Name: _Mitesh Pundalik Konkar____ **Roll no: 44**

Class: _M.Sc. Computer Science Part 1___ **Subject: Bioinformatics**

Academic Year: 2021-2022

Practical 1

Aim: Write a Python/Java code to perform pairwise alignment. Take 2 sequences from user and calculate the score.

Code:

```
se1=input("Enter the first sequence::")
se2=input("Enter the second sequence::")
seq1=list(se1)
seq2=list(se2)
score=[]

def Pairwise_alignment(a,b):
    gap(a,b)
    print(a)
    print(b)
    value=0
    length=len(a)
    for i in range(0,length):
        if(a[i]==b[i]):
            score.append('1')
            value=value+1
        else:
            score.append('0')
    print(score)
    print(value)

def gap(a,b):
    if(len(a)==len(b)):
        print()
    else:
        k=int(input("Enter the position to insert"))
        if(len(a)<len(b)):
            a.insert(k,'-')
        else:
            b.insert(k,'-')
    return(a,b)

Pairwise_alignment(seq1,seq2)
```

Output:

Enter the first sequence::abcvga
Enter the second sequence::abgvcf

['a', 'b', 'c', 'v', 'g', 'a']
['a', 'b', 'g', 'v', 'c', 'f']
['1', '1', '0', '1', '0', '0']
3

Practical 2

Aim: Write a Python/Java code to find the identity value of a given sequences. Take the sequence from user.

Code:

```
se1=input("Enter the first sequence::")
se2=input("Enter the second sequence::")
seq1=list(se1)
seq2=list(se2)

def find_identity(a,b):
    gap(a,b)
    print(a)
    print(b)
    score=0
    length=len(a)
    total_elements=len(a)*len(b)
    for i in range(0,length):
        for j in range(0,length):
            if(a[i]==b[j]):
                score=score+1
    identity=(score/total_elements)*100
    print("Matching Score::",score)
    print("Identity of the sequences::",identity)

def gap(a,b):
    if(len(a)==len(b)):
        print()
    else:
        k=int(input("Enter the position to insert gap::"))
        if(len(a)<len(b)):
            a.insert(k,'-')
        else:
            b.insert(k,'-')
    return(a,b)
find_identity(seq1,seq2)
```

Output:

Enter the first sequence::abcvfdg
Enter the second sequence::abvcgfd

['a', 'b', 'c', 'v', 'f', 'd', 'g']
['a', 'b', 'v', 'c', 'g', 'f', 'd']
Matching Score:: 7
Identity of the sequences:: 14.285714285714285

Practical 3

Aim: Write a Python/Java code to find the Similarity value of a given sequences. Take the sequence from user.

Code:

```
sequence_one=input("Enter the first sequence::")
sequence_two=input("Enter the second sequence::")
how_many=int(input("How many elements for similarity condition?"))
similarities=[]
for i in range(0,how_many):
    a=input("Enter an element:")
    c=int(input("How many elements is it similaar to?"))
    similarities.append([])
    similarities[i].append(a)

    for j in range(0,c):
        b=input("What is it similar to?")
        similarities[i].append(b)

def compare(o,t,s):
    print(o)
    print(t)
    print(s)
    score=0
    for i in range(len(o)):
        for j in range(len(s)):
            if o[i] in s[j] and t[i] in s[j] and o[i] != t[i]:
                score+=1

    similarity=(score*100)/len(o)
    return similarity

print(compare(list(sequence_one),list(sequence_two),similarities),"%")
```

Output:

Enter the first sequence::abcvdgfhijk
Enter the second sequence::abgcvfghji
How many elements for similarity condition?2
Enter an element:a
How many elements is it similaar to?2
What is it similar to?j
What is it similar to?i
Enter an element:c
How many elements is it similaar to?3
What is it similar to?v
What is it similar to?f
What is it similar to?g
['a', 'b', 'c', 'v', 'd', 'g', 'f', 'h', 'i', 'j', 'k']
['a', 'b', 'g', 'c', 'v', 'f', 'g', 'h', 'j', 'i']
[['a', 'j', 'i'], ['c', 'v', 'f', 'g']]
54.54545454545455 %

Practical 4

Aim: Write a Python/Java code to find the Percentage matching of the given sequences.

Code:

```
from random import choice, randint
from string import ascii_uppercase


# Function to generate sequences of similar or dissimilar lengths
# This function will be executed for Python v3.6+
def generate_sequences() -> tuple[list[str]]:
    char_sequence = list(ascii_uppercase)
    sequence_1 = [choice(char_sequence) for i in range(randint(8, 20))]
    sequence_2 = [choice(char_sequence) for i in range(randint(8, 20))]

    return sequence_1, sequence_2


# This function adds gaps to the sequence.
def add_gap(sequence : list[str], random_flag : bool) -> list[str]:
    index = randint(0, len(sequence) - 1)

    if not random_flag:
        index = int(input(f"Enter an integer from 0 to {len(sequence) - 1} where you want to insert the gap>\t"))

    sequence.insert(index, "-")

    return sequence


# Function to add multiple gaps in a sequence
def add_gaps(sequence_1 : list[str], sequence_2 : list[str],  random_flag : bool) -> tuple[list[str]]:
    while(len(sequence_1) != len(sequence_2)):
        if len(sequence_1) > len(sequence_2):
            sequence_2 = add_gap(sequence_2, random_flag)
```

```
        else:
            sequence_1 = add_gap(sequence_1, random_flag)

    return sequence_1, sequence_2


# Get the similar proteins from the user
def get_similar_proteins() -> list[list[str]]:
    number_of_similar_protein_sets = int(input("Enter the number of sets housing similar
proteins>\t"))
    similar_protein_list = []

    print("Enter similar proteins represented as single letter representation without any
delimiters.")

    while(number_of_similar_protein_sets > 0):
        similar_proteins = list(input("Enter similar proteins>\t"))
        similar_protein_list.append(similar_proteins)
        number_of_similar_protein_sets -= 1

    return similar_protein_list


# This function finds similarity between two sequences.
def find_similarity(sequence_1 : list[str], sequence_2 : list[str], similar_protein_list :
list[list[str]]) -> int:
    similarity = 0

    for i in range(len(sequence_1)):
        for j in range(len(similar_protein_list)):
            if(sequence_1[i] == sequence_2[i]):
                continue

            if(sequence_1[i] in similar_protein_list[j]) and (sequence_2[i] in similar_protein_list[j]):
                similarity += 1

    print(f"Similarity value for Sequence 1 and Sequence 2 is:\t{similarity}")

    return similarity
```

```
# This functions finds identity between the two sequences
def find_identity(sequence_1 : list[str], sequence_2 : list[str]) -> int:
    identity = 0

    for i in range(len(sequence_1)):
        if sequence_1[i] == sequence_2[i]:
            identity += 1

    print(f"Identity value for Sequence 1 and Sequence 2 is:\t{identity}")

    return identity


# This function calculates the total number of gaps in the sequences
def count_gaps(sequence_1 : list[str], sequence_2 : list[str]) -> int:
    return sequence_1.count('-') + sequence_2.count('-')


# This function calculates percent matches for the two sequences
def get_percent_matches(sequence_1 : list[str], sequence_2 : list[str]) -> float:
    similar_protein_list = get_similar_proteins()

    similarity = find_similarity(sequence_1, sequence_2, similar_protein_list)

    identity = find_identity(sequence_1, sequence_2)

    return ((similarity + identity) / (len(sequence_1) - count_gaps(sequence_1, sequence_2))) *
100


if __name__ == "__main__":
    print("Percent match calculator for two sequences in Python")
    is_random_sequences = input("Do you want the sequences to be randomly generated?
[Yes]/No >\t")

    sequence_1, sequence_2 = generate_sequences()
    if (is_random_sequences.lower() == "no"):
        print("Enter the two sequences. The two sequences should not have any delimiters.\nPlace
'-' as a gap (if necessary)")
```

```
        sequence_1 = list(input("Enter sequence 1>\t"))
        sequence_2 = list(input("Enter sequence 2>\t"))

    print("Sequence 1 is:\t", sequence_1)
    print("Sequence 2 is:\t", sequence_2)

    random_gaps = input("Do you want the gaps to be randomly added? [Yes]/No >\t")
    random_flag = True

    if (random_gaps.lower() == "no"):
        random_flag = False

    sequence_1, sequence_2 = add_gaps(sequence_1, sequence_2, random_flag)

    print("Sequence 1 after adding gaps:\t", sequence_1)
    print("Sequence 2 after adding gaps:\t", sequence_2)

    print(f"Percent matches in sequence 1 and sequence_2 is: {get_percent_matches(sequence_1,
sequence_2)}%")
```

Output:

```
Percent match calculator for two sequences in Python
Do you want the sequences to be randomly generated? [Yes]/No >       yes
Sequence 1 is:        ['B', 'S', 'S', 'Z', 'B', 'E', 'M', 'E', 'I', 'S', 'J', 'M', 'C', 'B']
Sequence 2 is:        ['B', 'U', 'H', 'E', 'V', 'B', 'P', 'F']
Do you want the gaps to be randomly added? [Yes]/No > yes
Sequence 1 after adding gaps:    ['B', 'S', 'S', 'Z', 'B', 'E', 'M', 'E', 'I', 'S', 'J', 'M', 'C', 'B']
Sequence 2 after adding gaps:    ['B', 'U', 'H', 'E', '-', '-', 'V', '-', '-', 'B', '-', '-', 'P', 'F']
Enter the number of sets housing similar proteins> 3
Enter similar proteins represented as single letter representation without any delimiters.
Enter similar proteins>    BU
Enter similar proteins>    WE
Enter similar proteins>    BF
Similarity value for Sequence 1 and Sequence 2 is: 1
Identity value for Sequence 1 and Sequence 2 is:    1
Percent matches in sequence 1 and sequence_2 is: 25.0%
```

Practical 5

Aim: Write a Python/Java code to create a scoring matrix for Global Alignment.

Code:

```
a=input("Enter seq 1").split()
print("Length of seq 1: ",len(a))
b=input("Enter seq 2").split()
print("Length of seq 2: ",len(b))
matrix=[]
print("Enter the entries rowwise")
for i in range(len(a)+1):
    a=[]
    for j in range(len(b)+1):
        a.append(int(input()))
    matrix.append(a)
for i in range(len(a)+1):
    for j in range(len(b)+1):
        print(matrix[i][j], end="")
    print()
```

Output:

```
Enter seq 1A D T G
Length of seq 1:  4
Enter seq 2A C E G
Length of seq 2:  4
Enter the entries rowwise
1
-9
-8
-3
3
6
-6
2
-9
3
6
```

-7
2
1
-5
7
-8
5
3
-5
-8
2
1
-2
-1
1-9-8-33
6-62-93
6-721-5
7-853-5
-821-2-1

Pratical 6

Aim: Enter genome of five different organism and write a python/java program to find consensus sequence using Multiple Sequence Alignment (MSA) technique

Code:

```
from collections import Counter

no_of_seq=int(input("Enter how many sequences is to be entered:"))
sequence=[]
lst2=[]
consensus=[]


def take_input(sequence):
    global no_of_element
    seq=input("Enter the seq in comma separated format:")
    element=seq.split(",")
    if not sequence:
        no_of_element=len(element)
    if len(element)==no_of_element:
        sequence.append(element)
        return True
    else:
        return False

for i in range(no_of_seq):
    check=take_input(sequence)
    if not check:
        print("Please enter the seq of correct length")
        check=take_iput(sequence)
        if not check:
            break

#creates a dynamic empty list to store value columnwise
for i in range(len(sequence[0])):
    lst2.append([])
```

```
#Creates list columnwise
for i in range(len(sequence[0])):   #length of first element of whole sequence
    for j in range(len(sequence)):  #no_of_ssequences
        lst2[i].append(sequence[j][i])

#counting the occurence of elements and appending as per need
for j in lst2:
    nul_list=[]
    j=[x for x in j if x != '-']
    counter=Counter(j)
    if len(list(counter.keys()))==1:
        consensus.append(list(counter.keys())[0].upper())
    elif len(list(counter.keys()))>1:
        max_value=max(counter.values())
        for i in range(len(list(counter.keys()))):
            if max_value==counter.get(list(counter.keys())[i]):
                nul_list.append(list(counter.keys())[i].upper())
        if len(nul_list)>1:
            consensus.append(nul_list)
        else:
            consensus.append(nul_list[0].lower())

#joining with /
for i in range(len(consensus)):
    final_str=''
    if type(consensus[i])==list:
        final_str='/'.join(consensus[i])
        consensus[i]=final_str
print("Consensus: ",consensus)
```

Output:

```
Enter how many sequences is to be entered:5
Enter the seq in comma separated format:A,C,T,G
Enter the seq in comma separated format:T,C,A,G
Enter the seq in comma separated format:T,A,T,G
Enter the seq in comma separated format:T,G,C,A
Enter the seq in comma separated format:-,-,T,A
Consensus:  ['t', 'c', 't', 'g']
```

Practical 7

Aim: Generate a regular expression enter three protein sequence of three different organism. Write Python/Java code to find regular expression for these sequences.

Code:

```
def gen_reg_exp(seq_list, no_of_col):
    final_list=[]
    for colnum in range(no_of_col):
        collist=[]
        for colseq in seq_list:
            collist.append(colseq[colnum])
        if len(set(collist))==len(collist):
            #print(final_list)
            final_list.append('x')
        else:
            if len(set(collist))==1:
                final_list.append(collist[0])
            else:
                final_list.append(''.join(set(collist)))
    display_output(final_list)

def display_output(final_list):
    print(*final_list, sep='-')
no_of_seq=int(input("Enter the number of sequence: "))
print("Enter all the sequences")
seq_list=[]
for _ in range(no_of_seq):
    seq_list.append(list(map(str, input("").split())))
gen_reg_exp(seq_list, len(seq_list[0]))
```

Output:

```
Enter the number of sequence: 4
Enter all the sequences
A D L G A V F A L C D R Y F Q
S D V G P R S C F C E R F Y Q
A D L G R T Q L R C D R Y Y Q
```

A D I G Q P H S L C E R Y F Q
SA-D-ILV-G-x-x-x-x-RLF-C-ED-R-YF-YF-Q

Practical 8

Aim: **Enter six sequence of different organism and write a program to find a fingerprint of sequence.**

Code:

```
def solve_fingerprint(seq_list,no_of_col):
    seq_dict=dict()
    for colnum in range(no_of_col):
        counta,countc,countt,countg=0,0,0,0
        for colseq in seq_list:
            if colseq[colnum]=='A':
                counta+=1
            elif colseq[colnum]=='T':
                countt+=1
            elif colseq[colnum]=='C':
                countc+=1
            elif colseq[colnum]=='G':
                countg+=1
        seq_dict[colnum]=[counta,countc,countt,countg]
    display_results(seq_dict)
def display_results(seq_dict):
    print("\tA \tC \tT \tG")
    for key in seq_dict:
        print("\n",*seq_dict[key],sep="\t")
no_of_seq=int(input("Enter the number of sequence: "))
print("Enter all the sequences")
seq_list=[]
for _ in range(no_of_seq):
    seq_list.append(list(map(str, input("").split())))
solve_fingerprint(seq_list,len(seq_list[0]))
```

Output:

Enter the number of sequence: 6
Enter all the sequences
A C T G A T G

A T C A G A A
A T A A G C A
A G T T A G C
G A T A C G T
T G C A T G A

A	C	T	G
4	0	1	1
1	1	2	2
1	2	3	0
4	0	1	1
2	1	1	2
1	1	1	3
3	1	1	1

Practical 9

Aim: Write a Python/Java code to find motif in a given sequence.

Code:

```
import random
l=int(input("Enter the length of motif"))
file=open("mot.txt","r")
r=file.read()
print("Sequence",r)
size=len(r)
print("Size of the sequence",size)
pos=random.randint(0,len(r)-5)
#pos=1
print("Position",pos)
motif=r[pos:pos+l]
print("Motif",motif)
i=pos+1
while(i<=size-1):
    if(motif==r[i:i+1]):
        str1=r[i:i+1]
        print("Match motif",str1)
        file1=open("motoutput.txt","a")
        file1.write(str1+" ")
    i+=1
```

Output:

```
Enter the length of motif4
Sequence AGAAGTTCGAGAAGCCGTAGT
Size of the sequence 21
Position 0
Motif AGAA
```

Practical 10

Aim: Perform BLAST search and Find the no of repetition of each nucleotide in the sequence.

Code:

```
file=open("genes.txt","r")
r=file.read()
size=len(r)
score_A=0
score_C=0
score_T=0
score_G=0
for i in range(size):
        if(r[i]=='A'):
                score_A+=1
        elif (r[i]=='C'):
                score_C+=1
        elif (r[i]=='T'):
                score_T+=1
        elif (r[i]=='G'):
                score_G+=1
print("score of A is ",score_A)
print("score of C is ",score_C)
print("score of T is ",score_T)
print("score of G is ",score_G)
```

Output:

```
score of A is 6
score of C is 4
score of T is 7
score of G is 6
```