Computational Biology Lab

Lab 1

1. Go to PDBTM database (http://pdbtm.enzim.hu/)
2. Download alpha helical membrane protein sequences (TMH) in FASTA format
3. Obtain non-redundant sequences using CD-HIT software (40%)
4. Repeat steps 2 and 3 for beta barrel membrane proteins (TMB)
5. Compute and tabulate the overall amino acid composition in TMH and TMB (20 values each).

TMH & TMB

Amino Acid	Composition (TMH)	Amino Acid	Composition (TMB)
A	0.084	A	0.076
C	0.015	C	0.007
D	0.04	D	0.064
E	0.048	E	0.048
F	0.056	F	0.042
G	0.069	G	0.089
H	0.019	H	0.015
I	0.068	I	0.045
K	0.045	K	0.05
L	0.118	L	0.081
M	0.027	M	0.017
N	0.037	N	0.061
P	0.042	P	0.038
Q	0.034	Q	0.043
R	0.046	R	0.05
S	0.062	S	0.076
T	0.054	T	0.067
V	0.077	V	0.064
W	0.018	W	0.017
Y	0.036	Y	0.049

The code that gives the above table is

```
# !pip install biopython
from Bio import SeqIO
from Bio.SeqUtils.ProtParam import ProteinAnalysis
import pandas as pd
import statistics
```

```python
# For Alpha
file = "C:\\Users\\shree\\Desktop\\ACADS\\Comp Bio Lab\\alpha_40.txt"
Big_sequence = ''
df = pd.DataFrame()

sequences = SeqIO.parse(open(file),'fasta')
for file in sequences:
    sequence = str(file.seq)
    x = ProteinAnalysis(sequence)
    df_small = pd.DataFrame([dict(x.get_amino_acids_percent())])
    df = pd.concat([df, df_small], ignore_index=True)
    Big_sequence += sequence
X = ProteinAnalysis(Big_sequence)

df_Big_a = pd.DataFrame([dict(X.get_amino_acids_percent())])
df_alpha = pd.concat([df, df_Big_a], ignore_index=True)
df_individual_alpha = df_alpha[:-1]

# For Beta
file = "C:\\Users\\shree\\Desktop\\ACADS\\Comp Bio Lab\\beta_40.txt"
Big_sequence = ''
df = pd.DataFrame()

sequences = SeqIO.parse(open(file),'fasta')
for file in  sequences:
    sequence = str(file.seq)
    x = ProteinAnalysis(sequence)
    df_small = pd.DataFrame([dict(x.get_amino_acids_percent())])
    df = pd.concat([df, df_small], ignore_index=True)
    Big_sequence += sequence
X = ProteinAnalysis(Big_sequence)

df_Big_b = pd.DataFrame([dict(X.get_amino_acids_percent())])
df_beta = pd.concat([df, df_Big_b], ignore_index=True)
df_individual_beta = df_beta[:-1]

print('The overall amino acid percentage in TMH is')
print(round(df_Big_a,3))

print('The overall amino acid percentage in TMB is')
print(round(df_Big_b,3))
```

6. Identify the amino acids, which are important for discrimination (use Fisher discriminant ratio). Ref:Bioinformatics Vol. 21, pages 4223–4229; https://sthalles.github.io/fisher-linear-discriminant/]. Include the data in the previous table.

The FDR is calculated using the formula given in the paper and attached to the above table

AA	Composition (TMH)	AA	Composition (TMB)	FDR Values
A	0.084	A	0.076	0.014
C	0.015	C	0.007	0.15
D	0.04	D	0.064	0.633
E	0.048	E	0.048	0
F	0.056	F	0.042	0.155
G	0.069	G	0.089	0.183
H	0.019	H	0.015	0.041
I	0.068	I	0.045	0.434
K	0.045	K	0.05	0
L	0.118	L	0.081	0.475
M	0.027	M	0.017	0.258
N	0.037	N	0.061	0.563
P	0.042	P	0.038	0.028
Q	0.034	Q	0.043	0.171
R	0.046	R	0.05	0.007
S	0.062	S	0.076	0.21
T	0.054	T	0.067	0.16
V	0.077	V	0.064	0.115
W	0.018	W	0.017	0.004
Y	0.036	Y	0.049	0.158

The code used to calculate is given below

```
# FDR Calculations
mean_alpha = pd.DataFrame([dict(df_individual_alpha.mean())])
std_alpha = pd.DataFrame([dict(df_individual_alpha.std())])
mean_beta = pd.DataFrame([dict(df_individual_beta.mean())])
std_beta = pd.DataFrame([dict(df_individual_beta.std())])

FDR = pd.concat([mean_alpha, mean_beta, std_alpha, std_beta], ignore_index=True)

FDR_list = []
for column in FDR:
    ma = FDR[column][0]
    mb = FDR[column][1]
    sa = FDR[column][2]
    sb = FDR[column][3]

    FDR_val = (ma-mb)**2/((sa**2)+(sb**2))
    FDR_list.append(round(FDR_val,3)) #Rounded off to 3rd decimal place

FDR.loc[4] = FDR_list
round(FDR,3)
FDR["Index"] = ["Mean alpha", "Mean Beta", "Std Alpha", "Std Beta", "FDR Value"]
FDR.insert(0,"Index", FDR.pop("Index"))

FDR = round(FDR,3)
print(FDR)
```

7. For each sequence in TMH
(a) Compute the composition
(b) Compare with overall composition of TMH and compute the absolute deviation and total for the 20 residues
 σ(TMH) = Σ |comp(x)-comp(TMH)|
(c) Compare with overall composition of TMB and compute the absolute deviation and total for the 20 residues
 σ(TMB) = Σ |comp(x)-comp(TMB)|
(d) If σ(TMH) < σ(TMB), the protein is TMH Otherwise, it is TMB
(e) Correctly predicted TMH are True Positives (TP)
(f) Wrongly predicted as TMB are False Negatives (FN)

8. Repeat the same with all TMB proteins. In this case,
(e) Correctly predicted TMB are True Negatives (TN)
(f) Wrongly predicted as TMH are False Positives (FP)

9. Compute sensitivity, specificity and accuracy
Sensitivity = TP/(TP+FN)
Specificity = TN/(TN+FP)
Accuracy = (TP+TN)/(TP+TN+FP+FN)

The code for performing the above three questions is given below, along with the output

```
# TMH Calculations
sigmaTHM_alpha = []
sigmaTHB_alpha = []
for i in range(len(df_individual_alpha)):
    comp1 = 0
    comp2 = 0
    for j in range(20):
        comp1 += abs(df_individual_alpha.iloc[i,j] - df_Big_a.iloc[0,j])
        comp2 += abs(df_individual_alpha.iloc[i,j] - df_Big_b.iloc[0,j])

    sigmaTHM_alpha.append(comp1)
    sigmaTHB_alpha.append(comp2)

TP = 0
FN = 0
delta = 0
for k in range(len(sigmaTHM_alpha)):
    if (sigmaTHM_alpha[k] + delta) < sigmaTHB_alpha[k]:
        TP += 1
    else:
        FN += 1

# TMB Calculations
sigmaTHB_beta = []
sigmaTHM_beta = []

for i in range(len(df_individual_beta)):
    comp1 = 0
    comp2 = 0
    for j in range(20):
        comp1 += abs(df_individual_beta.iloc[i,j] - df_Big_b.iloc[0,j])
        comp2 += abs(df_individual_beta.iloc[i,j] - df_Big_a.iloc[0,j])

    sigmaTHB_beta.append(comp1)
    sigmaTHM_beta.append(comp2)

TN = 0
FP = 0
delta = 0
for k in range(len(sigmaTHB_beta)):
    if (sigmaTHB_beta[k] + delta) < sigmaTHM_beta[k]:
        TN += 1
    else:
        FP += 1

# Computations
Sensitivity = round(TP/(TP+FN),3)
Specificity = round(TN/(TN+FP),3)
Accuracy = round((TP+TN)/(TP+TN+FP+FN),3)

print(f'Sensitivity = {Sensitivity}')
print(f'Specificity = {Specificity}')
print(f'Accuracy = {Accuracy}')
```

The output of which is

Sensitivity = 0.803
Specificity = 0.835
Accuracy = 0.806
10. Take 50% of TMH and 50% of TMB to compute the composition (step 5). For the remaining set of proteins follow steps 7 to 9 to assess the performance.

11. Change the split in question 10 to 30%, 40%, 60% and 70% and repeat the computation. Tabulate the data. (Optional)

The code to calculate the above two questions is given below, change the p value based on the percentage

```python
# p%
p = 50
k1 = len(df_alpha)
k2 = len(df_beta)

df_alpha_1 = df_individual_alpha[int(k1*p/100):]
df_beta_1 = df_individual_beta[int(k2*p/100):]

df_Big_a, df_Big_b = split(p)


# TMH Calculations
sigmaTHM_alpha = []
sigmaTHB_alpha = []
for i in range(len(df_alpha_1)):
    comp1 = 0
    comp2 = 0
    for j in range(20):
        comp1 += abs(df_alpha_1.iloc[i,j] - df_Big_a.iloc[0,j])
        comp2 += abs(df_alpha_1.iloc[i,j] - df_Big_b.iloc[0,j])

    sigmaTHM_alpha.append(comp1)
    sigmaTHB_alpha.append(comp2)

TP = 0
FN = 0
for k in range(len(sigmaTHM_alpha)):
    if sigmaTHM_alpha[k] < sigmaTHB_alpha[k]:
        TP += 1
    else:
        FN += 1

# TMB Calculations
sigmaTHB_beta = []
sigmaTHM_beta = []
```

```
for i in range(len(df_beta_1)):
    comp1 = 0
    comp2 = 0
    for j in range(20):
        comp1 += abs(df_beta_1.iloc[i,j] - df_Big_b.iloc[0,j])
        comp2 += abs(df_beta_1.iloc[i,j] - df_Big_a.iloc[0,j])

    sigmaTHB_beta.append(comp1)
    sigmaTHM_beta.append(comp2)

TN = 0
FP = 0
for k in range(len(sigmaTHB_beta)):
    if sigmaTHB_beta[k] < sigmaTHM_beta[k]:
        TN += 1
    else:
        FP += 1

# Computations
Sensitivity = round(TP/(TP+FN),3)
Specificity = round(TN/(TN+FP),3)
Accuracy = round((TP+TN)/(TP+TN+FP+FN),3)

print(f'Split = {p} %')
print(f'Sensitivity = {Sensitivity}')
print(f'Specificity = {Specificity}')
print(f'Accuracy = {Accuracy}')
```

The output with respect to the percentage split if given below

Percentage split	**30%**	**40%**	**50%**	**60%**	**70%**
Sensitivity	0.813	0.827	0.839	0.834	0.824
Specificity	0.853	0.829	0.818	0.794	0.74
Accuracy	0.817	0.827	0.837	0.83	0.816

12. In 7d include a deviation δ (E.g., σ(TMH) + 0.5) estimate the sensitivity, specificity and accuracy. (Optional)

Changing the delta value to 0.5 in the code for 7d, we get the following values
Sensitivity = 0.0
Specificity = 0.0
Accuracy = 0.0
* The code files can be found here.