HMDS ASSIGNMENT-2

Group Members
Daksh Sammi
Shyama Goel
Shaikh Shuhail
Shivam IIITD

PART A

Objective: To determine the top 10 microbial species that are most strongly and most weakly correlated with the Mediterranean Diet adherence score.

Methodology:

- Aligned the datasets by using common sample names in all three sheets.
- Dropped any rows with missing values or zero-variance variables.
- Conducted Shapiro-Wilk test to check normality for all species, food nutrients, and adherence scores.
- Choose the correct correlation test:
    - Pearson if both variables are normally distributed
    - Spearman otherwise (as most variables are non-normal)
- Computed correlation between species and adherence scores.
- Sorted correlation values and picked the Top 10 Positive and Top 10 Negative microbial species.

Code Snippet:

```
import pandas as pd
import numpy as np
from scipy.stats import shapiro, pearsonr, spearmanr


# file path: change as per your system
xls = pd.ExcelFile("/content/Assignment2_Data.xlsx")


species_df = xls.parse("SpeciesProfile")
adherence_df = xls.parse("AdherenceScores")
food_df = xls.parse("FoodIntakeProfile")
```

```
species_df.set_index("Unnamed: 0", inplace=True)
adherence_df.set_index("Unnamed: 0", inplace=True)
food_df.set_index("Unnamed: 0", inplace=True)

common_samples =
species_df.index.intersection(adherence_df.index).intersection(food_df.ind
ex)
species_common = species_df.loc[common_samples]
adherence_common = adherence_df.loc[common_samples]
food_common = food_df.loc[common_samples]
# checking normality; p-value 0.05
normality_status = {}

if np.std(adherence_common["AdherenceScores"]) > 0:
    normality_status["AdherenceScores"] =
shapiro(adherence_common["AdherenceScores"].values)[1] >= 0.05

for col in species_common.columns:
    if np.std(species_common[col]) > 0:
        normality_status[col] = shapiro(species_common[col].values)[1] >=
0.05

for col in food_common.columns:
    if np.std(food_common[col]) > 0:
        normality_status[col] = shapiro(food_common[col].values)[1] >=
0.05

print("Total variables tested:", len(normality_status))
print("Normal variables:", sum(normality_status.values()))
print("Non-normal variables:", len(normality_status) -
sum(normality_status.values()))

# Computing correlation of species with adherence score and using the
appropriate tests (pearson vs spearman)
correlation_values = {}

for species in species_common.columns:
    x = species_common[species]
    y = adherence_common["AdherenceScores"]
    if np.std(x) > 0 and np.std(y) > 0:
```

```
        if normality_status.get(species, False) and
normality_status.get("AdherenceScores", False):
            corr, _ = pearsonr(x, y)
        else:
            corr, _ = spearmanr(x, y)#using spearman as most data is
non-normal
        correlation_values[species] = corr

# selecting top 10 +ve and -ve species
corr_series = pd.Series(correlation_values).dropna().sort_values()
top_negative = corr_series.head(10)
top_positive = corr_series.tail(10)

print("Top 10 Negative Species:\n", top_negative)
print("\nTop 10 Positive Species:\n", top_positive)
```

Results:

The top 10 Negatively and Positively associated species are displayed along with their correlation coefficients.

```
Top 10 Negative Species:
 Collinsella_aerofaciens       -0.165982
Ruminococcus_torques           -0.163115
Dorea_formicigenerans          -0.138623
Coprococcus_comes              -0.125208
Mogibacterium_Unclassified     -0.124854
Eubacterium_biforme            -0.096323
Olsenella_Unclassified         -0.094810
Desulfovibrio_desulfuricans    -0.091978
Atopobium_parvulum             -0.091318
Howardella_ureilytica          -0.088517
dtype: float64

Top 10 Positive Species:
 Roseburia_intestinalis         0.086299
Veillonella_atypica             0.086457
Clostridium_saccharogumia       0.087999
Eubacterium_eligens             0.091626
Ruminococcus_flavefaciens       0.097174
Veillonella_dispar              0.104320
Eubacterium_xylanophilum        0.111981
Clostridium_populeti            0.113818
Haemophilus_parainfluenzae      0.123353
Anaerostipes_Unclassified       0.123462
dtype: float64
```

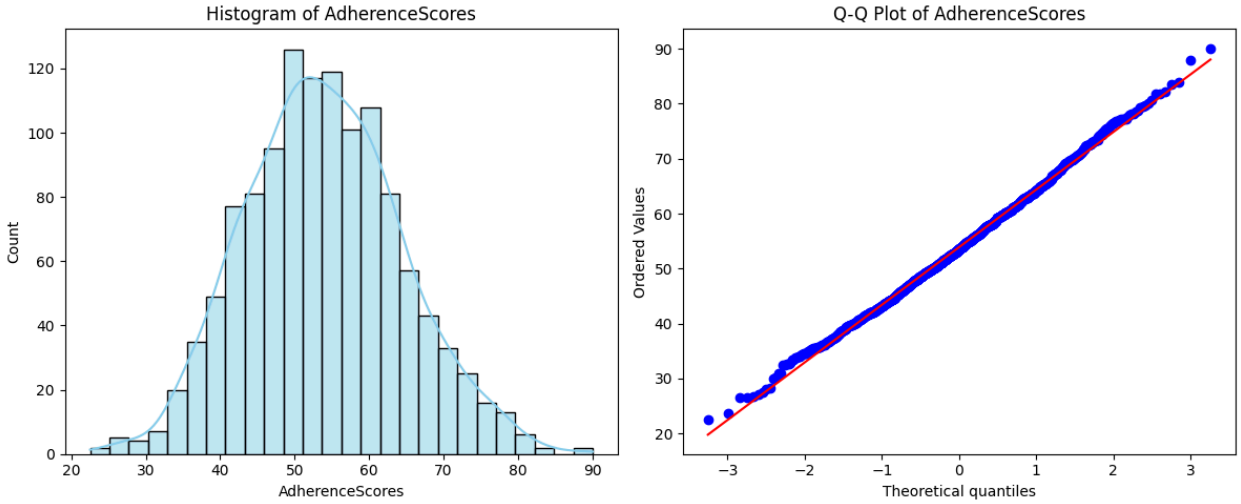
Normality Testing:
Only the adherence scores variable is normal. Rest all data variables are non-normal.

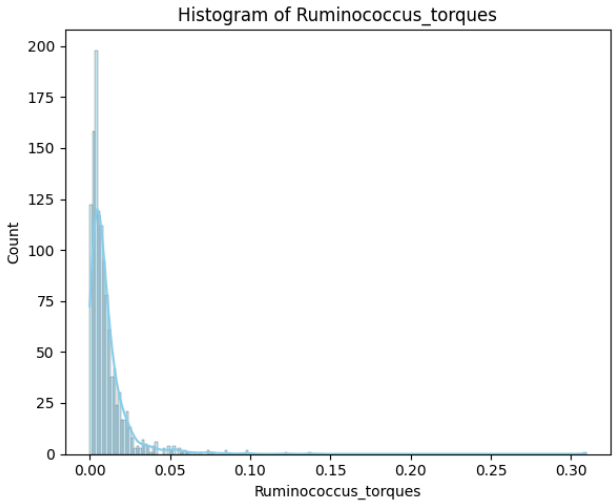
```
# checking normality; p-value 0.05
normality_status = {}

if np.std(adherence_common["AdherenceScores"]) > 0:
    normality_status["AdherenceScores"] =
shapiro(adherence_common["AdherenceScores"].values)[1] >= 0.05

for col in species_common.columns:
    if np.std(species_common[col]) > 0:
        normality_status[col] = shapiro(species_common[col].values)[1] >=
0.05

for col in food_common.columns:
    if np.std(food_common[col]) > 0:
        normality_status[col] = shapiro(food_common[col].values)[1] >=
0.05

print("Total variables tested:", len(normality_status))
print("Normal variables:", sum(normality_status.values()))
print("Non-normal variables:", len(normality_status) -
sum(normality_status.values()))
```

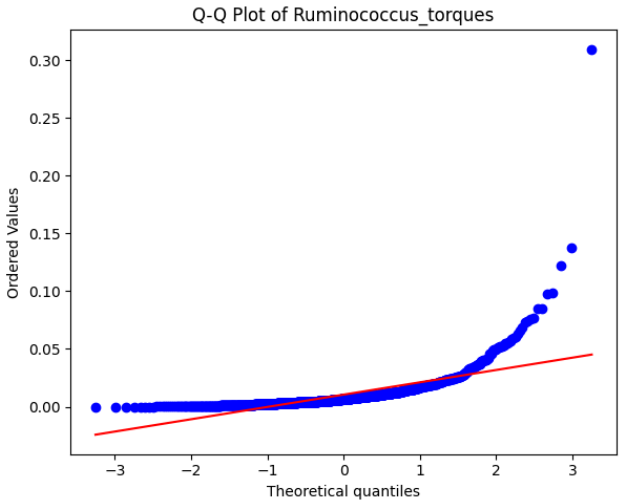
```
Total variables tested: 504
Normal variables: 1
Non-normal variables: 503
```

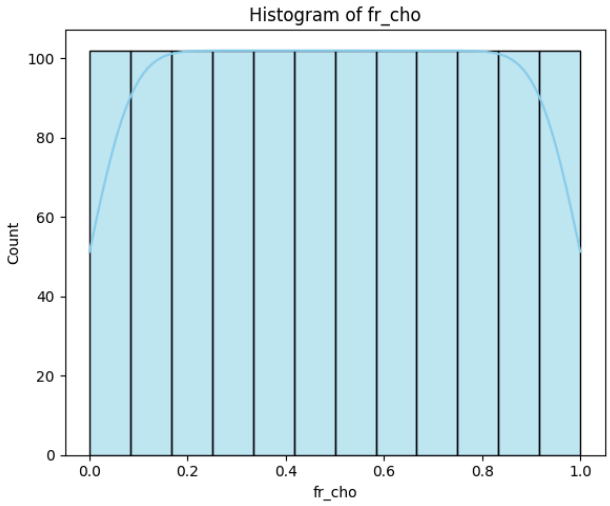
```
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats

def check_normality(data, column):
    values = data[column].dropna()

    # Shapiro Test
    stat, p = stats.shapiro(values)
    print(f"Shapiro-Wilk Test → {column}: stat={stat:.3f}, p={p:.3f}")
```

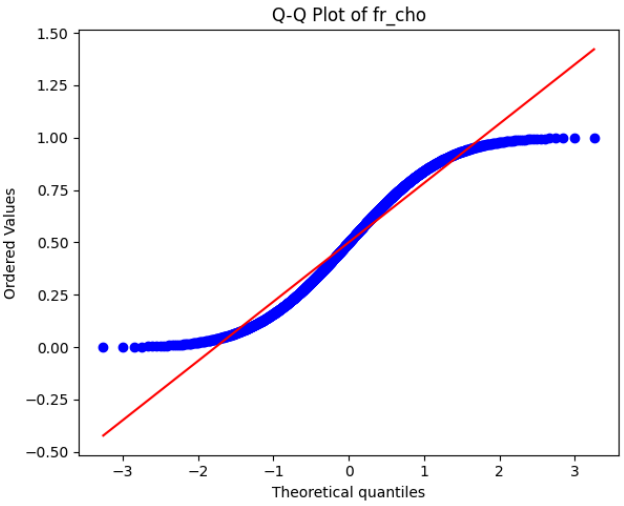
```
    # Histogram + KDE
    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    sns.histplot(values, kde=True, color='skyblue')
    plt.title(f"Histogram of {column}")

    # Q-Q Plot
    plt.subplot(1, 2, 2)
    stats.probplot(values, dist="norm", plot=plt)
    plt.title(f"Q-Q Plot of {column}")

    plt.tight_layout()
    plt.show()

check_normality(adherence_common, "AdherenceScores")
check_normality(species_common, "Ruminococcus_torques")
check_normality(food_common, "fr_cho")  # Carbohydrate freq
check_normality(food_common, "fr_prot") # Protein freq
```

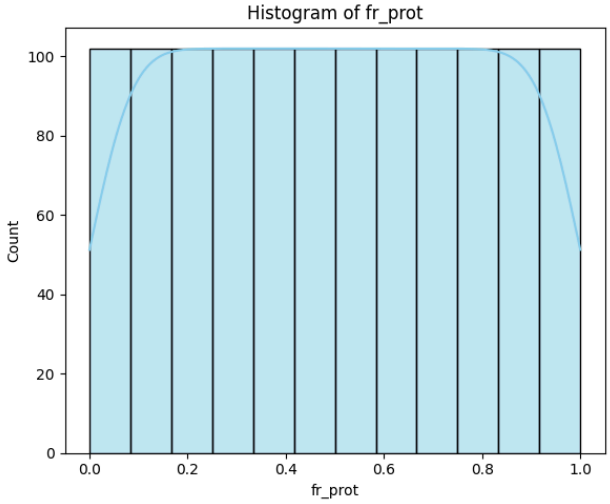


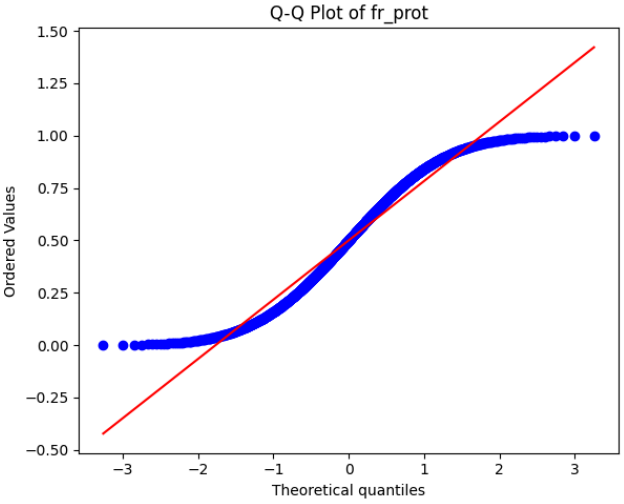

Histogram of Ruminococcus_torques
Count
Ruminococcus_torques


Q-Q Plot of Ruminococcus_torques
Ordered Values
Theoretical quantiles


Histogram of fr_cho
Count
fr_cho


Q-Q Plot of fr_cho
Ordered Values
Theoretical quantiles


Histogram of fr_prot
Count
fr_prot


Q-Q Plot of fr_prot
Ordered Values
Theoretical quantiles

PART B

Objective: Analyze the correlation between the intake of food macronutrients and microbial species discovered in Part A.

Methodology:

- To start, we reused the aligned **species_common and food_common** datasets.
- Directed solely at the 20 chosen species (10 positive, 10 negative from Part A
- For each species-nutrient pair:
    - Select Pearson if both variables normally distributed
    - Otherwise, employed Spearman
- Keep only statistically significant correlations (p-value < 0.05).
- Ensured that both variables had non-zero variance.
- The end product is a table of all nutrient-species pairs for which there are significant linear relationships, as well as the correlation coefficients and p-values.

Code Snippet:

```
import numpy as np
from scipy.stats import pearsonr

food_df = xls.parse("FoodIntakeProfile")
food_df.set_index('Unnamed: 0', inplace=True)

food_df

# gettin common samples for food and species
common_samples_b = food_df.index.intersection(species_df.index)
food_common = food_df.loc[common_samples_b]
species_common_b = species_df.loc[common_samples_b]

food_common

species_common_b

selected_species = list(top_negative.index) + list(top_positive.index)

selected_species
```

```
results = []
for species in selected_species:
    for nutrient in food_common.columns:
        x = food_common[nutrient]
        y = species_common[species]
        if np.std(x) > 0 and np.std(y) > 0:
            # Use appropriate correlation based on normality of BOTH
variables
            if normality_status.get(nutrient, False) and
normality_status.get(species, False):
                r, p = pearsonr(x, y)
            else:
                r, p = spearmanr(x, y)
            if p < 0.05:
                results.append({
                    'Species': species,
                    'Nutrient': nutrient,
                    'Correlation': r,
                    'P-Value': p
                })

results_df = pd.DataFrame(results)

# Summarization
unique_nutrients = results_df['Nutrient'].nunique()
total_associations = len(results_df)

print(f"Total significant associations: {total_associations}")
print(f"Unique nutrients involved: {unique_nutrients}\n")

print("Head of the associations:")
print(results_df.head())

print("\nTail of the associations:")
print(results_df.tail())

results_df.to_csv("significant_species_nutrient_associations.csv",
index=False)
```

Results:
Total significant association observed is **89** and **39** unique nutrients are involved.

```
Total significant associations: 89
Unique nutrients involved: 39

Head of the associations:
                  Species      Nutrient  Correlation   P-Value
0  Collinsella_aerofaciens       fr_fapu    -0.058762  0.039830
1  Collinsella_aerofaciens     fr_fapun3    -0.059411  0.037687
2  Collinsella_aerofaciens     fr_fapun6    -0.079458  0.005412
3  Collinsella_aerofaciens      fr_cartb    -0.067005  0.019054
4  Collinsella_aerofaciens  fr_vitamin_A    -0.071384  0.012488

Tail of the associations:
                      Species      Nutrient  Correlation   P-Value
84          Veillonella_dispar         fr_b8     0.068696  0.016228
85    Eubacterium_xylanophilum      fr_cartb     0.060136  0.035407
86    Eubacterium_xylanophilum  fr_vitamin_A     0.087209  0.002260
87  Haemophilus_parainfluenzae      fr_chorl    -0.069641  0.014813
88   Anaerostipes_Unclassified        fr_alc    -0.062363  0.029131
```