasta) and identify the α-helices and β-strands.
Hydrophobicity values:
A: 13.85 D: 11.61 C: 15.37 E: 11.38 F: 13.93 G: 13.34 H: 13.82 I: 15.28 K: 11.58 L: 14.13
M: 13.86 N: 13.02 P: 12.35 Q: 12.61 R: 13.10 S: 13.39 T: 12.70 V: 14.56 W: 15.48 Y: 13.88

2. Calcualte the amphiphatic index for the helices and strands found in Q1. Use stretch lengths of 8 and 6 for α-helices and β-strands, respectively.

3. Plot the hydrophobicity profile for the sequence (Q2.fasta) with window lengths 9 and 19 and list the transmembrane segments.

4. Use ScanProsite tool (https://prosite.expasy.org/scanprosite/ - select option 2), to search for the patterns a) [SV]-T-[VT]-[DERK](2)-{IL} and b) [FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY] in in UniProtKB (Include Swiss-Prot, isoforms). List the number of matches for each pattern.

5. Write a program to identify the patterns (refer Q4) in the sequence database (Q4.fasta). List the matches along with the sequence header and location of the matches in the sequence.

6. Identify the beta barrel membrane proteins with the following pattern:
[K,R,H,Q,F,E]-x-G-[I,V,L,F,A,C]-x-[I,V,L,F,M,Y,W]-x-[I,V,L,F,W]
Use: http://www.bioinformatics.org/sms2/protein_pattern.html
and
http://prosite.expasy.org/scanprosite/
Hint: Modify the patterns according to the input format of the server.

Deadline: 7th April 2024

BT 3040: BIOINFORMATICS

Assignment 8

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

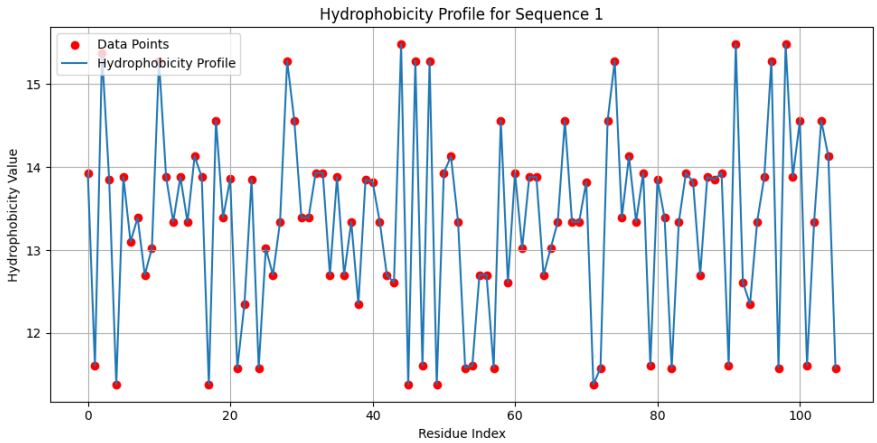
Indian Institute of Technology
Madras

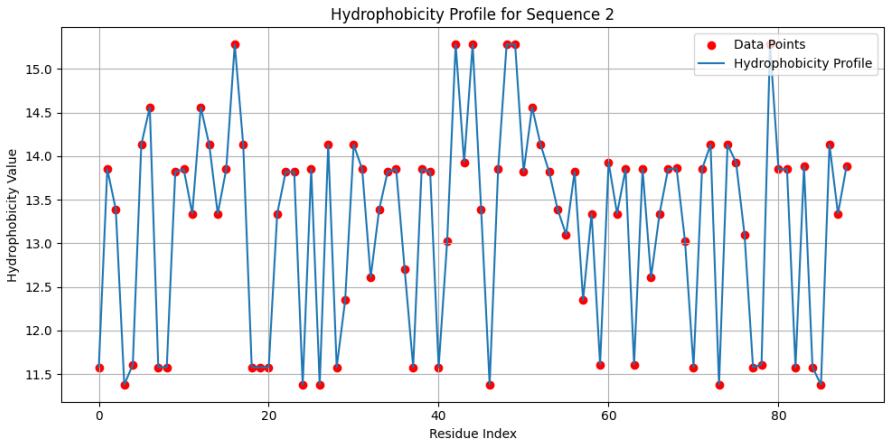
Q1) Compute the amino acid composition of the following sequences. Provide the output as a table of amino acid percentage values for each sequence and comment on the results.

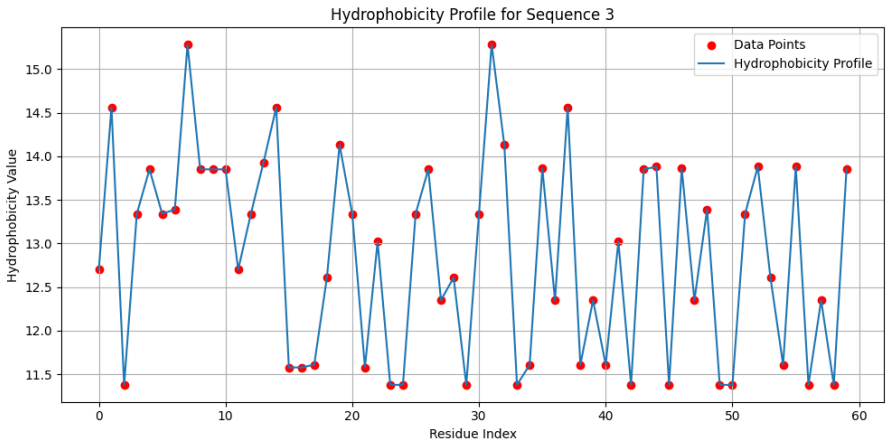
```
#BT3040 | Assignment 8 | Q1 |Atharva Mandar Phatak | BE21B009
import matplotlib.pyplot as plt

# hydrophobicity values dictionary
hydrophobicity_values = {
    'A': 13.85, 'D': 11.61, 'C': 15.37, 'E': 11.38, 'F': 13.93,
    'G': 13.34, 'H': 13.82, 'I': 15.28, 'K': 11.58, 'L': 14.13,
    'M': 13.86, 'N': 13.02, 'P': 12.35, 'Q': 12.61, 'R': 13.10,
    'S': 13.39, 'T': 12.70, 'V': 14.56, 'W': 15.48, 'Y': 13.88
}

# Function to calculate hydrophobicity values for a sequence
def calculate_hydrophobicity(sequence):
    return [hydrophobicity_values[residue] for residue in sequence]

# List of sequences
sequences = [q1_seq1,q1_seq2,q1_seq3]

# Create subplots for each sequence
fig, axs = plt.subplots(len(sequences), 1, figsize=(10, 5 * len(sequences)))

# Plot each sequence
for i, seq in enumerate(sequences):
    seq_val = calculate_hydrophobicity(seq)
    sequence = [i for i in seq]
    values = seq_val

    # Scatter plot
    axs[i].scatter(range(len(sequence)), values, marker='o', label='Data
Points',c='red')

    # Line plot
    axs[i].plot(range(len(sequence)), values ,label="Hydrophobicity Profile")

    # Customize subplot
    axs[i].set_title('Hydrophobicity Profile for Sequence {}'.format(i+1))
    axs[i].set_xlabel('Residue Index')
    axs[i].set_ylabel('Hydrophobicity Value')
    axs[i].grid(True)
    axs[i].legend()

# Adjust layout
```

```
plt.tight_layout()
plt.show()
```

Output







Seq1:

Alpha helix= [(5,12), (13,20), (25,32), (33,40), (60,66), (92,99)]

Beta sheet= [(0,5), (44,49)]

Seq2:

Alpha helix= [(1,8), (9,16), (32,39), (51,58)]

Beta sheet= [(40,45), (82,87)]

Seq3:

Alpha helix= [(3,10), (11,18), (19,26)]

Beta sheet= [(28,33), (40,45)]

Q2) Calcualte the amphiphatic index for the helices and strands found in Q1. Use stretch lengths of 8 and 6 for α-helices and β-strands, respectively.

```
#BT3040 | Assignment 8 | Q2 |Atharva Mandar Phatak | BE21B009

x1=[i for i in q1_seq1]
y1=calculate_hydrophobicity(q1_seq1)

x2=[i for i in q1_seq2]
y2=calculate_hydrophobicity(q1_seq2)

x3=[i for i in q1_seq3]
y3=calculate_hydrophobicity(q1_seq3)

def amphicity(x, y, A=None, B=None):
    # Beta sheet
    if B:
        for i in B:
            s, e = i
            ad1 = [sum(y[i] for i in range(s, e + 1) if i % 2 == 0)]
            ad2 = [sum(y[i] for i in range(s, e + 1) if i % 2 != 0)]
            re = abs((ad1[0] / 3) - (ad2[0] / 3))
            print(f"Beta sheet {i}: {re:.2f}")

    # Alpha helix
    if A:
        for i in A:
            s, e = i
            ad1 = [sum(y[i] for i in range(s, e + 1) if i % 4 == 0 or i % 4 ==
1)]
            ad2 = [sum(y[i] for i in range(s, e + 1) if i % 4 == 2 or i % 4 ==
3)]
            re = abs((ad1[0] / 4) - (ad2[0] / 4))
            print(f"Alpha helix {i}: {re:.2f}")


# seq1
A_1 = [(5, 12), (13, 20), (25, 32), (33, 40), (60, 66), (92, 99)]
B_1 = [(0, 5), (44, 49)]
# seq2 :
A_2 = [(1, 8), (9, 16), (32, 39), (51, 58)]
B_2 = [(40, 45), (82, 87)]
# seq3 :
A_3 = [(3, 10), (11, 18), (19, 26)]
B_3 = [(28, 33), (40, 45)]

print("For seq 1:")
```

```
amphicity(x1, y1, A_1, B_1)
print("\nFor seq 2:")
amphicity(x2, y2, A_2, B_2)
print("\nFor seq 3:")
amphicity(x3, y3, A_3, B_3)
```

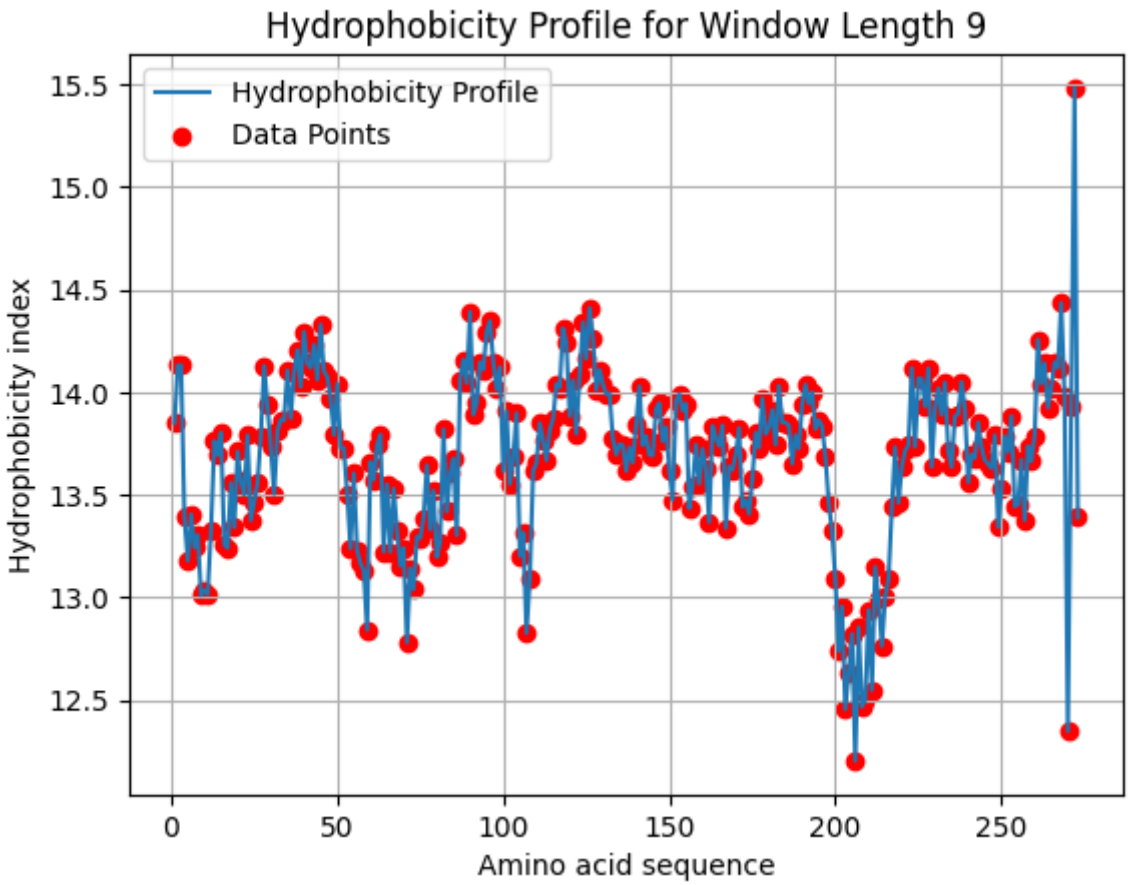
Output:

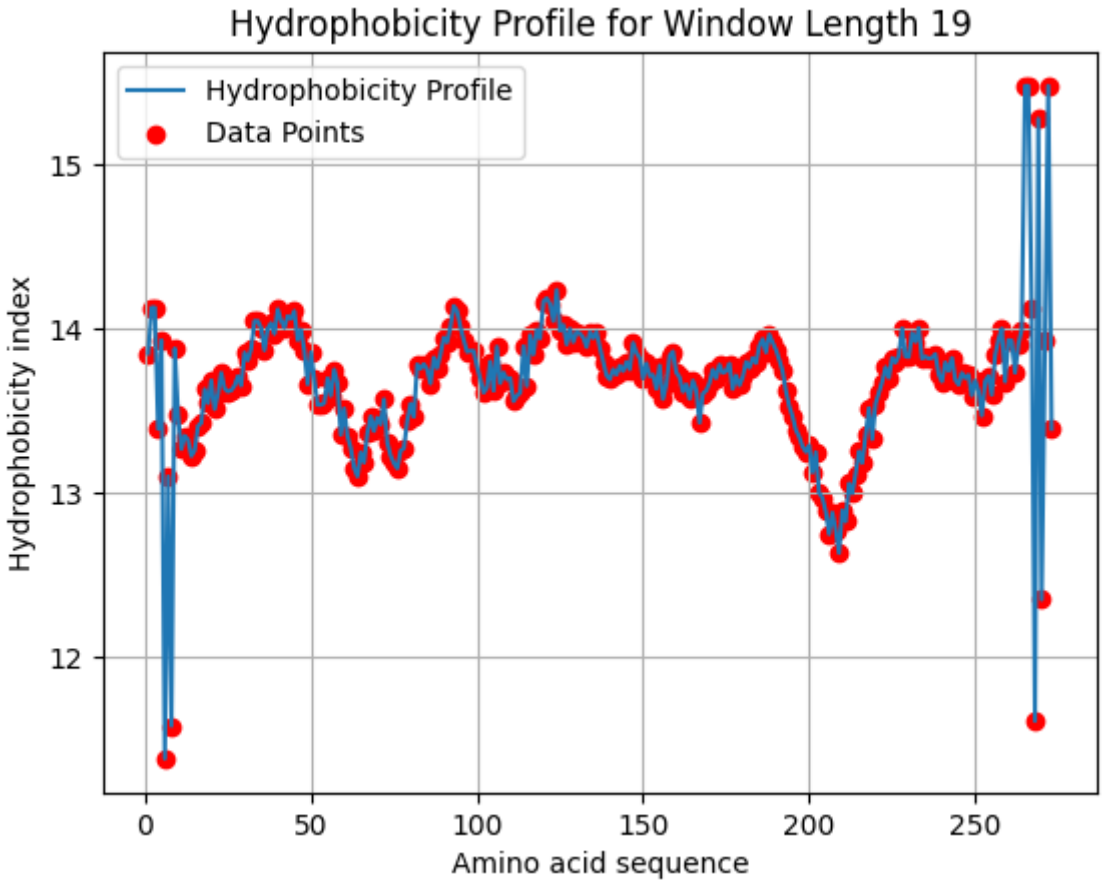
```
For seq 1:
Beta sheet (0, 5): 0.45
Beta sheet (44, 49): 3.89
Alpha helix (5, 12): 0.68
Alpha helix (13, 20): 0.61
Alpha helix (25, 32): 0.99
Alpha helix (33, 40): 0.25
Alpha helix (60, 66): 2.89
Alpha helix (92, 99): 1.19

For seq 2:
Beta sheet (40, 45): 0.60
Beta sheet (82, 87): 0.44
Alpha helix (1, 8): 0.06
Alpha helix (9, 16): 0.85
Alpha helix (32, 39): 1.27
Alpha helix (51, 58): 0.07

For seq 3:
Beta sheet (28, 33): 0.68
Beta sheet (40, 45): 0.46
Alpha helix (3, 10): 0.24
Alpha helix (11, 18): 0.25
Alpha helix (19, 26): 0.69
```

Q3) Plot the hydrophobicity profile for the sequence (Q2.fasta) with window lengths 9 and 19 and list the transmembrane segments

```
#BT3040 | Assignment 8 | Q3 |Atharva Mandar Phatak | BE21B009

import matplotlib.pyplot as plt

def my_h_r_plot(seq, w, wid_len):
    # Create x-axis values
    x = [i for i in range(1, len(seq) + 1)]
    # Create y-axis values
    y = [hydrophobicity_values[i] for i in seq]

    # Calculate moving average for hydrophobicity values
    yn = []
    for i in range(w):
        yn.append(y[i])
    for i in range(w, len(seq) - w):
        p = (sum(y[i - w:i]) + sum(y[i + 1:i + w + 1])) / (2 * w)
        yn.append(p)
    for i in range(len(seq) - w, len(seq)):
        yn.append(y[i])

    # Plot the hydrophobicity profile
    plt.figure()
    plt.plot(x, yn, label="Hydrophobicity Profile")
    plt.scatter(x, yn, c="red", label="Data Points")
    plt.grid(True)

    # Set axis labels and title
    plt.xlabel('Amino acid sequence')
    plt.ylabel('Hydrophobicity index')
    plt.title(f'Hydrophobicity Profile for {wid_len}')

    # Add legend
    plt.legend()

    # Show plot
    plt.show()


# Plot hydrophobicity profile with window length 4 and 9
my_h_r_plot(q2_seq1, 4, 'Window Length 9')
my_h_r_plot(q2_seq1, 9, 'Window Length 19')
```

Output:


Hydrophobicity Profile for Window Length 9
Hydrophobicity Profile
Data Points
Hydrophobicity index
Amino acid sequence
15.5
15.0
14.5
14.0
13.5
13.0
12.5
0
50
100
150
200
250



Transmembrane Segments: [9-49], [80-100], [107-193], [216-265]

Q4) Use ScanProsite tool (https://prosite.expasy.org/scanprosite/ - select option 2), to search for the patterns a) [SV]-T-[VT]-[DERK](2)-{IL} and b) [FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY] in UniProtKB (Include Swiss-Prot, isoforms). List the number of matches for each pattern.

a) Patten 1:

[ScanProsite] Results for job: Assignment8_Q4 External Inbox ×

Expasy <expasy@expasy.org> 10:13 AM (34 minutes ago)
to me

ScanProsite results:

WARNING: result output bigger than 4MB!
output truncated to 4MB
try to change your scan search parameters to get less results!

include splice variants (UniProtKB/Swiss-Prot)
Output format: Text

Hits for USERPAT1 "[SV]-T-[VT]-[DERK](2)-{IL}" on UniProtKB/Swiss-Prot sequences:
UniProtKB/Swiss-Prot (Release 2024_02 of 27-Mar-24) contains 571'282 entries.

found: 11007 hits in 10584 sequences

Hits for:
>USERPAT1 (user pattern):
Pattern: [SV]-T-[VT]-[DERK](2)-{IL}
Approximate number of expected random matches in ~ 100'000 sequences (50'000'000 residues): 1826

>sp|Q6GZS1|055L_FRG3G (431 aa)
Putative helicase 055L (EC 3.6.4.-). [Frog virus 3 (isolate Goorha) (FV-3)]
MAKLLRLNAIDGDMPGAGEADLFTLAPGGKAYVPFAWGSRVLGCKPPPAHGAARERGSVSLRPHQKGVLKEAWGHVTSKG
YCMLKCPPGFGKTFMALELWRRLGLPALVLTNRRVLATQWRDSATRFLPDSRVFTSGTPPPDALPRDLYVTGPASLRNRR
IKAKDSPAKFLLIVDEAHQLTSPVSCRVLLSVRPSHLLGLSATPMRYDDYHAALGAFFGREDSTVDRVDPRPHEVEILST
GVHIEPEFSKITGKMDWNSVIKAQSDNPERDAALADRMLLRPDVKWLVLCKRVDHVKRMAETLSSRSGKKVDVLHGSKDE
WDRDAWCVVGTYSKAGTGFDACERTGLCLAADVDRYFEQCLGRLRANGGTVLDPVDDLGVLRKHSKNREAVYIAAGCTIK
KTKCDASRPSQSTPTPTGSSQPAPRTRRPQR
223 - 228: STVDRV

11007 hits in 10584 sequences

b) Pattern 2

[ScanProsite] Results for job: Assignment8_Q4_Pattern2 External Inbox ×

Expasy <expasy@expasy.org> 10:53 AM (3 minutes ago)
to me

ScanProsite results:

WARNING: result output bigger than 4MB!
output truncated to 4MB
try to change your scan search parameters to get less results!

include splice variants (UniProtKB/Swiss-Prot)
Output format: Text

Hits for USERPAT1 "[FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY]" on UniProtKB/Swiss-Prot sequences:
UniProtKB/Swiss-Prot (Release 2024_02 of 27-Mar-24) contains 571'282 entries.

found: 3837 hits in 3774 sequences

Graphical View (graphical view with feature detection): https://p-prosite.expasy.org/cgi-bin/prosite/ScanView.cgi?scanfile=873655730501.scan.gz&sig=[FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY]
(link will be valid for 12hours)

Hits for:
>USERPAT1 (user pattern):
Pattern: [FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY]
Approximate number of expected random matches in ~ 100'000 sequences (50'000'000 residues): 40367724

>sp|Q4P2Q7|3HAO_USTMA (181 aa)
3-hydroxyanthranilate 3,4-dioxygenase (EC 1.13.11.6) (3-hydroxyanthranilate oxygenase) (3-HAO) (3-hydroxyanthranilic acid dioxygenase) (HAD) (Biosynthesis of nicotinic acid protein 1). [Ustilago maydis (strain 521 / FGSC 9021) (Corn smut fungus)]
MPFPLPLNFPKWLSENEHLLQPPVGNFCLFRTRDYTVMAVGGPNARSDYHYQPTEEFFYQYKGDMLLKVIDEDGKFQDIP
IKQGEMFMLPAHTPHSPVRFANTVGIVVERTRPDGSPDAMRWYCPNKEAHGETPTLVKEVHFQCTDLGTQLKPIIDAWVN
DEAGRQCSHCGYTQGARELPA
142 - 155: FQctdLGtqlKpil

>sp|Q6CFX1|3HAO_YARLI (171 aa)
3-hydroxyanthranilate 3,4-dioxygenase (EC 1.13.11.6) (3-hydroxyanthranilate oxygenase) (3-HAO) (3-hydroxyanthranilic acid dioxygenase) (HAD) (Biosynthesis of nicotinic acid protein 1). [Yarrowia lipolytica (strain CLIB 122 / E 150) (Yeast) (Candida lipolytica)]
MLQEPINLPKWLEENQHLLKPPVNNFCIQRGGYTVMIVGGPNARTDYHINQTPEWFHQKKGHMTLKVVDDGEFRDITINE
GDIFLLPANVPHNPVRYADTIGVVVEQDRPEGMKDALRWYCPNEKCREIVFENSFQLVDLGTQIKEAILDFDGDIEKRTC

3837 hits in 3774 sequences

Complete output in email. Can forward if required.

Q5) Write a program to identify the patterns (refer Q4) in the sequence database (Q4.fasta). List the matches along with the sequence header and location of the matches in the sequence.

```
import regex as re

pattern1 = "[SV]T[VT][DERK]{2}[^IL]"
pattern2 = "[FILV]Q...[^RK]G...[RK]..[FILVWY]"

with open("Q4.fasta") as file1:
    sequences = [line.strip() for line in file1]

for i in range(len(sequences)):
    matches1 = re.finditer(pattern1, sequences[i])
    matches2 = re.finditer(pattern2, sequences[i])

    for match in matches1:
        start_pos = match.start()
        end_pos = match.end()
        print(f"Pattern 1 matched at position {start_pos + 1} to {end_pos}
in:")
        print(sequences[i-1])
        print(sequences[i])
        print()

    for match in matches2:
        start_pos = match.start()
        end_pos = match.end()
        print(f"Pattern 2 matched at position {start_pos + 1} to {end_pos}
in:")
        print(sequences[i-1])
        print(sequences[i])
        print()
```

Output:

```
Pattern 1 matched at position 665 to 670 in:
>4A0C_2|Chains C,E|CULLIN-4B|HOMO SAPIENS (9606)
MHHHHHHVDENLYFQGGGRGSAKKLVIKNFKDKPKLPENYTDETWQKLKEAVEAIQNSTSIKYNLEELYQAVENLCSYKISANLYKQLRQICEDHIKAQIHQFREDSLDSVLFLKKIDRCWQNHCRQMIMIRSIFLFLDRTYVLQNSMLPSIWDMGLELFRAHIISDQKVQNKTIDGILLLIERI

Pattern 1 matched at position 665 to 670 in:
>4A0C_2|Chains C,E|CULLIN-4B|HOMO SAPIENS (9606)
MHHHHHHVDENLYFQGGGRGSAKKLVIKNFKDKPKLPENYTDETWQKLKEAVEAIQNSTSIKYNLEELYQAVENLCSYKISANLYKQLRQICEDHIKAQIHQFREDSLDSVLFLKKIDRCWQNHCRQMIMIRSIFLFLDRTYVLQNSMLPSIWDMGLELFRAHIISDQKVQNKTIDGILLLIERI

Pattern 1 matched at position 666 to 671 in:
>4A0K_1|Chain A|CULLIN-4A|HOMO SAPIENS (9606)
MHHHHHHVDEENLYFQGGGRGGSKKLVIKNFRDRPRLPDNYTQDTWRKLHEAVRAVQSSTSIRYNLEELYQAVENLCSHKVSPMLYKQLRQACEDHVQAQILPFREDSLDSVLFLKKINTCWQDHCRQMIMIRSIFLFLDRTYVLQNSTLPSIWDMGLELFRTHIISDKMVQSKTIDGILLLIEI

Pattern 2 matched at position 252 to 265 in:
>4FXG_2|Chains B,E|Complement C4-A alpha chain|Homo sapiens (9606)
NVNFQKAINEKLGQYASPTAKRCCQDGVTRLPMMRSCEQRAARVQQPDCREPFLSCCQFAESLRKKSRDKGQAGLQRALEILQEEDLIDEDDIPVRSFFPENWLWRVETVDRFQILTLWLPDSLTTWEIHGLSLSKTKGLCVATPVQLRVFREFHLHLRLPMSVRRFEQLELRPVLYNYLDKNL

Pattern 2 matched at position 252 to 265 in:
>4FXG_2|Chains B,E|Complement C4-A alpha chain|Homo sapiens (9606)
NVNFQKAINEKLGQYASPTAKRCCQDGVTRLPMMRSCEQRAARVQQPDCREPFLSCCQFAESLRKKSRDKGQAGLQRALEILQEEDLIDEDDIPVRSFFPENWLWRVETVDRFQILTLWLPDSLTTWEIHGLSLSKTKGLCVATPVQLRVFREFHLHLRLPMSVRRFEQLELRPVLYNYLDKNL

Pattern 2 matched at position 252 to 265 in:
>4FXK_2|Chain B|Complement C4-A Alpha chain|Homo sapiens (9606)
NVNFQKAINEKLGQYASPTAKRCCQDGVTRLPMMRSCEQRAARVQQPDCREPFLSCCQFAESLRKKSRDKGQAGLQRALEILQEEDLIDEDDIPVRSFFPENWLWRVETVDRFQILTLWLPDSLTTWEIHGLSLSKTKGLCVATPVQLRVFREFHLHLRLPMSVRRFEQLELRPVLYNYLDKNL
```

Complete output in ‘Q5_MatchingSequences.txt’ file

Q6) Identify the beta barrel membrane proteins with the following pattern: [K,R,H,Q,F,E]-x-G-[I,V,L,F,A,C]-x-[I,V,L,F,M,Y,W]-x-[I,V,L,F,W] Use: http://www.bioinformatics.org/sms2/protein_pattern.html and http://prosite.expasy.org/scanprosite/ Hint: Modify the patterns according to the input format of the server.

1) SMS Output

Protein Pattern Find results

```
Results for 284 residue sequence "3b07_A" starting "VTLYKTTATA"

no matches found for this sequence.


Results for 270 residue sequence "3b07_B" starting "AEIIKRTQDI"

no matches found for this sequence.


Results for 60 residue sequence "8b4i_C" starting "WFYHKYSTTT"

no matches found for this sequence.


Results for 38 residue sequence "8b4i_E" starting "ALSREELQAA"

no matches found for this sequence.


Results for 51 residue sequence "8b4i_I" starting "ALSEESKERI"

no matches found for this sequence.


Results for 211 residue sequence "7cbl_A" starting "CAWIPAKPLV"

no matches found for this sequence.


Results for 303 residue sequence "7cbl_a" starting "ERIRDLTSVQ"

no matches found for this sequence.
```

2) Prosite Output

(/)

Home (/)
Browse (/cgi-bin/prosite/prosite-list.pl)
Documentation (/prosite_doc.html)
About (/prosite_details.html)
ScanProsite (/scanprosite/)
ProRule (/prorule.html)
Downloads (https://ftp.expasy.org/databases/prosite)
Contact (/contact)

ScanProsite Res

Ouput format: Graphical view - this view shows ScanProsite results together with ProRule-based predicted intra-domain feat

Hits for USERPAT1 "[KRHQFE]-x-G-[IVLFAC]-x-[IVLFMYW]-x-[IVLFW]" on LGr

found: 94 hits in 70 sequences

Legend:

disulfide bridge active site other 'ranges' other sites

Please note that the graphical representations of domains displayed hereafter are for illustrative purposes only, and that thei
For more information about how these graphical representations are constructed, go to https://prosite.expasy.org/mydomain

hits by patterns: [94 hits (by 1 pattern) on 70 sequences]

ruler: 1 100 200 300 400 500 600 700 800 900 100

1bh3_A — View all PROSITE motifs hits on sequence (PSScan.cgi?seq=>1bh3_A%0A&output=nice)

USERPAT1 :
Pattern: **[KRHQFE]-x-G-[IVLFAC]-x-[IVLFMYW]-x-[IVLFW]**
Approximate number of expected random matches [Ref: PMID 11535175 (http://www.ncbi.nlm.nih.gov:80/entrez/query.fcgi?
26784

195 - 202: FvGAaYkF

Outputs attached as PDFs, 'Q6_SMS_Output.pdf' and 'Q6_prosite_match.pdf'

Practical 9

Questions

08 April 2024

1. Identify the pair of sequences which are close to each other using Hamming and Euclidean distance methods.

(i)AMENLNMDLLYMAAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA

(ii)AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI

(iii)MALLPAAPGAPARATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA

2. Get the non-redundant sequences of beta barrel membrane proteins with sequence identities of less than 40%, 50%, 75% and 90% using CD-HIT

3. Get the non-redundant sequences of the same type of proteins with sequence identities of less than 20%, 30%, 40% and 50% using PISCES (https://dunbrack.fccc.edu/pisces/)

4. Compare the results obtained with the cut-offs 40% and 50%.

5. Extract the data with the cut-off of 50% from Uniprot and compare with CD-HIT and PISCES.

Deadline: 14th April 2024

BT 3040: BIOINFORMATICS

Assignment 9

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) 1. Identify the pair of sequences which are close to each other using Hamming and Euclidean distance methods.

(i)AMENLNMDLLYMAAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIP LLRTQFFIVMGLVDAIPMIAVG

LGLYVMFAVA

(ii)AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAG YTYWEGGDEGAGKHSLSFAP

VFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFAN GQSVGVRAIHYSNAGLKQPN

DGIESYSLFYKIPI

(iii)MALLPAAPGAPARATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYA WTGLLTASKPSLHHATATPEY

LAALKQKSRHAA

```
#Q1 | Atharva Mandar Phatak | BE21B009 | Assignment 9 | BT3040 |

def calculate_aa_percentage(sequence):
    # Define the 20 standard amino acids
    amino_acids = "ACDEFGHIKLMNPQRSTVWY"

    # Initialize a dictionary to store the percentage occurrence of each amino acid
    aa_percentage = {aa: 0 for aa in amino_acids}

    # Count occurrences of each amino acid in the sequence
    total_aa_count = 0
    for aa in sequence:
        if aa in amino_acids:
            aa_percentage[aa] += 1
            total_aa_count += 1

    # Calculate percentage occurrence of each amino acid
    if total_aa_count > 0:
        for aa in aa_percentage:
            aa_percentage[aa] = (aa_percentage[aa] / total_aa_count) * 100

    # Sort the dictionary by keys
    sorted_aa_percentage = {k: v for k, v in sorted(aa_percentage.items())}

    return sorted_aa_percentage

def hamming(dict1, dict2):
    # Initialize the sum of differences
    sum_of_differences = 0
```

```
    # Iterate through each key in the dictionaries
    for key in dict1.keys():
        # Calculate the difference between the values of corresponding keys
        difference = dict1[key] - dict2[key]

        # Take the modulus of the difference
        difference_mod = abs(difference)  # Assuming you meant mod 100

        # Add the modulus to the sum
        sum_of_differences += difference_mod

    return sum_of_differences



def euclidean(dict1, dict2):
    # Initialize the sum of squared differences
    sum_of_squared_differences = 0

    # Iterate through each key in the dictionaries
    for key in dict1.keys():
        # Calculate the difference between the values of corresponding keys
        difference = dict1[key] - dict2[key]

        # Square the difference
        squared_difference = difference ** 2

        # Add the squared difference to the sum
        sum_of_squared_differences += squared_difference

    # Take the square root of the sum of squared differences
    sqrt_sum = (sum_of_squared_differences)**(0.5)
    return sqrt_sum



dict1 = calculate_aa_percentage(seq1)
dict2 = calculate_aa_percentage(seq2)
dict3 = calculate_aa_percentage(seq3)

hammering_1_2 = hamming(dict1, dict2)
hammering_1_3 = hamming(dict1, dict3)
hammering_2_3 = hamming(dict2, dict3)

eucledian_1_2= euclidean(dict1, dict2)
eucledian_1_3= euclidean(dict1, dict3)
eucledian_2_3= euclidean(dict2, dict3)

print(f"For pair 1 and 2, Hamming Distance is {hammering_1_2}")
```

```
print(f"For pair 1 and 3, Hamming Distance is {hammering_1_3}")
print(f"For pair 2 and 3, Hamming Distance is {hammering_2_3}")
print("")
print(f"For pair 1 and 2, Euclidean Distance is {eucledian_1_2}")
print(f"For pair 1 and 3, Euclidean Distance is {eucledian_1_3}")
print(f"For pair 2 and 3, Euclidean Distance is {eucledian_2_3}")
```

Output:

```
For pair 1 and 2, Hamming Distance is 66.5728476821192
For pair 1 and 3, Hamming Distance is 84.33544303797467
For pair 2 and 3, Hamming Distance is 72.66325760751111

For pair 1 and 2, Euclidean Distance is 20.1062168421535
For pair 1 and 3, Euclidean Distance is 22.086816691389576
For pair 2 and 3, Euclidean Distance is 20.112952107271116
```

Pair	Hamming Distance	Euclidean Distance
Seq 1 and Seq 2	66.57	20.11
Seq 1 and Seq 3	84.34	22.09
Seq 2 and Seq 3	72.66	20.11
Shortest	**66.57**	**20.11**

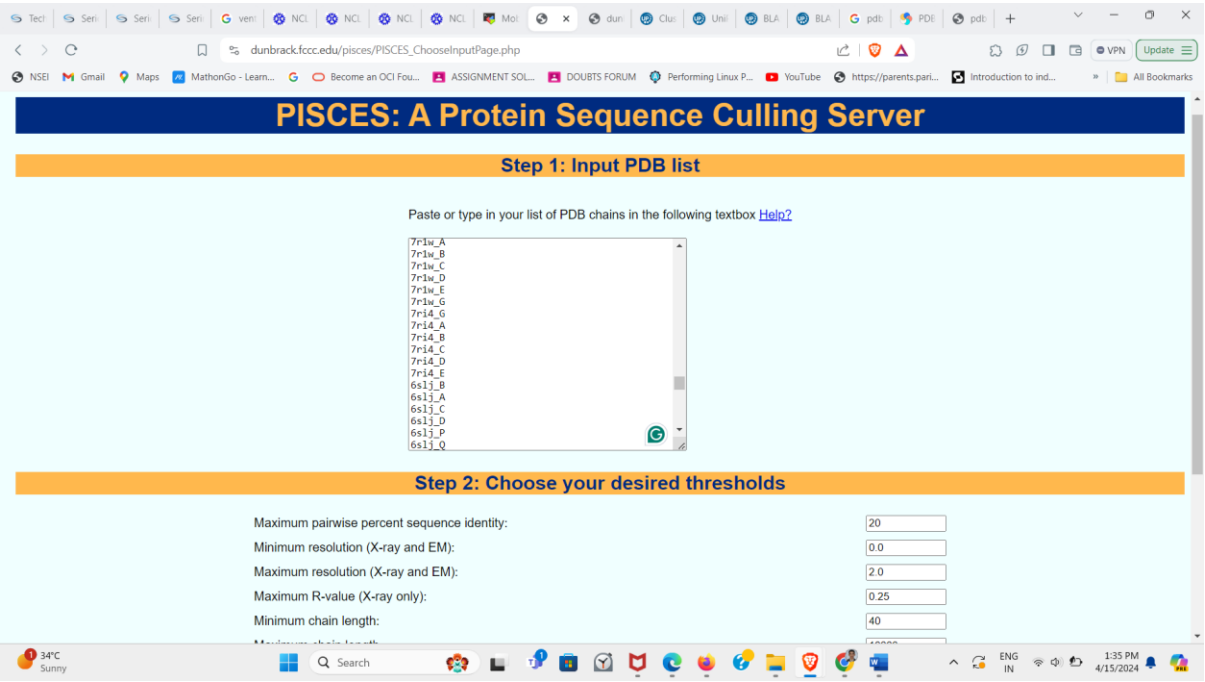
Q2) Get the non-redundant sequences of beta barrel membrane proteins with sequence identities of less than 40%, 50%, 75% and 90% using CD-HIT

CD-HIT	
Percentage Identity	**No. of Clusters**
40%	240
50%	265
75%	330
90%	370

The downloaded files are in the folder

Q3) Get the non-redundant sequences of the same type of proteins with sequence identities of less than 20%, 30%, 40% and 50% using PISCES (https://dunbrack.fccc.edu/pisces/

a. 20% (similarly for others, screenshot attaching only for one)



PISCES	
Percentage Identity	**No. of Clusters**
20%	31
30%	39
40%	42
50%	47

The downloaded files are in the folder

Q4) Compare the results obtained with the cut-offs 40% and 50%.

Comparision				
Resource	**Percentage Identity**	**Mean Length**	**Min Length**	**Max Length**
CD-HIT	40	341.6	11	2124
	50	341.6	11	2124
PISCES	40	390.8	99	77
	50	393.5	99	774

The number of chains for **40%** and **50%** are **42** and **47** respectively

Analysis:

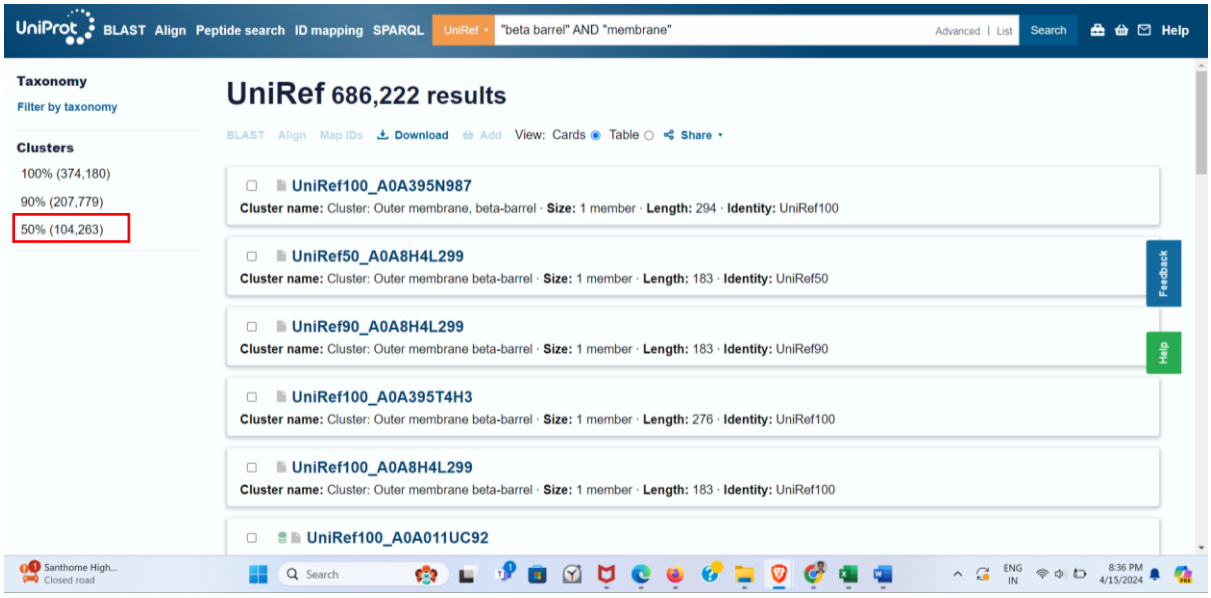
Comparing the length of sequences between PISCES and CD-HIT datasets reveals that, on average, PISCES tends to include longer sequences, possibly favouring complete protein structures, while CD-HIT encompasses a broader range of sequence lengths, from short fragments to longer ones.

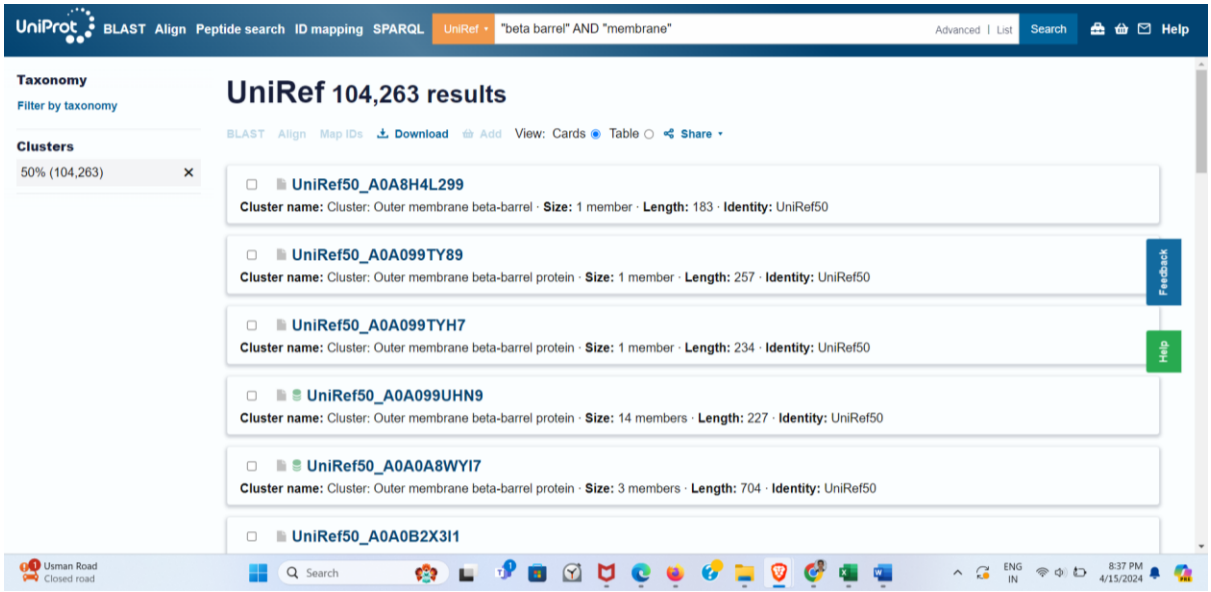
CD-HIT clusters formed with default cutoff values of 40% and 50% for redundant beta barrel membrane proteins resulted in smaller clusters with higher sequence similarity, suggesting tight groupings of very similar sequences.

PISCES selects sequences above a higher cutoff based on structural quality and non-redundancy, indicating a focus on structural integrity. The resolutions of sequences in PISCES datasets show minimal variation between the 40% and 50% cutoffs, indicating consistently high-quality structures.

In terms of unique contributions, CD-HIT is well-suited for studies requiring high sequence similarity, such as sequence homology or evolutionary analysis, while PISCES is more beneficial for structural biology, prioritizing high-quality and unique protein structures for applications like modelling and drug design.

Q5) Extract the data with the cut-off of 50% from UniProt and compare with CD-HIT and PISCES





Comparison:

Clusters with 50% cutoff	
UniProt	104263
CD-HIT	47
PISCES	265

Practical 10

12 April 2024

Questions

1. Obtain the consensus phylogenetic tree for the following two sets of sequences:

Set 1: tim.dat
Set 2: tim-hemo.dat

Hint:
(i) Multiple sequence alignment using MAFFT
(ii) Save in Phylip format
(iii) Install Phylip (windows) / Phylip (Mac OS)
(iv) Bootstrapping (Seqboot program)
(v) Maximum likelihood method (proml program)
(vi) Consensus tree (Consens program)
(vii) Use TreeView / MEGA-X to view the tree
(viii) NJ and UPGMA methods (protdist and neighbor programs)

Refer to practical 10.ppt for detailed steps.

2. Obtain the weight matrix for the following sequences

```
MVLSPADKTNVKGKVGAHAGEYGAAAW
MKRLPADPPCVKGKVKAKAGDYGATTW
MALSAADKTNVKSKVGGHAGEYGAATS
MVLSAADKTNVKSKAGGNAGEWWAAAW
MVLSAADKTNVKSKVLANAGEFGAAAW
ALLPIRTTYHKKCASGHIPEEKDLNNV
DEASSLKGHHIKKLEADALLIPLSASS
```

Deadline: 19 April 2024

BT 3040: BIOINFORMATICS

Assignment 10

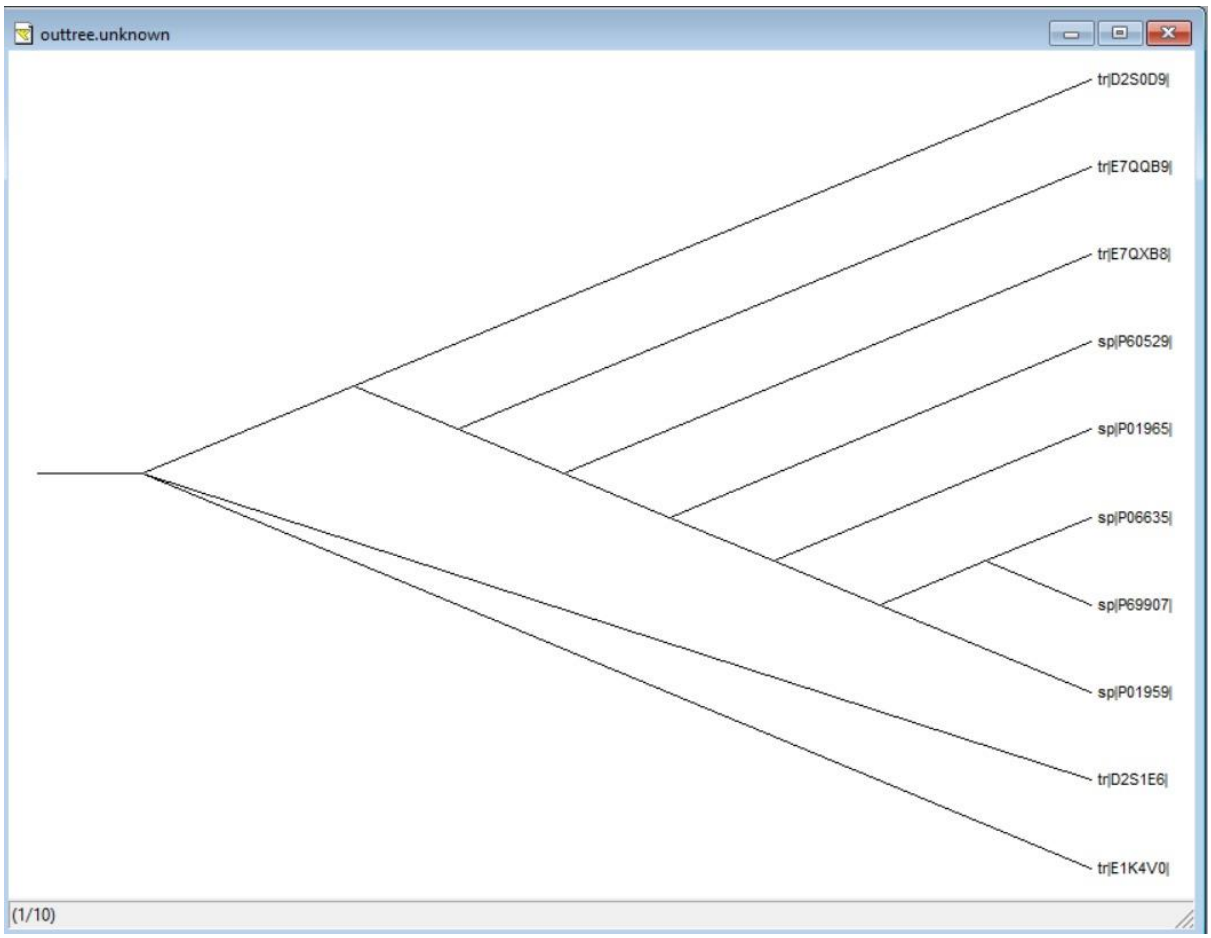
Atharva Mandar Phatak | BE21B009
Department of Biotechnology

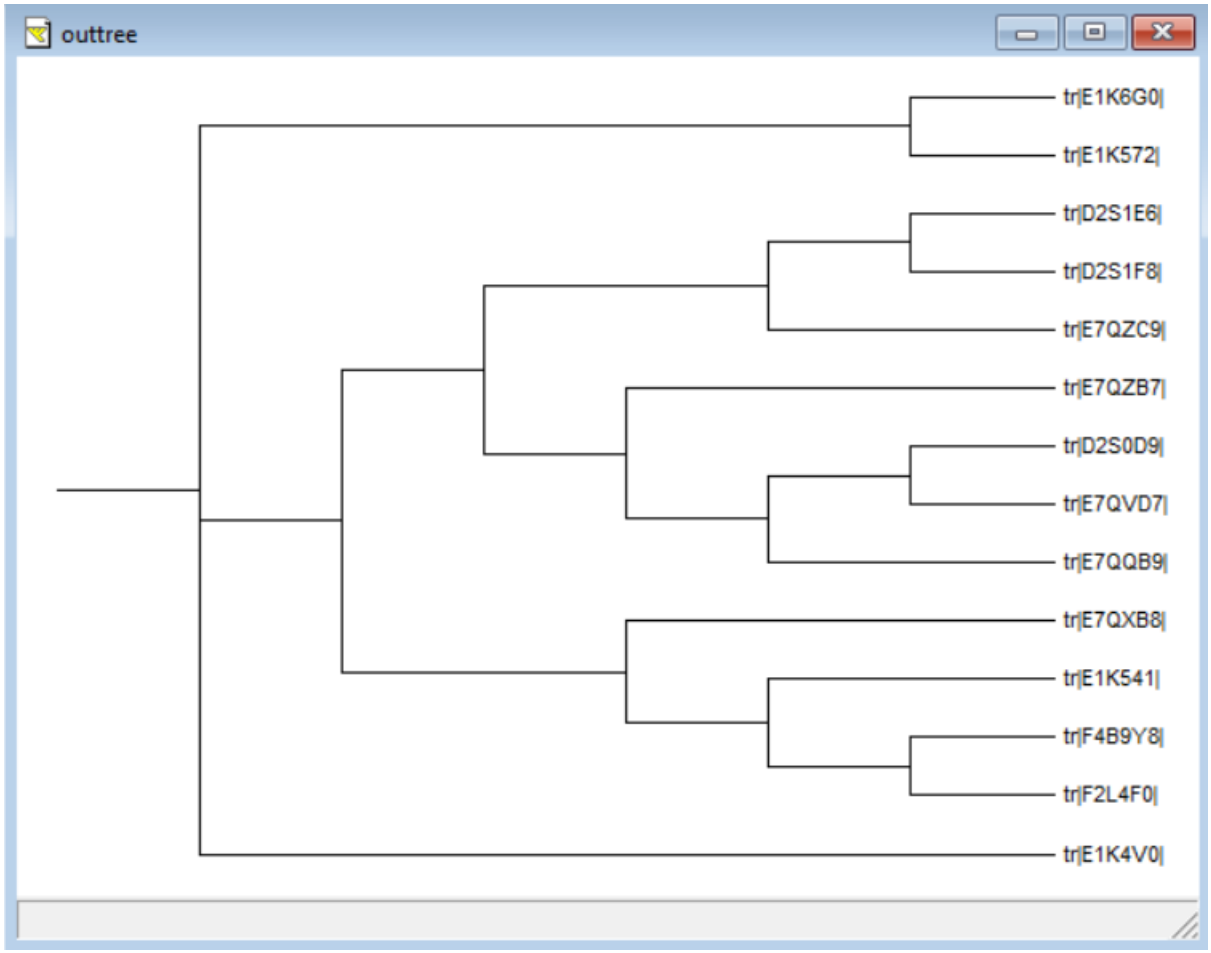
Indian Institute of Technology
Madras

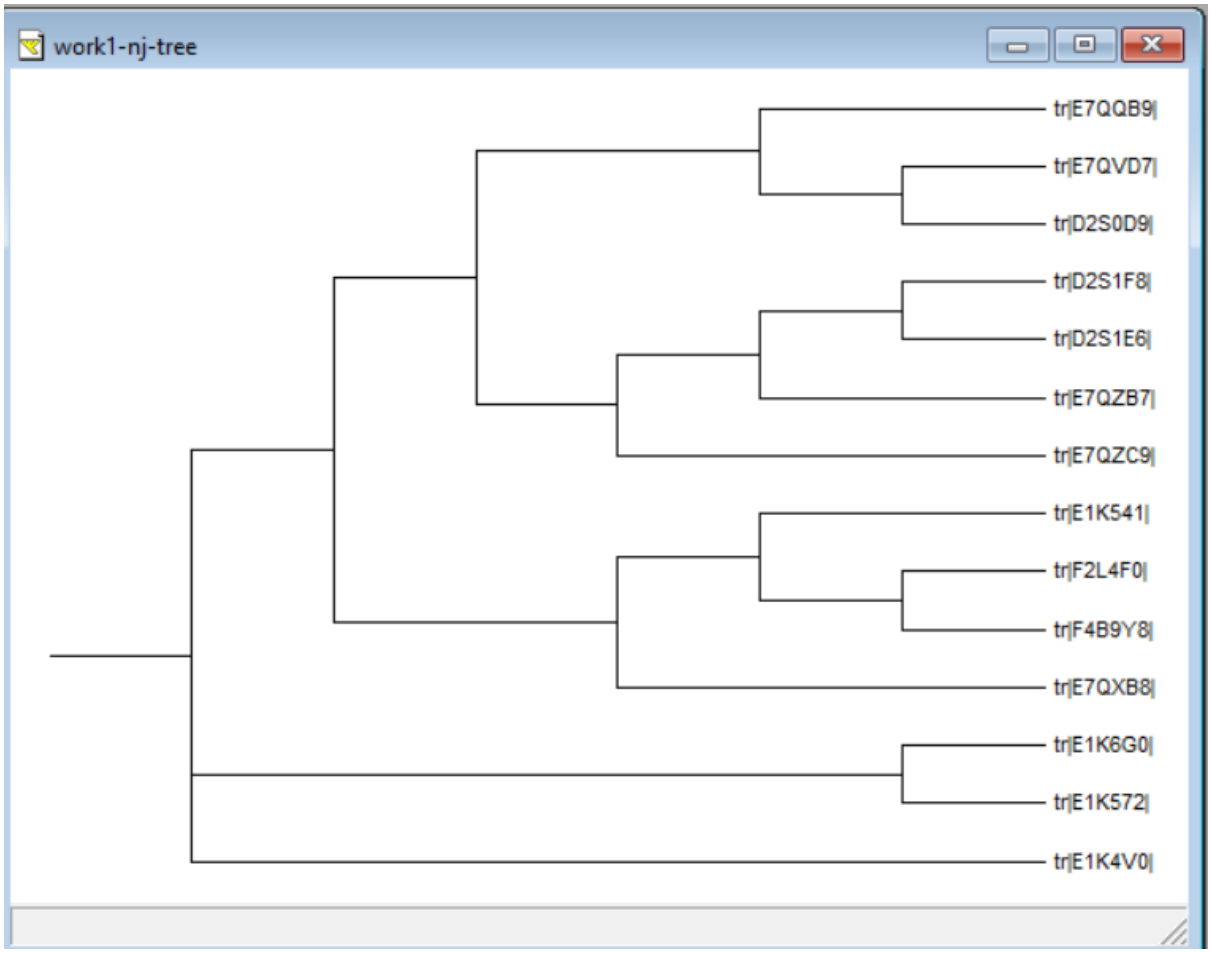
Q1) Obtain the consensus phylogenetic tree for the following two sets of sequences:

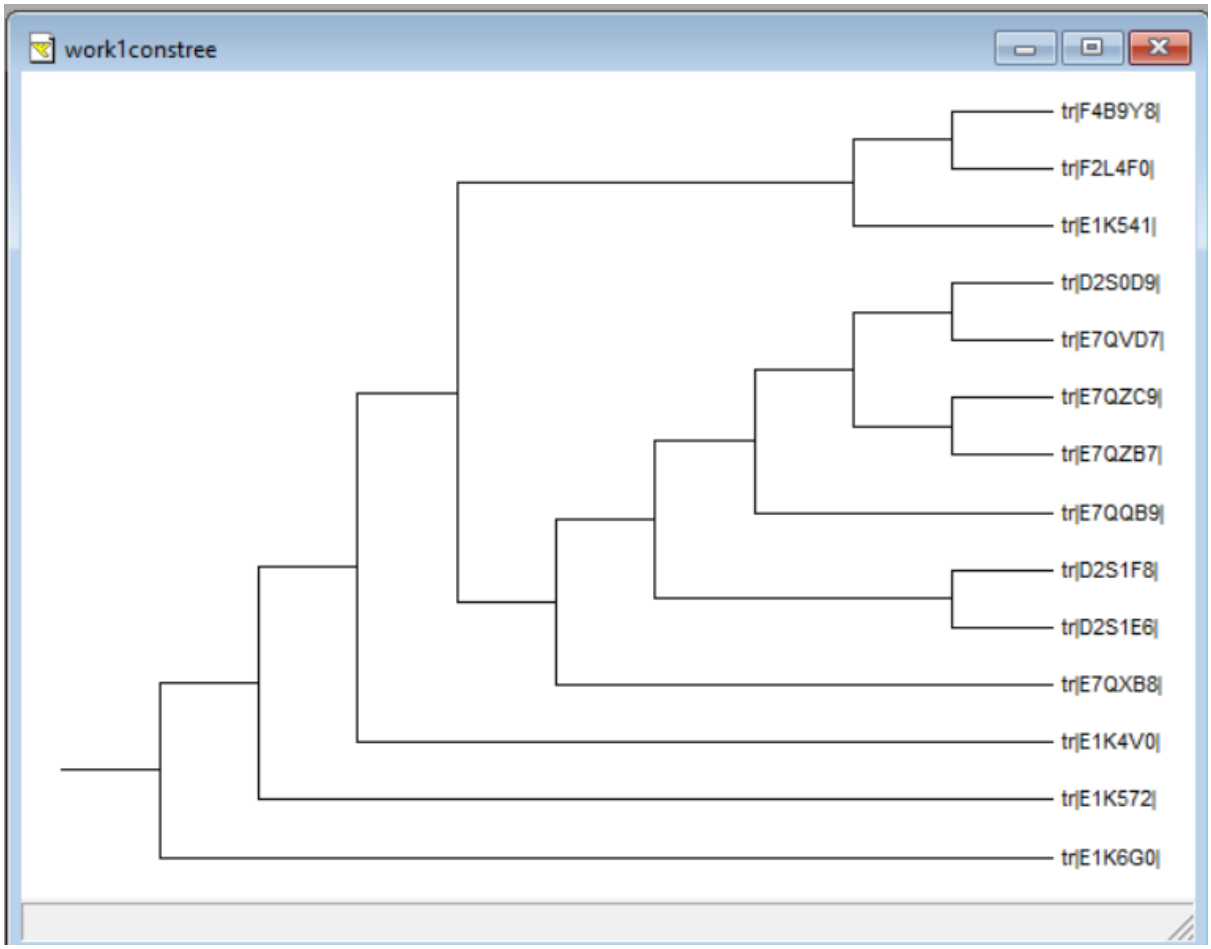
Set 1: tim.dat

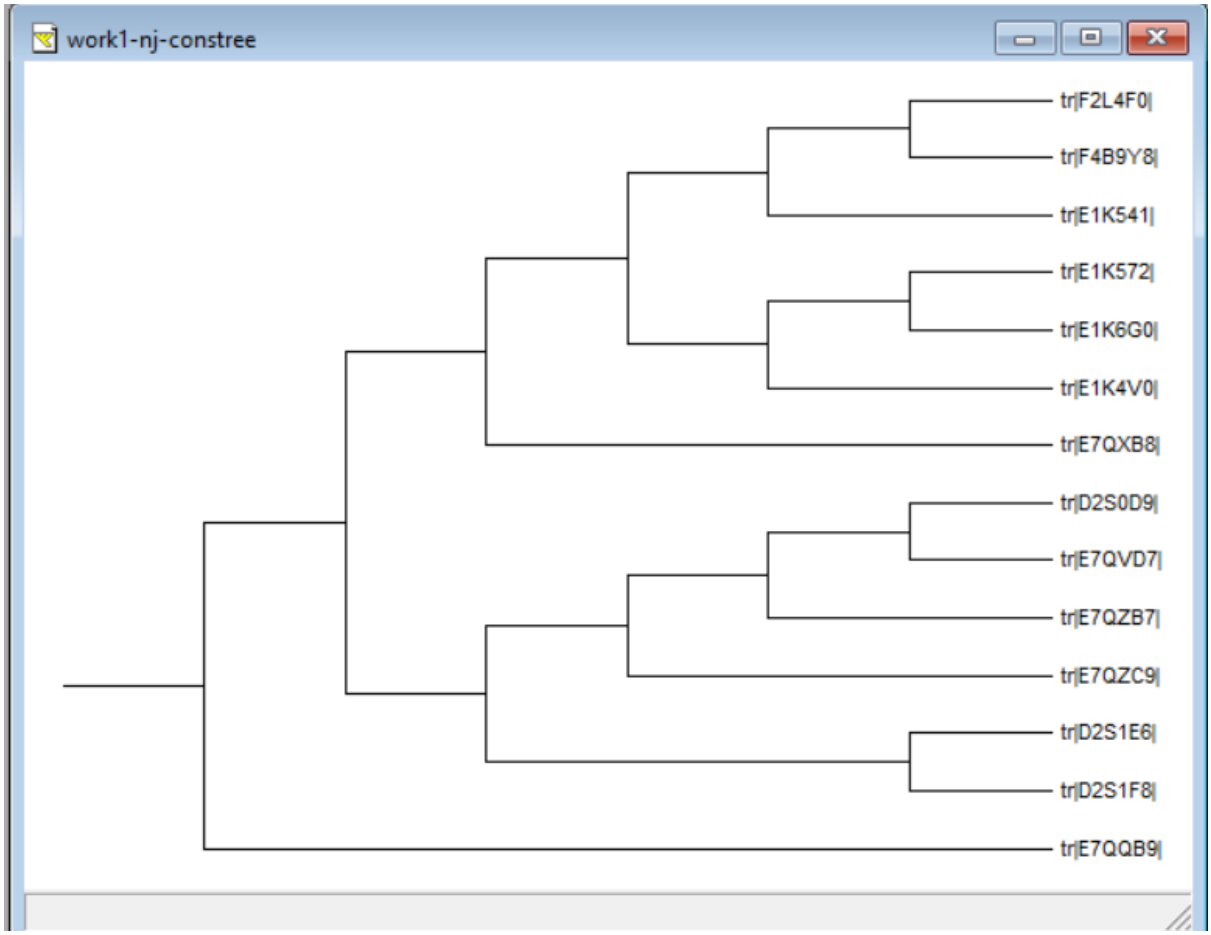
All the files in 'tim_dat' folder




outtree
tr|E1K6G0|
tr|E1K572|
tr|D2S1E6|
tr|D2S1F8|
tr|E7QZC9|
tr|E7QZB7|
tr|D2S0D9|
tr|E7QVD7|
tr|E7QQB9|
tr|E7QXB8|
tr|E1K541|
tr|F4B9Y8|
tr|F2L4F0|
tr|E1K4V0|

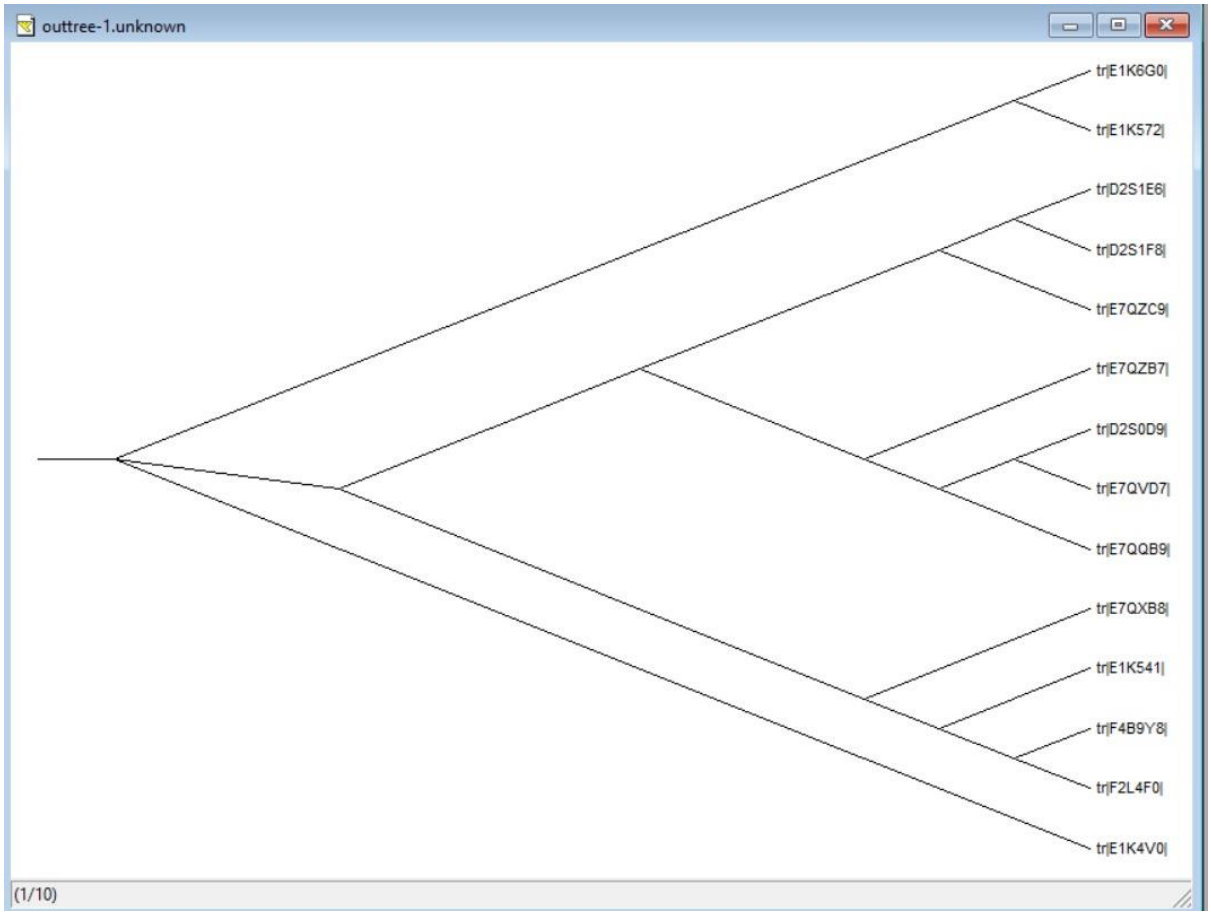

work1-nj-tree
tr|E7QQB9|
tr|E7QVD7|
tr|D2S0D9|
tr|D2S1F8|
tr|D2S1E6|
tr|E7QZB7|
tr|E7QZC9|
tr|E1K541|
tr|F2L4F0|
tr|F4B9Y8|
tr|E7QXB8|
tr|E1K6G0|
tr|E1K572|
tr|E1K4V0|

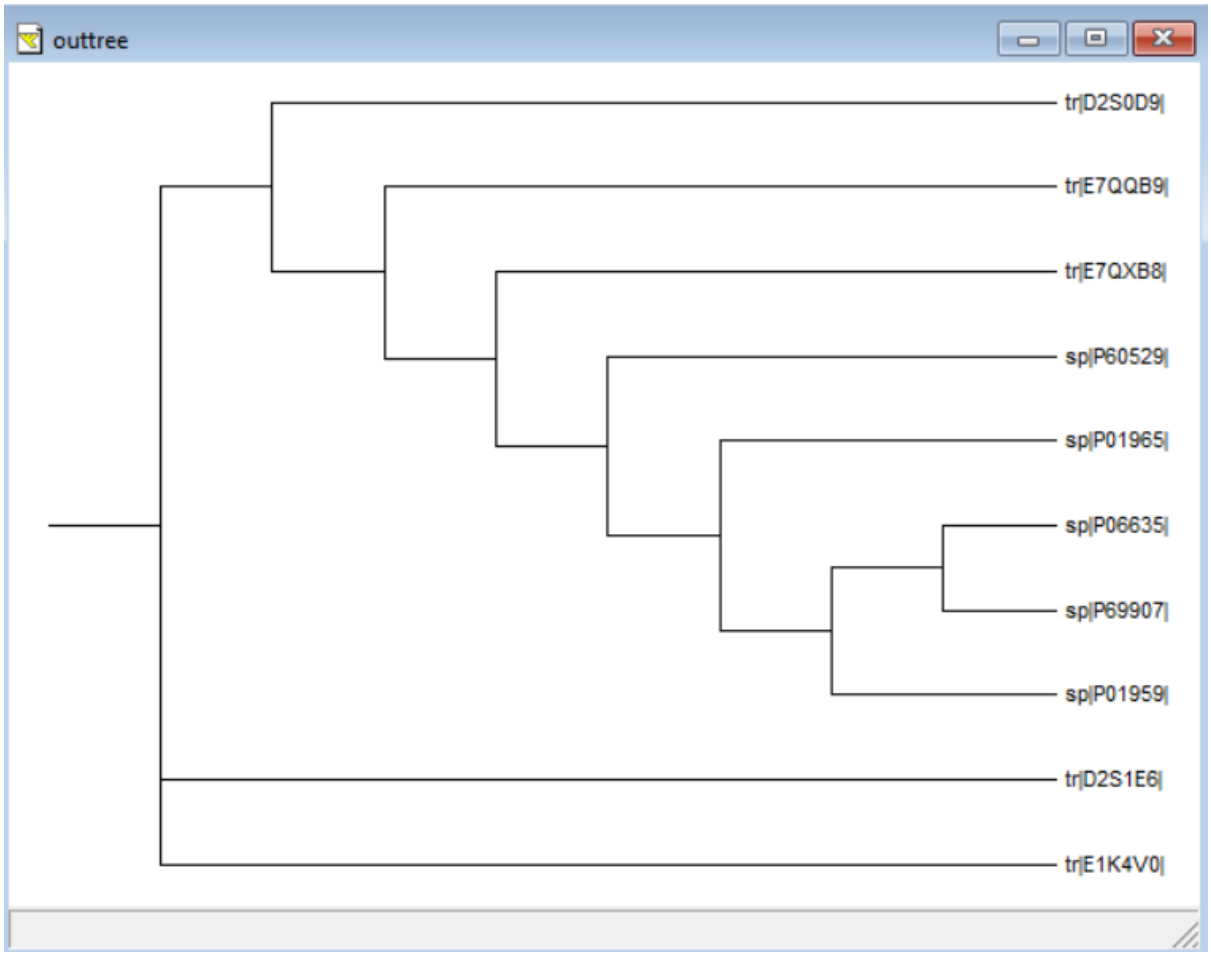

work1constree
tr|F4B9Y8|
tr|F2L4F0|
tr|E1K541|
tr|D2S0D9|
tr|E7QVD7|
tr|E7QZC9|
tr|E7QZB7|
tr|E7QQB9|
tr|D2S1F8|
tr|D2S1E6|
tr|E7QXB8|
tr|E1K4V0|
tr|E1K572|
tr|E1K6G0|

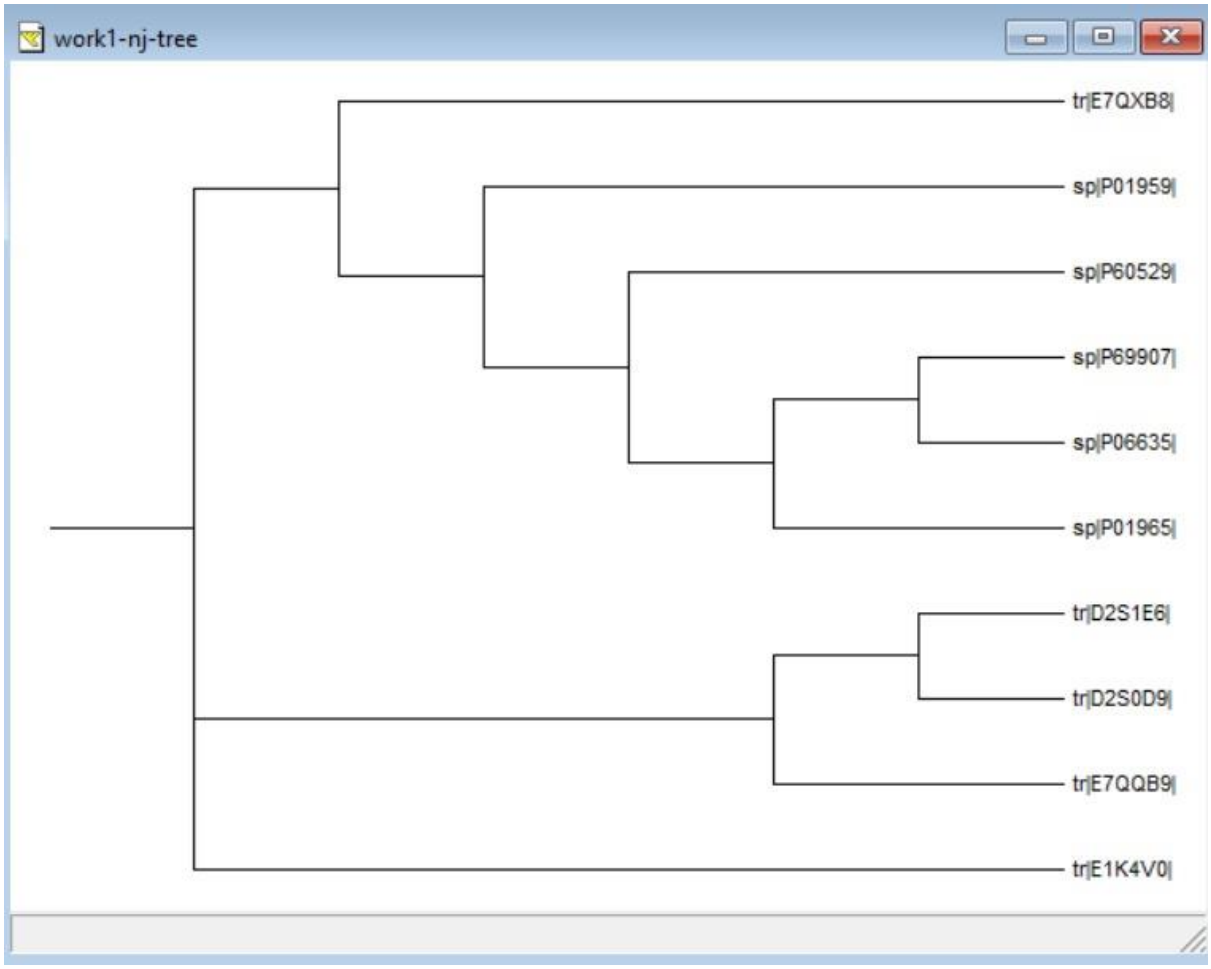

work1-nj-constree
tr|F2L4F0|
tr|F4B9Y8|
tr|E1K541|
tr|E1K572|
tr|E1K6G0|
tr|E1K4V0|
tr|E7QXB8|
tr|D2S0D9|
tr|E7QVD7|
tr|E7QZB7|
tr|E7QZC9|
tr|D2S1E6|
tr|D2S1F8|
tr|E7QQB9|

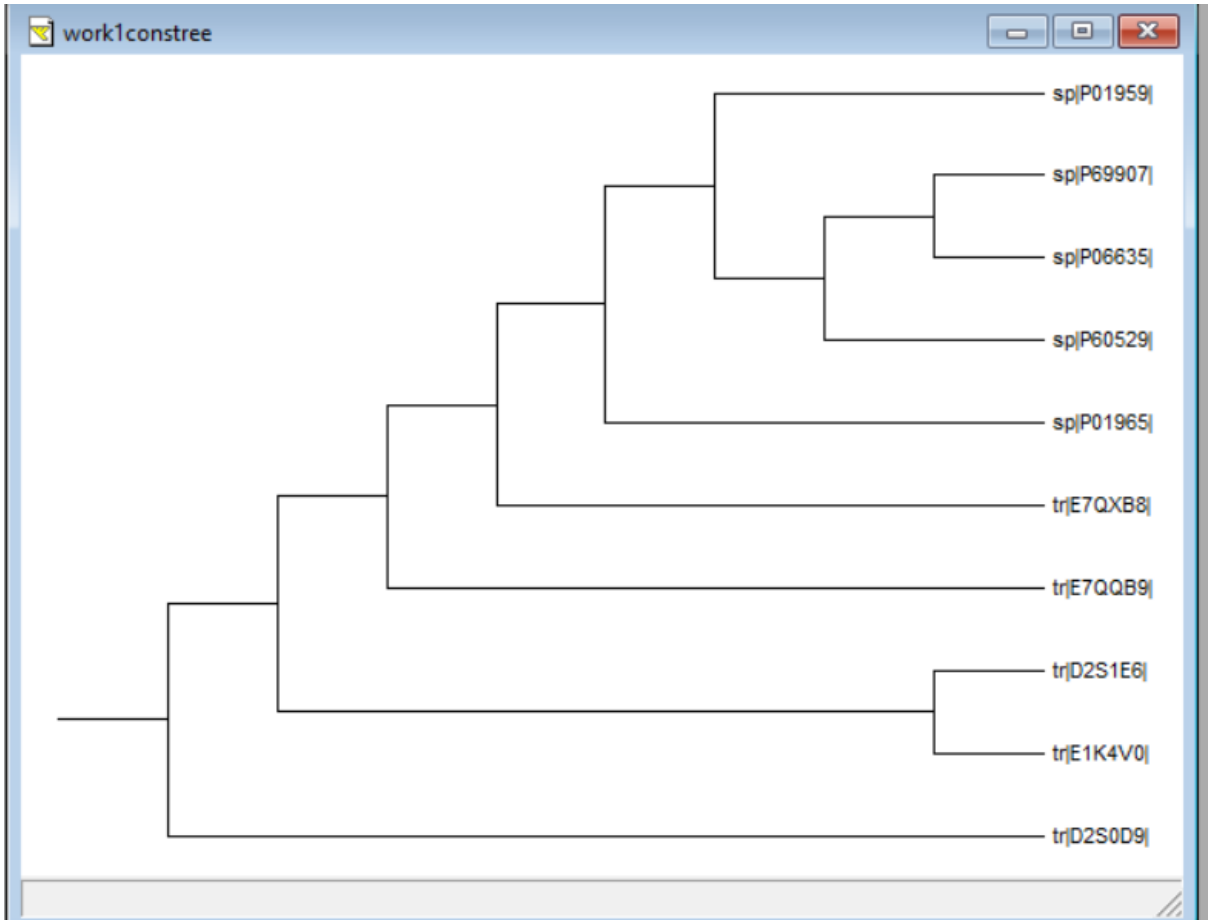
Set 2: tim-hemo.dat

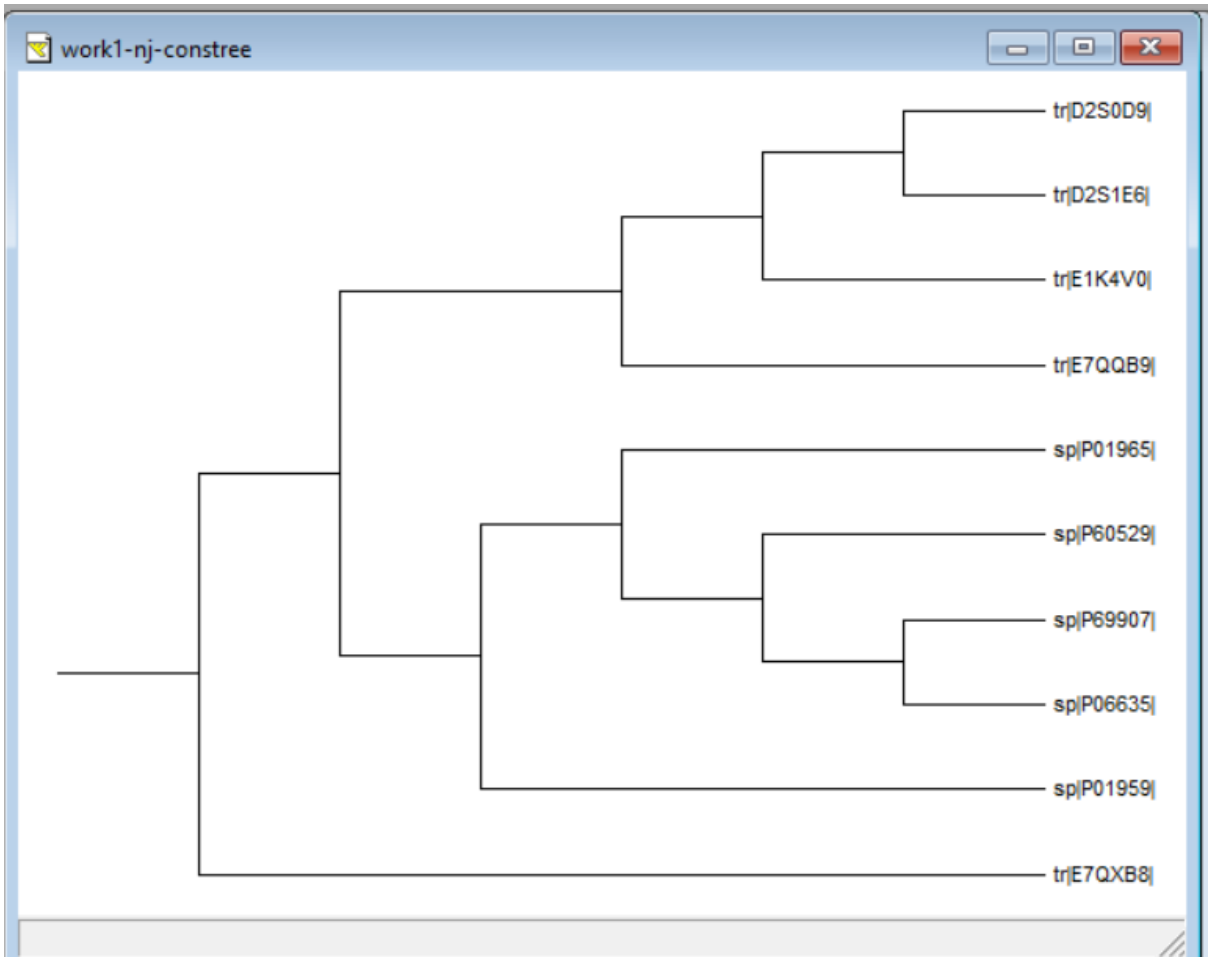
All the files in 'tim_dat_hemo' folder




outtree
tr|D2S0D9|
tr|E7QQB9|
tr|E7QXB8|
sp|P60529|
sp|P01965|
sp|P06635|
sp|P69907|
sp|P01959|
tr|D2S1E6|
tr|E1K4V0|


work1-nj-tree
tr|E7QXB8|
sp|P01959|
sp|P60529|
sp|P69907|
sp|P06635|
sp|P01965|
tr|D2S1E6|
tr|D2S0D9|
tr|E7QQB9|
tr|E1K4V0|


work1constree
sp|P01959|
sp|P69907|
sp|P06635|
sp|P60529|
sp|P01965|
tr|E7QXB8|
tr|E7QQB9|
tr|D2S1E6|
tr|E1K4V0|
tr|D2S0D9|


work1-nj-constree
tr|D2S0D9|
tr|D2S1E6|
tr|E1K4V0|
tr|E7QQB9|
sp|P01965|
sp|P60529|
sp|P69907|
sp|P06635|
sp|P01959|
tr|E7QXB8|

Q2) Obtain the weight matrix for the following sequences

MVLSPADKTNVKGKVGAHAGEYGAAAW
MKRLPADPPCVKGKVKAKAGDYGATTW
MALSAADKTNVKSKVGGHAGEYGAATS
MVLSAADKTNVKSKAGGNAGEWWAAAW
MVLSAADKTNVKSKVLANAGEFGAAAW
ALLPIRTTYHKKCASGHIPEEKDLNNV
DEASSLKGHHIKKLEADALLIPLSASS

Code:

```
#BT3040| Assignemnt 10 | Q2 | Atharva Mandar Phatak | BE21B009 |

import math

def sequence_to_matrix(input_sequence, sequence_length):
    amino_acids = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N',
'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
    matrix = []

    for i in range(0, len(input_sequence), sequence_length):
        subsequence = input_sequence[i:i+sequence_length]
        row = [0] * len(subsequence)
        for j, aa in enumerate(subsequence):
            row[j] = aa
        matrix.append(row)

    return matrix

input_sequence =
'MVLSPADKTNVKGKVGAHAGEYGAAAWMKRLPADPPCVKGKVKAKAGDYGATTWMALSAADKTNVKSKVGGHAGEYG
AATSMVLSAADKTNVKSKAGGNAGEWWAAAWMVLSAADKTNVKSKVLANAGEFGAAAWALLPIRTTYHKKCASGHIPE
EKDLNNVDEASSLKGHHIKKLEADALLIPLSASS'
sequence_length = 27
amino_acid_matrix = sequence_to_matrix(input_sequence, sequence_length)


def count_amino_acids(matrix):
    amino_acids = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L', 'M', 'N',
'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
    amino_acid_counts = [[0] * len(matrix[0]) for _ in
range(len(amino_acids))]
```

```python
    for row in matrix:
        for i, amino_acid in enumerate(row):
            amino_acid_counts[amino_acids.index(amino_acid)][i] += 1

    return amino_acid_counts

def calculate_weight_matrix(amino_acid_counts):
    num_rows = len(amino_acid_counts)
    num_cols = len(amino_acid_counts[0])

    weight_matrix = [[0] * num_cols for _ in range(num_rows)]

    for i in range(num_rows):
        for j in range(num_cols):
            n = amino_acid_counts[i][j]
            weight_matrix[i][j] = round(math.log((n + 1/20) / (8/20)),2)

    return weight_matrix


# Count amino acids
amino_acid_counts = count_amino_acids(amino_acid_matrix)

# Calculate weight matrix
weight_matrix = calculate_weight_matrix(amino_acid_counts)


from tabulate import  tabulate
print("Alignment Matrix")
print(tabulate(amino_acid_counts, headers=["{}".format(i+1) for i in
range(len(amino_acid_matrix[0]))], tablefmt="grid",  numalign="centre"))
print("")
print("")
print("Weight Matrix")
print(tabulate(weight_matrix, headers=["{}".format(i+1) for i in
range(len(weight_matrix[0]))], tablefmt="grid",  numalign="centre"))
```

Output:

a) Alignment Matrix

Alignment Matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	1	0	3	5	0	0	0	0	0	0	0	1	1	1	3	1	5	0	0	0	0	5	5	3
0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	5	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	2	0	0	4	2	0	0	5	0	0	4	0	0	0
0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
0	1	0	0	0	0	1	4	0	0	1	7	1	5	0	1	0	1	0	0	0	1	0	0	0	0
0	1	5	1	0	1	0	0	0	0	0	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	2	0	0	0	0	0	0	1	1
0	0	0	1	2	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	5	1	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2

b) Weight Matrix

Weight Matrix

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0.97	0.97	0.97	-2.08	2.03	2.54	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	0.97	0.97	2.03	0.97	2.54	-2.08	-2.08
-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08
0.97	-2.08	-2.08	-2.08	-2.08	-2.08	2.54	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	0.97
-2.08	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	0.97	2.54
-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08
-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	1.63	-2.08	-2.08	2.32	1.63	-2.08	-2.08	2.54	-2.08
-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	1.63	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	1.63	-2.08	-2.08	-2.08
-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08	0.97
-2.08	0.97	-2.08	-2.08	-2.08	-2.08	0.97	2.32	-2.08	-2.08	0.97	2.87	0.97	2.54	-2.08	0.97	-2.08	0.97	-2.08	-2.08	-2.08
-2.08	0.97	2.54	0.97	-2.08	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	0.97	-2.08	-2.08	0.97	0.97	-2.08
2.54	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08
-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	2.32	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	1.63	-2.08	-2.08	-2.08
-2.08	-2.08	-2.08	0.97	1.63	-2.08	-2.08	0.97	0.97	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	-2.08	0.97	-2.08	-2.08

Complete output in 'Q2_Output.txt'

BT 3040: BIOINFORMATICS

Assignment 11

Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

Q1) Obtain the consensus phylogenetic tree for the following two sets of sequences:

```
#Q1|Assignment 11 | BT3040 | Atharva Mandar Phatak | BE21B009
import numpy as np
import pandas as pd
import pprint as pprint

amino_acids = [
    "A",  # Alanine
    "R",  # Arginine
    "N",  # Asparagine
    "D",  # Aspartic Acid
    "C",  # Cysteine
    "Q",  # Glutamine
    "E",  # Glutamic Acid
    "G",  # Glycine
    "H",  # Histidine
    "I",  # Isoleucine
    "L",  # Leucine
    "K",  # Lysine
    "M",  # Methionine
    "F",  # Phenylalanine
    "P",  # Proline
    "S",  # Serine
    "T",  # Threonine
    "W",  # Tryptophan
    "Y",  # Tyrosine
    "V"   # Valine
]

prop_dict = {
    "A": 0,  # Alanine
    "R": 0,  # Arginine
    "N": 0,  # Asparagine
    "D": 0,  # Aspartic Acid
    "C": 0,  # Cysteine
    "Q": 0,  # Glutamine
    "E": 0,  # Glutamic Acid
    "G": 0,  # Glycine
    "H": 0,  # Histidine
    "I": 0,  # Isoleucine
    "L": 0,  # Leucine
    "K": 0,  # Lysine
    "M": 0,  # Methionine
    "F": 0,  # Phenylalanine
```

```
    "P": 0,  # Proline
    "S": 0,  # Serine
    "T": 0,  # Threonine
    "W": 0,  # Tryptophan
    "Y": 0,  # Tyrosine
    "V": 0   # Valine
}


def calculate_propensity(residues, structure):
    # Create a DataFrame to store residue counts
    df = pd.DataFrame({'Residue': list(residues), 'Structure': list(structure)})

#N
    N = len(df['Residue'])




    alpha_conformation_df = df[df['Structure'] == 'H']

#N_alpha
    n_alpha= len(alpha_conformation_df['Residue'])


#n_alpha_i
    n_alpha_i =alpha_conformation_df['Residue'].value_counts()

#N_alpha_i
    N_alpha_i =df['Residue'].value_counts()

    for i in amino_acids:
        if(i in n_alpha_i):
            percent_residue=n_alpha_i[i]/N_alpha_i[i]
            percent_all = n_alpha/N

            prop_of_i = percent_residue/percent_all

            prop_dict[i]=prop_of_i
        else: continue




propensities = calculate_propensity(seq1_prim, seq1_sec)

sorted_prop_dict = {k: v for k, v in sorted(prop_dict.items())}

pprint.pprint(sorted_prop_dict)
```

Output :

```
{'A': 1.5510204081632653,
 'C': 1.0204081632653061,
 'D': 0,
 'E': 0.40816326530612246,
 'F': 1.0204081632653061,
 'G': 1.0612244897959184,
 'H': 0.8746355685131195,
 'I': 1.2004801920768309,
 'K': 0,
 'L': 1.2244897959183674,
 'M': 1.530612244897959,
 'N': 1.3605442176870748,
 'P': 0.22675736961451246,
 'Q': 0,
 'R': 0.6802721088435374,
 'S': 1.3605442176870748,
 'T': 0.7653061224489796,
 'V': 0.5830903790087464,
 'W': 1.1131725417439702,
 'Y': 0.5830903790087464}
```

Q2) Find the propensity of alpha helices manually for the sequence in question 1

Total number of AA in the given sequence (nH): **200**

Total number of AA which have helical confirmation in the given sequence (N): **98**

Denominator value of Propensity formula = 98/200 = **0.49**

Residue	n_alpha_1	N_alpha_i	Percent_Residue	Percent_All_Residue	Ratio (Propensity)
A	19.00	25.00	0.76	0.49	1.55
C	2.00	4.00	0.50	0.49	1.02
D	0.00	0.00	0.00	0.49	0.00
E	1.00	5.00	0.20	0.49	0.41
F	10.00	20.00	0.50	0.49	1.02
G	13.00	25.00	0.52	0.49	1.06
H	3.00	7.00	0.43	0.49	0.87
I	10.00	17.00	0.59	0.49	1.20
K	0.00	0.00	0.00	0.49	0.00
L	12.00	20.00	0.60	0.49	1.22
M	3.00	4.00	0.75	0.49	1.53
N	2.00	3.00	0.67	0.49	1.36
P	1.00	9.00	0.11	0.49	0.23
Q	0.00	0.00	0.00	0.49	0.00
R	3.00	9.00	0.33	0.49	0.68
S	6.00	9.00	0.67	0.49	1.36
T	3.00	8.00	0.38	0.49	0.77
V	2.00	7.00	0.29	0.49	0.58
W	6.00	11.00	0.55	0.49	1.11
Y	2.00	7.00	0.29	0.49	0.58

Q3) Using the rules for helices and strands, identify the helical and strand segments in the following sequence

KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL

```
 CHOFAS predicts protein secondary structure
 version 2.0u66 September 1998
Please cite:
 Chou and Fasman (1974) Biochem., 13:222-245
```

```
 Chou-Fasman plot of @, 129 aa;
seq1


                .         .         .         .         .         .
       KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINS
 helix     <-------->             <-------->                 <>
 sheet                        EEEEEEE        EEEEEEE      EEEEEEEE
 turns T                                   T   T    T   T T        T

                .         .         .         .         .         .
       RWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDV
 helix                        <--------------->    <---->        <>
 sheet                 EEEE                                      EEE
 turns      TT   T T            T         T    T T         T


       QAWIRGCRL
 helix
 sheet EEEE
 turns

 Residue totals: H: 47   E: 33   T: 16
        percent: H: 36.4 E: 25.6 T: 12.4
```

Search Databases with FASTA | Find Duplications | Search Status

←--→ : Represents helix

EEE.. : Represents Sheet

TTT.. : Represents Turn

Helix:

RCELAAAMKR

WVCAAKFESN

IL

LSSDITASVNCAKKIVS

MNAWVA

DV

Sheet

YSLGNWV

FNTQATN

TDYGILQI

LCNI

TDV

Turns

K

S

T

R

G

T

S

DG

G

R

S

K

D

N

N

Q4) Verify one of the helical and strand segments, manually

श्री ॥

Assignment 11 / BT 3040 / Atharva. Mandar. Phatak / B621B004

→ Q4)

Alpha Helix

KVFGRC = ~~0+0.5+0.5~~ +
= 0.5 + 1 + 1 + (−1) + 0 + 0
= 1.5 (<4)

VFGRCE = 1 + 1 + (−1) + 0 + 0 + 1
= 2 (<4)

FGRCEL = 1 + (−1) + 0 + 0 + 1 + 1
= 2

GRCELA = (−1) + 0 + 0 + 1 + 1 + 1
= 2

RCELAA = 0 + 0 + 1 + 1 + 1 + 1
= ~~4~~ 4

Extendy further

ELAA → 1.53 + 1.34 + 1.45 + 1.45 = 5.77 (>4)
LAAA → 1.34 + 1.45 + 1.45 + 1.45 = 5.69 (>4)
AAAM → 1.45 + 1.45 + 1.45 + 1.20 = 5.55 (>4)
AAMK → 1.45 + 1.45 + 1.20 + 1.07 = 5.17 (>4)
AMKR → 1.45 + 1.20 + 1.07 + 0.77 = 4.51 (>4)
MKRH → 1.20 + 1.07 + 0.79 + 1.24 = 4.3 (>4)
KRHG → 1.07 + 0.79 + 1.24 + 0.53 = 3.63 (<4)

∴ The potential α-helix segment can be

RCELAAAMKRHG

for β-strand

KVFGR = −1+1+1+0+0 = 1 (<3)
VFGRC = 1+1+0+0+1 = 3 (=3)
condition satisfied

Extending Segment

GRC = 0.81 + 0.9 + 1.3 = 3.01 (>3)
RCG = 0.9 + 1.3 + 0.26 = 2.46 (<3)

∴ β-strand can be "VFGRC"

Practical 11

Questions

15 April 2024

1. Find the propensity of alpha helices using the following sequence and secondary structure assignments.

```
Sequence: LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQY
Sec. str: XHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXX

Sequence: GMGIPPLHDGGWWLMAGLFMTLSLGSWWIRVYSRARALGLGTHIAWNFAA
Sec. str: XXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHHHHHXXHHHHHHHH

Sequence: AIFFVLCIGCIHPTLVGSWSEGVPFGIWPHIDWLTAFSIRYGNFYYCPWH
Sec. str: HHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHH

Sequence: GFSIGFAYGCGLLFAAHGATILAVARFGGDREIEQITDRGTAVERAALFW
Sec. str: HHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

2. Find the propensity of alpha helices manually for the sequence in question 1.

3. Using the rules for helices and strands, identify the helical and strand segments in the following sequence

```
KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGRTPGSRNLCNIPC
SALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL
```

Hint: **Helix**: Assign 1 for H_α and h_α; 0.5 for I_α; 0 for i_α; -1 for B_α and b_α; identify 6-residue segments with score more than or equal to 4; extend it until the actual value (Table 5.2) for last four residues is less than 4. Continue the search.

Strand:

Assign 1 for H_β and h_β; 0.5 for I_β; 0 for i_β; -1 for B_β and b_β; identify 5-residue segments with score more than or equal to 3; extend it until the actual value for the last three residues is less than 3.

For conflicting situation: compare the values and assign the secondary structure based on the highest value

Table 5.2 Chou–Fasman parameters

Residue	P_α	Residue	P_β
Glu	Hα 1.53	Hβ Met	1.67
Ala	1.45	Val	1.65
Leu	1.34	Ile	1.60
His	hα 1.24	hβ Cys	1.30
Met	1.20	Tyr	1.29
Gln	1.17	Phe	1.28
Trp	1.14	Gln	1.23
Val	1.14	Leu	1.22
Phe	1.12	Thr	1.20
Lys	Iα 1.07	Trp	1.19
Ile	1.00	Iβ Ala	0.97
Asp	iα 0.98	iβ Arg	0.90
Thr	0.82	Gly	0.81
Ser	0.79	Asp	0.80
Arg	0.79	bβ Lys	0.74
Cys	0.77	Ser	0.72
Asn	bα 0.73	His	0.71
Tyr	0.61	Asn	0.65
Pro	Bα 0.59	Pro	0.62
Gly	0.53	Bβ Glu	0.26

4. Verify one of the helical and strand segments, manually.

Deadline: 21 April 2024

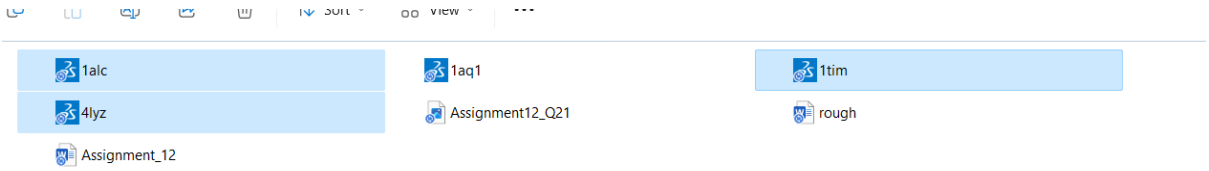
BT 3040: BIOINFORMATICS

Assignment 12

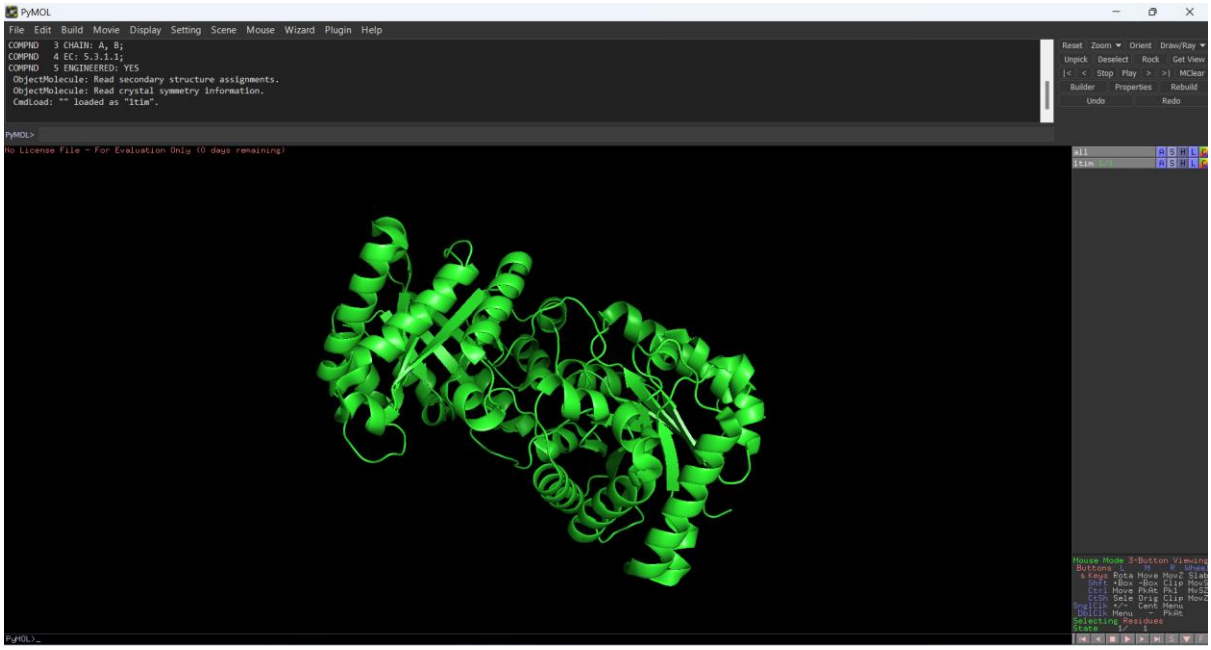
Atharva Mandar Phatak | BE21B009
Department of Biotechnology

Indian Institute of Technology
Madras

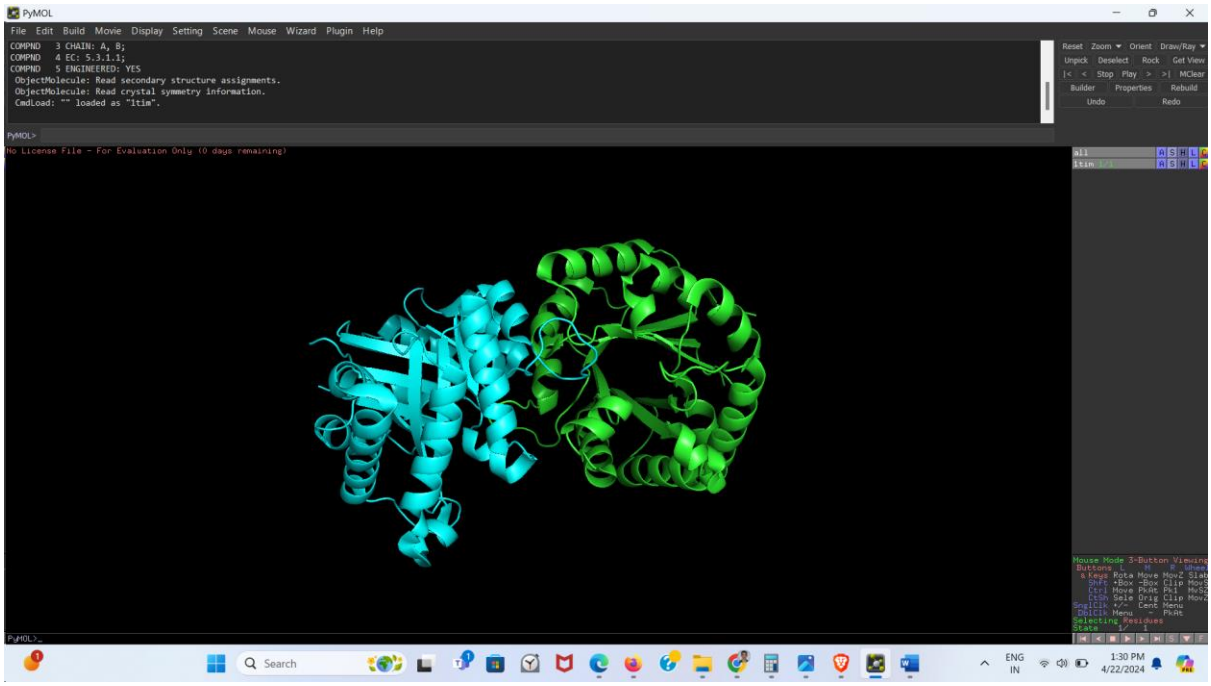
Q1) Get the PDB files, 1ALC, 4LYZ and 1TIM



Q2) Open the file 1TIM and show in cartoon style (hide line style)

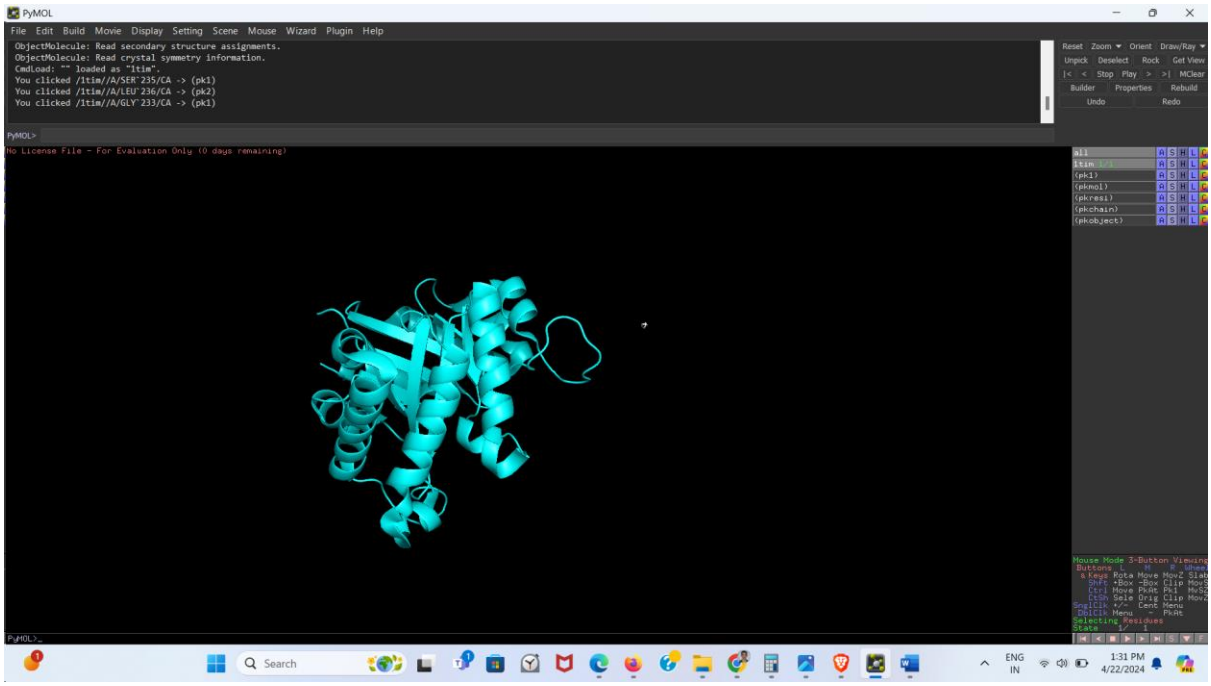


Q3) Give different colours for different chains. How many chains are there?



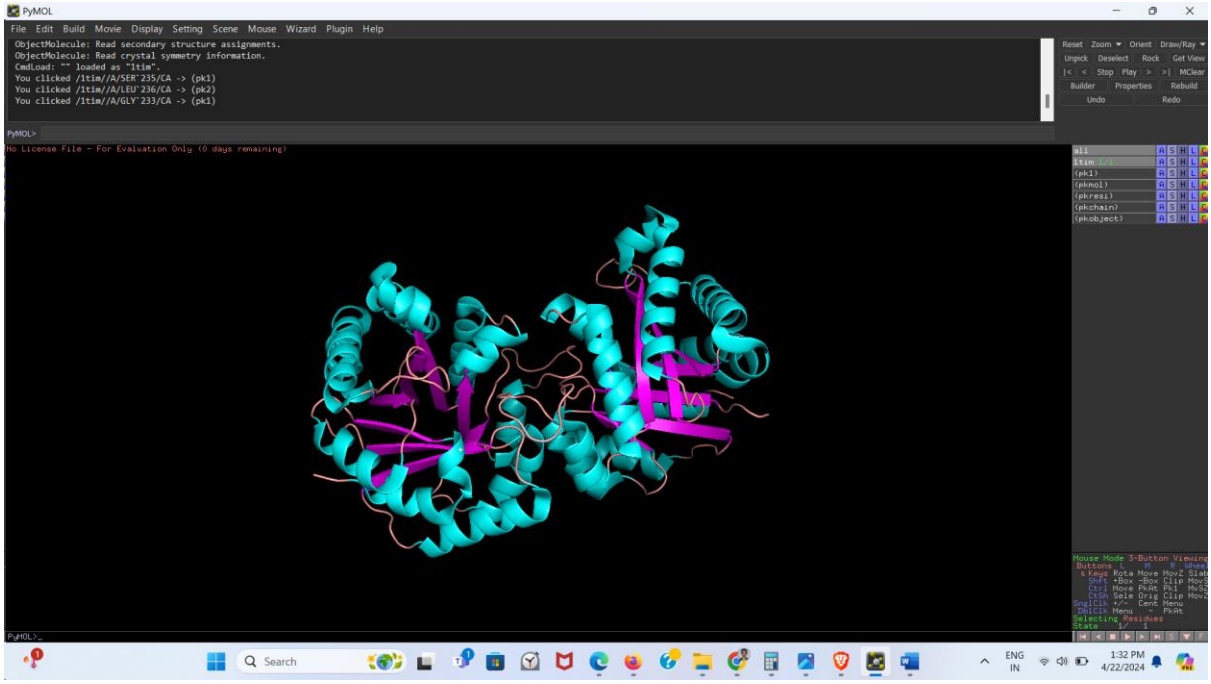
There are **two** chains

Q4) Remove one chain (right click on the chain and hide)



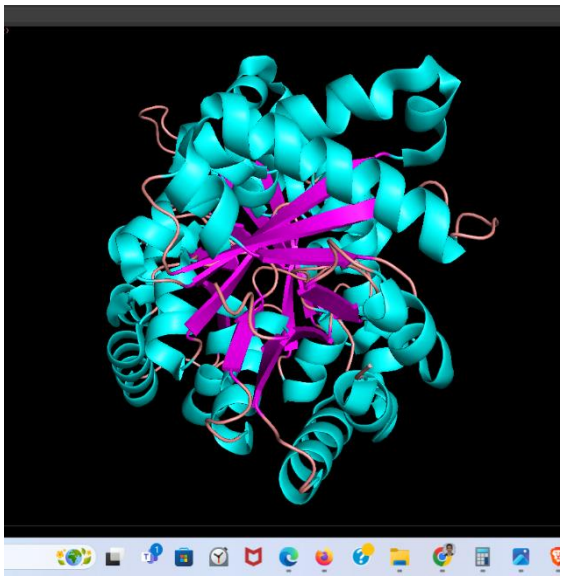
Green Coloured Chain Hidden

Q5) Identify the secondary structures with different colours

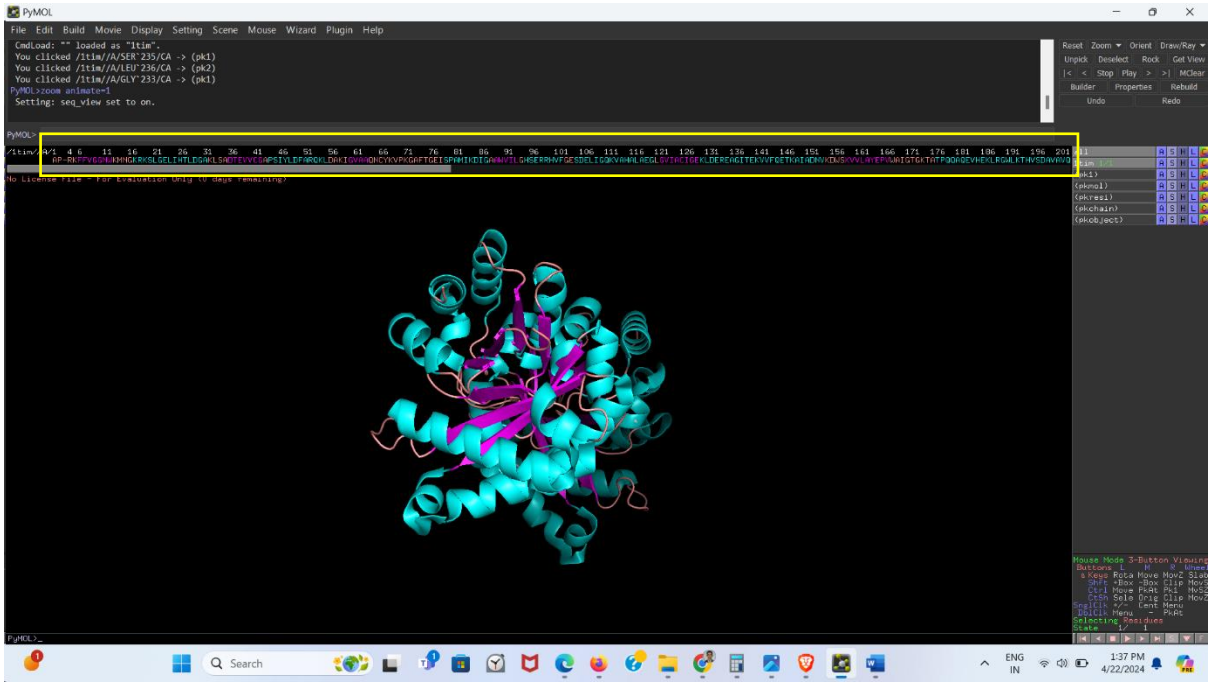


Helix have been represented by Light-Blue, Sheets by Pink and Loops by Orange

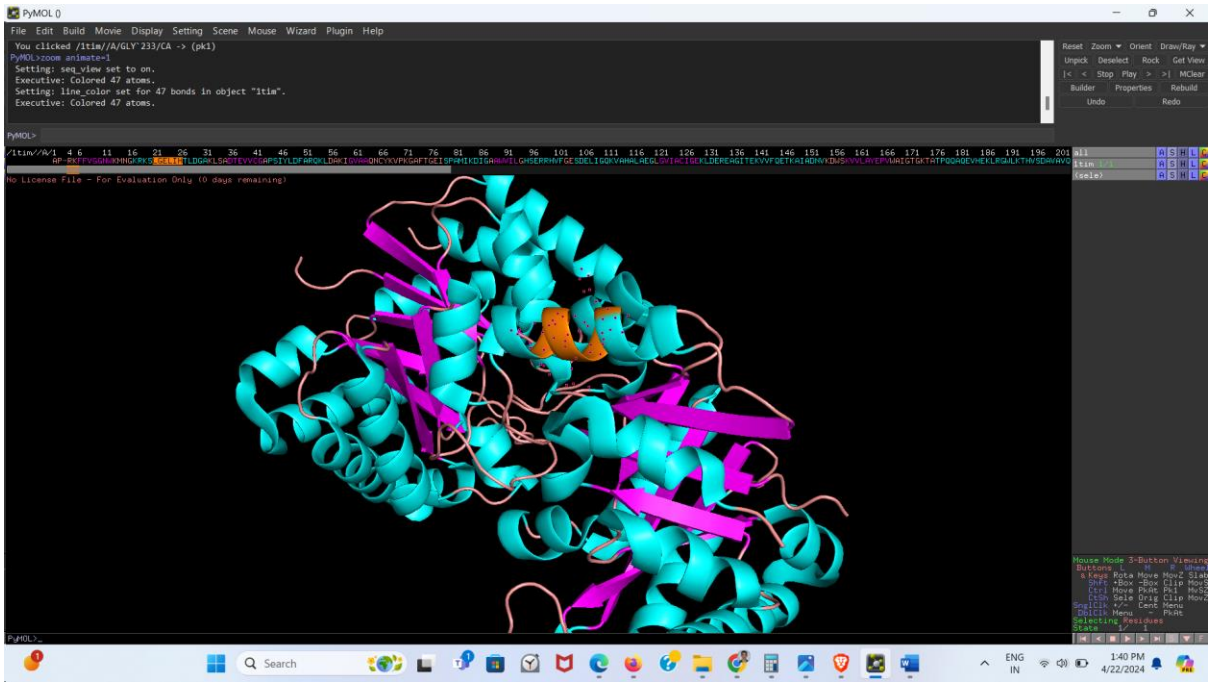
Q6) Zoom and rotate to get complete views



Q7) Show the sequence

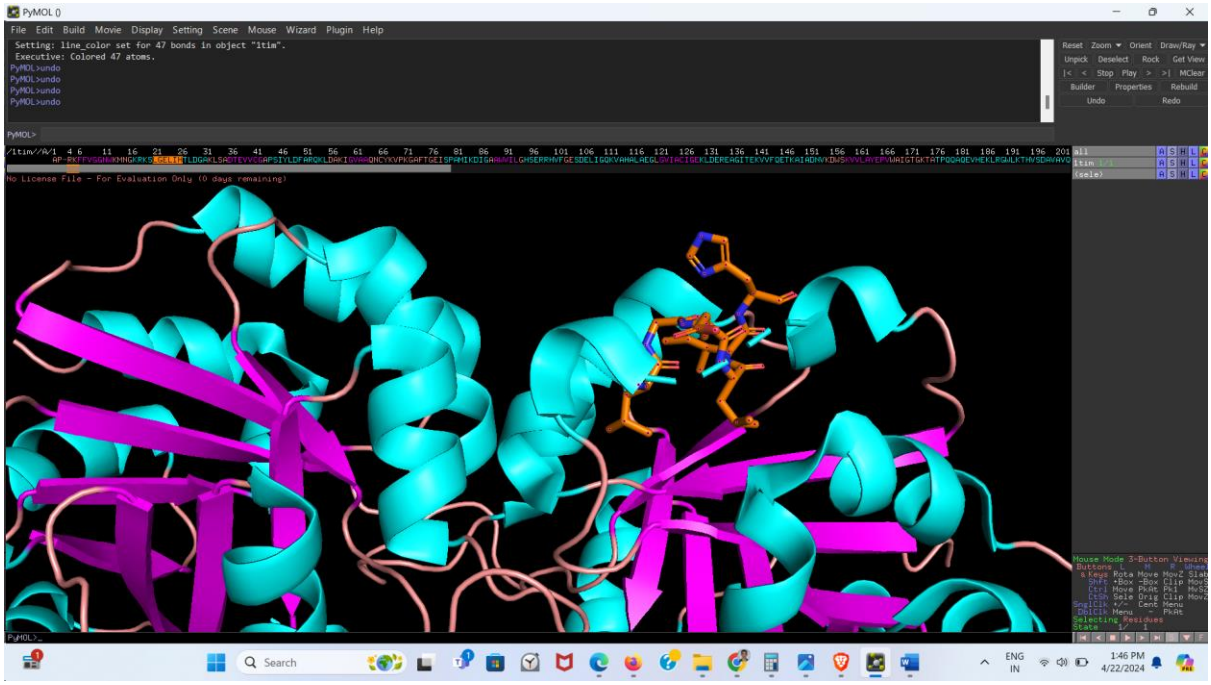


Q8) Select the residues 21 to 26: LGELIH

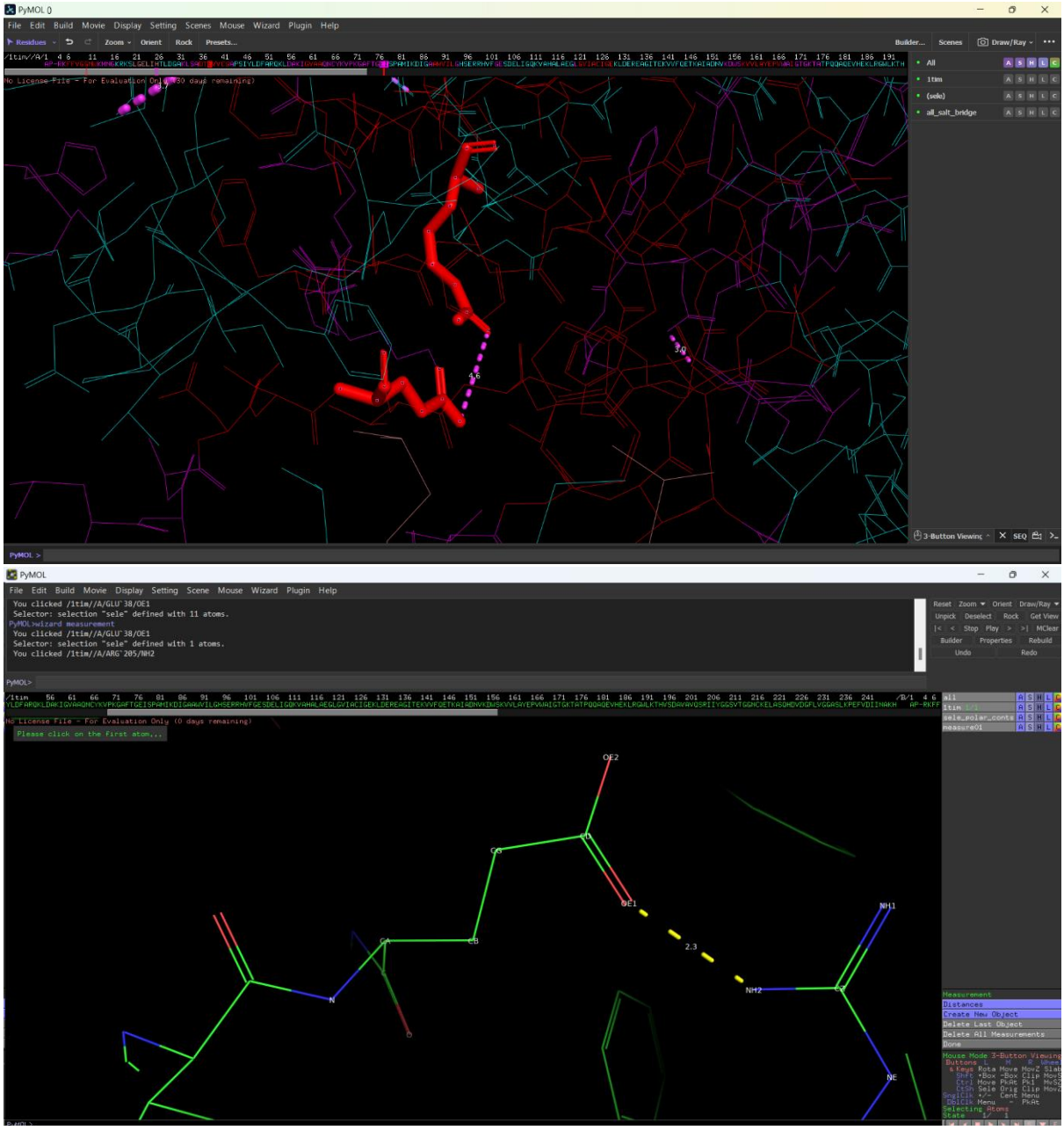


Q9) Hide cartoon diagrams

Q10) Colour the selected residues based on different atoms

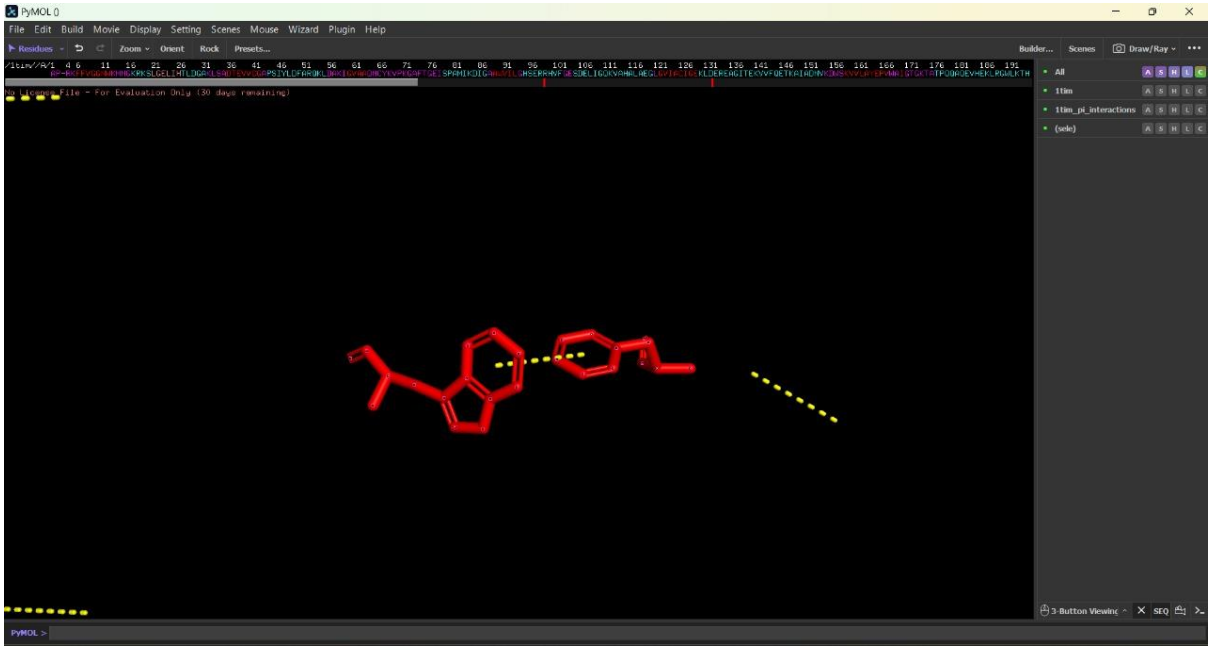


Q11) Identify at least one salt bridge within the protein and list the details of interaction.

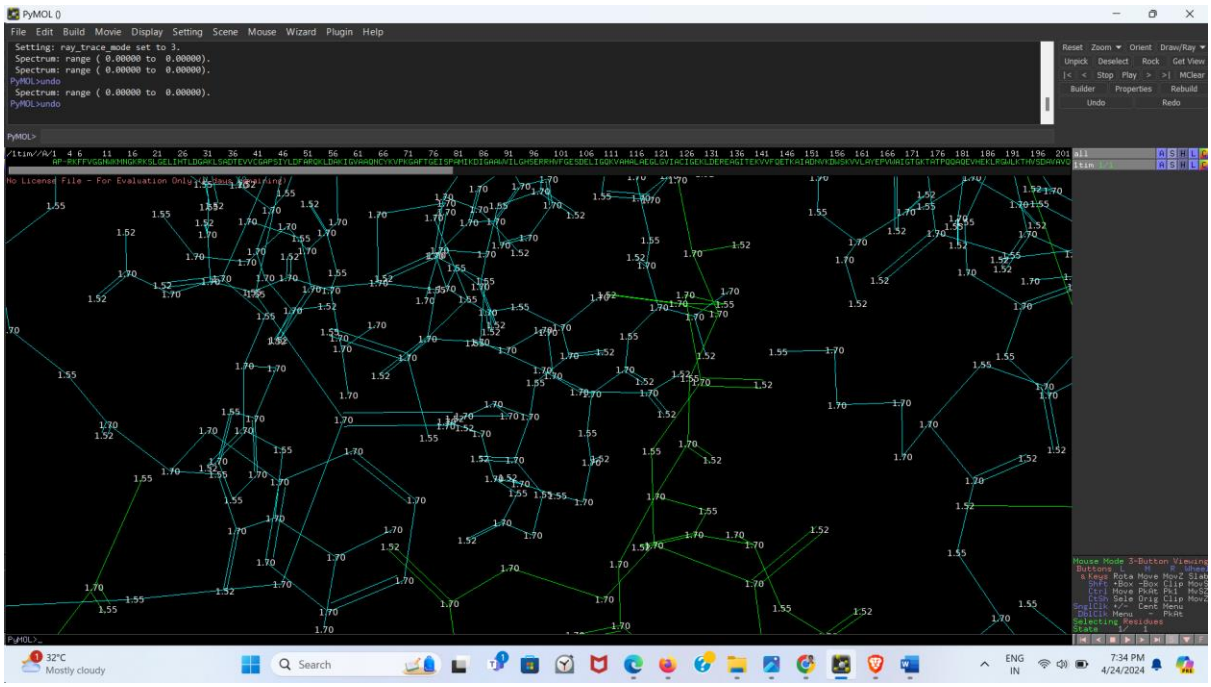


Description: Th interaction is seen between Glutamic Acid (E) at position 36 and Argnine (R) at position 205. The C=O group of Glutamic Acid forms a salt bridge with NH2 group of Arginine. The salt bridge is 2.3 A long.

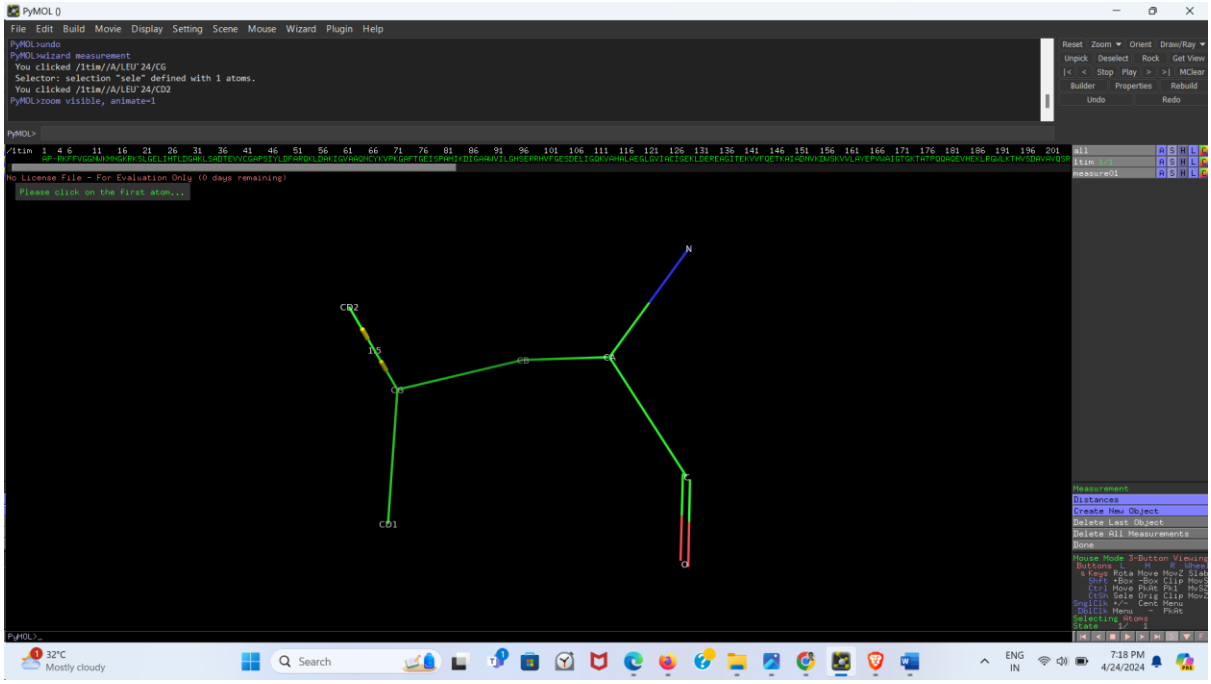
Q12) Identify any hydrophobic interaction / aromatic stacking interaction protein and list the details of interaction.



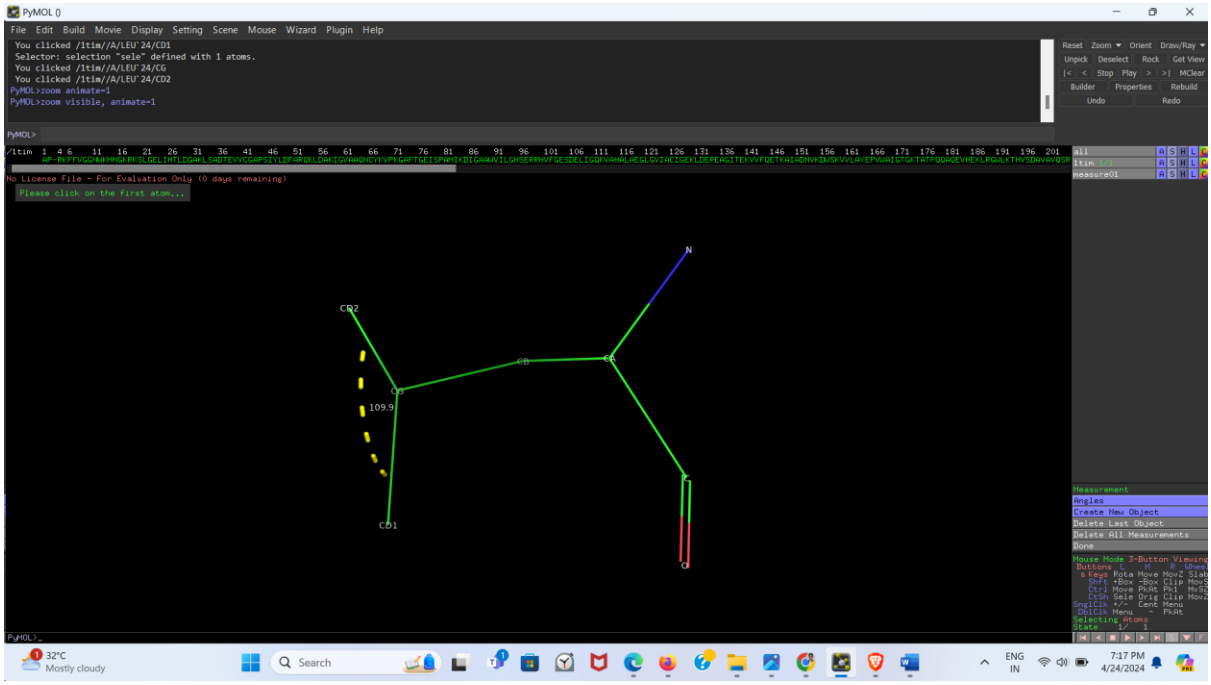
Q13) Label the residues using the atom name and show their van der Waal radii.



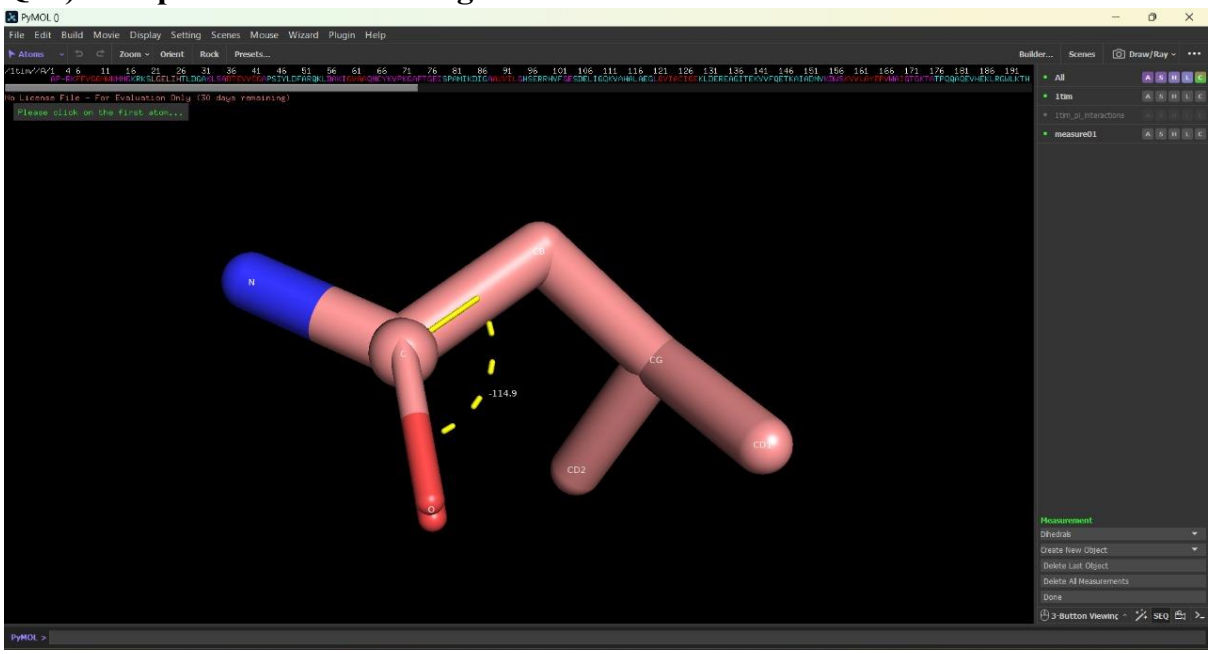
Q14) Compute the distance between CG and CD2 in Leu24



Q15) Compute the angle formed by the atoms CD1, CG and CD2 in Leu24.

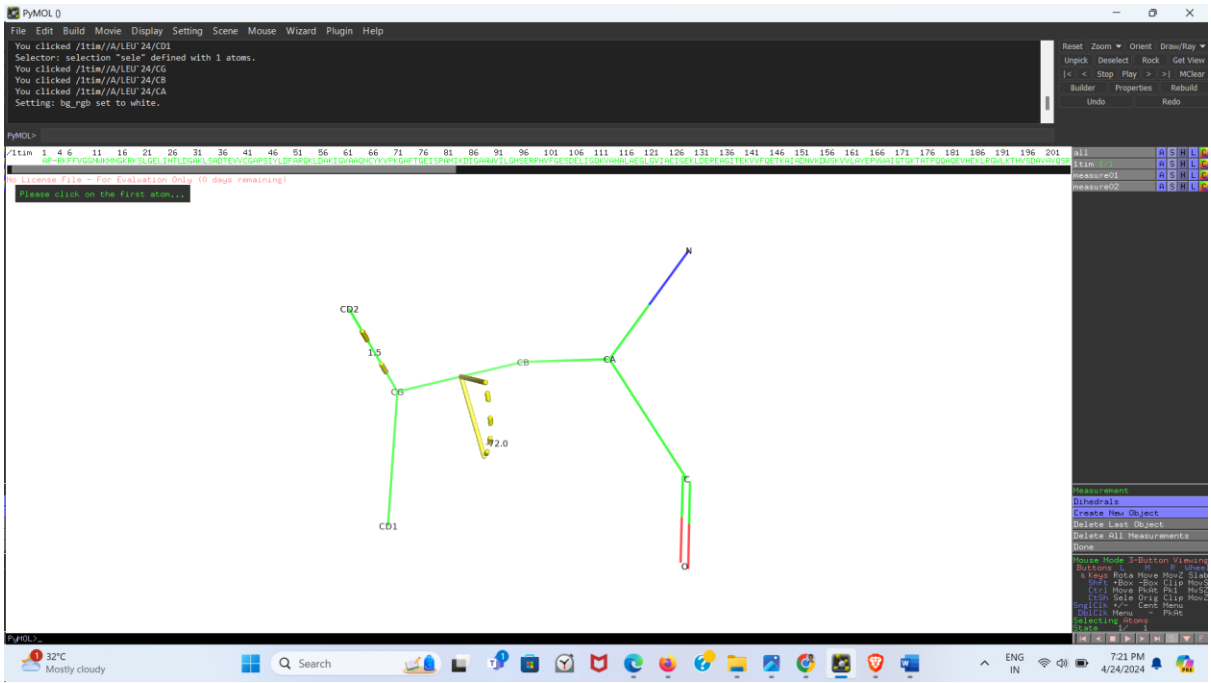


Q16) Compute the dihedral angles of Leu24

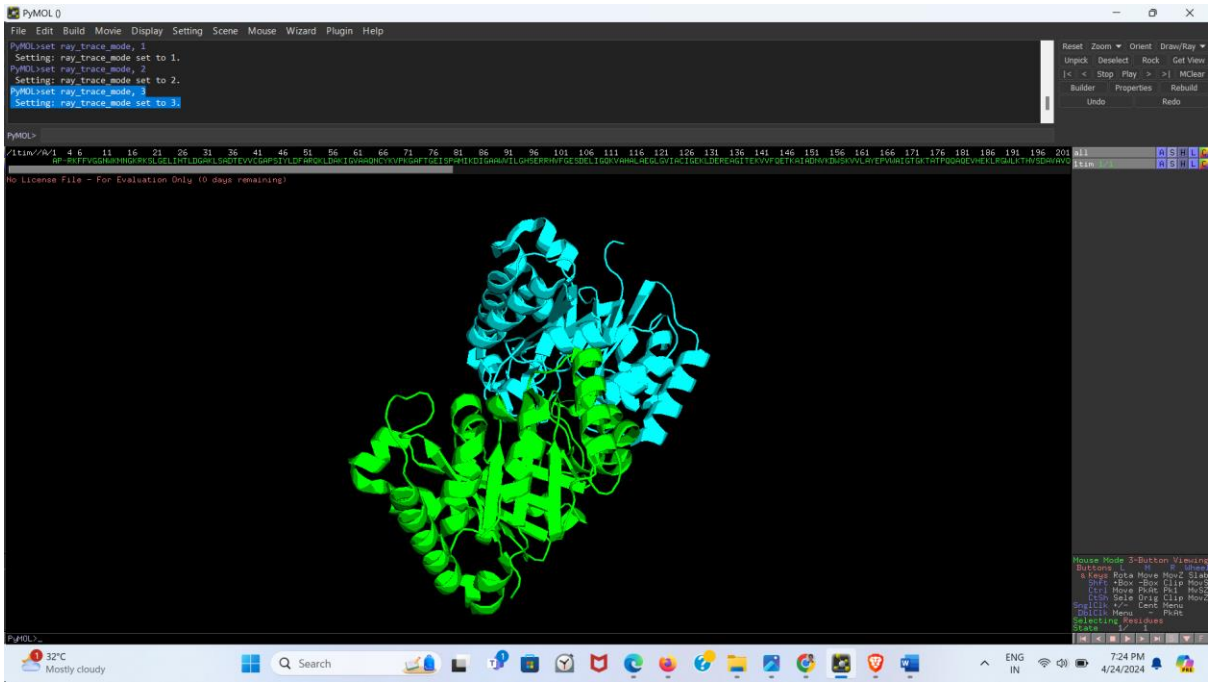


Dihedral Angle: -114.9

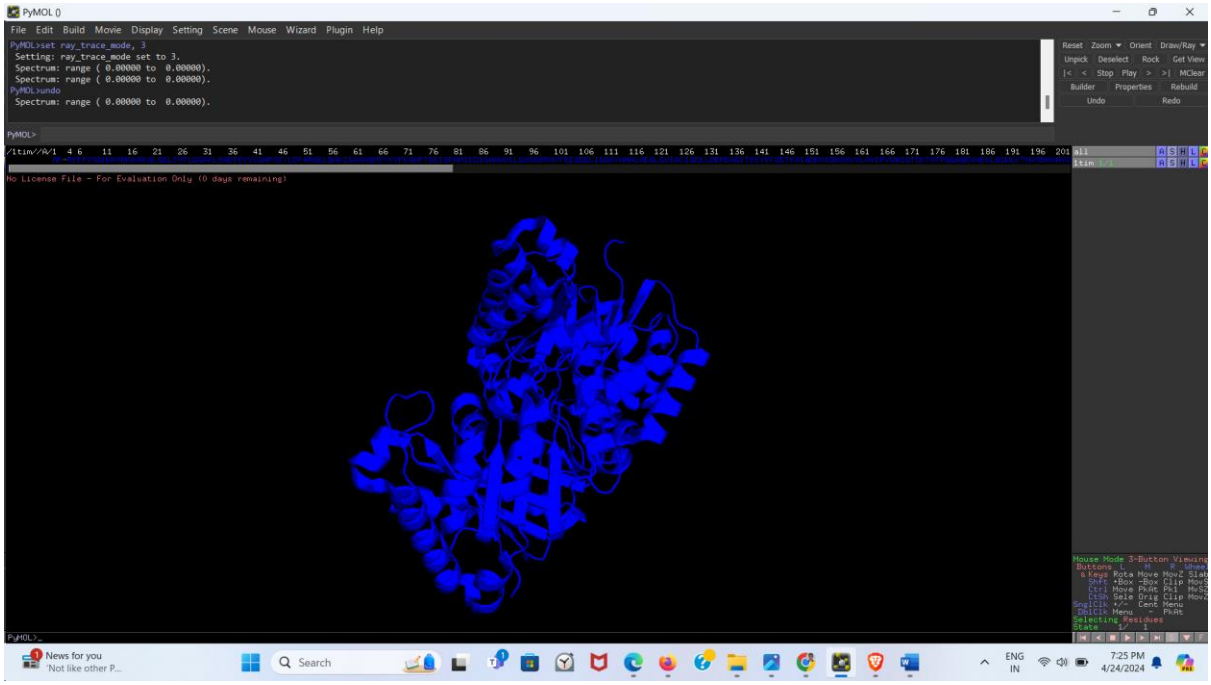
Q17) Change the background to white.



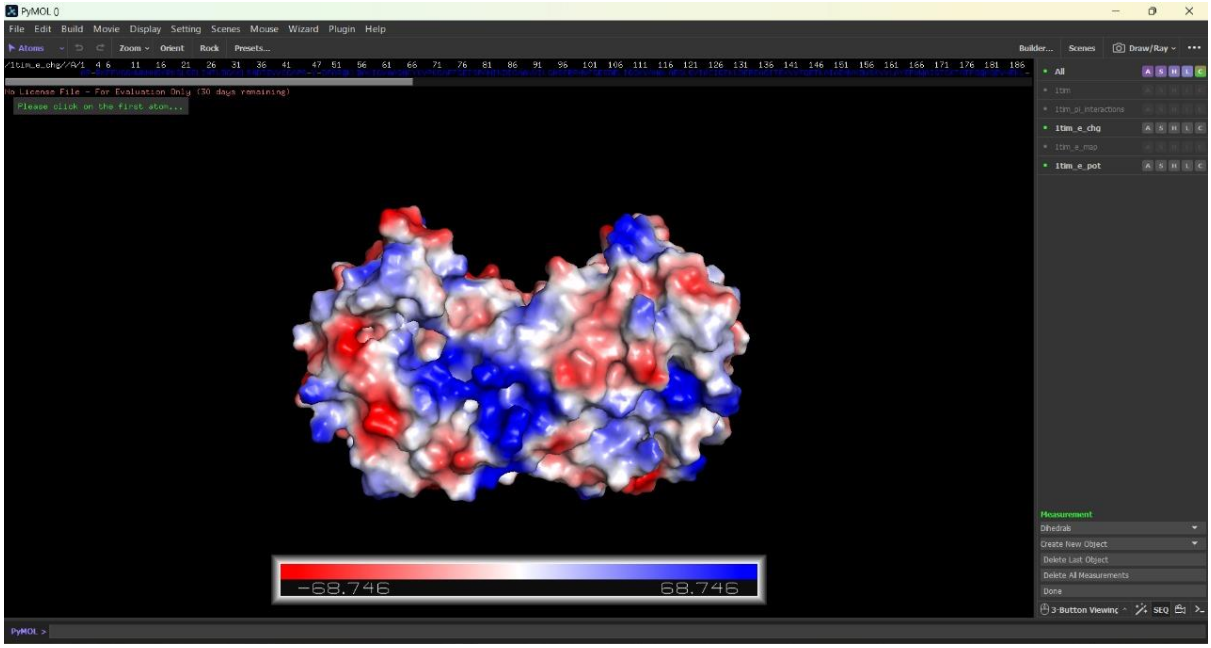
Q18) Make high quality picture (use ray)



Q19) Show B-factors to see flexible and rigid regions (color by spectrum, b-factor)



Q20) Compute the electrostatic potential (A -> generate -> vacuum statistics -> protein contact potential)



Q21) Save the image in PNG format

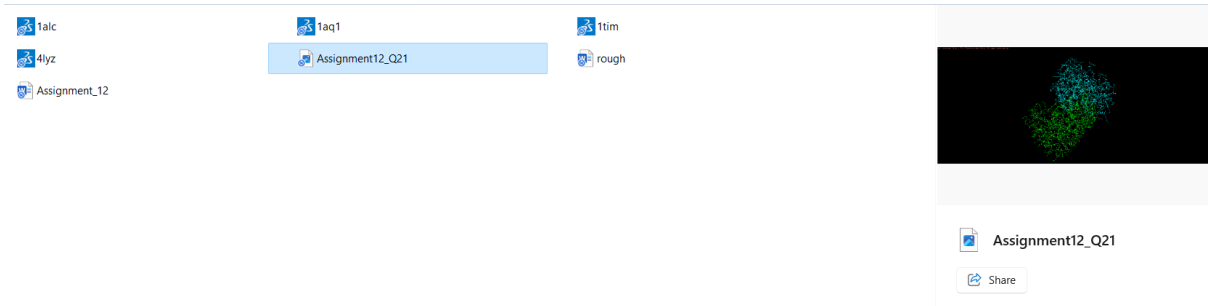
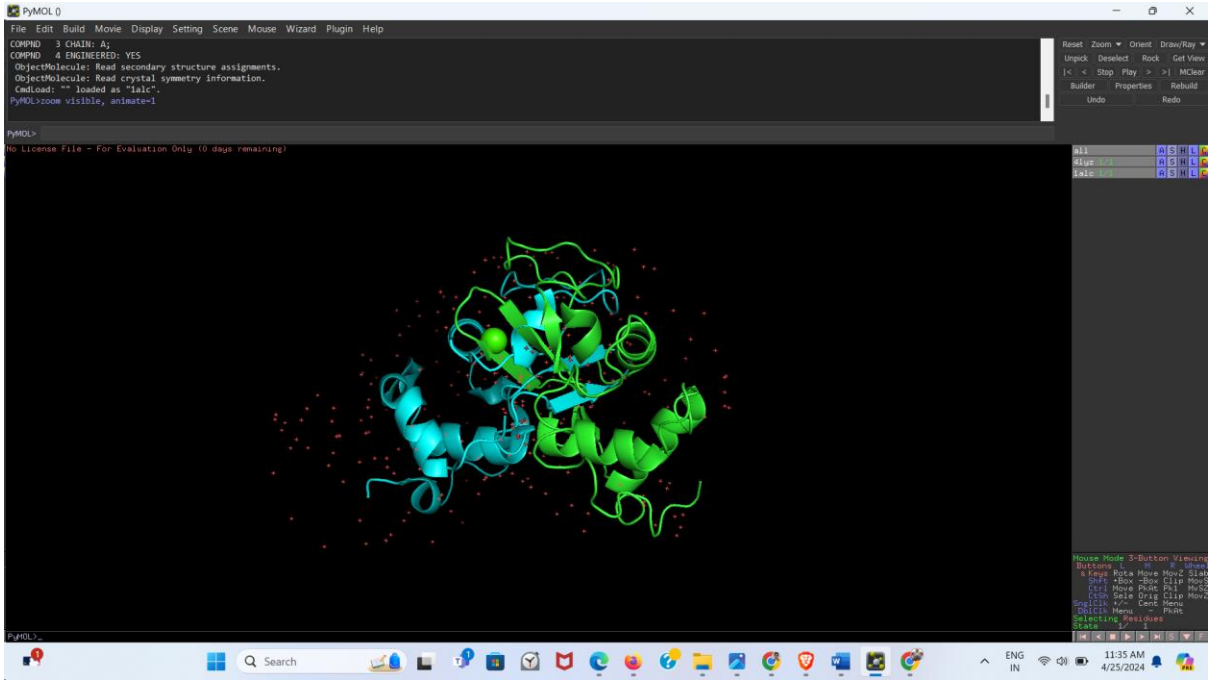
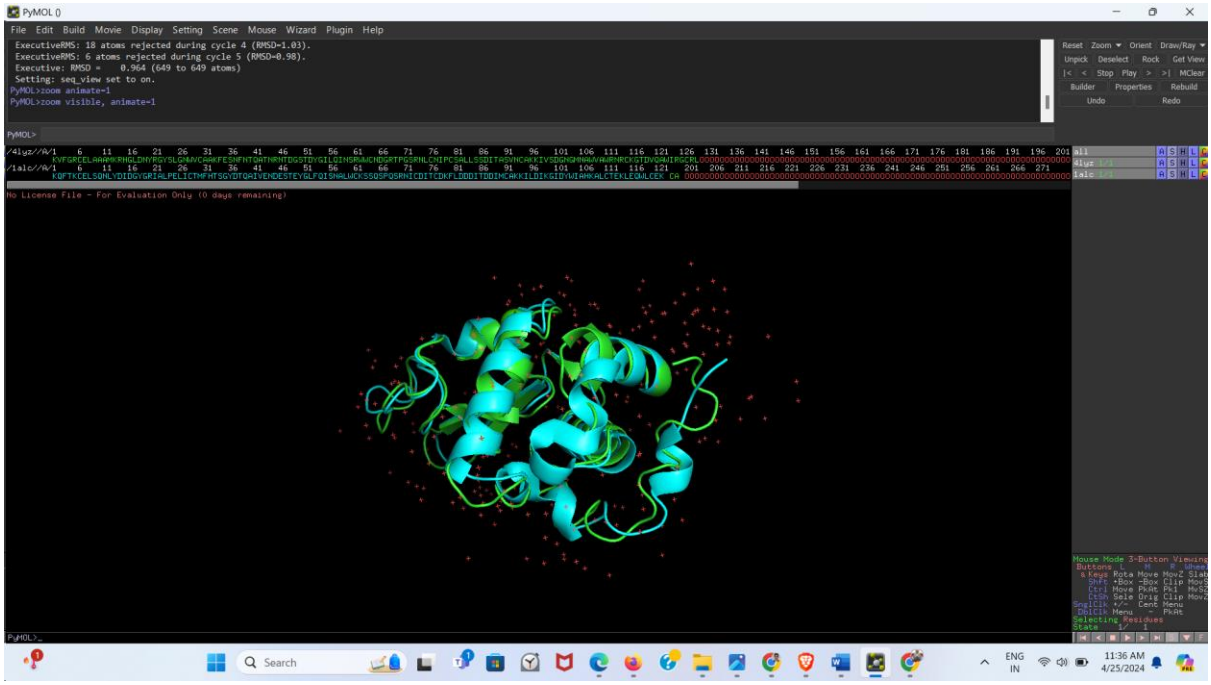


Image saved with file name 'Assignment12_Q21.png'

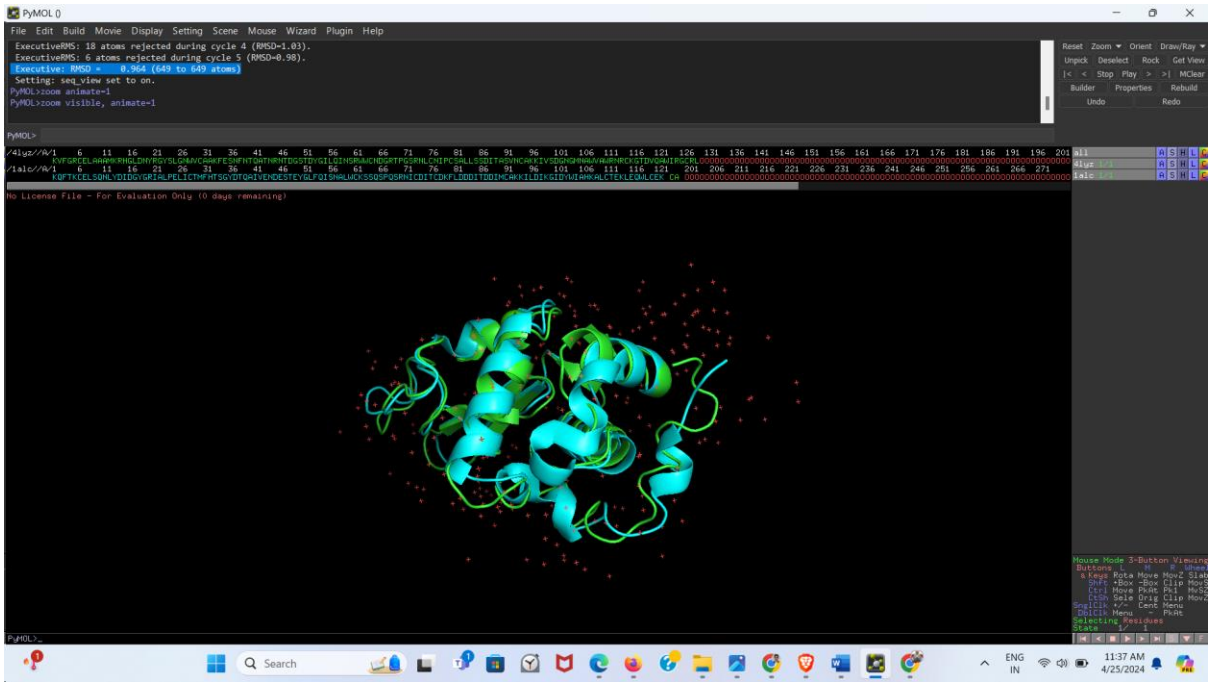
Q22) Open the files 1ALC and 4LYZ



Q23) Align the structures

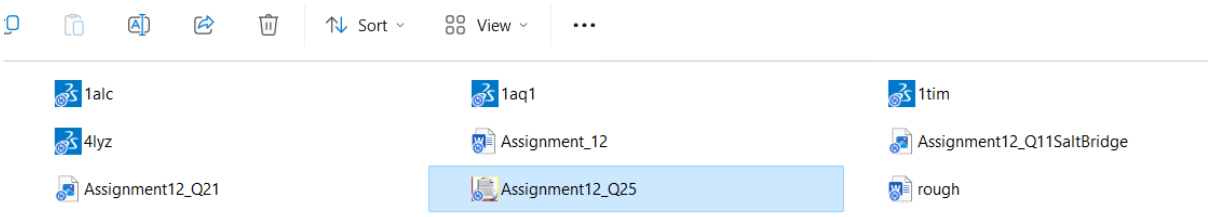


Q24) Compute RMSD.



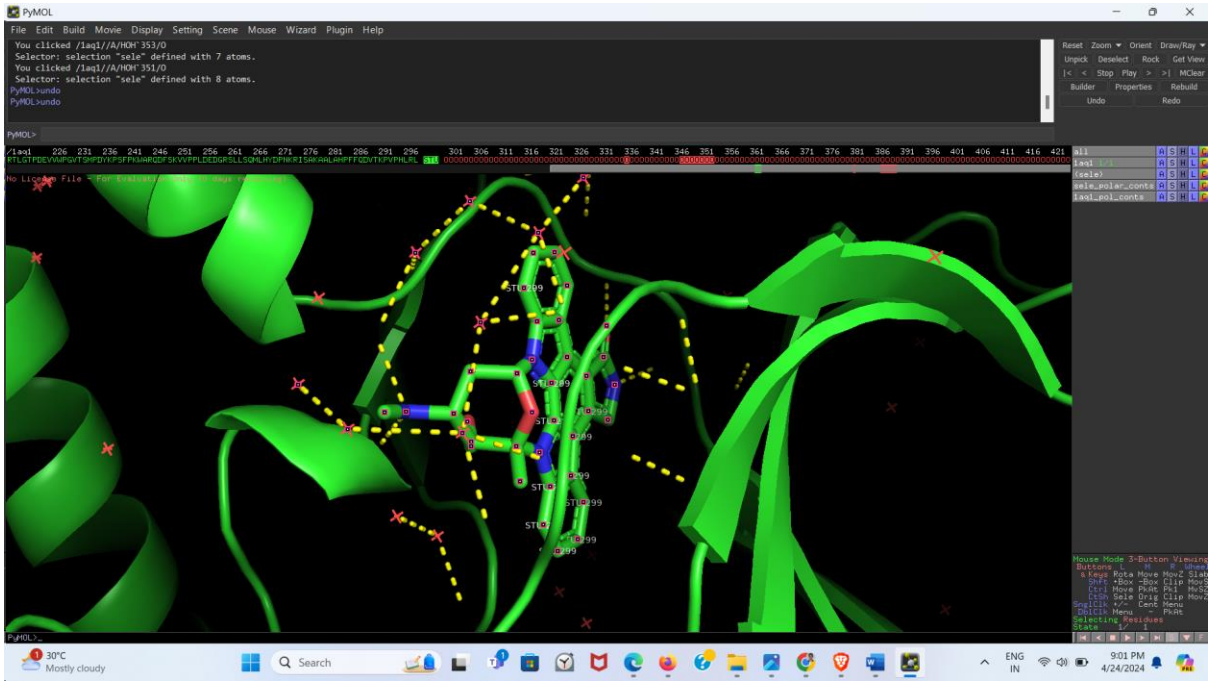
RMSD = 0.946

Q25) Save the aligned protein structure



File saved as ‘Assignment12_Q25’

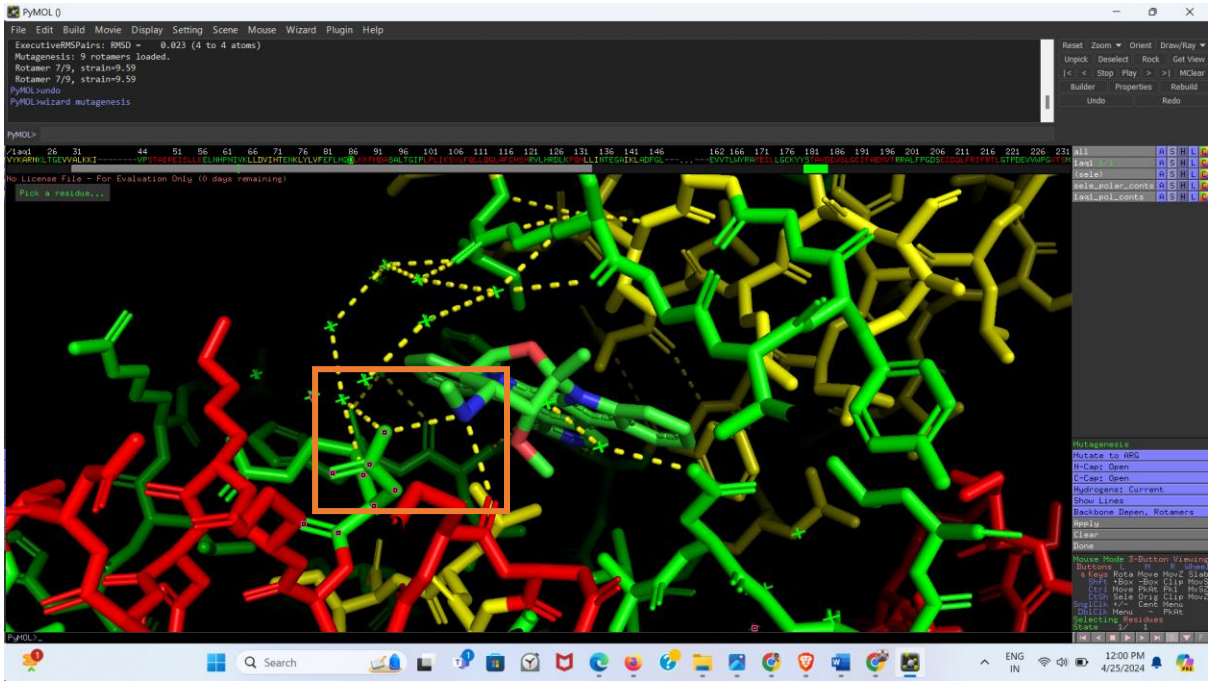
Q26) Download a protein structure with ligand(s). Identify the ligand and select residues within 3.5Å. Show any one type of interaction between them.



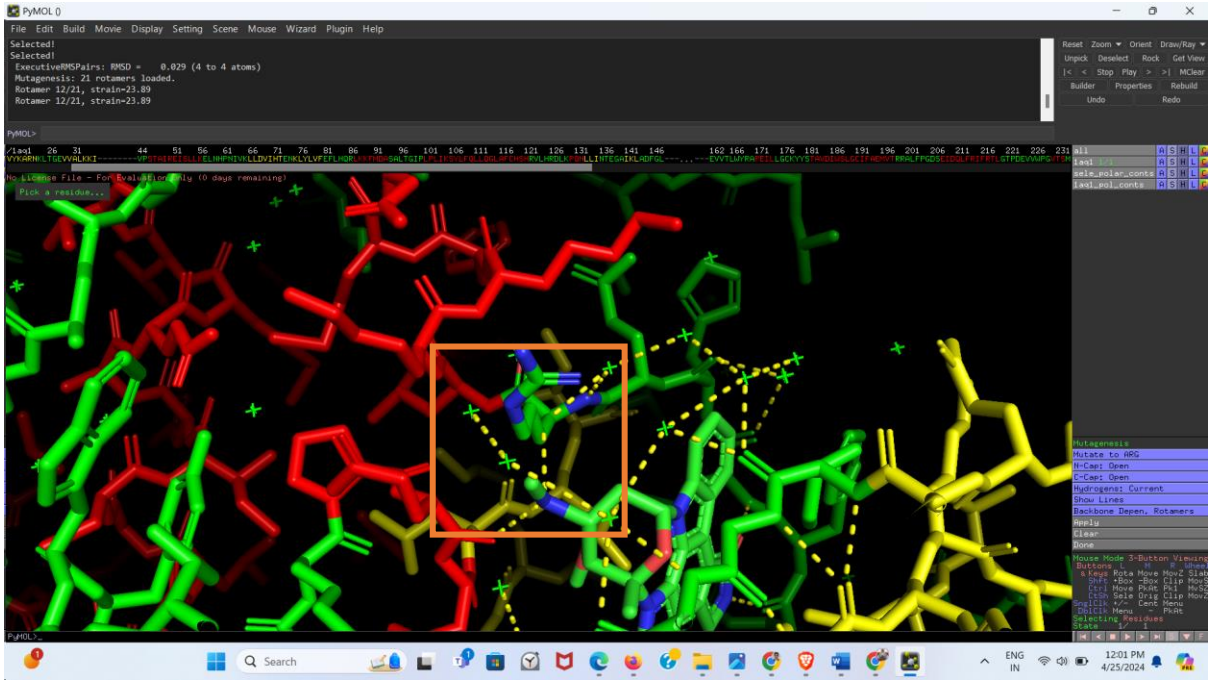
The yellow lines represent polar interactions.

Q27) Mutate any one of the active site residues and describe the change in the interaction with the ligand before and after mutation.

Before:



After:



Description: Aspartic Acid (D) was changed to Arginine (R), thus replacing a negative charge with positive charge. The interaction of protein-ligand at that residue is lost. Earlier the interaction, was between positively charged 'N' of ligand and 'O^-' of Aspartic Acid. Now there is no interaction, in fact repulsion between positively charged 'N' of both the groups

Practical 12
22 April 2024
Questions

1. Get the PDB files, 1ALC, 4LYZ and 1TIM
2. Open the file 1TIM and show in cartoon style (hide line style)
3. Give different colors for different chains. How many chains are there?
4. Remove one chain (right click on the chain and hide)
5. Identify the secondary structures with different colors
6. Zoom and rotate to get complete views
7. Show the sequence
8. Select the residues 21 to 26: LGELIH
9. Hide cartoon diagrams
10. Color the selected residues based on different atoms
11. Identify at least one salt bridge within the protein and list the details of interaction.
12. Identify any hydrophobic interaction / aromatic stacking interaction protein and list the details of interaction.
13. Label the residues using the atom name and show their van der Waal radii.
14. Compute the distance between CG and CD2 in Leu24
15. Compute the angle formed by the atoms CD1, CG and CD2 in Leu24.
16. Compute the dihedral angles of Leu24
17. Change background white
18. Make high quality picture (use ray)
19. Show B-factors to see flexible and rigid regions (color by spectrum, b-factor)
20. Compute the electrostatic potential (A -> generate -> vacuum statistics -> protein contact potential)
21. Save the image in PNG format
22. Open the files 1ALC and 4LYZ
23. Align the structures
24. Compute rmsd
25. Save the aligned protein structures
26. Download a protein structure with ligand(s). Identify the ligand and select residues within 3.5Å. Show any one type of interaction between them.
27. Mutate any one of the active site residues and describe the change in the interaction with the ligand before and after mutation.

Deadline: 24 April 2024