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

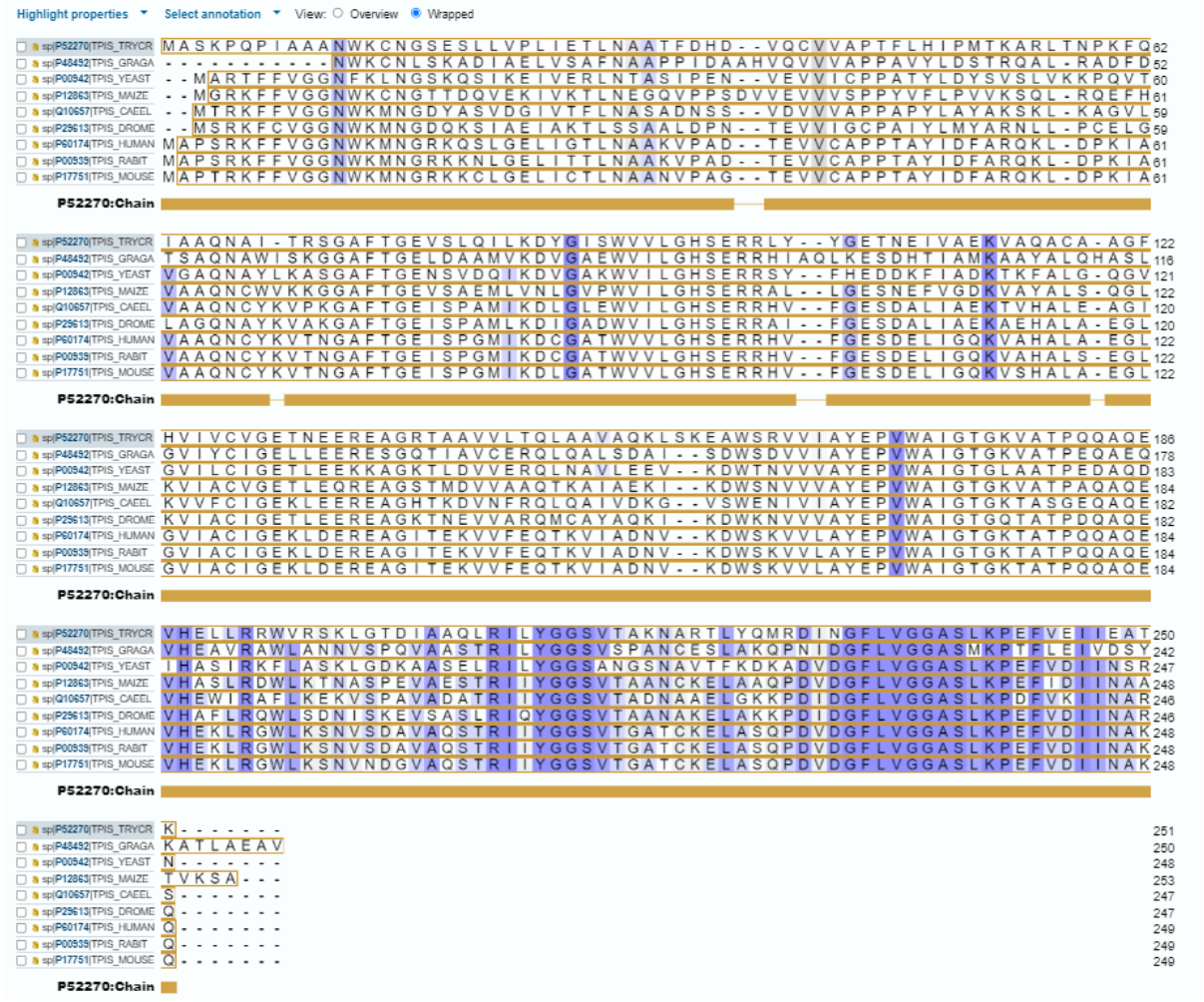
(iii) Unweighted frequency and sum of pairs measure

(iv) Weighted frequency and variance-based measure

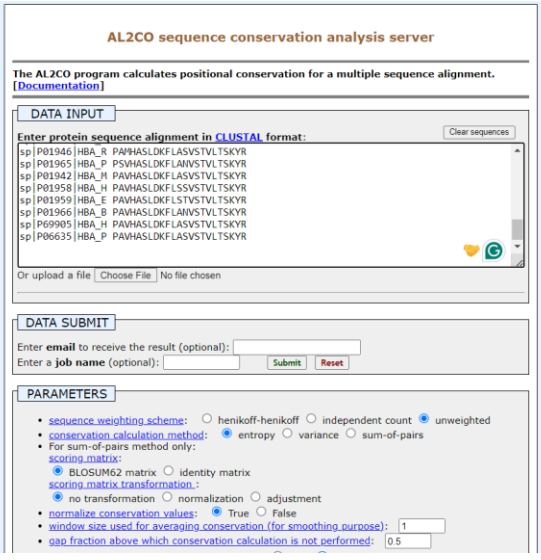
(v) Normalize the scores obtained with (i)

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, 1.1209503926735835, 1.159588814308626, 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',
```

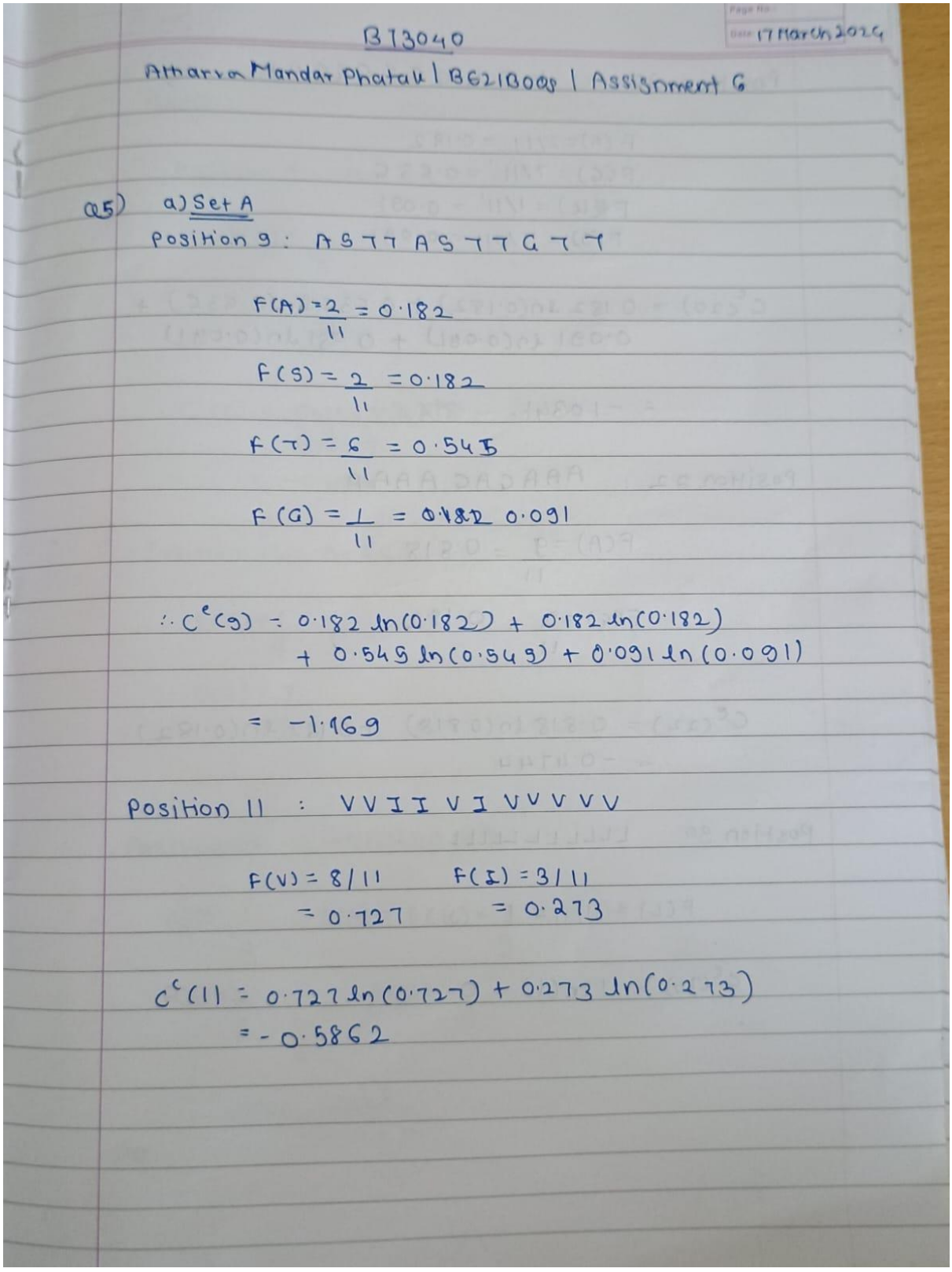
```
    'MVLSAADKTNVKAAWSKVGGNAGEFGAEALERMFLGFPTTKTYFPHF-DLSHGSAQV-
KAH',
    'MVLSAADKGNVKAAWGKVGGHAAEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
    'MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKGH',
    'MVLSPADKTNVKTAWGKVGAHAGDYGAEALERMFLSFPTTKTYFPHF-DLSH-
GSAQVKDH',]
  Clustal_Omega_entropy, 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=
 [-0.2316027135407524 3, -0.30463609734923813, 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, 0.9539876033057852, 0.9115633608815428, 0.26996556473829186, 0.31018595041322305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0

Sum of pairs=
 [1.6262312563634835, 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=
 [-0.2316027135407524 3, -0.30463609734923813, 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, 0.9539876033057852, 0.9115633608815428, 0.26996556473829186, 0.31018595041322305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0

Sum of pairs=
 [1.6262312563634835, 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=
 [-0.2316027135407524 3, -0.30463609734923813, 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, 0.9539876033057852, 0.9115633608815428, 0.26996556473829186, 0.31018595041322305, 0.9721694214876033, 0.8964118457300275, 0.2746487603305785, 0
...
[(0, 1.6262312563634835), (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, 4.661157024793
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

17 March 2024

Atharva Mandar Phatak | BG21B005 | Assignment 6

Q5) a) Set A

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

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

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

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

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

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

$= -1.169$

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

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

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

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

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

$C^e(20) = 0.182 \ln(0.182) + 0.636 \ln(0.635) + 0.091 \ln(0.091) + 0.091 \ln(0.091)$

$= -1.0341.$

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

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

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

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

$= 0.889$

$= -0.439$

Position 11 : A - G G G G G G

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

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

$C'(11) = 0.889 - 0.439$

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

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

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

$C'(20) = 0.889 - 1.756$

Pos

Position 22 LIIVVILLL

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

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

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

$c'(22) = 0.999 - 1.06$

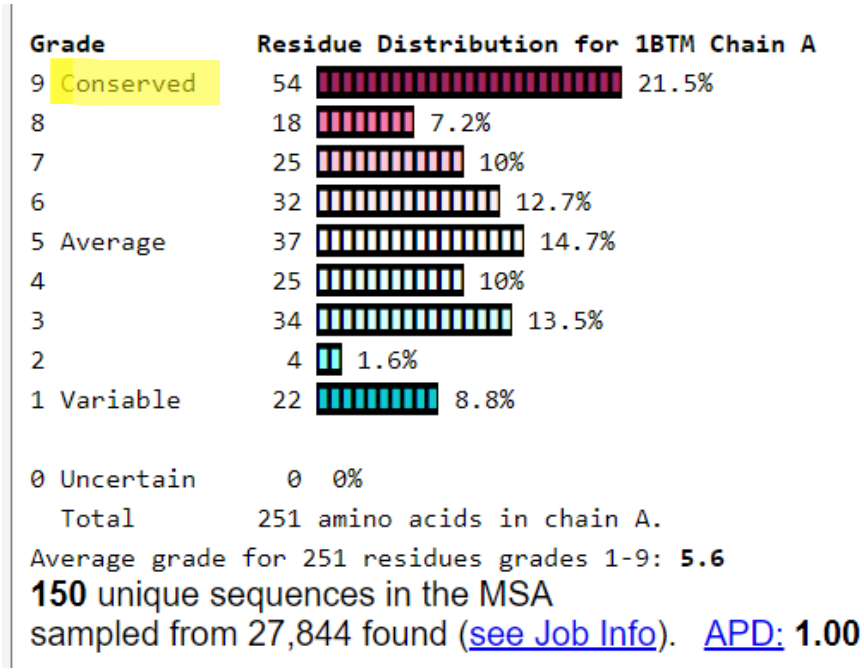
Position 30 : NNNNNNSNNN

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

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

$c'(30) = -0.349$

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'.