Korina Zaromytidou

Please find at the botom of the notebook my answers for Tasks 1 & 2 for Assigment 3 Five Personality Traits.

The notebook is based on the Week 7.2a_Five_Personality notbook.

I had to run this in colab, as my personal laptop was very slow, so I had to ajbust the code to import the dataset files.

Code provided.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

```
from google.colab import files
uploaded = files.upload()
```

Choose files data-small.csv

- **data-small.csv**(text/csv) - 10483304 bytes, last modified: 03/06/2023 - 100% done

```
Saving data-small.csv to data-small.csv
```

```
import pandas as pd

data = pd.read_csv('data-small.csv')
pd.options.display.max_columns = 150
```

```
#import pandas as pd

#data = pd.read_csv('data/data-small.csv')
#pd.options.display.max_columns = 150
```

```
print('Number of participants: ', len(data))
data.head()
```

```
Number of participants:  50000
```

	Unnamed: 0	EXT1	EXT2	EXT3	EXT4	EXT5	EXT6	EXT7	EXT8	EXT9	EXT10	EST1	EST2	EST3	EST4	EST5
0	549499	2.0	3.0	2.0	2.0	5.0	1.0	3.0	1.0	5.0	2.0	4.0	2.0	3.0	2.0	5.0
1	811367	2.0	3.0	2.0	4.0	3.0	3.0	2.0	4.0	2.0	4.0	4.0	4.0	4.0	2.0	4.0
2	450151	4.0	1.0	5.0	1.0	5.0	1.0	5.0	5.0	5.0	2.0	1.0	5.0	2.0	3.0	5.0
3	919073	1.0	4.0	3.0	5.0	1.0	2.0	1.0	5.0	1.0	4.0	2.0	4.0	2.0	3.0	2.0
4	894414	3.0	1.0	4.0	3.0	0.0	3.0	3.0	3.0	3.0	4.0	3.0	3.0	3.0	3.0	3.0

```
print('Is there any missing value? ', data.isnull().values.any())
print('How many missing values? ', data.isnull().values.sum())
data.dropna(inplace=True)
print('Number of participants after eliminating missing values: ', len(data))
```

```
Is there any missing value?  False
How many missing values?  0
Number of participants after eliminating missing values:  49906
```

```
# Defining a function to visualize the questions and answers distribution
def vis_questions(groupname, questions, color):
    plt.figure(figsize=(40,60))
    for i in range(1, 11):
        plt.subplot(10,5,i)
        plt.hist(data[groupname[i-1]], bins=14, color= color, alpha=.5)
        plt.title(questions[groupname[i-1]], fontsize=18)
```

```
from google.colab import files

uploaded = files.upload()
```

Choose files questions.json

- **questions.json**(application/json) - 1983 bytes, last modified: 03/06/2023 - 100% done

```
Saving questions.json to questions.json
```

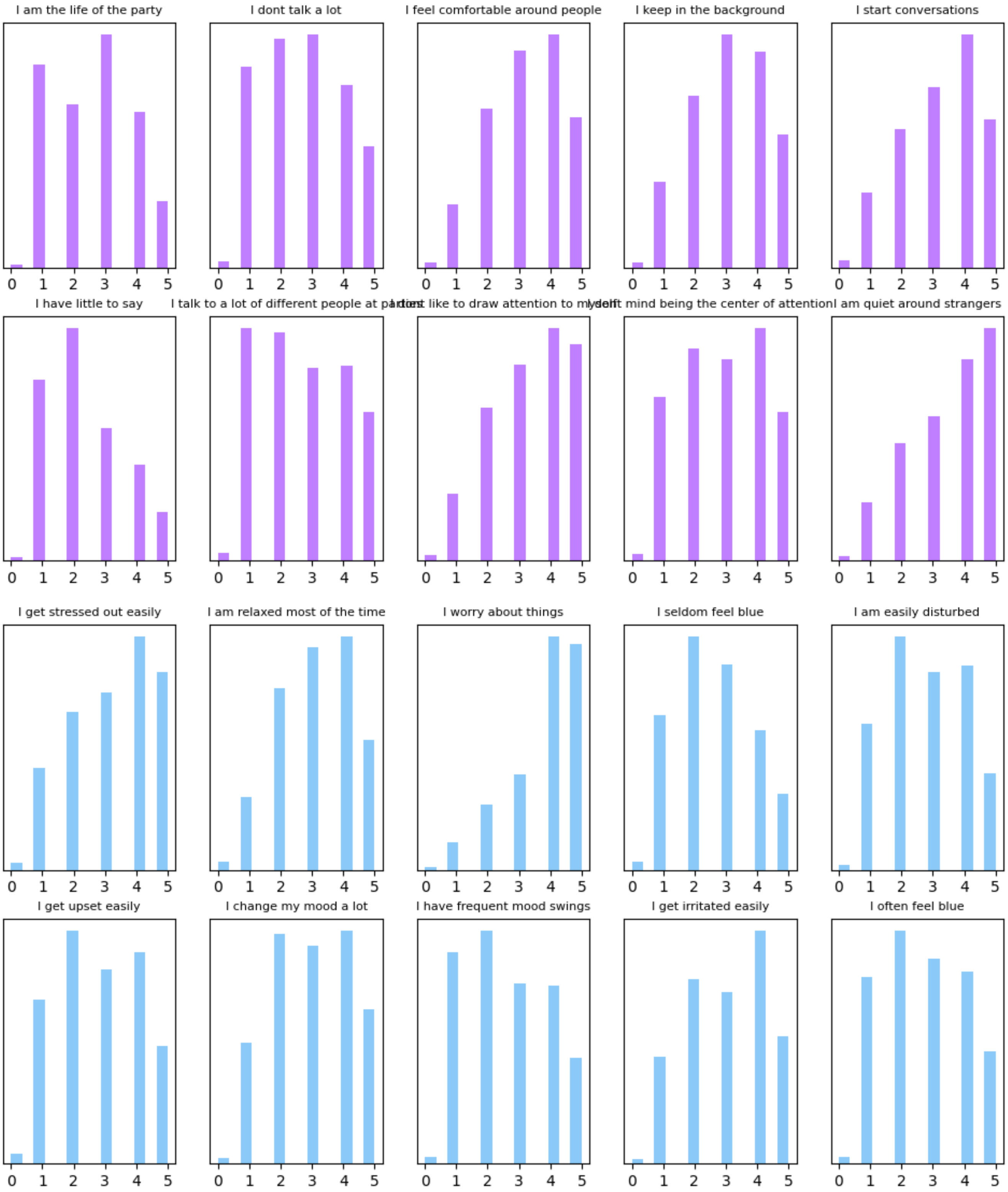
```
#Load in questions databank
import json
with open("questions.json", "r") as fp:
    questions = json.load(fp)
traits = list(questions.keys())
colours = plt.colormaps.get("rainbow")
```

```
#Helper functions
def keys_for_trait(trait):
    return list(questions[trait].keys())

def questions_for_trait(trait):
    return list(questions[trait].values())

#Plot histogram for responses to each question
def vis_questions(trait, color):
    fig, ax = plt.subplots(2,5,figsize=(12,2))
    plt.subplots_adjust(bottom=0, top=2.5)
    qs = questions_for_trait(trait)
    codes = keys_for_trait(trait)
    for i in range(10):
        plot = ax[int(np.floor(i/5)),i%5]
        plot.hist(data[codes[i]], bins=14, color= color, alpha=.5)
        plot.set_title(qs[i], fontsize=8)
        plot.set_yticks([])
        plot.set_xticks(np.arange(0,6))
```

```
#Plot all questions
for i,t in enumerate(traits):
    vis_questions(t, colours(i/5))
```


I am the life of the party
I dont talk a lot
I feel comfortable around people
I keep in the background
I start conversations
I have little to say
I talk to a lot of different people at parties
I dont like to draw attention to myself
I dont mind being the center of attention
I am quiet around strangers
I get stressed out easily
I am relaxed most of the time
I worry about things
I seldom feel blue
I am easily disturbed
I get upset easily
I change my mood a lot
I have frequent mood swings
I get irritated easily
I often feel blue
0 1 2 3 4 5


I feel little concern for others
I am interested in people
I insult people
I sympathize with others feelings
I am not interested in other peoples problems
I have a soft heart
I am not really interested in others
I take time out for others
I feel others emotions
I make people feel at ease
0 1 2 3 4 5

```
# For ease of calculation lets scale all the values between 0-1 and take a sample of 5000
from sklearn.preprocessing import MinMaxScaler

df = data.drop('country', axis=1)
columns = list(df.columns)

scaler = MinMaxScaler(feature_range=(0,1))
df = scaler.fit_transform(df)
df = pd.DataFrame(df, columns=columns)
df_sample = df[:5000]
```

```
%pip install yellowbrick
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: yellowbrick in /usr/local/lib/python3.10/dist-packages (1.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from yellowb
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-packages (from yellow
Requirement already satisfied: cycler>=0.10.0 in /usr/local/lib/python3.10/dist-packages (from yello
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from mat
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from ma
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from ma
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matp
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplo
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from mat
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-date
        0 1 2 3 4 5   0 1 2 3 4 5   0 1 2 3 4 5   0 1 2 3 4 5   0 1 2 3 4 5
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
# Visualize the elbow
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

kmeans = KMeans()
visualizer = KElbowVisualizer(kmeans, k=(2,15))
visualizer.fit(df_sample)
visualizer.poof()
```

*K- means *

```
# Creating K-means Cluster Model
from sklearn.cluster import KMeans

# Use the unscaled data but without the country column
df_model = data.drop('country', axis=1)

# Define 5 clusters and fit my model
kmeans = KMeans(n_clusters=5)
k_fit = kmeans.fit(df_model)
```

```
# Predicting the Clusters
pd.options.display.max_columns = 10
predictions = k_fit.labels_
df_model['Clusters'] = predictions
df_model.head()
```

	Unnamed: 0	EXT1	EXT2	EXT3	EXT4	...	OPN7	OPN8	OPN9	OPN10	Clusters
0	549499	2.0	3.0	2.0	2.0	...	4.0	3.0	3.0	4.0	2
1	811367	2.0	3.0	2.0	4.0	...	4.0	4.0	4.0	4.0	0
2	450151	4.0	1.0	5.0	1.0	...	5.0	5.0	5.0	5.0	2
3	919073	1.0	4.0	3.0	5.0	...	5.0	5.0	4.0	4.0	4
4	894414	3.0	1.0	4.0	3.0	...	3.0	2.0	4.0	4.0	4

5 rows × 52 columns

Analysing the Model and Predictions

```
df_model.Clusters.value_counts()
```

```
3    10365
1    10128
2    10001
0     9806
4     9606
Name: Clusters, dtype: int64
```

```
pd.options.display.max_columns = 150
df_model.groupby('Clusters').mean()
```

Unnamed: 0	EXT1	EXT2	EXT3	EXT4	EXT5	EXT6	EXT7	EXT8	EXT9

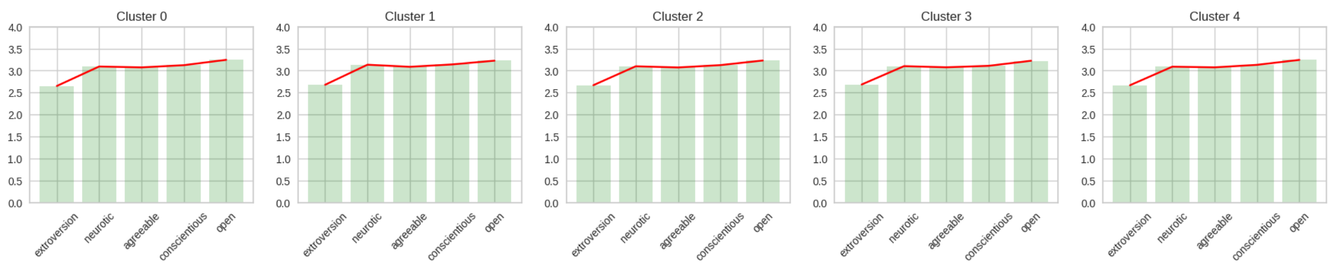
Clusters

```
# Summing up the different questions groups
col_list = list(df_model)
ext = col_list[1:10]
est = col_list[10:20]
agr = col_list[20:30]
csn = col_list[30:40]
opn = col_list[40:50]

data_sums = pd.DataFrame()
data_sums['extroversion'] = df_model[ext].sum(axis=1)/10
data_sums['neurotic'] = df_model[est].sum(axis=1)/10
data_sums['agreeable'] = df_model[agr].sum(axis=1)/10
data_sums['conscientious'] = df_model[csn].sum(axis=1)/10
data_sums['open'] = df_model[opn].sum(axis=1)/10
data_sums['clusters'] = predictions
data_sums.groupby('clusters').mean()
```

clusters	extroversion	neurotic	agreeable	conscientious	open
0	2.650296	3.093157	3.072619	3.123343	3.244636
1	2.677488	3.133906	3.086157	3.140383	3.225415
2	2.668423	3.098110	3.071123	3.124858	3.229177
3	2.683936	3.101245	3.074867	3.108538	3.223666
4	2.667052	3.088955	3.073621	3.130887	3.243442

```
# Visualizing the means for each cluster
dataclusters = data_sums.groupby('clusters').mean()
plt.figure(figsize=(22,3))
for i in range(0, 5):
    plt.subplot(1,5,i+1)
    plt.bar(dataclusters.columns, dataclusters.iloc[i,:], color='green', alpha=0.2)
    plt.plot(dataclusters.columns, dataclusters.iloc[i,:], color='red')
    plt.title('Cluster ' + str(i))
    plt.xticks(rotation=45)
    plt.ylim(0,4)
```



Visualizing the Cluster Predictions

PCA

```
# In order to visualize in 2D graph, PCA will be used
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
```

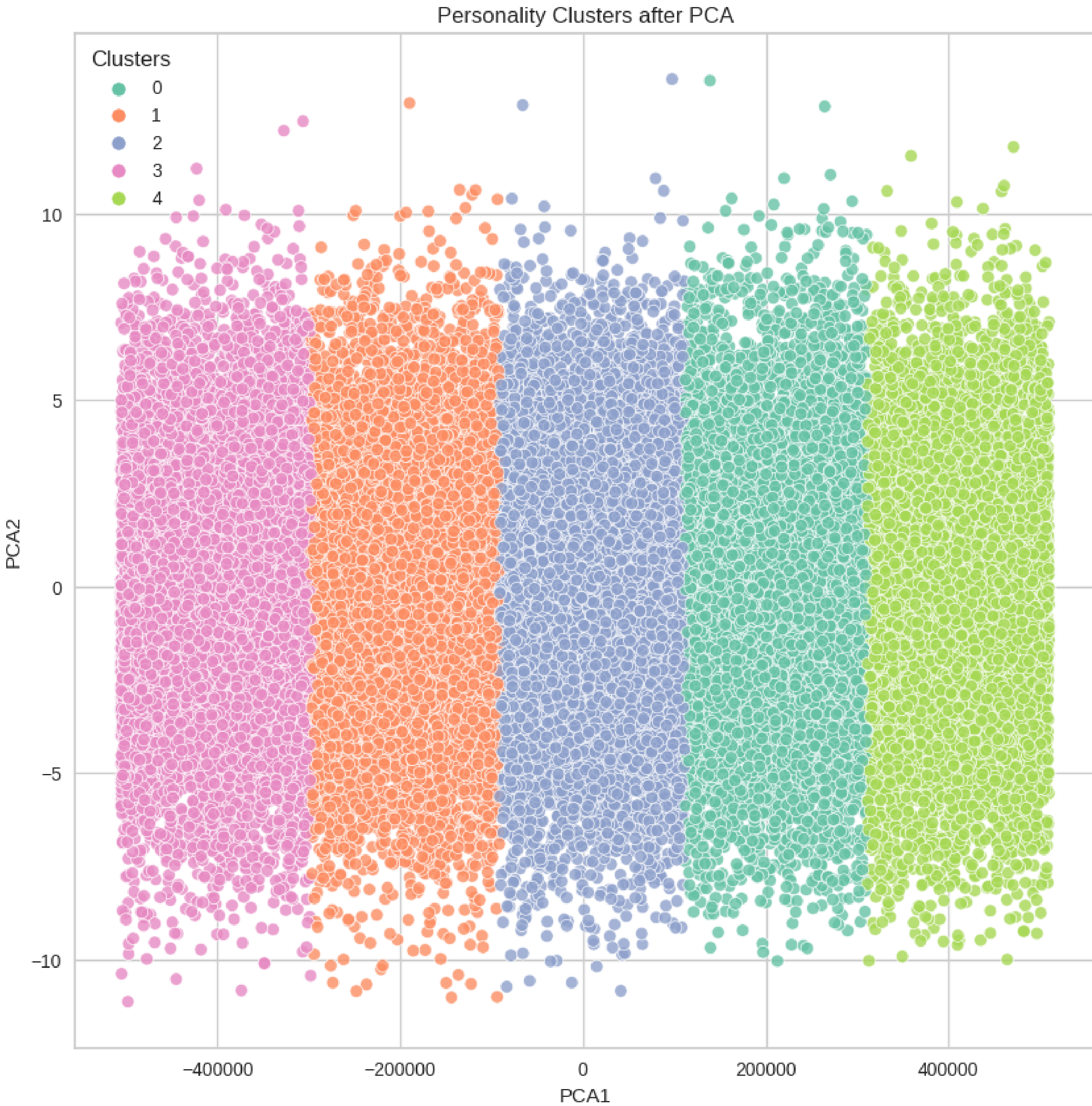
```
pca_fit = pca.fit_transform(df_model)

df_pca = pd.DataFrame(data=pca_fit, columns=['PCA1', 'PCA2'])
df_pca['Clusters'] = predictions
df_pca.head()
```

	PCA1	PCA2	Clusters
0	44046.698173	0.314354	2
1	305914.698172	2.163925	0
2	-55301.301828	-3.926704	2
3	413620.698172	3.312405	4
4	388961.698174	-0.858619	4

```
plt.figure(figsize=(10,10))
sns.scatterplot(data=df_pca, x='PCA1', y='PCA2', hue='Clusters', palette='Set2', alpha=0.8)
plt.title('Personality Clusters after PCA')
```

```
Text(0.5, 1.0, 'Personality Clusters after PCA')
```



Assignment 3

*Task 1 *

```
#Task 1. Probably this could have been done in a quicker way.

# Adding  5 extra columns, each with the aggregated trait score (the mean of the 10 questions for that tr
#Code taken from https://www.statology.org/pandas-average-selected-columns/


data['mean_Ext'] = data[['EXT1', 'EXT2', 'EXT3', 'EXT4', 'EXT5', 'EXT6', 'EXT7', 'EXT8', 'EXT9', 'EXT10']
print(data['mean_Ext'])
data['mean_EST'] = data[['EST1', 'EST2', 'EST3', 'EST4', 'EST5', 'EST6', 'EST7', 'EST8', 'EST9', 'EST10']
print(data['mean_EST'])
data['mean_AGR'] = data[['AGR1', 'AGR2', 'AGR3', 'AGR4', 'AGR5', 'AGR6', 'AGR7', 'AGR8', 'AGR9', 'AGR10']
print(data['mean_AGR'])
data['mean_CSN'] = data[['CSN1', 'CSN2', 'CSN3', 'CSN4', 'CSN5', 'CSN6', 'CSN7', 'CSN8', 'CSN9', 'CSN10']
print(data['mean_CSN'])
data['mean_OPN'] = data[['OPN1', 'OPN2', 'OPN3', 'OPN4', 'OPN5', 'OPN6', 'OPN7', 'OPN8', 'OPN9', 'OPN10']
print(data['mean_OPN'])
```

```
0        2.6
1        2.9
2        3.4
3        2.7
4        2.7
        ...
49995    3.8
49996    2.5
49997    3.3
49998    3.6
49999    3.0
Name: mean_Ext, Length: 49906, dtype: float64
0        3.7
1        3.4
2        2.5
3        2.4
4        3.1
        ...
49995    4.2
49996    3.8
49997    3.2
49998    1.9
49999    2.6
Name: mean_EST, Length: 49906, dtype: float64
0        3.0
1        3.3
2        3.6
3        2.6
4        2.7
        ...
49995    3.4
49996    3.3
49997    3.8
49998    3.3
49999    3.1
Name: mean_AGR, Length: 49906, dtype: float64
0        2.0
1        3.2
2        2.6
3        2.8
4        2.8
        ...
49995    3.0
49996    3.2
49997    3.5
49998    3.6
49999    3.2
```

```
Name: mean_CSN, Length: 49906, dtype: float64
0        2.6
1        3.5
2        3.6
3        3.4
4        2.7
        ...
49995    2.5
49996    2.9
49997    3.6
49998    3.7
```

```
# Showing the mean of each trait over the whole dataset. Code taken from https://sparkbyexamples.com/pand
mean_ext = data['mean_Ext'].mean()
print("The mean of the mean_Ext over the whole dataset:", mean_ext)

mean_est = data['mean_EST'].mean()
print("The mean of the mean_EST over the whole dataset:", mean_est)

mean_agr = data['mean_AGR'].mean()
print("The mean of the mean_AGR over the whole dataset:", mean_agr)

mean_csn = data['mean_CSN'].mean()
print("The mean of the mean_CSN over the whole dataset:", mean_csn)

mean_opn = data['mean_OPN'].mean()
print("The mean of the mean_OPN over the whole dataset:", mean_opn)
```

```
The mean of the mean_Ext over the whole dataset: 3.0248547268865464
The mean of the mean_EST over the whole dataset: 3.025952791247545
The mean of the mean_AGR over the whole dataset: 3.156991143349497
The mean of the mean_CSN over the whole dataset: 3.1257804672784837
The mean of the mean_OPN over the whole dataset: 3.2701037951348533
```

*Task 2 *

I first wanted to create list in which I would store all the skew values for each question of each trait. I used the existing trait and keys_for_trait() as this was not as simple as above. I used a for loop to iterate through the data and store the vales.

```
from scipy.stats import skew

for trait in traits:   # Iterate over each trait and find the related questions
    trait_questions = keys_for_trait(trait)

    trait_skewness_list = []  # Create a list to store the trait skew values

    for question in trait_questions:  # calaculate the skew values for each questions and append to the l
        question_skewness = skew(data[question])
        trait_skewness_list.append(question_skewness)

# I wanted to have a look at the data, so I printed it.

    print(f"Skewness values for {trait}:")
    for i, skewness_value in enumerate(trait_skewness_list):
        print(f"Question {i+1}: {skewness_value}")
    print()  # Print an empty line between traits
```

```
Question 2: 0.15259657587671677
Question 3: -0.279295509138290З
Question 4: -0.17944379495610588
Question 5: -0.3732787551073541
Question 6: 0.5489333151896756
Question 7: 0.15056549502376185
Question 8: -0.3893711204227024
Question 9: -0.04883700220534794
Question 10: -0.5399730070658465
```

```
Question 1: -0.327408622349892
Question 2: -0.22179281101984893
Question 3: -0.953549292560431
Question 4: 0.2378309079734583
Question 5: 0.07559877865341474
Question 6: 0.048220478839306025
Question 7: -0.07243291504176515
Question 8: 0.22429204776082115
Question 9: -0.16621069631669297
Question 10: 0.13724153055905913

Skewness values for AGR:
Question 1: 0.7073888928479241
Question 2: -0.9354186617049717
Question 3: 0.593800012338043
Question 4: -1.0989636022906157
Question 5: 0.6828598326685161
Question 6: -0.8629854157353127
Question 7: 0.7124241149593249
Question 8: -0.7937644050038022
Question 9: -0.9035002353717004
Question 10: -0.6208035569070954

Skewness values for CSN:
Question 1: -0.4707216018701447
Question 2: -0.029596083061092016
Question 3: -1.0632000767118233
Question 4: 0.27396715466143895
Question 5: 0.26091654005697895
Question 6: 0.1117859607434505
Question 7: -0.7816177683733613
Question 8: 0.28015458289514206
Question 9: -0.287633995735212
Question 10: -0.5955884067027184

Skewness values for OPN:
Question 1: -0.7119656732955313
Question 2: 0.8112063191490683
Question 3: -1.087725552063303
Question 4: 0.8928069703281235
Question 5: -0.8099891574691805
Question 6: 1.1573115557410132
Question 7: -1.1185316636644924
Question 8: -0.26278892179596125
Question 9: -1.3682532103399743
Question 10: -0.9396693172689798
```

I then wanted to distinguish between which of the found value could be considered as positive, negative or symmetrical (no skew). I found this charategorisation of skewness (https://www.analyticsvidhya.com/blog/2021/05/shape-of-data-skewness-and-kurtosis/): If the skewness is between -0.5 & 0.5, the data are nearly symmetrical. If the skewness is between -1 & -0.5 (negative skewed) or between 0.5 & 1(positive skewed), the data are slightly skewed. If the skewness is lower than -1 (negative skewed) or greater than 1 (positive skewed), the data are extremely skewed.

I thefore created a loop that would categorise the value based on the above figures.

```
trait_skewness_list = []


trait_skewness_types = []
for question in trait_questions:
        question_skewness = skew(data[question])
        trait_skewness_list.append(question_skewness)

        if question_skewness < -0.5:
            skewness_type = "negative"
        elif question_skewness > 0.5:
            skewness_type = "positive"
        else:
```

```
            skewness_type = "symmetrical"

        trait_skewness_types.append(skewness_type)

skewness_types.append(trait_skewness_types)

# Print the skewness types for each question of each trait
for i, trait in enumerate(traits):
    print(f"Skewness types for Trait {i+1} ({trait}):")
    for j, question in enumerate(keys_for_trait(trait)):
        skewness_type = skewness_types[i][j]
        print(f"Question {j+1}: {skewness_type}")
    print()
```

```
Question 2: positive
Question 3: negative
Question 4: positive
Question 5: negative
Question 6: positive
Question 7: negative
Question 8: symmetrical
Question 9: negative
Question 10: negative

Skewness types for Trait 2 (EST):
Question 1: negative
Question 2: positive
Question 3: negative
Question 4: positive
Question 5: negative
Question 6: positive
Question 7: negative
Question 8: symmetrical
Question 9: negative
Question 10: negative

Skewness types for Trait 3 (AGR):
Question 1: negative
Question 2: positive
Question 3: negative
Question 4: positive
Question 5: negative
Question 6: positive
Question 7: negative
Question 8: symmetrical
Question 9: negative
Question 10: negative

Skewness types for Trait 4 (CSN):
Question 1: negative
Question 2: positive
Question 3: negative
Question 4: positive
Question 5: negative
Question 6: positive
Question 7: negative
Question 8: symmetrical
Question 9: negative
Question 10: negative

Skewness types for Trait 5 (OPN):
Question 1: negative
Question 2: positive
Question 3: negative
Question 4: positive
Question 5: negative
Question 6: positive
Question 7: negative
Question 8: symmetrical
Question 9: negative
Question 10: negative
```

Once I had the types for each of the questions of each trait, I used a for loop to firstly iterate over the list and then increment the types of positive/negative.

I then had to sort the values.

```
trait_ranks = []
for trait_skewness_types in skewness_types:
    rank = 0
    for skewness_type in trait_skewness_types:
        if skewness_type in ("positive", "negative"):
            rank += 1
    trait_ranks.append(rank)

# Sorting  the traits based on their ranksr
ranked_traits = [trait for _, trait in sorted(zip(trait_ranks, traits), reverse=True)]   # I was having i


print("Ranking of Traits:")
for i, trait in enumerate(ranked_traits):
    print(f"{i+1}. {trait}")
```

```
Ranking of Traits:
1. OPN
2. EXT
3. EST
4. CSN
5. AGR
```

Comparing the ranking to the printed trait_skewness_types results, it seems to be correct, with AGR having the most skewed answers to the questions.

*Thank you for reading through - I am aware that possible this could have been done in much fewer coding lines - getting there! 😀

✓ 0s completed at 8:48 PM