Practicals – 4

-BS19B032

-R. Vasantha Kumar

1) The human hemoglobin beta chain is:

>sp|P68871|HBB_HUMAN Hemoglobin subunit beta OS=Homo sapiens OX=9606 GN=HBB PE=1 SV=2

MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPK

VKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFG

KEFTPPVQAAYQKVVAGVANALAHKYH

The chicken hemoglobin beta chain is:

>sp|P02112|HBB_CHICK Hemoglobin subunit beta OS=Gallus gallus OX=9031 GN=HBB PE=1 SV=2

MVHWTAEEKQLITGLWGKVNVAECGAEALARLLIVYPWTQRFFASFGNLSSPTAILGNPM

VRAHGKKVLTSFGDAVKNLDNIKNTFSQLSELHCDKLHVDPENFRLLGDILIIVLAAHFS

KDFTPECQAAWQKLVRVVAHALARKYH

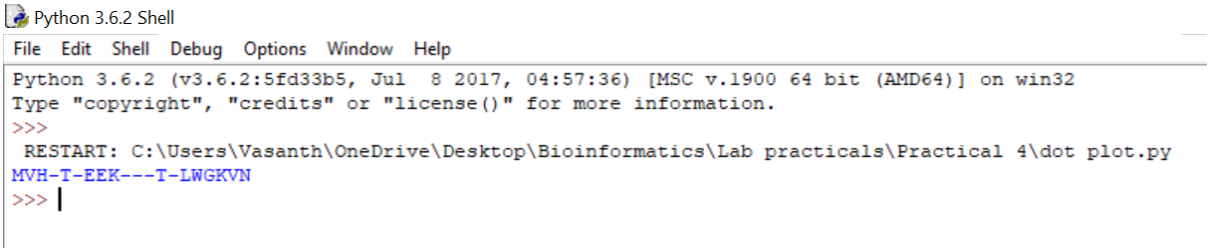
a) The code for constructing dot plot for alignment of human and chicken hemoglobin beta chain is:

```
#importing required libraries

import numpy as np

import matplotlib as plt

import matplotlib.pyplot as pyplt

#given sequences

human_seq = "MVHLTPEEKSAVTALWGKVN"

chick_seq = "MVHWTAEEKQLITGLWGKVN"


#function to check matches between given sequences

def seq_check(seq1,seq2):

    mat=np.zeros((len(seq1),len(seq2)))

    for i in range(len(seq1)):

        for j in range(len(seq2)):

            if(seq1[i]==seq2[j]):

                if(i==j):
```

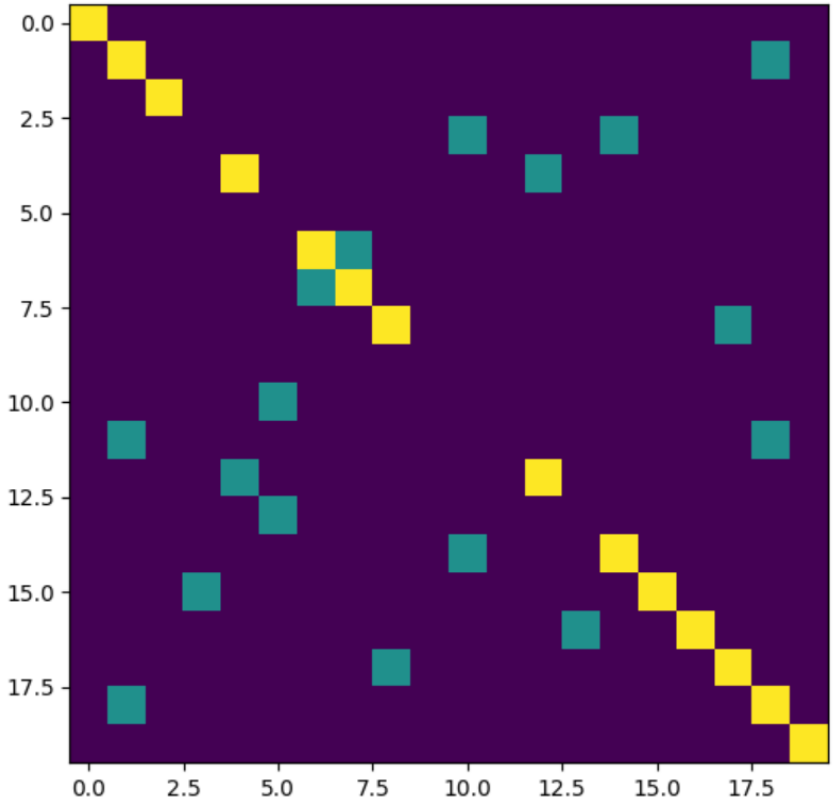
```
            mat[i][j]=2
        else:
            mat[i][j]=1
    return mat

#function to print matched parts of seuence
def align_seq(seq1,seq2):
    seq = ""
    for i in range(len(seq1)):
        if(seq1[i]==seq2[i]):
            seq = seq + seq1[i]
        else:
            seq = seq + "-"
    return seq

aligned_seq = align_seq(human_seq,chick_seq)
print(aligned_seq)
#plotting the graph
mat = seq_check(human_seq,chick_seq)
pyplt.imshow(mat)
pyplt.show()
```

The aligned parts of the given sequences for length of 20 residues is:

MVH-T-EEK---T-LWGKVN



The dot plot obtained is:

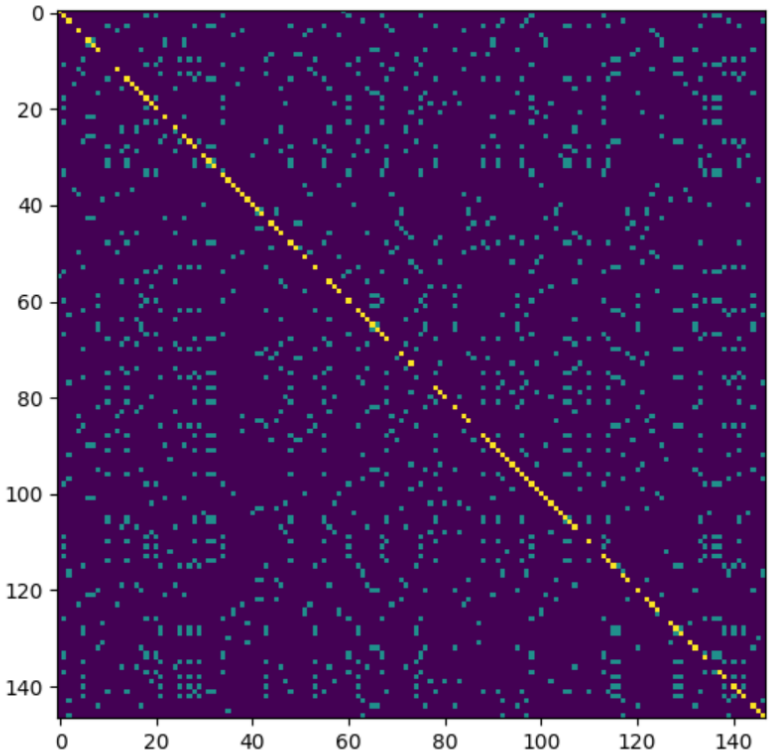


Yellow dots = Same segments in both sequences

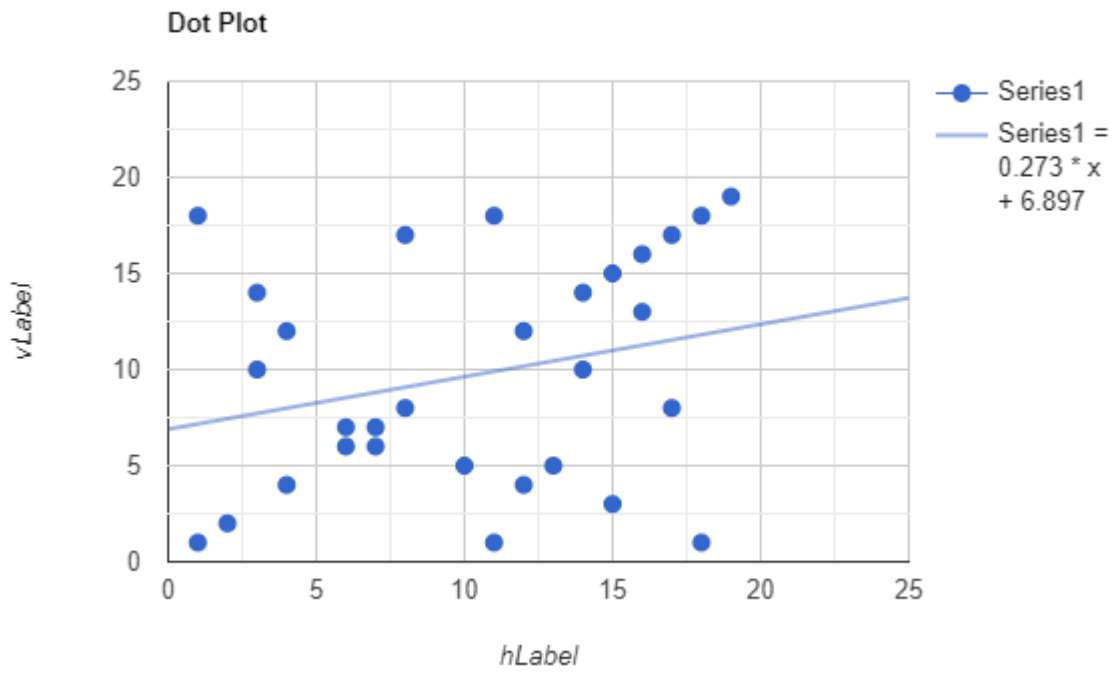
Green dots = Matched pairs in sequences

Violet dots = Unmatched pairs in sequences

Dot plot for complete sequences is:



b) The dot plot for first 20 residues of human and chicken sequences is:



This dot plot is identical to the dot plot constructed form the code. But only the y-axis is reversed.

Hence, verified.

2) The code for calculating the score is:

```
#getting sequence input from user
seq1 = input("Enter your sequence 1:")
seq2 = input("Enter your sequence 2:")

match_score = 1
mismatch_score = 0
origination_penalty = -2
length_penalty = -1

#function to find number of gaps
def gap_find(seq):
    gap_count=0
    flag=0
```

```
    for i in range(len(seq)):
        if(seq[i]=='-'):
            if(flag==0):
                gap_count=gap_count+1
                flag=1
        else:
            flag=0
    return gap_count

#function to count total gaps(including consecutive gaps)
def gap_num(seq):
    gaps = 0
    for i in range(len(seq)):
        if(seq[i]=='-'):
            gaps=gaps+1
    return gaps

#initialising total match and mismatch counts to 0
match=0
mismatch=0

#calculating length penalty(including the gaps)
if(len(seq1)==len(seq2)):
    length_diff=abs(gap_num(seq1)-gap_num(seq2))
else:
    length_diff=abs(abs(len(seq1)-len(seq2)) - abs(gap_num(seq1)-gap_num(seq2)))

for i in range(len(seq1)):
    if(seq1[i]==seq2[i]):
        match=match+1
    else:
        mismatch=mismatch+1

#calculating gap penalty
```

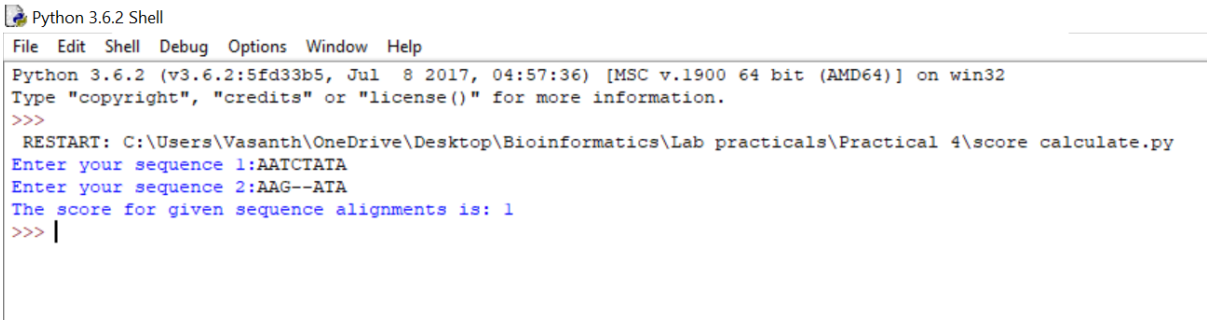
```
gap_count = gap_find(seq1) + gap_find(seq2)

#calculating total score for alignment with given match scores
score = match*match_score + mismatch*mismatch_score + gap_count*origination_penalty + length_diff*length_penalty

print("The score for given sequence alignments is:",score)
```

Using the above code, I calculated the sequence alignment score for given sequences:

Seq1: AATCTATA

Seq2: AAG—ATA

*The result score is **1**.*



3) Given sequences,

Seq1: AATCTATA

Seq2: AAG—ATA

Total matches = 5

position 1: A=A

position 2: A=A

position 6: A=A

position 7: T=T

position 8: A=A

Total mismatches = 1

position 3: T ~= G

Total Gaps = 1

At position 4,5.

Length penalty = length of seq1 – length of seq2

= 8-6

=2

Match score = 1; Mismatch score = 0;

Origination penalty = -2; Length penalty = -1

Therefore,

*Total score = (5*1) + (1*0) + (1*-2) + (2*-1)*

=1

Hence, the final score is 1, and it is also the same score we got from code.

Hence, verified.

4) The code for calculating partial alignment score table using Needleman and Wunsch dynamic programming method is:

```
import numpy as np

#given sequences
seq1 = "ACAGTCGAACG"
seq2 = "ACCGTCCG"

#given scores
match_score = 2
mismatch_score = -1
```

```
gap_penalty = -2

#initialisng a matrix of 0 of size-length of seq1+1 and length of seq2+1
mat = np.zeros((len(seq2)+1,len(seq1)+1))
#filling gap penalties in first row and column
for i in range(1,len(seq1)+1):
    mat[0][i]=mat[0][i-1] + gap_penalty
for j in range(1,len(seq2)+1):
    mat[j][0]=mat[j-1][0] + gap_penalty

#function to calculate left score
def left_score(matrix,n,m):
    a = matrix[n][m-1] + gap_penalty
    return a
#function to calculate top score
def top_score(matrix,n,m):
    b = matrix[n-1][m] + gap_penalty
    return b
#function to calculate diaagonal score
def diag_score(matrix,n,m):
    if seq2[n-1]==seq1[m-1]:
        c = matrix[n-1][m-1] + match_score
    else:
        c = matrix[n-1][m-1] + mismatch_score
    return c

#filling the matrix using formula: maximum of top,left,diagonal score
for i in range(1,len(seq2)+1):
    for j in range(1,len(seq1)+1):
        a=left_score(mat,i,j)
        b=top_score(mat,i,j)
        c=diag_score(mat,i,j)
        mat[i][j] = max(a,b,c)
```

```
x=len(seq2)
y=len(seq1)
#initialising traceback matrix
traceback_mat=[]

#tracing back
while(x>=0):
    while(y>=0):
        if(seq2[x-1]==seq1[y-1]):
            traceback_mat.append(1)
            x=x-1
            y=y-1
        else:
            if(mat[x-1][y-1]>=mat[x-1][y] and mat[x-1][y-1]>=mat[x][y-1]):
                traceback_mat.append(0)
                x=x-1
                y=y-1
            elif(mat[x][y-1]>=mat[x-1][y] and mat[x][y-1]>mat[x-1][y-1]):
                traceback_mat.append(-1)
                y=y-1
            elif(mat[x-1][y]>mat[x][y-1] and mat[x-1][y]>mat[x-1][y-1]):
                traceback_mat.append(-2)
                x=x-1

#reversing the traceback matrix
traceback_mat = traceback_mat[::-1]
#initialsing new akignment sequence
align_seq1=""
align_seq2=""
#creating aligned sequence using traceback matrix
for i in range(1,len(traceback_mat)):
    if(traceback_mat[i]==1):
        align_seq1=align_seq1+seq1[i-1]
        align_seq2=align_seq2+seq1[i-1]
```

```
    elif(traceback_mat[i]==0):
        align_seq1=align_seq1+seq1[i-1]
        align_seq2=align_seq2+seq2[i-1]
    elif(traceback_mat[i]==-1):
        align_seq1=align_seq1+seq1[i-1]
        align_seq2=align_seq2+"-"
    else:
        align_seq1=align_seq1+"-"
        align_seq2=align_seq2+seq2[i-1]

#printing the values
print("The aligned sequence 1 is:")
print(align_seq1)
print("The aligned sequence 2 is:")
print(align_seq2)
print("The partial alignment matrix is:")
print(mat)
```

```
The aligned sequences are:
ACAGTCGAACG
ACCGTC---CG
The alignment table is:
[[  0.  -2.  -4.  -6.  -8. -10. -12. -14. -16. -18. -20. -22.]
 [ -2.   2.   0.  -2.  -4.  -6.  -8. -10. -12. -14. -16. -18.]
 [ -4.   0.   4.   2.   0.  -2.  -4.  -6.  -8. -10. -12. -14.]
 [ -6.  -2.   2.   3.   1.  -1.   0.  -2.  -4.  -6.  -8. -10.]
 [ -8.  -4.   0.   1.   5.   3.   1.   2.   0.  -2.  -4.  -6.]
 [-10.  -6.  -2.  -1.   3.   7.   5.   3.   1.  -1.  -3.  -5.]
 [-12.  -8.  -4.  -3.   1.   5.   9.   7.   5.   3.   1.  -1.]
```

[-14. -10. -6. -5. -1. 3. 7. 8. 6. 4. 5. 3.]

[-16. -12. -8. -7. -3. 1. 5. 9. 7. 5. 3. 7.]]

Python 3.6.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Users\Vasanth\OneDrive\Desktop\Bioinformatics\Lab practicals\Practical 4\needleman-wunsh.py
The aligned sequence 1 is:
ACAGTCGAACG
The aligned sequence 2 is:
ACCGTC---CG
The partial alignment matrix is:
[[  0.  -2.  -4.  -6.  -8. -10. -12. -14. -16. -18. -20. -22.]
 [ -2.   2.   0.  -2.  -4.  -6.  -8. -10. -12. -14. -16. -18.]
 [ -4.   0.   4.   2.   0.  -2.  -4.  -6.  -8. -10. -12. -14.]
 [ -6.  -2.   2.   3.   1.  -1.   0.  -2.  -4.  -6.  -8. -10.]
 [ -8.  -4.   0.   1.   5.   3.   1.   2.   0.  -2.  -4.  -6.]
 [-10.  -6.  -2.  -1.   3.   7.   5.   3.   1.  -1.  -3.  -5.]
 [-12.  -8.  -4.  -3.   1.   5.   9.   7.   5.   3.   1.  -1.]
 [-14. -10.  -6.  -5.  -1.   3.   7.   8.   6.   4.   5.   3.]
 [-16. -12.  -8.  -7.  -3.   1.   5.   9.   7.   5.   3.   7.]]
>>>
```

5) *The partial alignment table for given sequences is:*

File Edit Table Column Row Cell Help

	A	B	C	D	E	F	G	H	I	J	K	L	M
1			A	C	A	G	T	C	G	A	A	C	G
2		0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22
3	A	-2	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
4	C	-4	0	4	2	0	-2	-4	-6	-8	-10	-12	-14
5	C	-6	-2	2	3	1	-1	0	-2	-4	-6	-8	-10
6	G	-8	-4	0	1	5	3	1	2	0	-2	-4	-6
7	T	-10	-6	-2	-1	3	7	5	3	1	-1	-3	-5
8	C	-12	-8	-4	-3	1	5	9	7	5	3	1	-1
9	C	-14	-10	-6	-5	-1	3	7	8	6	4	5	3
10	G	-16	-12	-8	-7	-3	1	5	9	7	5	3	7

Generate

Result (click "Generate" to refresh)

So, the aligned sequences are:

ACAGTCGAACG

ACCGTC---CG

The score for these sequences are:

Matches = 7

Mismatches = 1

Gaps = 3

Score = matches(2) + mismatches*(-1) + gaps*(-2)*

*= 7*2 + 1*(-1) + 3*(-2)*

= 7

Total score is 7.

Hence, verified.