Principal Component Analysis (PCA)

Data Visualization

In this project, we explored Principal Component Analysis (PCA)'s application in Dimensionality Reduction and Data Visualization. In addition, PCA can also be used in Noise Filtering, Feature Extraction, Data Compression and Anomaly Detection. It is easily one of the favorite tools for Data Analysts.

The main idea is to find a new set of uncorrelated varaibles called principal components (PCs) that are linear combinations of the original variables to explain the variance in the data.

The sample data used is a representation of food consumed in European countries.

Load Packages

In [3]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Load Sample Data

In [4]:

```
df = pd.read_csv('food-consumption.csv',header=0)
```

In [5]:

```
df.head(2)
```

Out[5]:

	Country	Real coffee	Instant coffee	Tea	Sweetener	Biscuits	Powder soup	Tin soup	Potatoes	Froz f
0	Germany	90	49	88	19	57	51	19	21	
1	Italy	82	10	60	2	55	41	3	2	

2 rows × 21 columns

In [6]:

```
df.describe(include='all')
```

Out[6]:

	Country	Real coffee	Instant coffee	Tea	Sweetener	Biscuits	Powde sou
count	16	16.000000	16.000000	16.000000	16.000000	16.000000	16.00000
unique	16	NaN	NaN	NaN	NaN	NaN	Na
top	Germany	NaN	NaN	NaN	NaN	NaN	Na
freq	1	NaN	NaN	NaN	NaN	NaN	Na
mean	NaN	78.562500	39.250000	78.500000	18.000000	60.687500	49.00000
std	NaN	23.145824	23.147354	18.540047	10.532489	19.168442	15.42724
min	NaN	27.000000	10.000000	40.000000	2.000000	22.000000	27.00000
25%	NaN	71.500000	17.000000	62.500000	11.000000	56.500000	36.25000
50%	NaN	89.000000	39.000000	84.500000	18.500000	62.000000	47.00000
75%	NaN	96.000000	54.250000	92.250000	25.750000	74.500000	58.00000
max	NaN	98.000000	86.000000	99.000000	35.000000	91.000000	75.00000

11 rows × 21 columns

Check if there is NULL values:

In [16]:

```
df.isnull().sum()
```

Out[16]:

```
Country           0
Real coffee       0
Instant coffee    0
Tea               0
Sweetener         0
Biscuits          0
Powder soup       0
Tin soup          0
Potatoes          0
Frozen fish       0
Frozen veggies    0
Apples            0
Oranges           0
Tinned fruit      0
Jam               0
Garlic            0
Butter            0
Margarine         0
Olive oil         0
Yoghurt           0
Crisp bread       0
dtype: int64
```

In [17]:

```
food_array = df.columns[1::].to_numpy()
food_array
```

```
Out[17]: array(['Real coffee', 'Instant coffee', 'Tea', 'Sweetener', 'Biscuits',
                'Powder soup', 'Tin soup', 'Potatoes', 'Frozen fish',
                'Frozen veggies', 'Apples', 'Oranges', 'Tinned fruit', 'Jam',
                'Garlic', 'Butter', 'Margarine', 'Olive oil', 'Yoghurt',
                'Crisp bread'], dtype=object)
```

```
In [18]: country_array = df['Country'].to_numpy()
         country_array
```

```
Out[18]: array(['Germany', 'Italy', 'France', 'Holland', 'Belgium', 'Luxembourg',
                'England', 'Portugal', 'Austria', 'Switzerland', 'Sweden',
                'Denmark', 'Norway', 'Finland', 'Spain', 'Ireland'], dtype=object)
```

```
In [19]: country_data = df.iloc[:, 1:].to_numpy()
         country_data
```

```
Out[19]: array([[90, 49, 88, 19, 57, 51, 19, 21, 27, 21, 81, 75, 44, 71, 22, 91,
                 85, 74, 30, 26],
                [82, 10, 60,  2, 55, 41,  3,  2,  4,  2, 67, 71,  9, 46, 80, 66,
                 24, 94,  5, 18],
                [88, 42, 63,  4, 76, 53, 11, 23, 11,  5, 87, 84, 40, 45, 88, 94,
                 47, 36, 57,  3],
                [96, 62, 98, 32, 62, 67, 43,  7, 14, 14, 83, 89, 61, 81, 15, 31,
                 97, 13, 53, 15],
                [94, 38, 48, 11, 74, 37, 23,  9, 13, 12, 76, 76, 42, 57, 29, 84,
                 80, 83, 20,  5],
                [97, 61, 86, 28, 79, 73, 12,  7, 26, 23, 85, 94, 83, 20, 91, 94,
                 94, 84, 31, 24],
                [27, 86, 99, 22, 91, 55, 76, 17, 20, 24, 76, 68, 89, 91, 11, 95,
                 94, 57, 11, 28],
                [72, 26, 77,  2, 22, 34,  1,  5, 20,  3, 22, 51,  8, 16, 89, 65,
                 78, 92,  6,  9],
                [55, 31, 61, 15, 29, 33,  1,  5, 15, 11, 49, 42, 14, 41, 51, 51,
                 72, 28, 13, 11],
                [73, 72, 85, 25, 31, 69, 10, 17, 19, 15, 79, 70, 46, 61, 64, 82,
                 48, 61, 48, 30],
                [97, 13, 93, 31, 61, 43, 43, 39, 54, 45, 56, 78, 53, 75,  9, 68,
                 32, 48,  2, 93],
                [96, 17, 92, 35, 66, 32, 17, 11, 51, 42, 81, 72, 50, 64, 11, 92,
                 91, 30, 11, 34],
                [92, 17, 83, 13, 62, 51,  4, 17, 30, 15, 61, 72, 34, 51, 11, 63,
                 94, 28,  2, 62],
                [98, 12, 84, 20, 64, 27, 10,  8, 18, 12, 50, 57, 22, 37, 15, 96,
                 94, 17, 21, 64],
                [70, 40, 40, 18, 62, 43,  2, 14, 23,  7, 59, 77, 30, 38, 86, 44,
                 51, 91, 16, 13],
                [30, 52, 99, 11, 80, 75, 18,  2,  5,  3, 57, 52, 46, 89,  5, 97,
                 25, 31,  3,  9]])
```

```
In [20]: country_data.shape
```

```
Out[20]: (16, 20)
```

Therefore, this is a 16x20 matrix: 16 countries and 20 foods. In another word,

> $\mu = \frac{1}{m} \sum x_i$

> $c = \frac{1}{m} \sum (x - \mu)(x - \mu)^T$

where m=16, and there are 16 x features

In [21]:

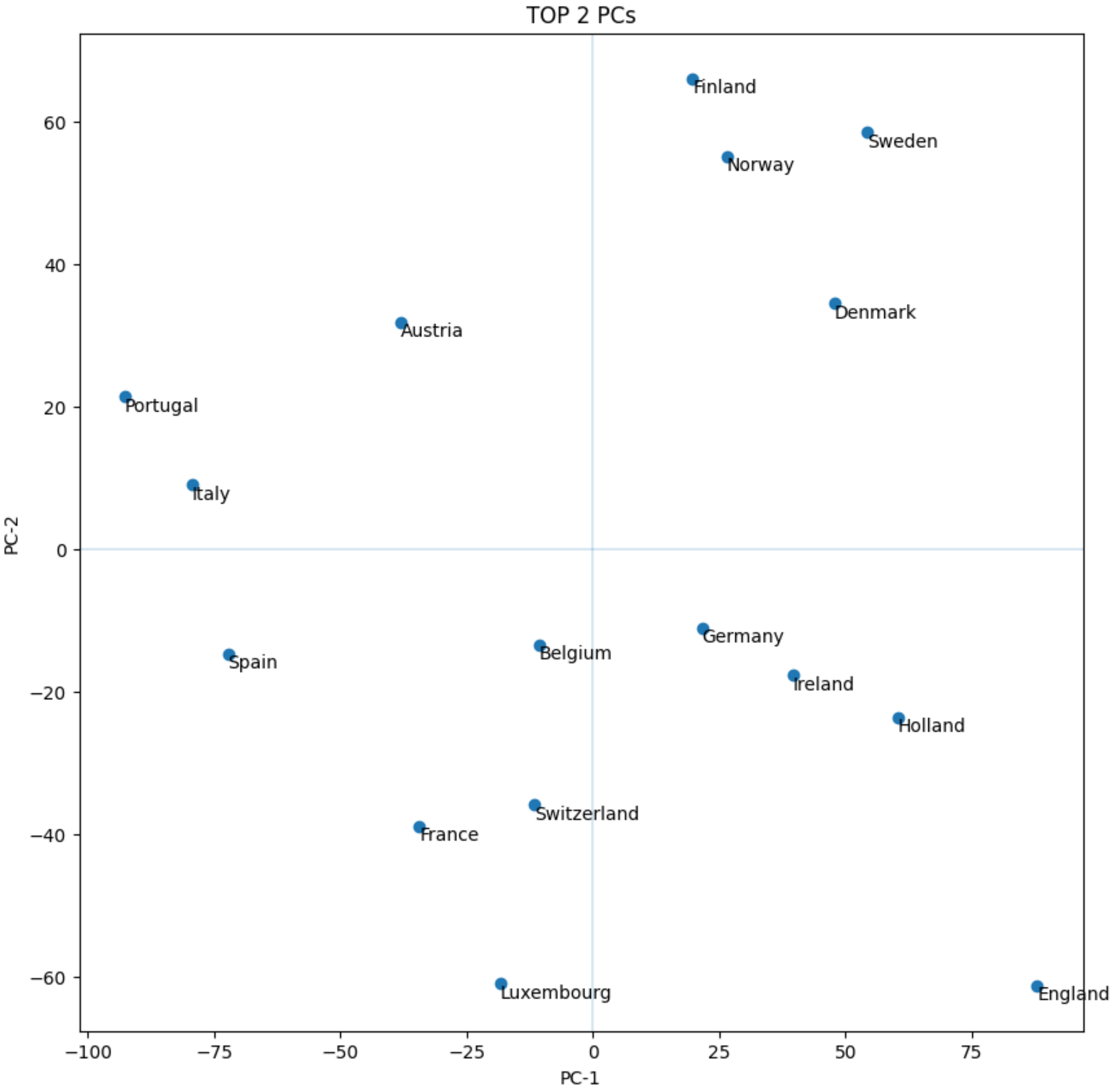
```
############################################################################
# Create PCA
############################################################################
def myPCA(X, d=2):

    mean_X = np.mean(X, axis=0)
    centered_X = X - mean_X
    conv_X = np.cov(centered_X.T)

    w, v = np.linalg.eig(conv_X)
    ids = np.argsort(w)[::-1]
    sort_v = v[:, ids]

    top_d_PC = (sort_v[:,:d]).real

    projected_X = np.dot(centered_X, top_d_PC)
    rebuild_X = np.dot(projected_X, top_d_PC.T) + mean_X

    return projected_X, rebuild_X, top_d_PC
```

In [22]:

```
projected_X1, rebuild_X1, top_d_PC1 = myPCA(country_data, d=2)
```

In [25]:

```
plt.figure(figsize=(10, 10))
plt.scatter(projected_X1[:, 0], projected_X1[:, 1])
plt.axvline(0, linewidth = 0.25)
plt.axhline(0, linewidth = 0.25)
for i in range(len(country_array)):
    plt.text(projected_X1[i, 0], projected_X1[i, 1], country_array[i], ha='l
plt.xlabel('PC-1')
plt.ylabel('PC-2')
plt.title('TOP 2 PCs')
plt.show()
```



Now the data tells a very interesting story by itself - some of the countries are clustered together as they consume similar food!