Analysis of the indicator 'GDP per capita (current US$)' benchmarking with 12 countries from 2015 to 2020 (based on data availability)

Data Source: A direct access of the world bank data using an API

Indicator Id = ["NY.GDP.PCAP.CD"] = GDP per capita (current US$)

Clustering Analysis

The dataframe created from the above was Inspected to determine the correlation between the variables across the years of consideration. A close range of values were observed measuring the correlation between the variables in the dataframe across the period of consideration which shows that variables in the dataframe as positively correlated(correlation values ranges between positive value 0.99 to 1) as shown below

```
#print(df.describe())
print(df.corr())
print()
```

	YR2015	YR2016	YR2017	YR2018	YR2019	YR2020
YR2015	1.000000	0.996622	0.995845	0.996089	0.995246	0.993357
YR2016	0.996622	1.000000	0.999128	0.998645	0.998857	0.998436
YR2017	0.995845	0.999128	1.000000	0.999656	0.999538	0.998650
YR2018	0.996089	0.998645	0.999656	1.000000	0.999344	0.998554
YR2019	0.995246	0.998857	0.999538	0.999344	1.000000	0.999568
YR2020	0.993357	0.998436	0.998650	0.998554	0.999568	1.000000

A standard technique of rescaling (normalise) data variables at the interval [0, 1] using the formula (xscale = x –min/ max –min) where the min and max are the minimum and maximum value of the variables in the dataset was used to normalise the dataframe as well as its summary statistics before clustering the data as seen below.

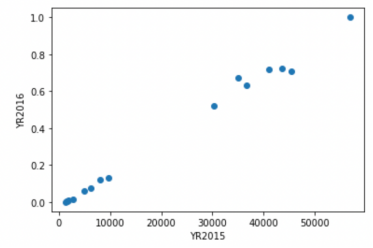
```
# clustering rely on differences apart, so we need to normalize our data to make them have the same range
def norm(column):
    col_min = np.min(column)
    col_max = np.max(column)
    col_norm = (column - col_min) / (col_max - col_min)
# the above is from maths X_norm = X-min(X)/max(x)-min(x) to normalize data to be numerically comparable
    return col_norm
for col in df.columns[1:]:
    df[col] = norm(df[col])

print(df.describe())
df_plot = df.describe()
```

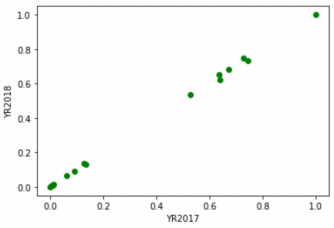
	YR2015	YR2016	YR2017	YR2018	YR2019	YR2020
count	16.000000	16.000000	16.000000	16.000000	16.000000	16.000000
mean	20401.096597	0.337132	0.336692	0.339617	0.329654	0.325603
std	19891.840876	0.354213	0.352025	0.352921	0.341253	0.340256
min	1356.667831	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2459.128735	0.013354	0.009992	0.011698	0.014906	0.015956
50%	8816.538497	0.124460	0.129381	0.135006	0.136872	0.132110
75%	37741.765909	0.680582	0.646568	0.659673	0.626061	0.632415
max	56863.371496	1.000000	1.000000	1.000000	1.000000	1.000000

A function was used to create three exploratory plots by splitting the indicator dataframe into 2 consecutive years (YR2015 & YR2016, YR2017 & YR2018, YR2019 & YR2020).See below.

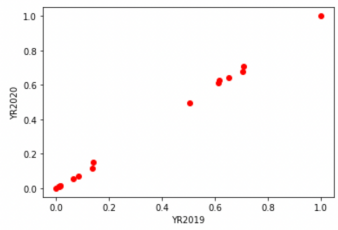
```
def makeplot(df,col1, col2):
    plt.figure()
    plt.scatter(df[col1], df[col2])
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.show()
makeplot(df, "YR2015", "YR2016")
```




```
def makeplot(df,col1, col2):
    plt.figure()
    plt.scatter(df[col1], df[col2], c = 'green')
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.show()
#without defining the makeplot the plts wont run
makeplot(df, "YR2017", "YR2018")
```




```
def makeplot(df,col1, col2):
    plt.figure()
    plt.scatter(df[col1], df[col2], c = 'red')
    plt.xlabel(col1)
    plt.ylabel(col2)
    plt.show()
#without defining the makeplot the plts wont run
makeplot(df, "YR2019", "YR2020")
```

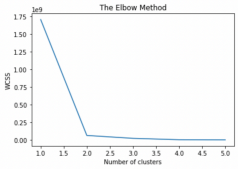


Clustering using KMean(the elbow method) and AgglomerativeClustering was carried out. A new dataframe (X) was created which extracted the two-variable used for clustering from the dataframe. Fits of time series was created using the range of values 1-6 (5 years) to visualize the cluster plots.

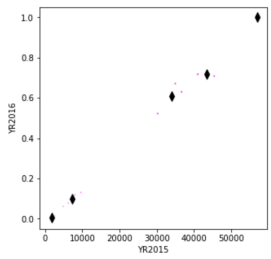
```
X = df[1:6].values
X
```

```
array([[4.35961355e+04, 7.22771890e-01, 7.44559715e-01, 7.31809215e-01,
        7.03851084e-01, 6.78345543e-01],
       [8.01643143e+03, 1.18721684e-01, 1.25366095e-01, 1.36778618e-01,
        1.38383298e-01, 1.49085200e-01],
       [3.66529223e+04, 6.30048246e-01, 6.36308217e-01, 6.51333979e-01,
        6.13994229e-01, 6.10282008e-01],
       [4.54045678e+04, 7.08367562e-01, 6.71719858e-01, 6.84690304e-01,
        6.52934942e-01, 6.42886368e-01],
       [1.77407477e+03, 1.06531032e-02, 1.03907809e-02, 1.26440340e-02,
        1.49719439e-02, 1.63932215e-02]])
```

```
#Performing my clustering using KMean(the elbow method) and AgglomerativeClustering to find the optimal number of clust
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 6):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 6), wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```




```
##### set up agglomerative clustering for 5 clusters
ac = cluster.AgglomerativeClustering(n_clusters=5)
df_fit = df[["YR2015", "YR2016", "YR2017", "YR2018", "YR2019", "YR2020"]].copy()
ac.fit(df_fit)
plt.figure()
labels = ac.labels_
#compute the cluster centre
xcen = []
ycen = []
for ic in range(6):
    xc = np.average(df_fit["YR2015"][labels==ic])
    yc = np.average(df_fit["YR2016"][labels==ic])
    xcen.append(xc)
    ycen.append(yc)
# plot using the labels to select colour
plt.figure(figsize=(5.0,5.0))
plt.scatter(df_fit["YR2015"], df_fit["YR2016"], df_fit["YR2017"], c='magenta', label = 'Value cluster')
# show cluster centres
for ic in range(6):
    plt.plot(xcen[ic], ycen[ic], "dk", markersize=8)
plt.xlabel("YR2015")
plt.ylabel("YR2016")
plt.show()
```



Upper & lower limits of the fitting were estimated using a fitting and plotting function to produce a curve fit plot of the dataset of consideration.

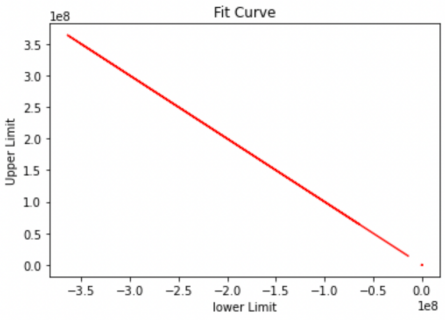
```
#Determine the curve fit
from scipy.optimize import curve_fit
import itertools as iter

def func(X , a , b):
  return a+b*X
popt , pcov = curve_fit(func , df_fit["YR2015"] , df_fit["YR2016"] )
print(popt)
def err_ranges(x, func, param, sigma):
# initiate arrays for lower and upper limits
    lower = func(x, *param)
    upper = lower
    uplow = []
# list to hold upper and lower limits for parameters
    for p,s in zip(param, sigma):
            pmin = p - s
            pmax = p + s
            uplow.append((pmin, pmax))
    pmix = list(iter.product(*uplow))
    for p in pmix:
            y = func(x, *p)
            lower = np.minimum(lower, y)
            upper = np.maximum(upper, y)
    return lower,upper
lower_limit,upper_limit = err_ranges(X,func,popt,np.sum(X,axis=1))
lower_limit,upper_limit
```

```
[-2.49231293e-02  1.77468271e-05]

(array([[-3.49558167e+08, -4.93942761e+04, -4.95689513e+04,
         -4.94667293e+04, -4.92425861e+04, -4.90381057e+04],
        [-6.43121302e+07, -4.45515454e+04, -4.46048143e+04,
         -4.46963096e+04, -4.47091745e+04, -4.47949727e+04],
        [-2.93893734e+08, -4.86509014e+04, -4.87010882e+04,
         -4.88215513e+04, -4.85221948e+04, -4.84924335e+04],
        [-3.64056549e+08, -4.92787952e+04, -4.89849869e+04,
         -4.90889723e+04, -4.88343864e+04, -4.87538259e+04],
        [-1.42665341e+07, -4.36851488e+04, -4.36830457e+04,
```

```
fitting = np.arange(0.0,0.09,0.001)
plt.plot(lower_limit, upper_limit,'r')
plt.xlabel('lower Limit')
plt.ylabel('Upper Limit')
plt.title('Fit Curve')
plt.show()
```

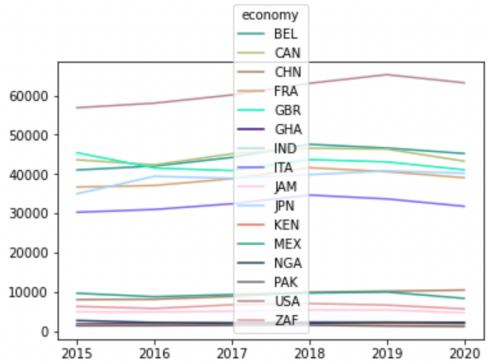


Comparative Analysis

This was done by grouping the country of consideration by continent (four countries per continent) and comparing each country's GDP per capital (current US$) for relevant inferences as follows.

On the overall dataframe:

- amongst the 12 countries, USA top the chart, being the country with the highest GDP per capital (current US$) across the years of consideration.
- Although United Kingdom (GBR) recorded the second highest in 2015, Canada took over the second country with highest GDP per capital (current US$) amongst the 12 countries, in 2016 & 2017
- Belgium However, took over the second highest GDP per capital (current US$) from 2018 to 2020 of the investigated data



On the grouped dataframe,

- South Africa (ZAF) top the chart as the country with the highest GDP per capital (current US$) amongst the four grouped African countries. Nigeria on the other hand, emerged the second African country with highest GDP per capital (current US$) in 2015 & 2016 before Ghana took over from 2017 to 2020
- With an aggregated sum of the GDP per capital (current US$) for the period in view, USA has the highest, followed by Canada (as earlier confirmed by the overall observation above).
- For European grouped countries, United Kingdom top the chart in 2015 which, however, was taken over by Belgium across the years of consideration
- Japan tops the chart as the highest GDP per capital (current US$) amongst the Asian grouped countries followed by China