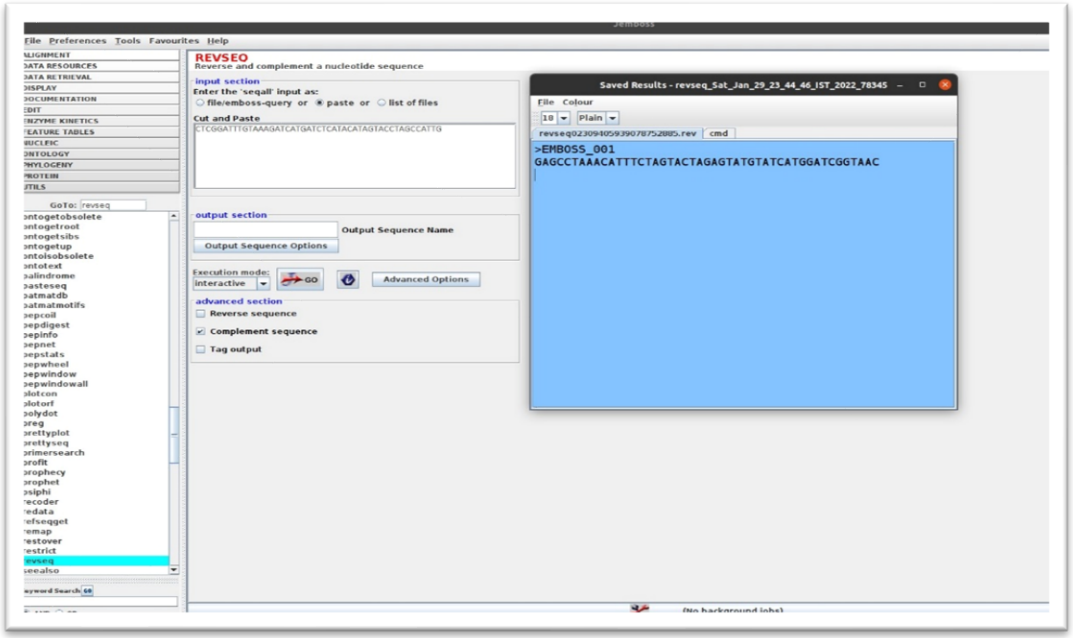
BIOINFORMATICS
PRACTICAL-1

Vineet Patel
BS19B033

1. Installed Emboss in Linux using Command line: sudo apt-get install jemboss.

2. Use REVSEQ to find complementary strand

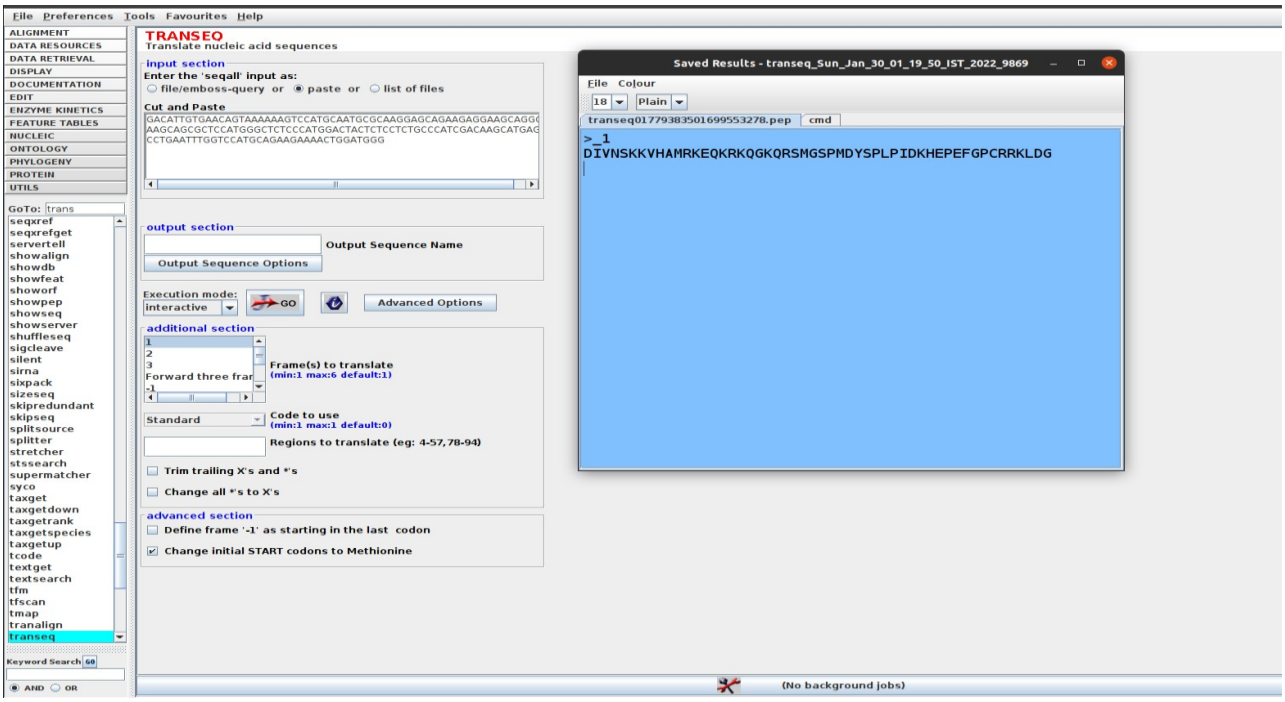


3. Screenshot of Python Code for complementary strand

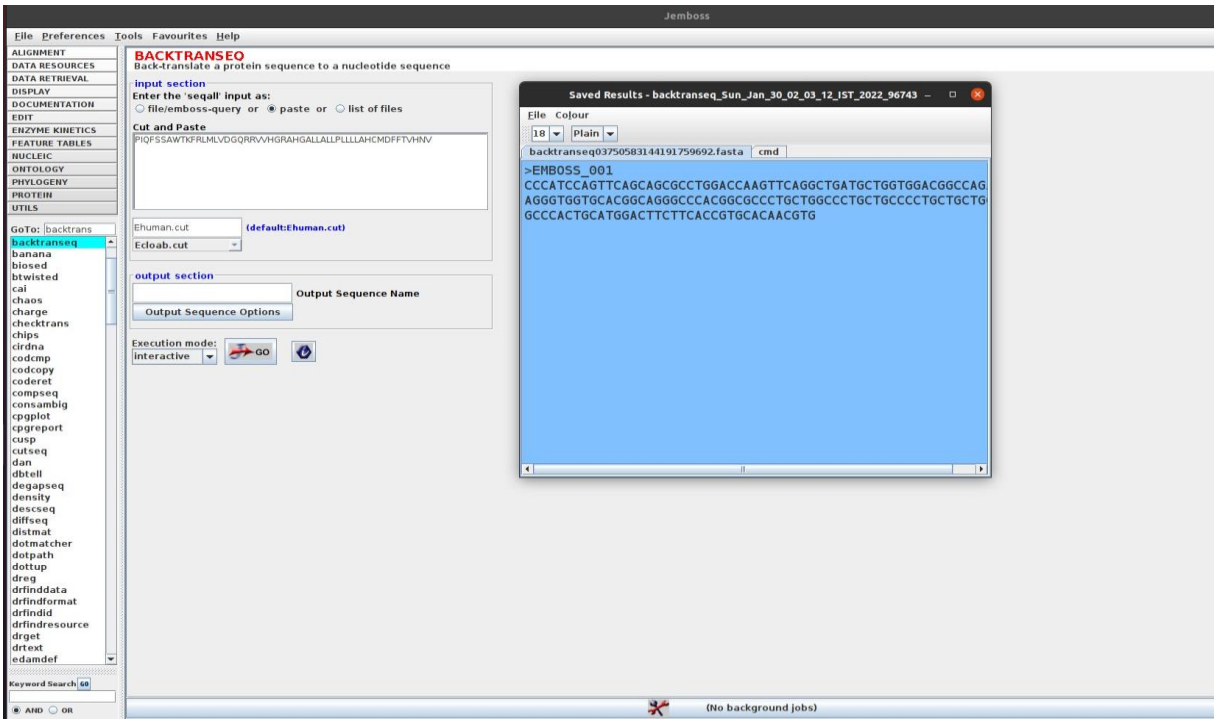
```
In [1]: s =str(input("Enter Sequence:"))
        s =list(s)
        for i in range(len(s)):
            if s[i]== 'A':
                s[i]="T"
            elif s[i] =="T":
                s[i] ="A"
            elif s[i] =="C":
                s[i]="G"
            else:
                s[i]="C"
        print("Complimentary Sequence:","".join(s))

        Enter Sequence:CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG
        Complimentary Sequence: GAGCCTAAACATTTCTAGTACTAGAGTATGTATCATGGATCGGTAAC
```

4. Protein Sequence

(i) protein sequence using Emboss



(ii) DNA sequence for given protein sequence



5. Code to find Protein sequence for given DNA Sequence

```
In [1]: s = str(input("Enter Sequence :"))
        s = list(s)
        protein_sequence =""
        d = {"UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L",
             "UCU":"S", "UCC":"s", "UCA":"S", "UCG":"S",
             "UAU":"Y", "UAC":"Y", "UAA":"", "UAG":"",
             "UGU":"C", "UGC":"C", "UGA":"STOP", "UGG":"W",
             "CUU":"L", "CUC":"L", "CUA":"L", "CUG":"L",
             "CCU":"P", "CCC":"P", "CCA":"P", "CCG":"P",
             "CAU":"H", "CAC":"H", "CAA":"Q", "CAG":"Q",
             "CGU":"R", "CGC":"R", "CGA":"R", "CGG":"R",
             "AUU":"I", "AUC":"I", "AUA":"I", "AUG":"M",
             "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T",
             "AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K",
             "AGU":"S", "AGC":"S", "AGA":"R", "AGG":"R",
             "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V",
             "GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A",
             "GAU":"D", "GAC":"D", "GAA":"E", "GAG":"E",
             "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G",
             "ATT":"I","GTG":"V",'ATA':'I', 'ATC':'I',
             'ATT':'I', 'ATG':'M','ACA':'T', 'ACC':'T',
             'AAC':'N', 'AAT':'N', 'AAA':'K', 'AAG':'K',
             'AGC':'S', 'AGT':'S', 'AGA':'R', 'AGG':'R',
             'CTA':'L', 'CTC':'L', 'CTG':'L', 'CTT':'L',
             'CCA':'P', 'CCC':'P', 'CCG':'P', 'CCT':'P',
             'CAC':'H', 'CAT':'H', 'CAA':'Q', 'CAG':'Q',
             'CGA':'R', 'CGC':'R', 'CGG':'R', 'CGT':'R',
             'GTA':'V', 'GTC':'V', 'GTG':'V', 'GTT':'V',
             'GCA':'A', 'GCC':'A', 'GCG':'A', 'GCT':'A',
             'GAC':'D', 'GAT':'D', 'GAA':'E', 'GAG':'E',
             'GGA':'G', 'GGC':'G', 'GGG':'G', 'GGT':'G',
             'TCA':'S', 'TCC':'S', 'TCG':'S', 'TCT':'S',
             'TTC':'F', 'TTT':'F', 'TTA':'L', 'TTG':'L',
             'TAC':'Y', 'TAT':'Y', 'TAA':'', 'TAG':'',
             'TGC':'C', 'TGT':'C', 'TGA':'_', 'TGG':'W',
             'ACG':'T', 'ACT':'T',}
        for i in range(0,len(s),3):
            codon = f"{s[i]}{s[i+1]}{s[i+2]}"
            k = d[codon]
            protein_sequence += d[codon]
        print("Protein Sequence:",protein_sequence)

Enter Sequence :GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTC
TGCCCATCGACAAGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG
Protein Sequence: DIVNSKKVHAMRKEQKRKQGKQRSMGSPMDYSPLPIDKHEPEFGPCRRKLDG
```

6. Code for finding the pattern. Position of match with **indexing starting from 1**

```
In [1]: 1 s = str(input("Enter the Sequence:"))
        2 pattern = str(input("Enter the Pattern:"))
        3 count =0
        4 for i in range(len(s)-len(pattern)+1):
        5     if s[i:i+len(pattern)] == pattern:
        6         count +=1
        7         print(i+1,end =" ")
        8 print()
        9 print(count,"Matches Found")

Enter the Sequence:GACATTGTGAACAGTAAAAAAGTCCATGCAATGCGCAAGGAGCAGAAGAGGAAGCAGGGCAAGCAGCGCTCCATGGGCTCTCCCATGGACTACTCTCCTCTGCCCATC
GACAAGCATGAGCCTGAATTTGGTCCATGCAGAAGAAAACTGGATGGG
Enter the Pattern:AAG
20 37 46 52 61 112 141
7 Matches Found
```

Similarly, “ACTA “was found once in given sequence at index 89.

7. Features of Emboss:

- **Dan** : used for calculating the melting temperature of a nucleic acid.

- **remap**: used to display the binding sites in each nucleotide sequence.
- **density**: used to draw density plot of a nucleic acid.

8. Code for calculation of Average Base Stacking Energy **:**

```
In [1]: s = str(input("Enter the sequence: "))
        s = list(s)
        d ={'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}
        energy=0
        for i in range(len(s)-1):
            k = f"{s[i]}{s[i+1]}"
            energy =energy +d[k]
        print("Averge Base Stacking Energy:",energy/(len(s)-1))

        Enter the sequence: CTCGGATTTGTAAAGATCATGATCTCATACATAGTACCTAGCCATTG
        Averge Base Stacking Energy: -5.804347826086956
```

9.

(i) For sequence ATATATATA :

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

Average Melting Temperature for ATATATATAT: 48.0022 degree

(ii) For sequence GCGCGCGCGC :

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 molA-1 AA-2	6.49111
Flexibility_slide	kJ molA-1 AA-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

Average Melting Temperature for GCGCGCGCGC: 107.867 degrees

GC pairs have a higher stacking energy than AT pairs because they make three hydrogen bonds in water whereas AT only forms two hydrogen bonds that's why GC pair have high melting point**.**

10. Sequence: AAATGGCCCTA

Your input seq is:

AAATGGCCCTAA

Iucleotide 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

AT Content: 58.333333 %
GC Content: 41.666667 %