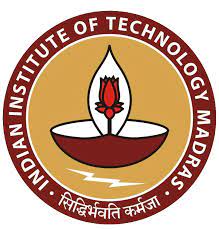
**BT 3040: Bioinformatics**

**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

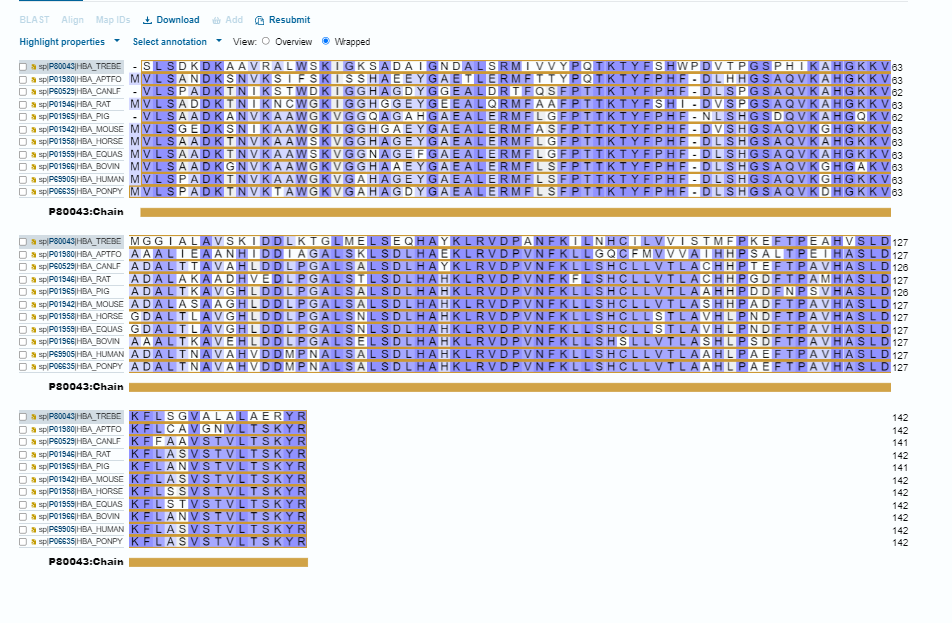
#### (ii) Unweighted frequency and variance-based measure

#### (iii) Unweighted frequency and sum of pairs measure

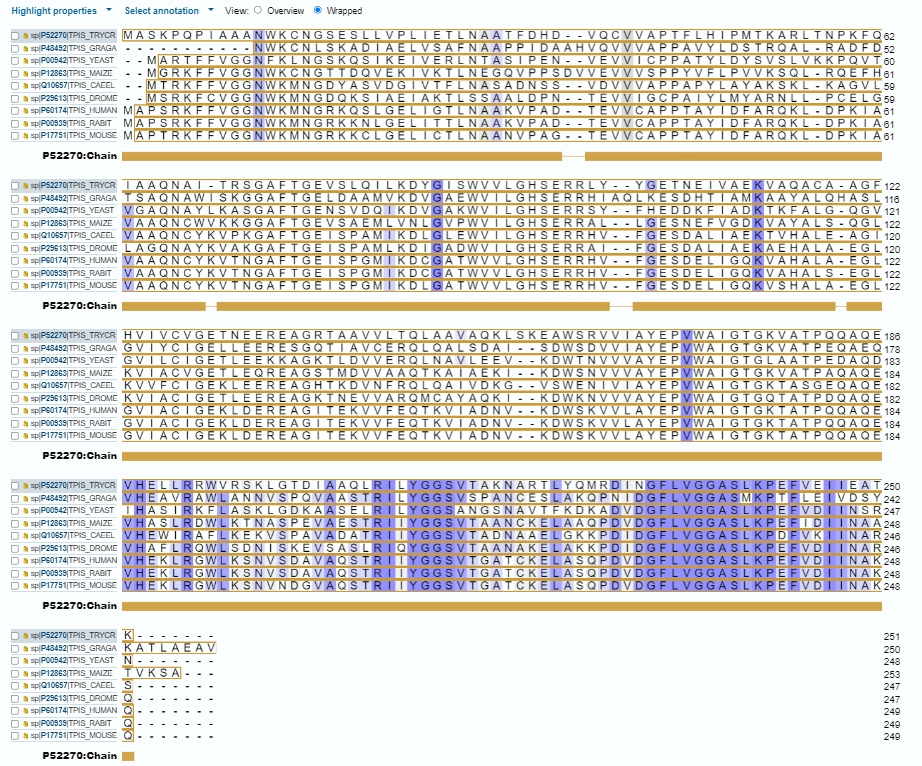
#### (iv) Weighted frequency and variance-based measure

#### (v) Normalize the scores obtained with (i)

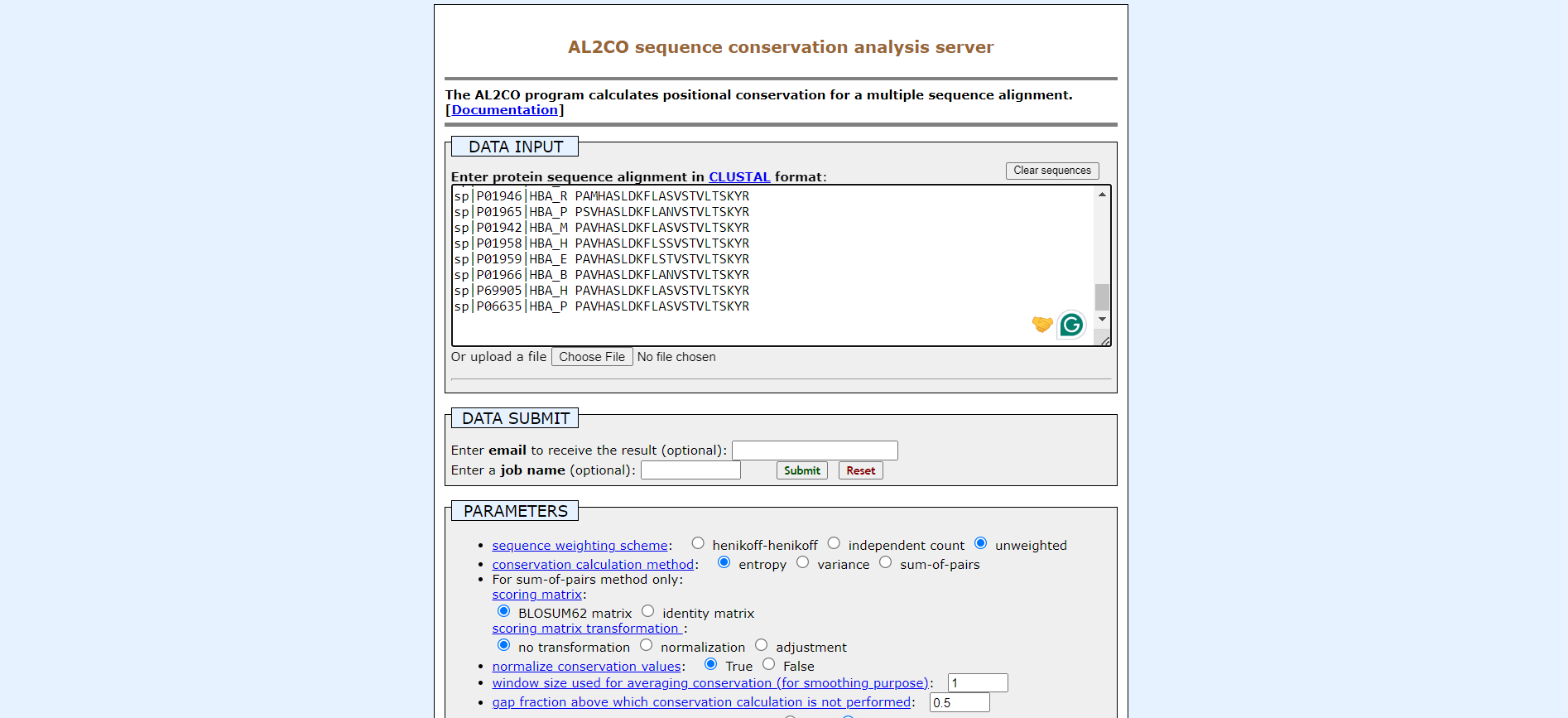
### Set 1 Alignment

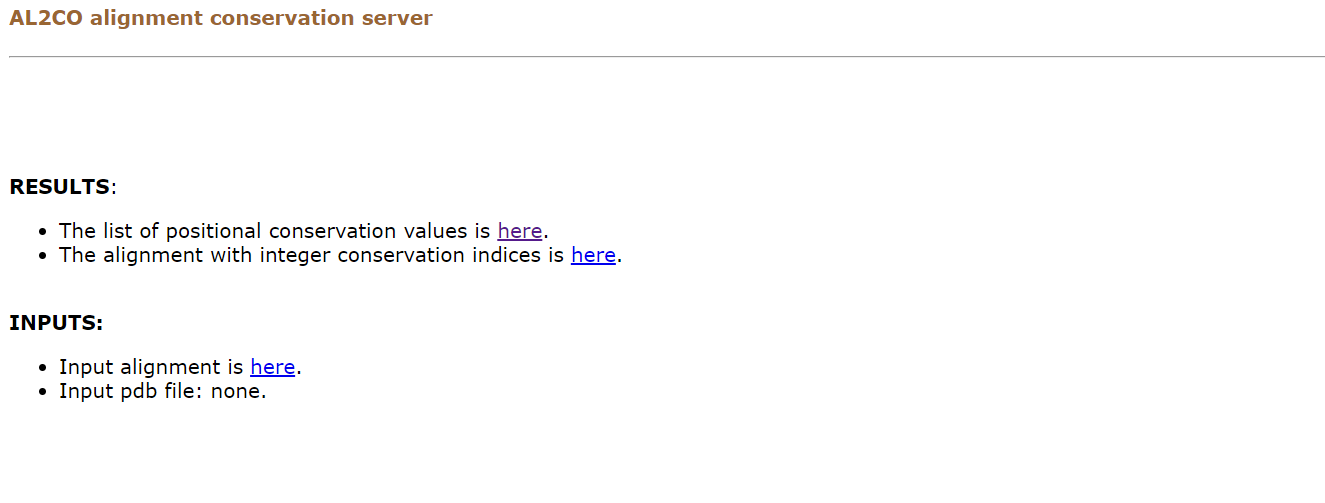


### Set 2 Alignment



### AL2CO interface

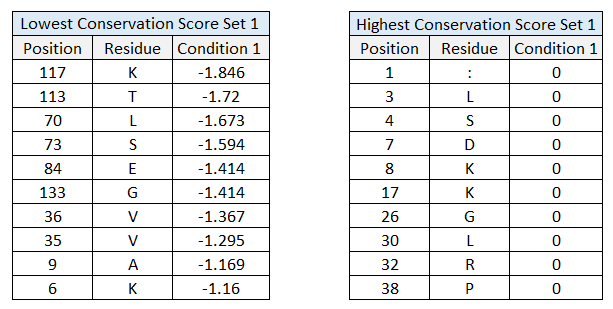




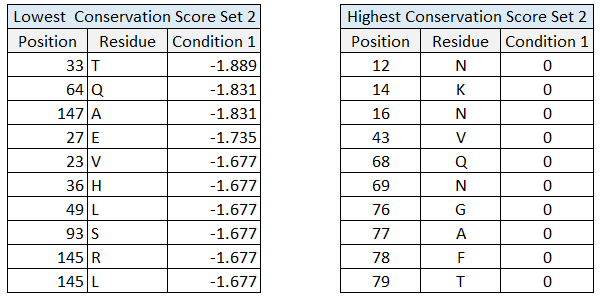
#### # Based on the question, we can tune the parameters and obtain the results. The outputs are mentioned in separate folders for Set 1 and Set 2

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

### Set 1



### Set 2



## **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 = ['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHIKAH',

            'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',

            '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',

            'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',

            '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQVKAH',

            'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',

            'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

            'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

            'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

            'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

            'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKDH']

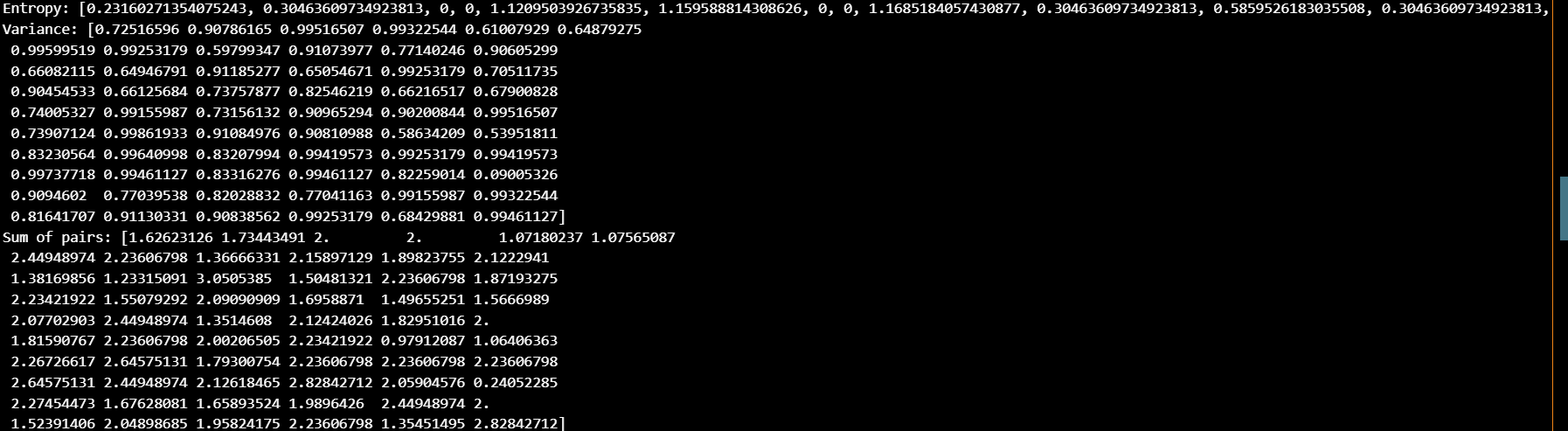
    entropy, variance, sum\_of\_pairs = Con\_score\_MSA(data)

    print('Entropy:', entropy)

    print('Variance:', variance)

    print('Sum of pairs:', sum\_of\_pairs)

#### Output:



## **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 =['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHIKAH',

            'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',

            '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',

            'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',

            '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQVKAH',

            'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',

            'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

            'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

            'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

            'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

            'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKDH']

    MAFFT = ['MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

         'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKDH',

         'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

         'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

         'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

         'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',

         '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQVKAH',

         'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',

         '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',

         'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',

         '-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHIKAH']

    MUSCLE = ['-SLSDKDKAAVRALWSKIGKSADAIGNDALSRMIVVYPQTKTYFSHWPDVTPGSPHIKAH',

         'MVLSANDKSNVKSIFSKISSHAEEYGAETLERMFTTYPQTKTYFPHF-DLHHGSAQVKAH',

         '-VLSPADKTNIKSTWDKIGGHAGDYGGEALDRTFQSFPTTKTYFPHF-DLSPGSAQVKAH',

         'MVLSADDKTNIKNCWGKIGGHGGEYGEEALQRMFAAFPTTKTYFSHI-DVSPGSAQVKAH',

         '-VLSAADKANVKAAWGKVGGQAGAHGAEALERMFLGFPTTKTYFPHF-NLSHGSDQVKAH',

         'MVLSGEDKSNIKAAWGKIGGHGAEYGAEALERMFASFPTTKTYFPHF-DVSHGSAQVKGH',

         'MVLSAADKTNVKAAWSKVGGHAGEYGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

         'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQVKAH',

         'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

         'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKGH',

         'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSHGSAQVKDH',]

    Clustal\_Omega\_entropy, Clustal\_Omega\_variance, Clustal\_Omega\_sum=Con\_score\_MSA(Clustal\_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)

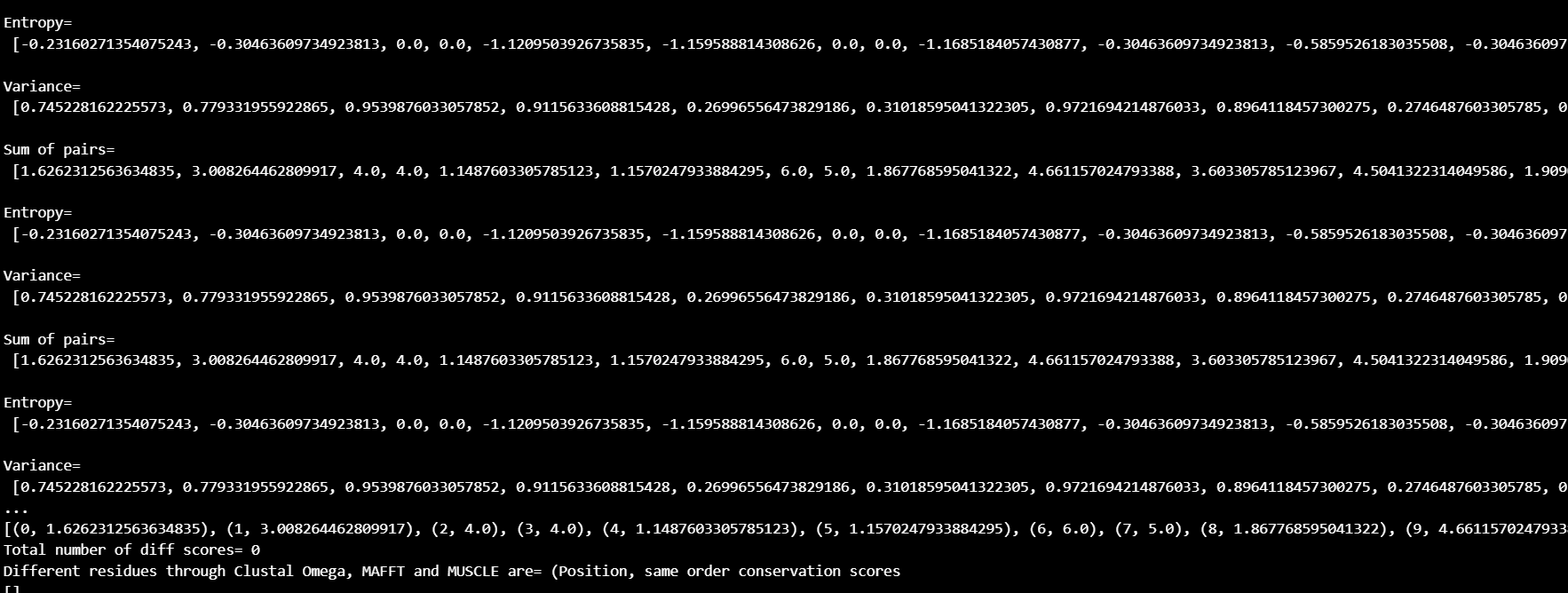
    print('Comparing sum of all pairs-based conservation scores...')

    compare\_msa(Clustal\_Omega\_sum, MAFFT\_sum, MUSCLE\_sum)

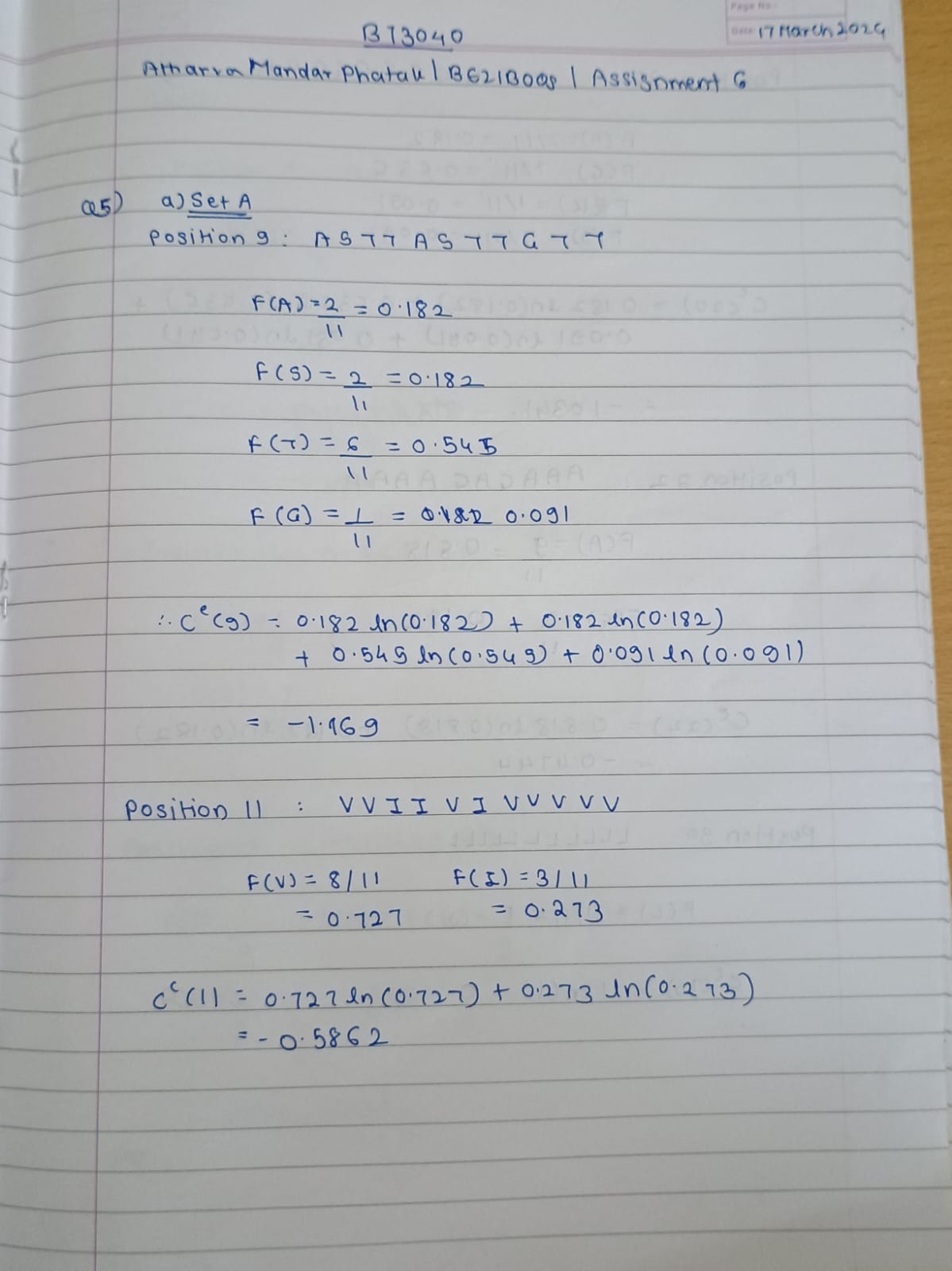
    return None

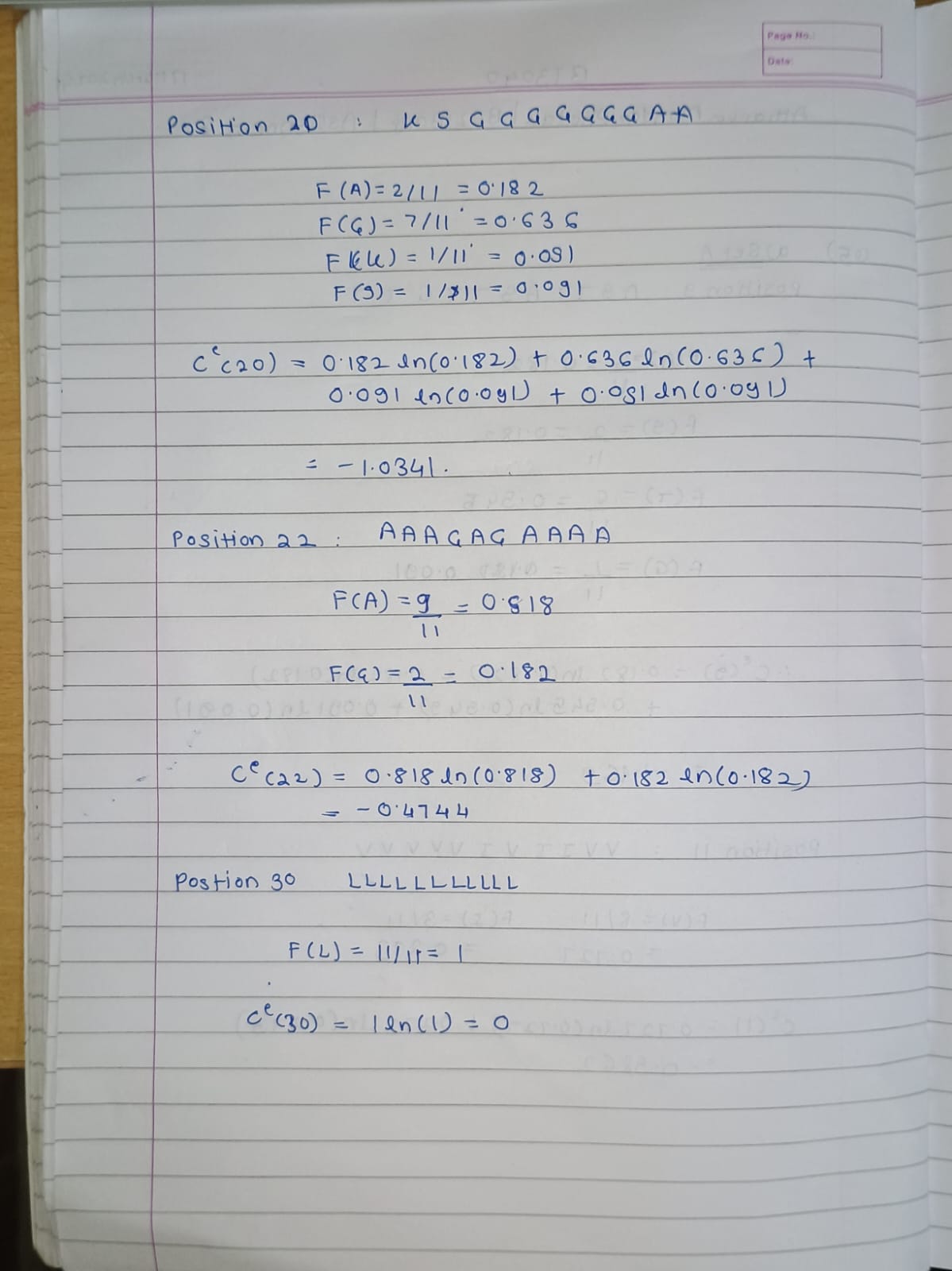
compare\_results()

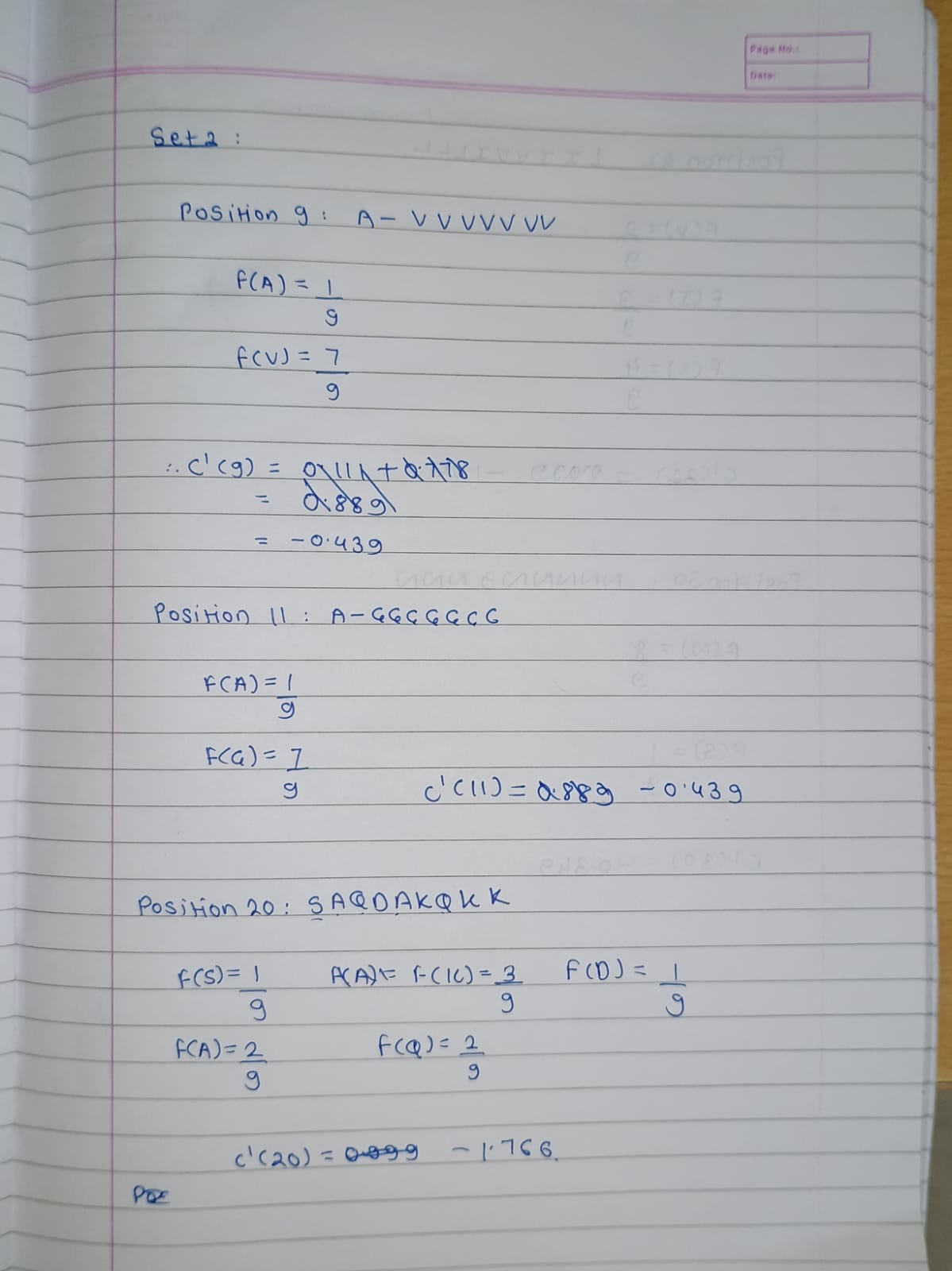
#### Output\*:

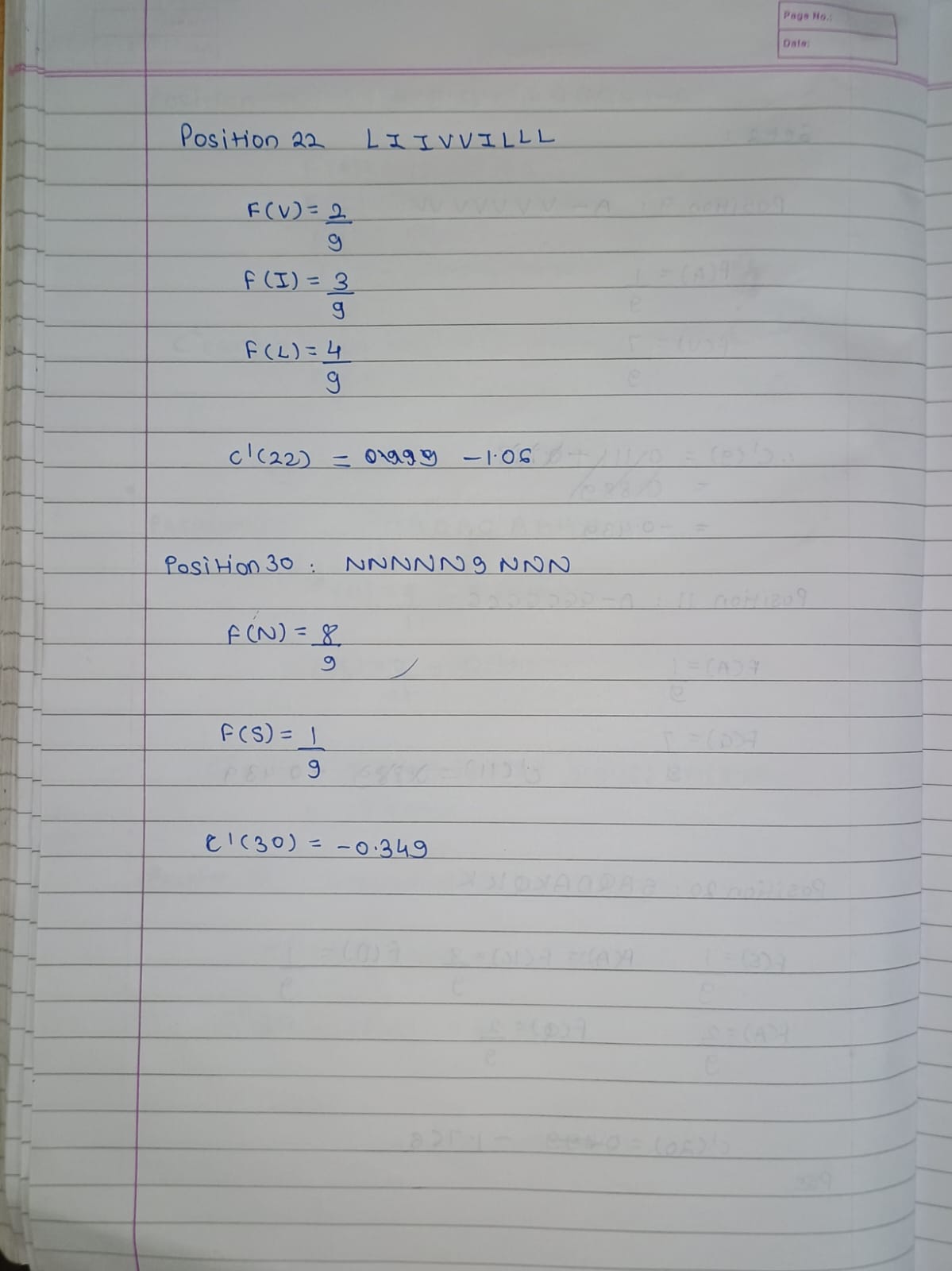
*\*Only a small snippet is shown. The complete output is saved in text file named “Assignment6\_Q4\_Output*”

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



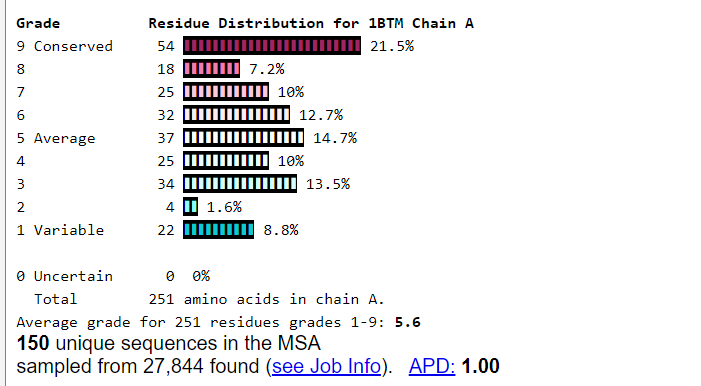




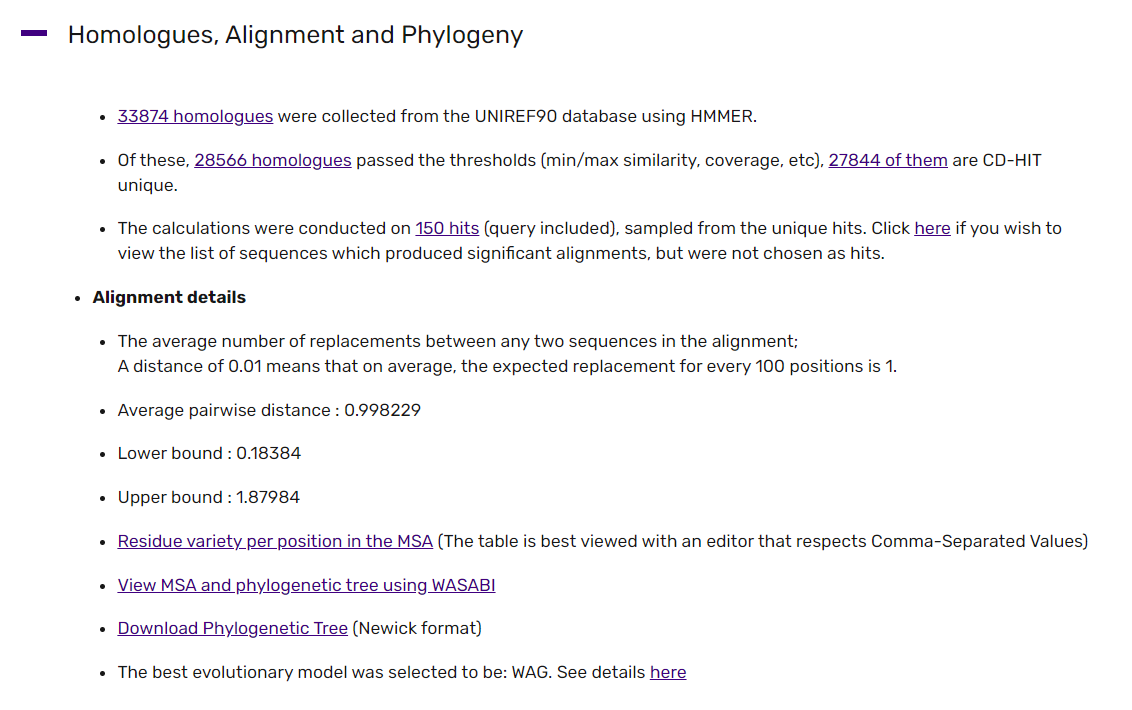


**Q6) Obtain the conservation score of 1BTM, A-chain using Consurf server (**[**https://consurf.tau.ac.il/**](https://consurf.tau.ac.il/)



****



****

#### # MSA Amino Acid Percentage, check excel file named ‘Assignment6\_Q6\_msa\_aa\_variety\_percentage’.