PRACTICAL 11

Question 1. Find the propensity of alpha helices using the following sequence and secondary structure assignments.

Q1 set of sequences with their corresponding locations and type of secondary structure

```
Sequence: LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQY
Sec  str: XHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXX

Sequence: GMGIPPLHDGGWWLMAGLFMTLSLGSWWIRVYSRARALGLGTHIAWNFAA
Sec. str: XXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHXXHHHHHHHH

Sequence: AIFFVLCIGCIHPTLVGSWSEGVPFGIWPHIDWLTAFSIRYGNFYYCPWH
Sec. str: HHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHH

Sequence: GFSIGFAYGCGLLFAAHGATILAVARFGGDREIEQITDRGTAVERAALFW
Sec. str: HHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXX
```

LISTING 1. Input sequence with corresponding secondary structure

Solution. Propensities, one for each type of secondary structure, are obtained from statistical analysis of proteins of known secondary structure, as ratio of the fractional occurrence of the residue in secondary structure elements of a given type to the fractional occurrence in all structures. The formula for propensity is:

$$propensity_\alpha(i) = \frac{\%\_of\_residue\_in\_\alpha\_helix}{\%\_of\_all\_residue\_in\_\alpha\_helix} \tag{1}$$

$$\%\_of\_residue\_in\_\alpha\_helix = \frac{n_\alpha(i)}{N(i)} \tag{2}$$

$$\%\_of\_all\_residue\_in\_\alpha\_helix = \frac{n_\alpha}{N} \tag{3}$$

$propensity_\alpha(i)$ = propensity of amino acid i in given conformation alpha
$n_\alpha(i)$ = number of residues of type i in $\alpha - helix$
$N(i)$ = number of residues of type i in the whole dataset
n_α = total number of residues in a-helix
N = total number of residues in the whole dataset

```
seq =
"LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQYGMGIPPLHDGGWWLMAGL
FMTLSLGSWWIRVYSRARALGLGTHIAWNFAAAIFFVLCIGCIHPTLVGSWSEGVPFGIWPHIDWLTAF
SIRYGNFYYCPWHGFSIGFAYGCGLLFAAHGATILAVARFGGDREIEQITDRGTAVERAALFW"
type_seq=
"XHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHXXHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXX"

amino_acid_list =
{"A": 0,"R": 1,"N": 2,"D": 3,"C": 4,
 "Q": 5,"E": 6,"G": 7,"H": 8,"I": 9,
 "L":10,"K":11,"M":12,"F":13,"P":14,
 "S":15,"T":16,"W":17,"Y":18,"V":19}
```

```
15
16 amino_acids = list(amino_acid_list.keys())
17
18 amino_helix = [0 for i in range(20)]
19 total_seq = [0 for i in range(20)]
20 propensity = [0 for i in range(20)]
21
22 total_helix = type.count("H")
23 total_data = len(type)
24
25 # Computing the frequency of each amino acid and their corresponding helix
      structure
26 for i in range(len(seq)):
27     total_seq[amino_acid_list[seq[i]]] += 1
28     if type_seq[i] == "H":
29         amino_helix[amino_acid_list[seq[i]]] += 1
30
31 # Calculating the propensity
32 for i in range(len(propensity)):
33     numerator = amino_helix[i] / total_seq[i]
34     denominator = total_helix / total_data
35     propensity[i] = numerator / denominator
36
37 data = {"Amino Acid Symbol": amino_acids, "Propensity": propensity}
38 import pandas as pd
39 df = pd.DataFrame(data)
```

Listing 2. Code to compute the propensity of alpha helices

Amino Acid Symbol	Propensity
A	1.551020
R	0.680272
N	1.360544
D	0.000000
C	1.020408
Q	0.000000
E	0.408163
G	1.061224
H	0.874636
I	1.200480

(a) Amino acids 1-10

Amino Acid Symbol	Propensity
L	1.224490
K	0.000000
M	1.530612
F	1.020408
P	0.226757
S	1.360544
T	0.765306
W	1.113173
Y	0.583090
V	0.583090

(b) Amino acids 10-20

Figure 1. Propensity of alpha helices

Amino acid	Propensity	Amino acid	Propensity
A	1.551020	L	1.224490
R	0.680272	K	0.000000
N	1.360544	M	1.530612
D	0.000000	F	1.020408
C	1.020408	P	0.226757
Q	0.000000	S	1.360544
E	0.408163	T	0.765306
G	1.061224	W	1.113173
H	0.874636	Y	0.583090
I	1.200480	V	0.583090

Question 2. Find the propensity of alpha helices manually for the sequence in question 1.

Solution. The following images show the manual calculation of propensity of α helices. In the manual manner, I have only shown the calculation of a few amino acids. The values match those obtained from the code. The same way, propensity can be calculated for all the 20 amino acids.

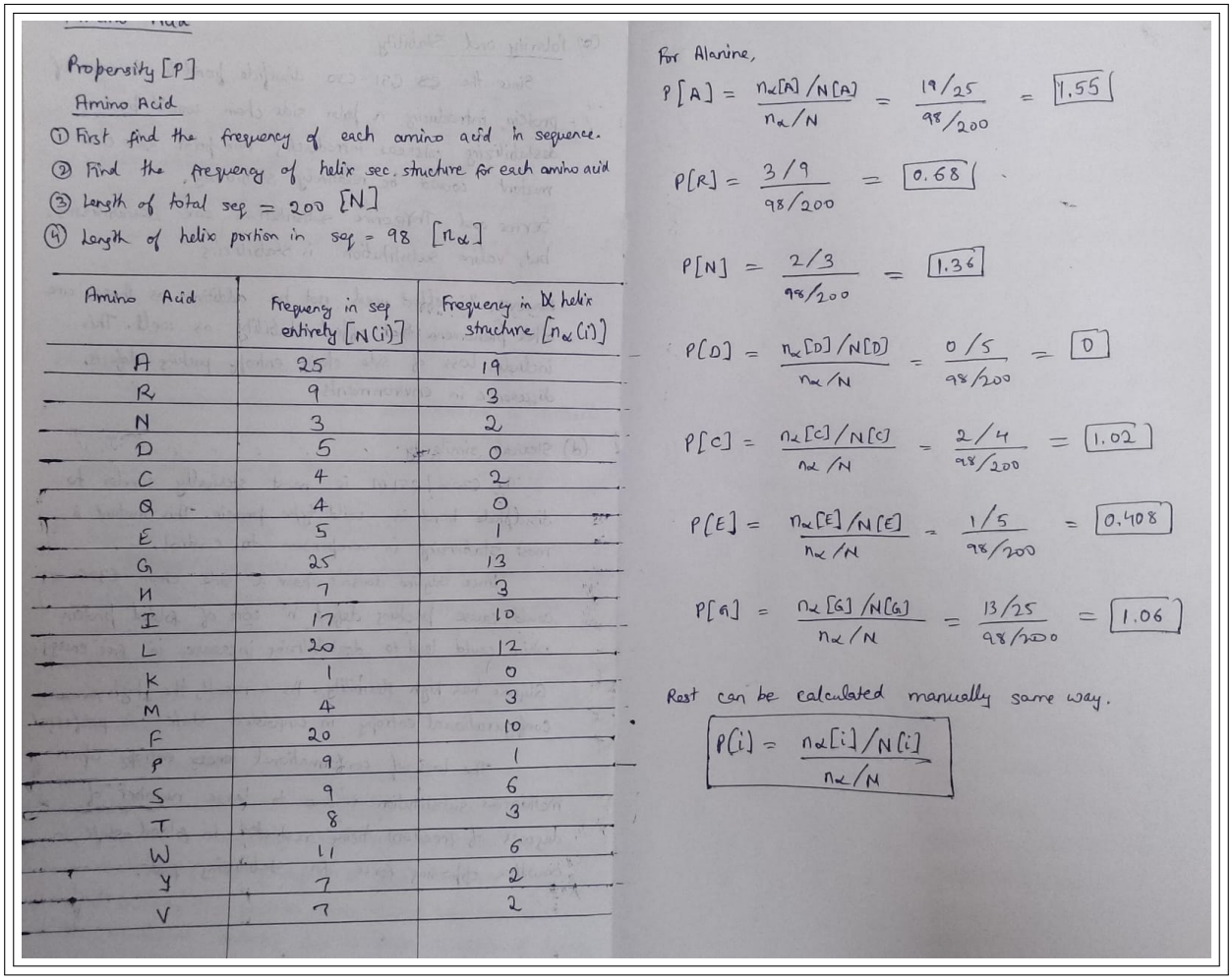

FIGURE 2. Frequency table (left) and Propensity calculation (right)

Question 3. Using the rules for helices and strands, identify the helical and strand segments in the following sequence

```
seq =
"KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCN
 DGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL"
```

Listing 3. Given sequence input

Solution. The different stages of code to compute the above are as follows:

```
seq =
"KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGR
 TPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL"

chou_alpha =
{"E": 1, "A": 1, "L": 1, "H": 1, "M": 1,
 "Q": 1, "W": 1, "V": 1, "F": 1, "K": 0.5,
 "I":0.5,"D": 0, "T": 0, "S": 0, "R": 0,
 "C": 0, "N":-1, "Y":-1, "P":-1, "G":-1}
chou_alpha_val =
{"E": 1.53, "A": 1.45, "L": 1.34, "H": 1.24, "M": 1.20,
 "Q": 1.17, "W": 1.14, "V": 1.14, "F": 1.12, "K": 1.07,
 "I": 1.00, "D": 0.98, "T": 0.82, "S": 0.79, "R": 0.79,
 "C": 0.77, "N": 0.73, "Y": 0.61, "P": 0.59, "G": 0.53}

min_amino_len = 6

chou_beta =
{"M": 1, "V": 1, "I": 1, "C": 1, "Y": 1,
 "F": 1, "Q": 1, "L": 1, "T": 1, "W": 1,
 "A":0.5,"R": 0, "G": 0, "D": 0, "K":-1,
 "S":-1, "H":-1, "N":-1, "P":-1, "E":-1}
chou_beta_val =
{"M": 1.67, "V": 1.65, "I": 1.60, "C": 1.30, "Y": 1.29,
 "F": 1.28, "Q": 1.23, "L": 1.22, "T": 1.20, "W": 1.19,
 "A": 0.97, "R": 0.90, "G": 0.81, "D": 0.80, "K": 0.74,
 "S": 0.72, "H": 0.71, "N": 0.65, "P": 0.62, "E": 0.26}

min_beta_len = 5

type_sec_alpha = ["-"]*len(seq)
type_sec_beta = ["-"]*len(seq)
```

Listing 4. Storing the Chou-Fasman parameters

Rules for identifying secondary structures

- **Rules for identifying α helices**: The values of the six parameters are given. Scan for window of 6 residues, where score ≥ 4, i.e. at least four helix formers and not more than one helix breaker; extend the length in both directions until 4-residue window has the average propensity ¡ 1; continue the search and locate all helical regions in the sequence.
- **Rules for identifying β strands**: The values of the six parameters are given. Scan for window of 5 residues, where score ≥ 3, i.e. at least three formers and not more than one strand breaker; extend the length in both directions until 3-residue window has the average propensity ¡ 1; continue the search and locate all strand regions in the sequence.
- **Conflict situation**: A region containing overlapping helical and strand assignments is considered as a helix (or strand) if average propensity of α-helix (β-strand) is greater than that of β-strand (α-helix)

```
def extend_alpha(type_sec, index):
    for i in range(index+3, 3, -1):
        # print(i, seq[i-4:i])
        extend_score = 0
        temp_seq = seq[i-4:i]
        for a in range(len(temp_seq)):
            extend_score += chou_alpha_val[temp_seq[a]]
        # print(temp_seq, extend_score)
        if extend_score >= 4:
            type_sec[i-4] = "H"
        else:
            break

    for i in range(index+min_amino_len-4, len(seq)-3):
        # print(i, seq[i:i+4])
        extend_score = 0
        temp_seq = seq[i:i+4]
        for a in range(len(temp_seq)):
            extend_score += chou_alpha_val[temp_seq[a]]
        # print(temp_seq, extend_score)
        if extend_score >= 4:
            type_sec[i+3] = "H"
        else:
            break

    return type_sec

def extend_beta(type_sec, index):
    for i in range(index+3, 2, -1):
        # print(i, seq[i-4:i])
        extend_score = 0
        temp_seq = seq[i-3:i]
        for a in range(len(temp_seq)):
            extend_score += chou_beta_val[temp_seq[a]]
        # print(temp_seq, extend_score)
        if extend_score >= 3:
            type_sec[i-3] = "B"
        else:
            break

    for i in range(index+min_beta_len-3, len(seq)-2):
        # print(i, seq[i:i+4])
        extend_score = 0
        temp_seq = seq[i:i+3]
        for a in range(len(temp_seq)):
            extend_score += chou_beta_val[temp_seq[a]]
        # print(temp_seq, extend_score)
        if extend_score >= 3:
            type_sec[i+2] = "B"
        else:
            break

    return type_sec
```

LISTING 5. Extend functions for α helices and β sheets

```
while(j<len(seq)-min_amino_len+1):
    temp_seq = seq[j:j+min_amino_len]
    temp_score = 0
    for k in range(len(temp_seq)):
        temp_score += chou_alpha[temp_seq[k]]
    if temp_score >= 4:
        for l in range(j, j+min_amino_len):
            type_sec_alpha[l] = "H"
        type_sec_alpha = extend_alpha(type_sec_alpha, j)
        j += min_amino_len
        continue
    j += 1
while(j<len(seq)-min_beta_len+1):
    temp_seq = seq[j:j+min_beta_len]
    temp_score = 0
    for k in range(len(temp_seq)):
        temp_score += chou_beta[temp_seq[k]]
    if temp_score >= 3:
        for l in range(j, j+min_beta_len):
            type_sec_beta[l] = "B"
        type_sec_beta = extend_beta(type_sec_beta, j)
        j += min_beta_len
        continue
    j += 1

type_final = ""
segment = ""
start_segment = 0
for i in range(len(seq)):
    if type_sec_alpha[i] == "H" and type_sec_beta[i] == "B":
        segment = segment + seq[i]
    else:
        score_a = 0
        score_b = 0
        for k in range(len(segment)):
            score_a += chou_alpha_val[segment[k]]
            score_b += chou_beta_val[segment[k]]
        if score_a >= score_b:
            type_final = type_final + "H"*len(segment)
        else:
            type_final = type_final + "B"*len(segment)
        segment = ""

        if type_sec_alpha[i] == "H" and type_sec_beta[i] != "B":
            type_final = type_final + "H"

        elif type_sec_alpha[i] != "H" and type_sec_beta[i] == "B":
            type_final = type_final + "B"

        else:
            type_final = type_final + "-"

print(seq)
print(type_final)
```

Listing 6. Applying individual rules and resolving conflict situations

First line of code output is the given sequence. Second line is the positions of α helices obeying only rules of α helices. Third line is the positions of β helices obeying only rules of β strands. Fourth line is obtained after addressing conflicts.

α **helix regions**	**positions**
ELAAAMKRH	(7,8,9,10,11,12,13,14,15)
KFESNF	(33,34,35,36,37,38)
MNAWVAWRN	(105,106,107,108,109,110,111,112,113)

β **helix regions**	**positions**
KVFGRC	(1,2,3,4,5,6)
NWVCAA	(27,28,29,30,31,32)
TDYGILQIN	(51,52,53,54,55,56,57,58,59)
GTDVQAWIRGCRL	(117,118,119,120,121,122,123,124,125,126,127,128,129)

Question 4. Verify one of the helical and strand segments, manually.

Solution. The following images show the manual verification of one of α helix and beta strand segments. In the manual manner, I have only shown the calculation of one of each. The values match those obtained from the code.

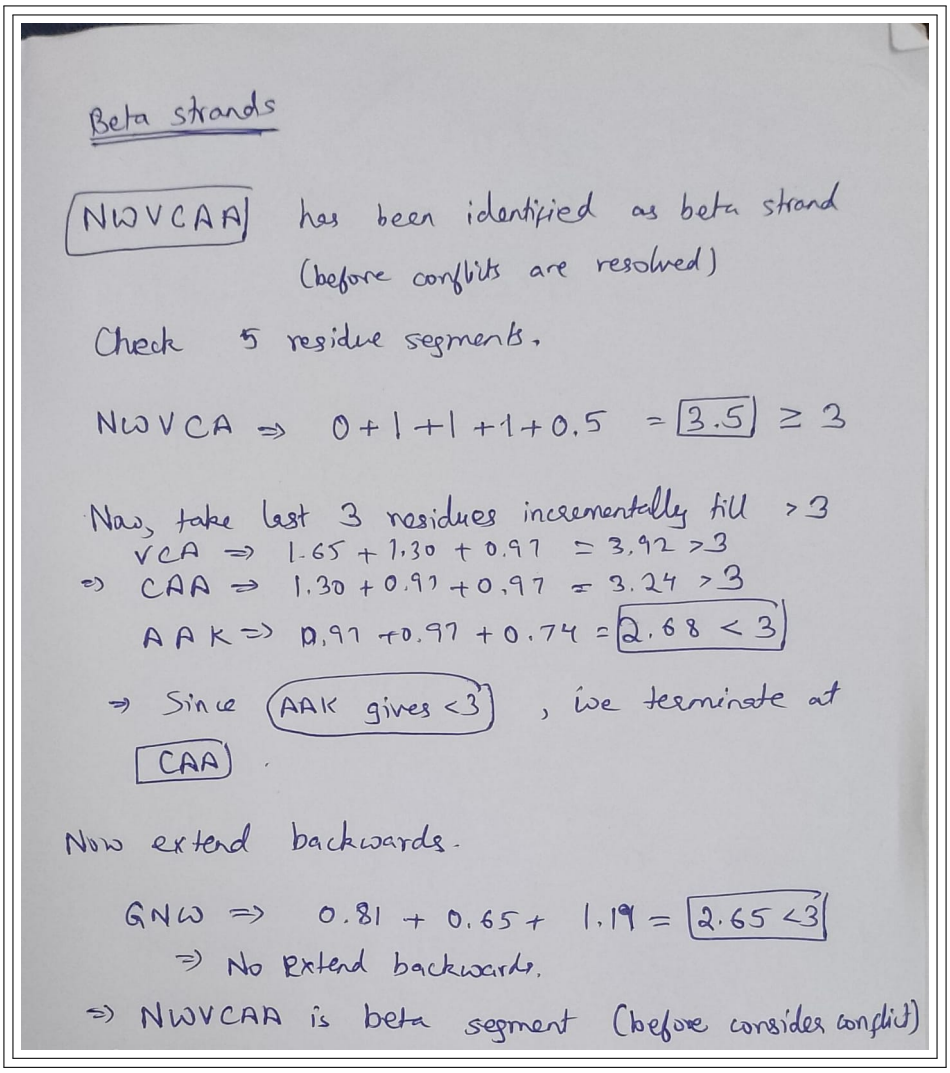

FIGURE 3. Verification of β strand

Helical strands

Position (5-15) was classified helix (before the conflict for α and β segments were resolved).

⇒ RCELAAAMKRH

First checking 6 segment RCELAA

RCELAA ⇒ 0+0+1+1+1+1 = 4 (Using Chou-Fasman parameters)

Now, take last 4 incrementally till >4.

⇒ ELAA ⇒ 1.53 + 1.34 + 1.45 + 1.45 = 5.77 > 4

LAAA ⇒ 1.34 + 1.45 + 1.45 + 1.45 = 5.69 > 4

AAAM ⇒ 1.45 + 1.45 + 1.45 + 1.20 = 5.55 > 4

AAMK ⇒ 1.45 + 1.45 + 1.20 + 1.07 = 5.17 > 4

AMKR ⇒ 1.45 + 1.20 + 1.07 + 0.79 = 4.51 > 4

MKRH ⇒ 1.20 + 1.07 + 0.79 + 1.24 = 4.3 > 4

KRHG ⇒ 1.07 + 0.79 + 1.24 + 0.53 = 3.63 < 4

Since KRHG gives < 4, we terminate at MKRH.

⇒ RCELAAAMKRH is helical segment (without considering conflicts)

[Extending backward

GRCE ⇒ 0.53 + 0.79 + 0.77 + 1.53 = 3.62 < 4

⇒ No backward extension.

FIGURE 4. Verification of α helix