import bct

import mat73

import pandas as pd

import numpy as np

from scipy import stats

data = mat73.loadmat('subject\_data.mat')['all\_matrices']

groups = pd.read\_excel('subj\_information\_4\_students.xlsx')

# Dimensions of data

n\_len\_x = len(data)

n\_len\_y = len(data[0])

networks = len(data[0][0])

subjects = len(data[0][0][0])

status1 = 'Status 1'

status2 = 'Status 2'

status3 = 'Status 3'

status4 = 'Status 4'

# note: transitivity and assortativity are GLOBAL features

dgs = 'degrees'

sth = 'strength'

clc = 'clustering\_coefficient'

btw = 'betweenness centrality'

eig = 'eigenvector centrality'

features = [dgs, sth, clc, btw, eig]

def feature\_comparison(feature):

group1 = []

group2 = []

group3 = []

group4 = []

for network in range(0, networks):

for subject in range(0, subjects):

subject\_data = []

# get all the data for an individual subject

for n\_x in range(0, n\_len\_x):

for n\_y in range(0, n\_len\_y):

subject\_data.append(data[n\_x][n\_y][network][subject])

# reshape to be 113x113 numpy array

subject\_data = np.asarray(subject\_data)

subject\_data = np.reshape(subject\_data, (-1, n\_len\_x))

# get feature - use weighted network version

if feature == dgs:

subject\_data = bct.degrees\_und(subject\_data)

elif feature == sth:

subject\_data = bct.strengths\_und(subject\_data)

elif feature == clc:

subject\_data = bct.clustering\_coef\_wu(subject\_data)

elif feature == btw:

subject\_data = bct.betweenness\_wei(subject\_data)

elif feature == eig:

subject\_data = bct.eigenvector\_centrality\_und(subject\_data)

# append subject feature data to group's overall data

status = groups['Status'][groups['Subj\_ID'] == subject+1].item()

if status == status1:

group1.append(subject\_data)

elif status == status2:

group2.append(subject\_data)

elif status == status3:

group3.append(subject\_data)

else:

group4.append(subject\_data)

# get average features to compare between groups

feature\_len = n\_len\_x

avg1 = np.zeros((feature\_len, 1))

avg2 = np.zeros((feature\_len, 1))

avg3 = np.zeros((feature\_len, 1))

avg4 = np.zeros((feature\_len, 1))

for group, avg in zip([group1, group2, group3, group4], [avg1, avg2, avg3, avg4]):

for i in range(0, feature\_len):

for subject in group:

avg[i][0] += subject[i]

# perform t-tests for different groups

p1 = stats.ttest\_ind(avg1, avg2).pvalue

p2 = stats.ttest\_ind(avg2, avg3).pvalue

p3 = stats.ttest\_ind(avg3, avg4).pvalue

p\_value = 0.05

# perform Bonferroni correction if necessary

p\_value = p\_value / feature\_len

# report the results

print(f'Feature to be tested is: {feature}.')

print(f'Network to be tested is: {network + 1}.')

print(f'P value for comparison between groups 1 and 2 is {p1}.')

if p\_value > p1:

print('Means DO differ between groups 1 and 2.')

else:

print('Means DO NOT differ between groups 1 and 2.')

print(f'P value for comparison between groups 2 and 3 is {p2}.')

if p\_value > p2:

print('Means DO differ between groups 2 and 3.')

else:

print('Means DO NOT differ between groups 2 and 3.')

print(f'P value for comparison between groups 3 and 4 is {p3}.')

if p\_value > p3:

print('Means DO differ between groups 3 and 4.')

else:

print('Means DO NOT differ between groups 3 and 4.')

print('\n')

def main():

for feature in features:

feature\_comparison(feature)

if \_\_name\_\_ == "\_\_main\_\_":

main()