Basics of cluster analysis

In this notebook we explore the issue of selecting the right number of clusters

Import the relevant libraries

In [7]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Set the styles to Seaborn
sns.set()
# Import the KMeans module so we can perform k-means clustering with sklearn
from sklearn.cluster import KMeans
```

Load the data

In [8]:
```python
# Load the country clusters data
data = pd.read_csv('3Country_clusters.csv')
```

In [9]:
```python
# Check out the data manually
data
```

Out[9]:

	Country	Latitude	Longitude	Language
0	USA	44.97	-103.77	English
1	Canada	62.40	-96.80	English
2	France	46.75	2.40	French
3	UK	54.01	-2.53	English
4	Germany	51.15	10.40	German
5	Australia	-25.45	133.11	English

Map the data

In [10]:
```python
# Create a copy of the original dataset
data_mapped = data.copy()
# Map languages with 0, 1, and 2. Note that this is not the best way to do that, but for now we will use it
data_mapped['Language']=data_mapped['Language'].map({'English':0,'French':1,'German':2})
# Check if we did it correctly
data_mapped
```

Out[10]:

	Country	Latitude	Longitude	Language
0	USA	44.97	-103.77	0
1	Canada	62.40	-96.80	0
2	France	46.75	2.40	1
3	UK	54.01	-2.53	0
4	Germany	51.15	10.40	2
5	Australia	-25.45	133.11	0

Select the features

In [11]:
```python
# iloc is a method used to 'slice' data
# 'slice' is not technically correct as there are methods 'slice' which are a bit different
# The term used by pandas is 'selection by position'
# The first argument of identifies the rows we want to keep
# The second - the columns
# When choosing the columns, e.g. a:b, we will keep columns a,a+1,a+2,...,b-1 ; so column b is excluded
x = data_mapped.iloc[:,1:4]
# for this particular case, we are choosing columns 1 and 2
# Note column indices in Python start from 0
```

In [12]:
```python
# Check if we worked correctly
x
```

Out[12]:

	Latitude	Longitude	Language
0	44.97	-103.77	0
1	62.40	-96.80	0
2	46.75	2.40	1
3	54.01	-2.53	0
4	51.15	10.40	2
5	-25.45	133.11	0

Clustering

In [13]:
```python
# Create an object (which we would call kmeans)
# The number in the brackets is K, or the number of clusters we are aiming for
kmeans = KMeans(2)
```

In [14]:
```python
# Fit the input data, i.e. cluster the data in X in K clusters
kmeans.fit(x)
```

Out[14]:
```
KMeans(n_clusters=2)
```

Clustering results

In [15]:
```python
# Create a variable which will contain the predicted clusters for each observation
identified_clusters = kmeans.fit_predict(x)
# Check the result
identified_clusters
```

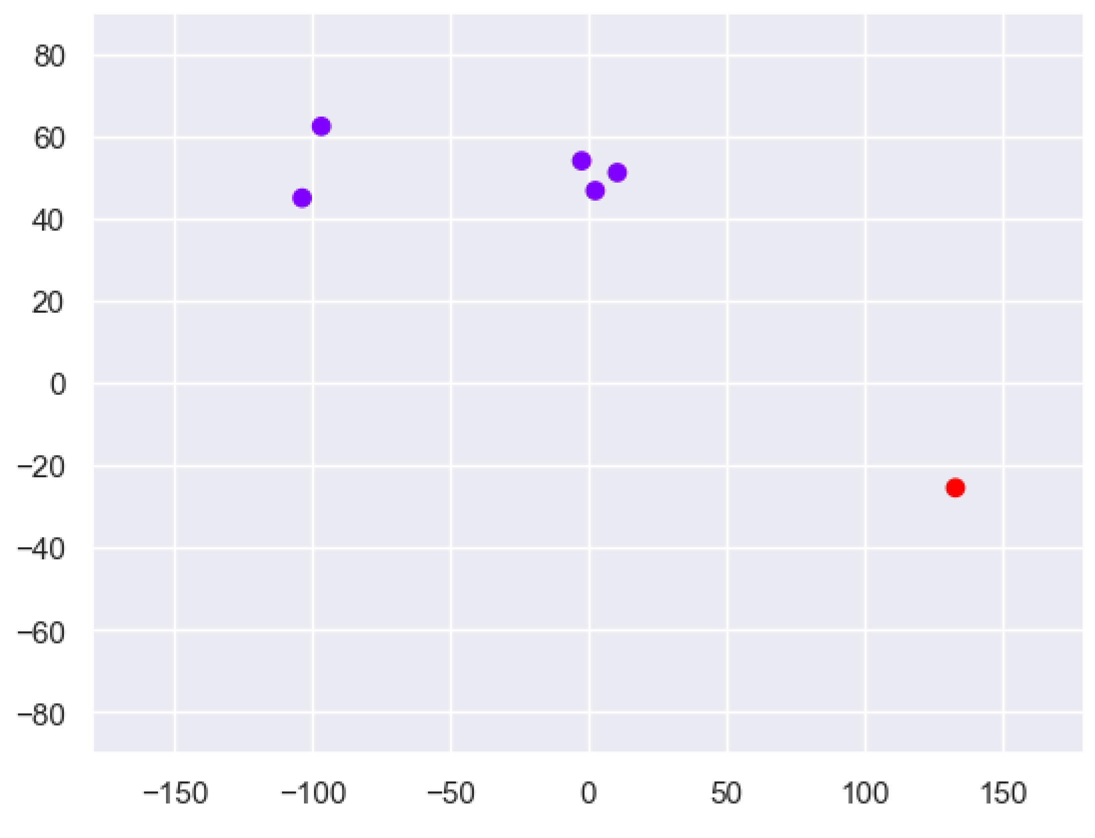
Out[15]:
```
array([0, 0, 0, 0, 0, 1])
```

In [16]:
```python
# Create a copy of the mapped data
data_with_clusters = data_mapped.copy()
# Create a new Series, containing the identified cluster for each observation
data_with_clusters['Cluster'] = identified_clusters
# Check the result
data_with_clusters
```

Out[16]:

	Country	Latitude	Longitude	Language	Cluster
0	USA	44.97	-103.77	0	0
1	Canada	62.40	-96.80	0	0
2	France	46.75	2.40	1	0
3	UK	54.01	-2.53	0	0
4	Germany	51.15	10.40	2	0
5	Australia	-25.45	133.11	0	1

In [17]:
```python
# Plot the data using the longitude and the latitude
# c (color) is an argument which could be coded with a variable
# The variable in this case has values 0,1,2, indicating to plt.scatter, that there are three colors (0,1,2)
# All points in cluster 0 will be the same colour, all points in cluster 1 - another one, etc.
# cmap is the color map. Rainbow is a nice one, but you can check others here: https://matplotlib.org/users/col
plt.scatter(data_with_clusters['Longitude'],data_with_clusters['Latitude'],c=data_with_clusters['Cluster'],cmap
plt.xlim(-180,180)
plt.ylim(-90,90)
plt.show()
```



Selecting the number of clusters

WCSS (within-cluster sum of squares)

WCSS is a measure developed within the ANOVA framework. It gives a very good idea about the different distance between different clusters and within clusters, thus providing us a rule for deciding the appropriate number of clusters.

In [18]:
```python
# Get the WCSS for the current solution
kmeans.inertia_
```

Out[18]:
```
13208.958119999996
```

In [19]:
```python
# Create an empty list
wcss=[]

# Create all possible cluster solutions with a loop
for i in range(1,7):
    # Cluster solution with i clusters
    kmeans = KMeans(i)
    # Fit the data
    kmeans.fit(x)
    # Find WCSS for the current iteration
    wcss_iter = kmeans.inertia_
    # Append the value to the WCSS list
    wcss.append(wcss_iter)
```

```
C:\Users\agawr\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to hav
e a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

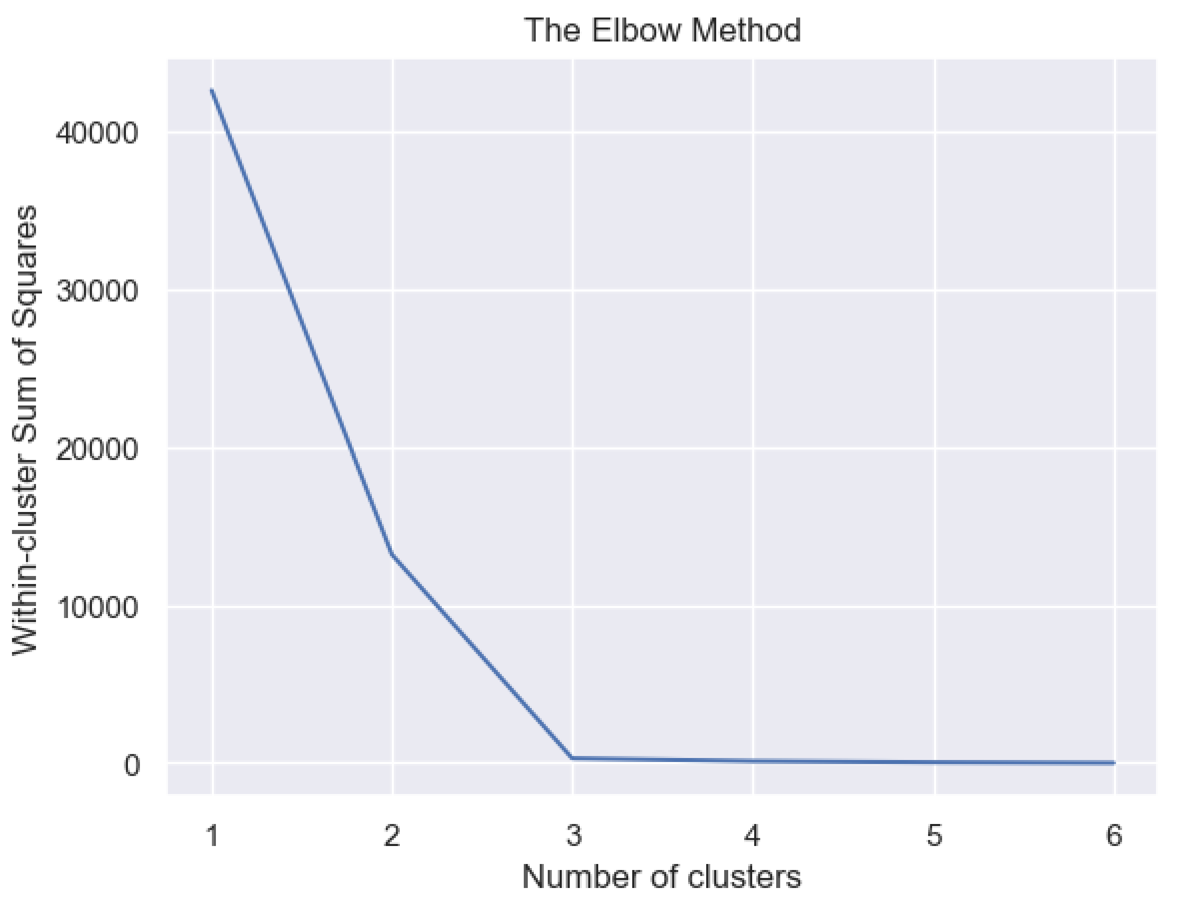
In [20]:
```python
# Let's see what we got
wcss
```

Out[20]:
```
[42605.41356666667,
 13208.958119999996,
 290.10523333333333,
 113.91233333333332,
 39.00624999999998,
 0.0]
```

The Elbow Method

In [21]:
```python
# Create a variable containing the numbers from 1 to 6, so we can use it as X axis of the future plot
number_clusters = range(1,7)
# Plot the number of clusters vs WCSS
plt.plot(number_clusters,wcss)
# Name your graph
plt.title('The Elbow Method')
# Name the x-axis
plt.xlabel('Number of clusters')
# Name the y-axis
plt.ylabel('Within-cluster Sum of Squares')
```

Out[21]:
```
Text(0, 0.5, 'Within-cluster Sum of Squares')
```



Choosing 3 clusters seems to be the best option