Anirudh Rao
BE21B004

# BT3040 – Bioinformatics

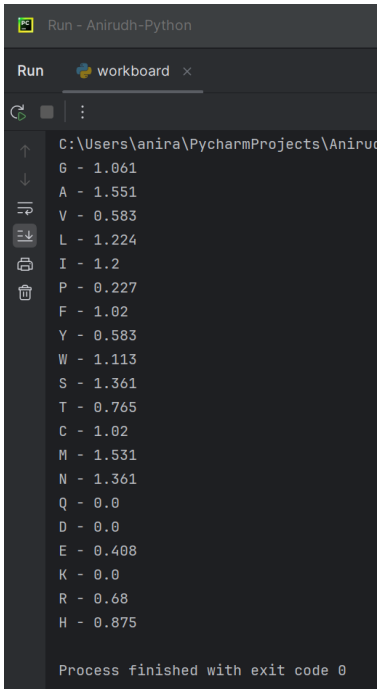
## Practical 11

### 1

The Python code to find the propensity of alpha helices in the given sequence is shown below:

```
def propensity(seq, secondary):

    residues = ["G", "A", "V", "L", "I", "P", "F", "Y", "W", "S", "T", "C", "M", "N",
"Q", "D", "E", "K", "R", "H"]
    helix = {residue:0 for residue in residues}
    frac_helix = secondary.count("H")/len(seq)

    for i in range(len(seq)):
        if structure[i] == "H":
            helix[seq[i]] += 1

    helix = {residue:helix[residue]/seq.count(residue) for residue in residues}

    propensities = {residue:round(helix[residue]/frac_helix,3) for residue in residues}

    return propensities


sequence =
"LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQYGMGIPPLHDGGWWLMAGLFMTLSLGSWWIRVYSRAR
ALGLGTHIAWNFAAAIFFVLCIGCIHPTLVGSWSEGVPFGIWPHIDWLTAFSIRYGNFYYCPWHGFSIGFAYGCGLLFAAHGATILA
VARFGGDREIEQITDRGTAVERAALFW"
structure =
"XHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHXXHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHH
XXXXXXXXXXXXXXXXXXXXXXXXXXX"
for residue, prop in propensity(sequence,structure).items():
    print(f"{residue} - {prop}")
```

The output of this is:



2

Total no. of residues in helix = 98

Total no. of residues in sequence = 200

Fraction of residues in helix $= \frac{98}{200} = 0.49$

| Residue | n (Count in helix) | N (Overall count) | $\frac{n}{0.49N}$ Propensity |
|---|---|---|---|
| G | 13 | 25 | 1.061 |
| A | 19 | 25 | 1.551 |
| V | 2 | 7 | 0.583 |
| L | 12 | 20 | 1.224 |
| I | 10 | 17 | 1.2 |
| P | 1 | 9 | 0.227 |
| F | 10 | 20 | 1.02 |
| Y | 2 | 7 | 0.583 |
| W | 6 | 11 | 1.113 |
| S | 6 | 9 | 1.361 |
| T | 3 | 8 | 0.765 |
| C | 2 | 4 | 1.02 |
| M | 3 | 4 | 1.531 |
| N | 2 | 3 | 1.361 |
| Q | 0 | 4 | 0 |
| D | 0 | 5 | 0 |
| E | 1 | 5 | 0.408 |
| K | 0 | 1 | 0 |
| R | 3 | 9 | 0.68 |
| H | 3 | 7 | 0.875 |

3

The Python code to identify the helical and strand segments in the given sequence is shown below:

```
import numpy as np

sequence =
"KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGRTPGSRNLCNIPCSALLSS
DITASVNCAKKIVSDGNGMNAWVAWRNRCKGTDVQAWIRGCRL"

helix_params = {residue:1 for residue in ["E","A","L","H","M","Q","W","V","F"]}
helix_params.update({residue:0.5 for residue in ["K","I"]})
helix_params.update({residue:0 for residue in ["D","T","S","R","C"]})
helix_params.update({residue:-1 for residue in ["N","Y","P","G"]})
```

```
helix_propensities = {
    '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}

sheet_params = {residue:1 for residue in ["M","V","I","C","Y","F","Q","L","T","W"]}
sheet_params.update({residue:0.5 for residue in ["A"]})
sheet_params.update({residue:0 for residue in ["R","G","D"]})
sheet_params.update({residue:-1 for residue in ["K","S","H","N","P","E"]})

sheet_propensities = {
    '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
}

helices = {}
i = 0
while i < len(sequence)-6+1:
    window = sequence[i:i+6]
    score = np.sum([helix_params[residue] for residue in window])
    if score >= 4:
        j = i-1
        while np.sum([helix_propensities[residue] for residue in
sequence[j]+window[:3]]) >= 4 and j > 0:
            window = sequence[j] + window
            j = j-1
        k = i+6
        while np.sum([helix_propensities[residue] for residue in window[-
3:]+sequence[k]]) >= 4 and k < len(sequence):
            window = window + sequence[k]
            k = k+1
```

```
        helices[sequence.find(window)+1] = (window, np.sum([helix_propensities[residue]
for residue in window]))
        i = i + len(window)
    else:
        i = i + 1

sheets = {}
i = 0
while i < len(sequence)-5+1:
    window = sequence[i:i+5]
    score = np.sum([sheet_params[residue] for residue in window])
    if score >= 3:
        j = i-1
        while np.sum([sheet_propensities[residue] for residue in
sequence[j]+window[:2]]) >= 3:
            window = sequence[j] + window
            j = j-1
            if j < 0:
                break
        k = i+5
        while np.sum([sheet_propensities[residue] for residue in window[-
2:]+sequence[k]]) >= 3:
            window = window + sequence[k]
            k = k+1
            if k >= len(sequence):
                break
        sheets[sequence.find(window)+1] = (window, np.sum([sheet_propensities[residue]
for residue in window]))
        i = i + len(window)
    else:
        i = i + 1

for helix_pos, (helix_seq, helix_prop) in list(helices.items()):
    for sheet_pos, (sheet_seq, sheet_prop) in list(sheets.items()):
        common =
set(range(helix_pos,helix_pos+len(helix_seq))).intersection(set(range(sheet_pos,sheet_p
os+len(sheet_seq))))
        if len(common) != 0:
            matching = sequence[min(common)-1:max(common)]
            if np.sum([helix_propensities[residue] for residue in matching]) <
np.sum([sheet_propensities[residue] for residue in matching]):
                new_helix = helix_seq.replace(matching,"")
                new_prop = np.sum([helix_propensities[residue] for residue in
new_helix])
                new_pos = sequence.find(new_helix) + 1
                del helices[helix_pos]
                if len(new_helix) > 0:
                    helices[new_pos] = (new_helix,new_prop)
            elif np.sum([helix_propensities[residue] for residue in matching]) >
np.sum([sheet_propensities[residue] for residue in matching]):
                new_sheet = sheet_seq.replace(matching,"")
                new_prop = np.sum([sheet_propensities[residue] for residue in
new_sheet])
                new_pos = sequence.find(new_sheet) + 1
                del sheets[sheet_pos]
                if len(new_sheet) > 0:
                    sheets[new_pos] = (new_sheet,new_prop)

helices = dict(sorted(helices.items()))
sheets = dict(sorted(sheets.items()))

print("Helices")
for helix_pos, (helix_seq, helix_prop) in list(helices.items()):
    print(f"{helix_seq} at {helix_pos} with propensity {helix_prop:.3f}")
print("\nSheets")
for sheet_pos, (sheet_seq, sheet_prop) in list(sheets.items()):
    print(f"{sheet_seq} at {sheet_pos} with propensity {sheet_prop:.3f}")
```

The output of this is:

```
Run - Anirudh-Python
Run    workboard ×

C:\Users\anira\PycharmProjects\Anirudh-Python\.
Helices
ELAAAMKRH at 7 with propensity 11.520
KFESNF at 33 with propensity 6.360
MNAWVAWRN at 105 with propensity 9.770

Sheets
KVFGRC at 1 with propensity 6.680
NWVCAA at 27 with propensity 6.730
TDYGILQIN at 51 with propensity 10.400
GTDVQAWIRGCRL at 117 with propensity 14.580

Process finished with exit code 0
```



**Helix**

At position 105, MNAWVA has a score 4.

Extending to the left, score(GMNA) = 3.91 < 4

Extending to the right, score(WAW) = 4.87 > 4

score(VAWR) = 4.52 > 4

score(AWRN) = 4.11 > 4

score(WRNR) = 3.45 < 4

⇒ stop extending

∴ final helix is MNAWVAWRN

**Sheet**

At position 2, VFGRC has a score 3

Extending to the left, score(KVF) = 3.67 > 3

Extending to the right, score(RCE) = 2.46 < 3

∴ final sheet is KVFGRC