```python
%matplotlib inline
%load_ext autoreload
%autoreload 2

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
from scipy.cluster import hierarchy
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

*Assignment Summary

You can find a dataset dealing with European employment in 1979 at http://www.dm.unibo.it/~simoncin/EuropeanJobs.html. This dataset gives the percentage of people employed in each of a set of areas in 1979 for each of a set of European countries. Notice this dataset contains only 26 data points. That's fine; it's intended to give you some practice in visualization of clustering.

1. Use agglomerative clustering to cluster this data. Produce a dendrogram of this data for each of single link, complete link, and group average clustering. You should label the countries on the axis. What structure in the data does each method expose? You should see dendrograms that "make sense" (at least if you remember some European history), and have interesting differences.
2. Using k-means, cluster this dataset. What is a good choice of k for this data and why?

0. Data

0.1 Description

You can find a dataset dealing with European employment in 1979 at http://www.dm.unibo.it/~simoncin/EuropeanJobs.html. This dataset gives the percentage of people employed in each of a set of areas in 1979 for each of a set of European countries. Notice this dataset contains only 26 data points. That's fine; it's intended to give you some practice in visualization of clustering.

0.2 Loading

```python
df = pd.read_csv("../Clustering-lib/EuropeanJobs.dat", sep='\t', header=0)
```

```python
df
```

	Country	Agr	Min	Man	PS	Con	SI	Fin	SPS	TC
0	Belgium	3.3	0.9	27.6	0.9	8.2	19.1	6.2	26.6	7.2
1	Denmark	9.2	0.1	21.8	0.6	8.3	14.6	6.5	32.2	7.1
2	France	10.8	0.8	27.5	0.9	8.9	16.8	6.0	22.6	5.7
3	W. Germany	6.7	1.3	35.8	0.9	7.3	14.4	5.0	22.3	6.1
4	Ireland	23.2	1.0	20.7	1.3	7.5	16.8	2.8	20.8	6.1
5	Italy	15.9	0.6	27.6	0.5	10.0	18.1	1.6	20.1	5.7
6	Luxembourg	7.7	3.1	30.8	0.8	9.2	18.5	4.6	19.2	6.2
7	Netherlands	6.3	0.1	22.5	1.0	9.9	18.0	6.8	28.5	6.8
8	United Kingdom	2.7	1.4	30.2	1.4	6.9	16.9	5.7	28.3	6.4
9	Austria	12.7	1.1	30.2	1.4	9.0	16.8	4.9	16.8	7.0
10	Finland	13.0	0.4	25.9	1.3	7.4	14.7	5.5	24.3	7.6
11	Greece	41.4	0.6	17.6	0.6	8.1	11.5	2.4	11.0	6.7
12	Norway	9.0	0.5	22.4	0.8	8.6	16.9	4.7	27.6	9.4
13	Portugal	27.8	0.3	24.5	0.6	8.4	13.3	2.7	16.7	5.7
14	Spain	22.9	0.8	28.5	0.7	11.5	9.7	8.5	11.8	5.5
15	Sweden	6.1	0.4	25.9	0.8	7.2	14.4	6.0	32.4	6.8
16	Switzerland	7.7	0.2	37.8	0.8	9.5	17.5	5.3	15.4	5.7
17	Turkey	66.8	0.7	7.9	0.1	2.8	5.2	1.1	11.9	3.2
18	Bulgaria	23.6	1.9	32.3	0.6	7.9	8.0	0.7	18.2	6.7
19	Czechoslovakia	16.5	2.9	35.5	1.2	8.7	9.2	0.9	17.9	7.0
20	E. Germany	4.2	2.9	41.2	1.3	7.6	11.2	1.2	22.1	8.4
21	Hungary	21.7	3.1	29.6	1.9	8.2	9.4	0.9	17.2	8.0
22	Poland	31.1	2.5	25.7	0.9	8.4	7.5	0.9	16.1	6.9
23	Rumania	34.7	2.1	30.1	0.6	8.7	5.9	1.3	11.7	5.0
24	USSR	23.7	1.4	25.8	0.6	9.2	6.1	0.5	23.6	9.3
25	Yugoslavia	48.7	1.5	16.8	1.1	4.9	6.4	11.3	5.3	4.0

Here is a description of the columns in the data:

- Country: name of the country
- Agr: percentage employed in agriculture
- Min: percentage employed in mining
- Man: percentage employed in manufacturing
- PS: percentage employed in power supply industries
- Con: percentage employed in construction
- SI: percentage employed in service industries
- Fin: percentage employed in finance
- SPS: percentage employed in social and personal services
- TC: percentage employed in transport and communications

```python
feature_cols = ['Agr','Min','Man','PS','Con','SI','Fin','SPS','TC']
X = df[feature_cols].values
Y = df['Country'].tolist()
```

1. Agglomerative Clustering

Task 1

Write a function `single_linkage` that produces a single-link agglomerative clustering. This function should take the data matrix `X` as input, which is a numpy array with the shape of (N, d) where N is the number of samples and d is the number of features. The output of the function should be a linkage matrix. Use the Euclidean distance as a metric.

You may find scipy's hierarchical clustering methods (https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html) useful here. The utilization of the `optimal_ordering` option makes interpretations of the resulting trees an easier job.

Student Notes

Recall that with agglomerative clustering you start with each obs its own cluster then agglomerate them based on distance. You can use the minimum, maximum, or average distance (single-link, complete-link, group average clustering) between every combination of observations in two clusters.

scipy.cluster.hierarchy.linkage(y, method='single', metric='euclidean', optimal_ordering=False): Perform hierarchical/agglomerative clustering.
The input y may be .. a 2-D array of observation vectors.

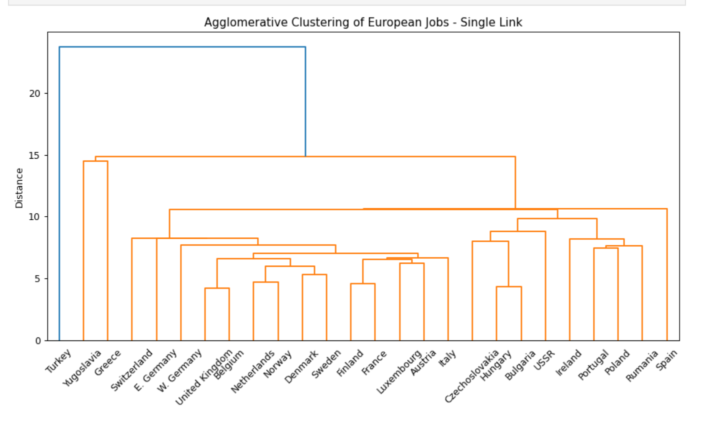
```python
def single_linkage(X):
    """
    Produce a single-link agglomerative clustering.

        Parameters:
                X (np.array): A numpy array of the shape (N,d) where N is the number o
        
        Returns:
                single_link (np.array): The single-link agglomerative clustering of X
    """
    
    # Mo code 0
    single_link = hierarchy.linkage(X, method='single', metric='euclidean', optimal_o
    # Mo code 1
    
    return single_link
```

```python
single_link = single_linkage(X)
assert single_link[:,2].min().round(3) == 4.234
```

```python
# Next, we will plot the dendogram for the first task.
```

```python
single_link = single_linkage(X)
plt.figure(figsize=(12,6), dpi=90)
plt.ylabel("Distance")
plt.title("Agglomerative Clustering of European Jobs - Single Link")
dn_single = hierarchy.dendrogram(single_link, labels=Y)
```



Task 2

Write a function `complete_linkage` that produces a complete-link agglomerative clustering. This function should take the data matrix `X` as input, which is a numpy array with the shape of (N, d) where N is the number of samples and d is the number of features. The output of the function should be a linkage matrix. Use the Euclidean distance as a metric.

You may find scipy's hierarchical clustering methods (https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html) useful here. The utilization of the `optimal_ordering` option makes interpretations of the resulting trees an easier job.

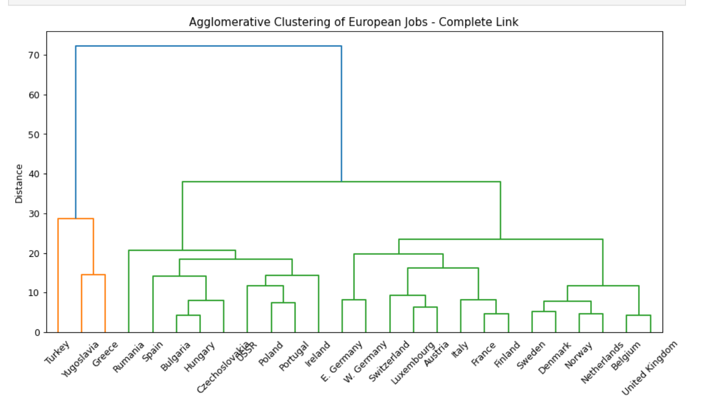
```python
def complete_linkage(X):
    """
    Produce a complete-link agglomerative clustering.

        Parameters:
                X (np.array): A numpy array of the shape (N,d) where N is the number o
        
        Returns:
                comp_link (np.array): The complete-link agglomerative clustering of X
    """
    
    # Beginning of Mo's code
    comp_link = hierarchy.linkage(X, method='complete', metric='euclidean', optimal_o
    # End of Mo's Code
    
    return comp_link
```

```python
comp_link = complete_linkage(X)
assert comp_link[:,2].max().round(3) == 72.278
```

```python
# Next, we will plot the dendogram for the second task.
```

```python
complete_link = complete_linkage(X)
plt.figure(figsize=(12,6), dpi=90)
plt.ylabel("Distance")
plt.title("Agglomerative Clustering of European Jobs - Complete Link")
dn_complete = hierarchy.dendrogram(complete_link,labels=Y)
```



Task 3

Write a function `group_avg_linkage` that produces an average-link agglomerative clustering. This function should take the data matrix `X` as input, which is a numpy array with the shape of (N, d) where N is the number of samples and d is the number of features. The output of the function should be a linkage matrix. Use the Euclidean distance as a metric.

You may find scipy's hierarchical clustering methods (https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html) useful here. The utilization of the `optimal_ordering` option makes interpretations of the resulting trees an easier job.

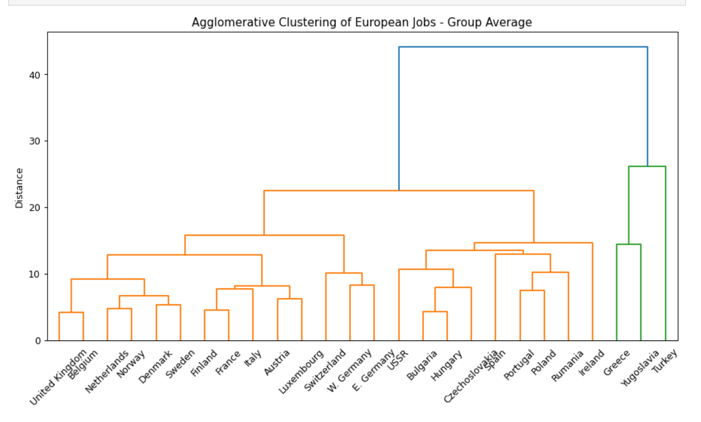
```python
def group_avg_linkage(X):
    """
    Produce an average-link agglomerative clustering.

        Parameters:
                X (np.array): A numpy array of the shape (N,d) where N is the number o
        
        Returns:
                avg_link (np.array): The average-link agglomerative clustering of X en
    """
    
    # Beginning of Mo's code
    avg_link = hierarchy.linkage(X, method='average', metric='euclidean', optimal_orde
    
    # End of Mo's Code
    
    return avg_link
```

```python
avg_link = group_avg_linkage(X)
assert avg_link[:,2].max().round(3) == 44.172
```

```python
# Next, we will plot the dendogram for the third task.
```

```python
average_link = group_avg_linkage(X)
plt.figure(figsize=(12,6), dpi=90)
plt.ylabel("Distance")
plt.title("Agglomerative Clustering of European Jobs - Group Average")
dn_average = hierarchy.dendrogram(average_link,labels=Y)
```



2. K-Means Clustering

In this part, we perform the K-Means clustering algorithm on the dataset, and evalute the effect of the parameter k (the number of clusters) on the outcome.

For this, we use the `KMeans` class from `sklearn.cluster`. You should familiarize yourself with this class. You can find the documentation of it here: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

In the following code, we run the K-Means algorithm for $2 \leq k \leq 25$ clusters.

Attention: Although you are not implementing this part, for the follow-up quiz of this assignment, you will need to come back here, write some code and do some calculations to get the answer of some questions in the quiz. For now, try to read the documentation for the `KMeans` class and try to understand what the following code is doing.

Student Notes:

class sklearn.cluster.KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='auto')

Quiz Question 2 *Requires coding*. For k=2, what is the cluster center for the cluster with label 1?

```python
k_list = list(range(2,26))
k_inertias = []
k_scores = []
model_list = []
for k in k_list:
    #fit model and append to model list
    model = KMeans(n_clusters=k, random_state=12345).fit(X)
    model_list.append(model)
    #extract labels of each point (n,)
    cluster_assignments = model.labels_
    #calculate silhouette_score from lablels
    score = silhouette_score(X, cluster_assignments, metric='euclidean')
    #append silhouette_score to list of scores
    k_scores.append(score)
    #extract inertia
    inertia = model.inertia_
    #append inertial to list of inertias
    k_inertias.append(inertia)
```

Quiz Question 2 *Requires coding*. For k=2, what is the cluster center for the cluster with label 1?

Documentation Notes:
Attributes:

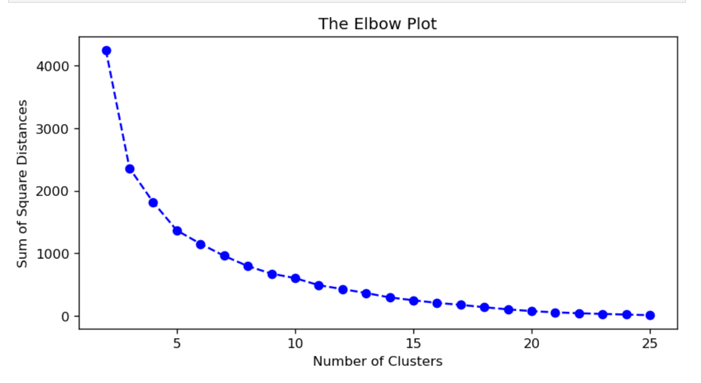
- **cluster*centers*: ndarray of shape (n_clusters, n_features): Coordinates of cluster centers.**
- **labels_: ndarray of shape (n_samples,): Labels of each point**
  - **There's one model per value of k so k points to a model. Model with k = 2 is model_list[0]**
  - **Within this model we need the cluster with label 1. So we need model*list[0].labels* to find the labels.**
  - **Labels are 0 and 1.**
  - **How to we get the center from the cluster with label 1.**
  - **We get all the cluster using cluster*centers* to see what it looks like.**
  - **It is (2, 9) a row for each of cluster, with labels 0 and 1.**

```python
# model_list[0].labels_ #(n,) # k = 2 => labels are 0 and 1, for the first and second
#model_list[0].cluster_centers_#(2, 9)
model_list[0].cluster_centers_[1, :]
```

```
array([44.54,  1.48, 19.62,  0.66,  6.58,  7.3 ,  3.4 , 11.2 ,  5.16])
```

Now, we plot the sum of square distances of samples to their closest cluster center as a function of k, the numebr of clusters.

```python
plt.figure(figsize=(8,4), dpi=120)
plt.title('The Elbow Plot')
plt.xlabel('Number of Clusters')
plt.ylabel('Sum of Square Distances')
_=plt.plot(k_list, k_inertias,'bo--')
```

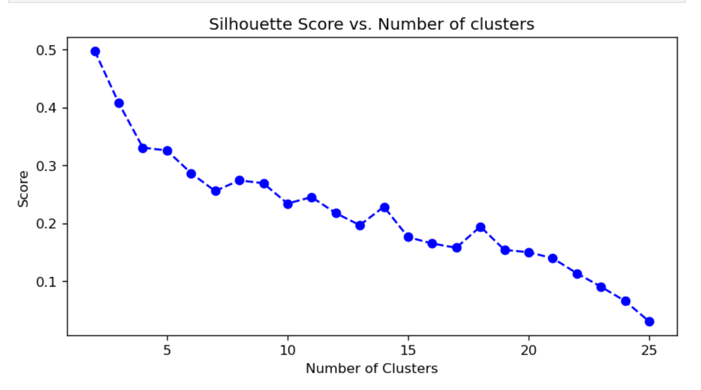


Look at the above *Elbow plot*. Based on this plot, what do you think is a reasonable choice of k?

Next, we plot the so called "*silhouette*" score for the result of the K-Means clustering algorithm for the values of k we implemented above. The silhouette score is a measure of how similar an object is to its cluster compared to other clusters.

- **Try to learn how the silhouette score is defined. For instance, you can look at this Wikipedia page: https://en.wikipedia.org/wiki/Silhouette_(clustering).**
- **What is the range of the silhouette score?**
- **Is a larger value of the silhouette score better or worse?**

```python
plt.figure(figsize=(8,4), dpi=120)
plt.title('Silhouette Score vs. Number of clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Score')
_=plt.plot(k_list, k_scores,'bo--')
```



Based on the silhouette measure, what do you think is a reasonable value for k? Is this the same value that the above elbow plot suggests? Why do think so?