Which are the most productive countries?

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly as py
import plotly.graph_objs as go

from plotly.offline import init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)

from sklearn.cluster import KMeans, AgglomerativeClustering,
AffinityPropagation
from sklearn.preprocessing import StandardScaler
import os
```

```
df = pd.read_csv("FAO.csv",  encoding = "ISO-8859-1")
df.head()
```

```
  Area Abbreviation  Area Code         Area  Item Code
0               AFG          2  Afghanistan       2511  \
1               AFG          2  Afghanistan       2805
2               AFG          2  Afghanistan       2513
3               AFG          2  Afghanistan       2513
4               AFG          2  Afghanistan       2514

                       Item  Element Code Element         Unit
latitude
0         Wheat and products          5142    Food  1000 tonnes
33.94  \
1  Rice (Milled Equivalent)          5142    Food  1000 tonnes
33.94
2        Barley and products          5521    Feed  1000 tonnes
33.94
3        Barley and products          5142    Food  1000 tonnes
33.94
4         Maize and products          5521    Feed  1000 tonnes
33.94

   longitude  ...   Y2004   Y2005   Y2006   Y2007   Y2008   Y2009
Y2010
0      67.71  ...  3249.0  3486.0  3704.0  4164.0  4252.0  4538.0
4605.0  \
1      67.71  ...   419.0   445.0   546.0   455.0   490.0   415.0
442.0
2      67.71  ...    58.0   236.0   262.0   263.0   230.0   379.0
```

```
315.0
3      67.71  ...   185.0    43.0    44.0    48.0    62.0    55.0
60.0
4      67.71  ...   120.0   208.0   233.0   249.0   247.0   195.0
178.0

    Y2011  Y2012  Y2013
0  4711.0   4810   4895
1   476.0    425    422
2   203.0    367    360
3    72.0     78     89
4   191.0    200    200

[5 rows x 63 columns]
```

DATA DESCRIPTION

Each row of this dataset contains the amount (values represent 1000 tonnes) of Feed/Food produced by each country ('Area') from 1961 to 2013 for a particular Item.

More metadata are included such as Area Abbreviation, Area/Item/Element Code, latitude, longitude, not used in this analysis.

The dataset is reduced containg only columns: Area, Item, Element, Y1961-Y2013

```
df.dtypes[:20]
```

```
Area Abbreviation     object
Area Code              int64
Area                  object
Item Code              int64
Item                  object
Element Code           int64
Element               object
Unit                  object
latitude             float64
longitude            float64
Y1961                float64
Y1962                float64
Y1963                float64
Y1964                float64
Y1965                float64
Y1966                float64
Y1967                float64
Y1968                float64
Y1969                float64
Y1970                float64
dtype: object
```

ADDITION OF POPULATION AND SURFACE DIMENSIONS

In order to make analysis richer I've decided to add to this dataset information about the population and the surface area of each country.

Data is taken again from FAO stats website, considering the year of 2013.

Population data is specify as Million of people. Surface area instead as 1000 acres.

Datasets are merged with existing dataset considering 'Area' as key.

```
#In order to not have problems of consistency:
df['Area'].replace(['Swaziland'], 'Eswatini', inplace=True)
df['Area'].replace(['The former Yugoslav Republic of Macedonia'],
'North Macedonia', inplace=True)

#GET NEW DATA
df_pop = pd.read_csv("FAOSTAT.csv")
df_area = pd.read_csv("countries.csv")
df_pop = pd.DataFrame({'Area': df_pop['Area'] , 'Population':
df_pop['Value'] })
df_area = pd.DataFrame({'Area' : df_area['Area'], 'Surface':
df_area['Value']})
#add missing line
##df_area = df_area.append({'Area' : 'Sudan' , 'Surface' : 1886} ,
ignore_index=True)

#MERGE OF TABLES
d1 = pd.DataFrame(df.loc[:, ['Area', 'Item', 'Element']])
data = pd.merge(d1, df_pop, on='Area', how='left')
new_data = pd.merge(data, df_area, on='Area', how='left')

d2 = df.loc[:, 'Y1961':'Y2013']
data = new_data.join(d2)
data.head()
```

```
          Area                      Item Element  Population  Surface
Y1961
0  Afghanistan        Wheat and products    Food   35530.081  65286.0
1928.0  \
1  Afghanistan  Rice (Milled Equivalent)    Food   35530.081  65286.0
183.0
2  Afghanistan       Barley and products    Feed   35530.081  65286.0
76.0
3  Afghanistan       Barley and products    Food   35530.081  65286.0
237.0
4  Afghanistan        Maize and products    Feed   35530.081  65286.0
210.0

    Y1962   Y1963   Y1964   Y1965  ...   Y2004   Y2005   Y2006   Y2007
```

```
0  1904.0  1666.0  1950.0  2001.0  ...  3249.0  3486.0  3704.0  4164.0
\
1   183.0   182.0   220.0   220.0  ...   419.0   445.0   546.0   455.0

2    76.0    76.0    76.0    76.0  ...    58.0   236.0   262.0   263.0

3   237.0   237.0   238.0   238.0  ...   185.0    43.0    44.0    48.0

4   210.0   214.0   216.0   216.0  ...   120.0   208.0   233.0   249.0


    Y2008   Y2009   Y2010   Y2011  Y2012  Y2013
0  4252.0  4538.0  4605.0  4711.0   4810   4895
1   490.0   415.0   442.0   476.0    425    422
2   230.0   379.0   315.0   203.0    367    360
3    62.0    55.0    60.0    72.0     78     89
4   247.0   195.0   178.0   191.0    200    200

[5 rows x 58 columns]
```

```
print('Number of different Countries: ' , df['Area'].unique().size)
print('Number of different Items: ' , df['Item'].unique().size)
```

```
Number of different Countries:  174
Number of different Items:  115
```

DATA CLEANING

Datasets of this kind mostly of the time contains missing values, represented by NaN.

Let's see if there are some missing values on this dataset. Each yellow line represent some missing values, is possible to understand that there are a lot of them, the biggest part are before 1991. This is because in those period (end of cold war) a lot of new countries were born.

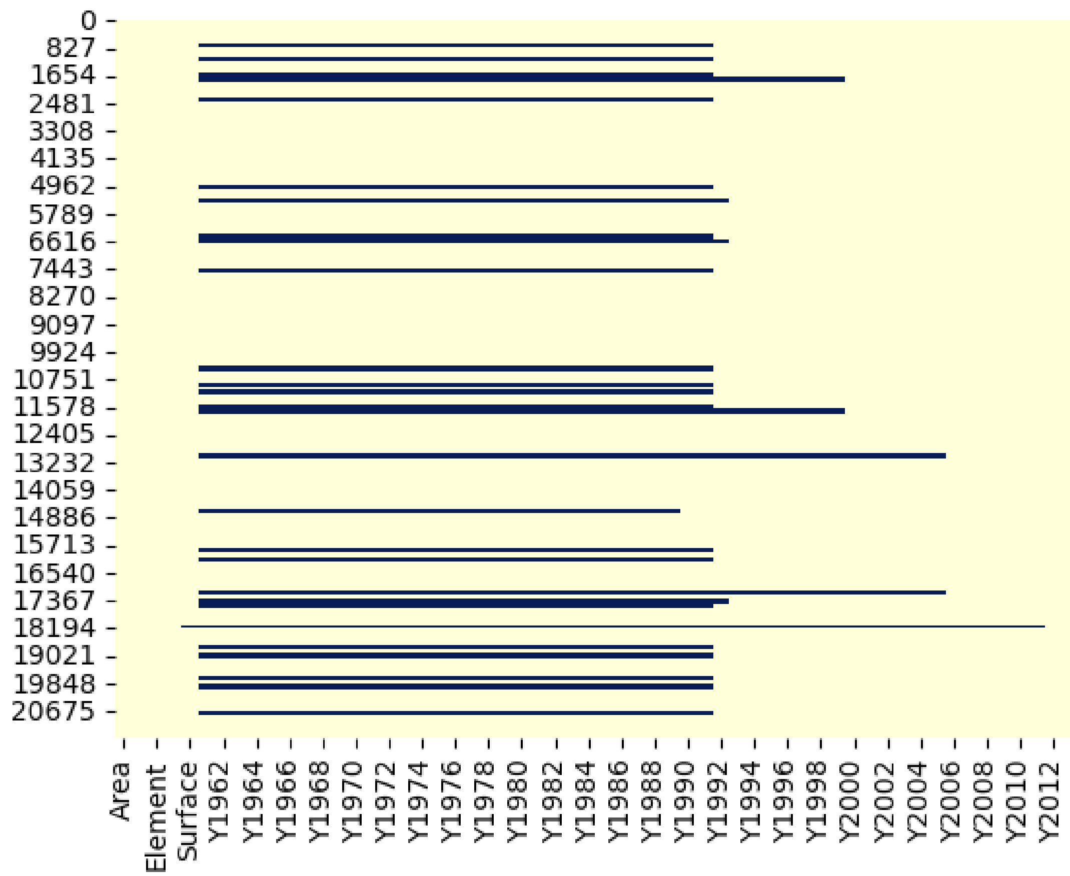
According to this constraint different ways of proceeding may be taken into consideration, **one is to compute analysis only from 1993 where there are a less amount of missing values and computing analisys only for the last 2 decade 1993-2013.**

The other way is to considering all the years removing from analysis the missing rows and so the missing countries. There is even the possibility to substitute NaN with 0

I decide to substitute missing values with 0 in order to make ranking and then considering only the last 20 years for the clustering analysis, due to a less amount of missing values limiting this constraint.

Let's visualize bettere those missing values.

```
#Graph of missing values
sns.heatmap(data.isnull(),cbar=False,cmap='YlGnBu')
plt.show()
```



From this graph is possible to visualize missing values represented by blue lines, only the last two years doesn't contains missing values.

```
# Total number of missing values per year
print('YEAR  MISSING VALUES')
print(df.shape)
print (df.loc[:, 'Y1961':'Y2013'].isnull().sum())
```

```
YEAR  MISSING VALUES
(21477, 63)
Y1961    3539
Y1962    3539
Y1963    3539
Y1964    3539
Y1965    3539
Y1966    3539
Y1967    3539
Y1968    3539
```

```
Y1969    3539
Y1970    3539
Y1971    3539
Y1972    3539
Y1973    3539
Y1974    3539
Y1975    3539
Y1976    3539
Y1977    3539
Y1978    3539
Y1979    3539
Y1980    3539
Y1981    3539
Y1982    3539
Y1983    3539
Y1984    3539
Y1985    3539
Y1986    3539
Y1987    3539
Y1988    3539
Y1989    3539
Y1990    3415
Y1991    3415
Y1992     987
Y1993     612
Y1994     612
Y1995     612
Y1996     612
Y1997     612
Y1998     612
Y1999     612
Y2000     349
Y2001     349
Y2002     349
Y2003     349
Y2004     349
Y2005     349
Y2006     104
Y2007     104
Y2008     104
Y2009     104
Y2010     104
Y2011     104
Y2012       0
Y2013       0
dtype: int64
```

```
df1 = data[data.isna().any(axis=1)]
df1.head()
```

```
        Area                       Item Element  Population  Surface
Y1961
679  Armenia        Wheat and products    Feed     2930.45   2847.0
NaN  \
680  Armenia        Wheat and products    Food     2930.45   2847.0
NaN
681  Armenia  Rice (Milled Equivalent)    Feed     2930.45   2847.0
NaN
682  Armenia  Rice (Milled Equivalent)    Food     2930.45   2847.0
NaN
683  Armenia       Barley and products    Feed     2930.45   2847.0
NaN

     Y1962  Y1963  Y1964  Y1965  ...  Y2004  Y2005  Y2006  Y2007
Y2008
679    NaN    NaN    NaN    NaN  ...   69.0   59.0   46.0   67.0
57.0  \
680    NaN    NaN    NaN    NaN  ...  490.0  433.0  445.0  412.0
428.0
681    NaN    NaN    NaN    NaN  ...    0.0    0.0    0.0    0.0
0.0
682    NaN    NaN    NaN    NaN  ...   11.0   14.0   17.0   15.0
13.0
683    NaN    NaN    NaN    NaN  ...   68.0   57.0   33.0   86.0
76.0

     Y2009  Y2010  Y2011  Y2012  Y2013
679   56.0   61.0   65.0     92     93
680  391.0  372.0  386.0    377    389
681    0.0    0.0    0.0      0      0
682   13.0   11.0    9.0      9      9
683  102.0   86.0  124.0    121    137

[5 rows x 58 columns]
```

```python
#Total number of missing values for Area
values_per_area = data.pivot_table(index=['Area'], aggfunc='size')
df1 = data[data.isna().any(axis=1)]
df_missing_area = df1.pivot_table(index=['Area'], aggfunc='size')
df_missing_area.sort_values()
```

```
Area
Turkmenistan               90
Tajikistan                102
Sudan                     104
Ethiopia                  116
Montenegro                118
Uzbekistan                123
Bosnia and Herzegovina    124
Oman                      124
```

```
Kyrgyzstan                 124
Azerbaijan                 124
Serbia                     127
Luxembourg                 127
Czechia                    129
Croatia                    129
Slovakia                   130
North Macedonia            130
Republic of Moldova        130
Belarus                    131
Slovenia                   132
Armenia                    133
Georgia                    133
Ukraine                    134
Estonia                    135
Latvia                     136
Belgium                    136
Russian Federation         137
Lithuania                  140
Kazakhstan                 141
dtype: int64
```

Countries shown in the list above represent the one for which there are missing values.

The biggest part are countries born after the dissolve of Jugoslavia and URSS

RANKING

```
year_list = list(df.iloc[:,10:].columns)
df_new = df.pivot_table(values=year_list,columns = 'Element',
index=['Area'], aggfunc='sum') #for each country sum over years
separatly Food&Feed
df_fao = df_new.T
df_fao.head()

Area           Afghanistan  Albania  Algeria  Angola  Antigua and
Barbuda
      Element

Y1961 Feed           720.0     94.0     83.0   118.0
2.0  \
      Food          8761.0   1612.0   7405.0  4716.0
90.0
Y1962 Feed           720.0    108.0     94.0   118.0
2.0
      Food          8694.0   1641.0   7141.0  4657.0
92.0
Y1963 Feed           736.0    124.0     63.0   116.0
```

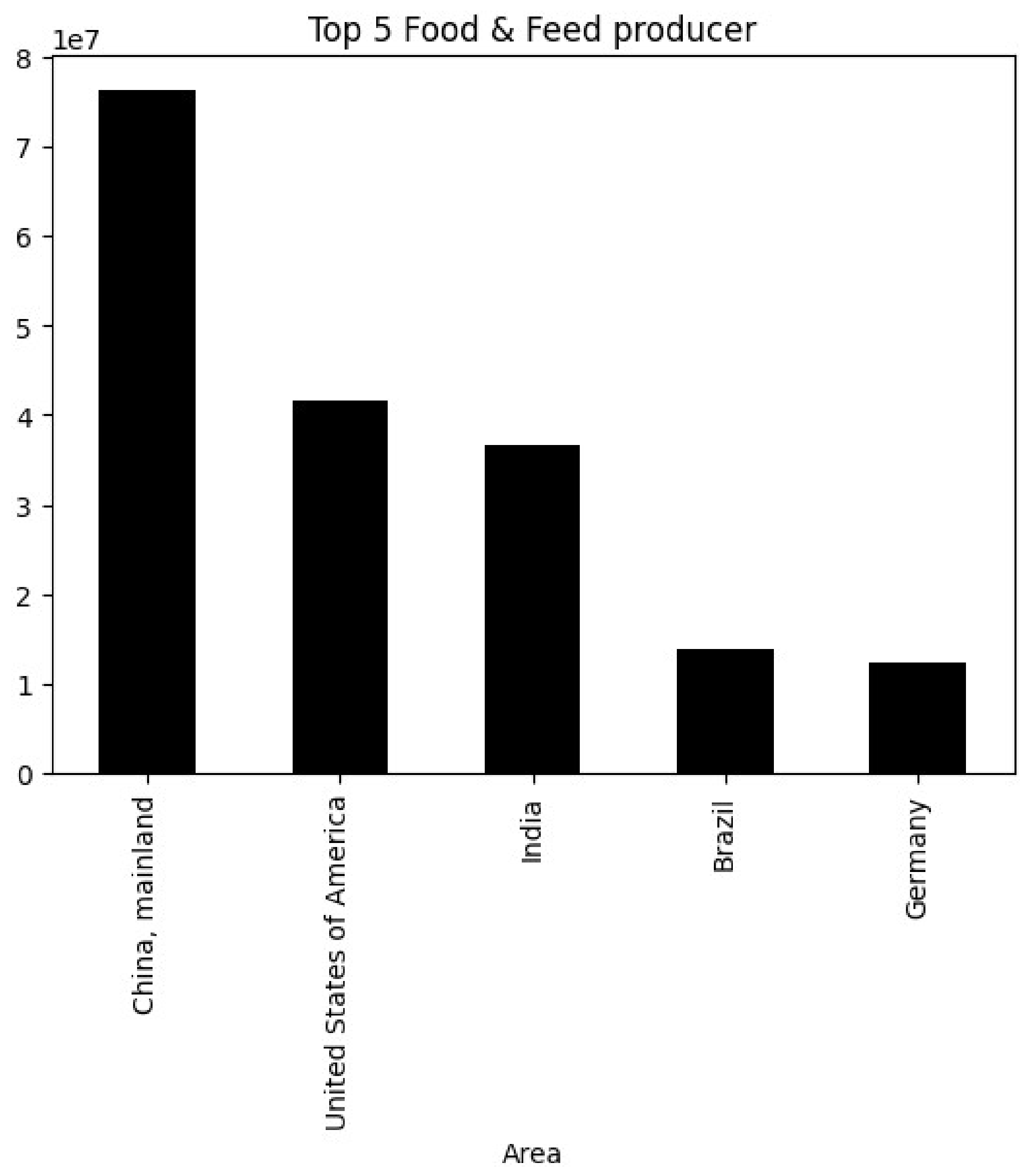
```
2.0

Area             Argentina  Armenia  Australia  Austria  Azerbaijan  ...

      Element                                                        ...

Y1961 Feed          9552.0      0.0     7813.0   9539.0         0.0  ...
\
      Food         33850.0      0.0    17982.0  13003.0         0.0  ...

Y1962 Feed          7553.0      0.0     8982.0   9807.0         0.0  ...

      Food         33231.0      0.0    18636.0  12820.0         0.0  ...

Y1963 Feed          6527.0      0.0     9556.0  10229.0         0.0  ...


Area             United Republic of Tanzania  United States of America
Uruguay
      Element

Y1961 Feed                             134.0                  234413.0
975.0  \
      Food                           12233.0                  324934.0
3656.0
Y1962 Feed                             138.0                  228541.0
970.0
      Food                           12672.0                  327778.0
3478.0
Y1963 Feed                             163.0                  223570.0
1004.0

Area             Uzbekistan  Vanuatu  Venezuela (Bolivarian Republic of)

      Element

Y1961 Feed              0.0      4.0                               360.0
\
      Food              0.0     93.0                              9163.0

Y1962 Feed              0.0      6.0                               291.0

      Food              0.0     95.0                              9078.0

Y1963 Feed              0.0      6.0                               321.0


Area             Viet Nam   Yemen  Zambia  Zimbabwe
      Element
Y1961 Feed         2104.0   167.0    90.0     180.0
      Food        21752.0  2815.0  2886.0    3080.0
```

```
Y1962 Feed        2512.0   168.0    90.0     216.0
      Food       22708.0  2870.0  2967.0    3287.0
Y1963 Feed        2614.0   172.0    70.0     190.0

[5 rows x 174 columns]
```

RANK FOR FOOD AND FEED

```python
# Finding the Top 5 producer of Feed and Food from 1961 to 2013
df_fao_tot = df_fao.sum(axis=0).sort_values(ascending=False).head()
df_fao_tot.plot(kind='bar', title='Top 5 Food & Feed producer',
color='black')
```

```
<Axes: title={'center': 'Top 5 Food & Feed producer'}, xlabel='Area'>
```



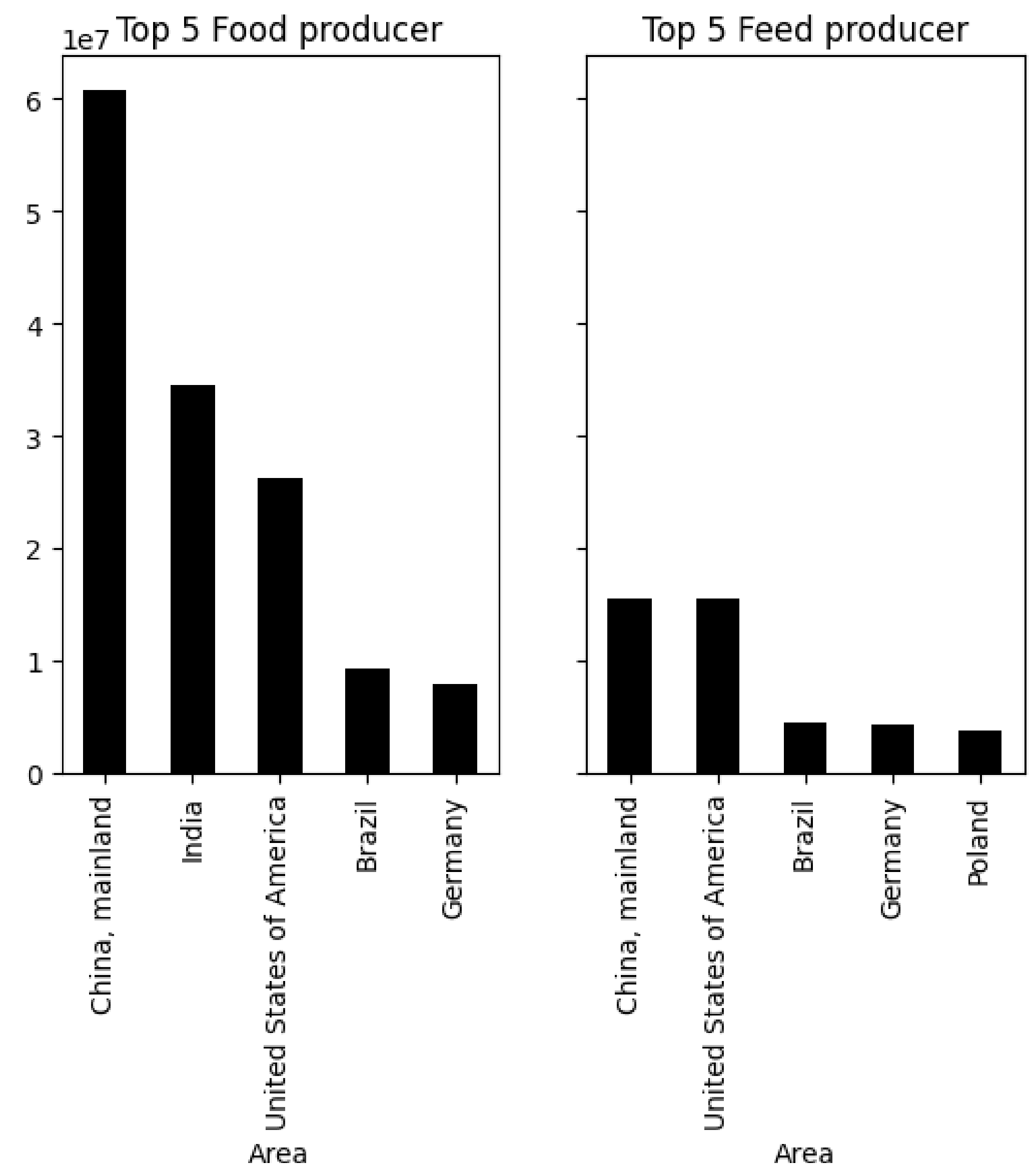
COMMENT:

It not surprisingly appeared as China is the country with the biggest amount of Food and Feed production, it is the most populous country in the world and also one of the biggest, with over 1 billion people to feed, there are a lot of food to produce. China is followed by USA and India respectively 3rd and 2nd most populous countries. It is logical to think that this rank isn't affected only by the amount of population of a country, but also by GDP, total cultivable area, and geographical position.

RANK FOR JUST FOOD OR FEED

```
#Producer of just Food
df_food = df_fao.xs('Food', level=1, axis=0)
df_food_tot = df_food.sum(axis=0).sort_values(ascending=False).head()
#Producer of just Feed
df_feed = df_fao.xs('Feed', level=1, axis=0)
df_feed = df_feed.sum(axis=0).sort_values(ascending=False).head()

#Plot
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
df_food_tot.plot(kind='bar', title='Top 5 Food producer',
color='black', ax=ax1)
df_feed.plot(kind='bar', title='Top 5 Feed producer', color='black',
ax=ax2 )
```

```
<Axes: title={'center': 'Top 5 Feed producer'}, xlabel='Area'>
```



COMMENT:

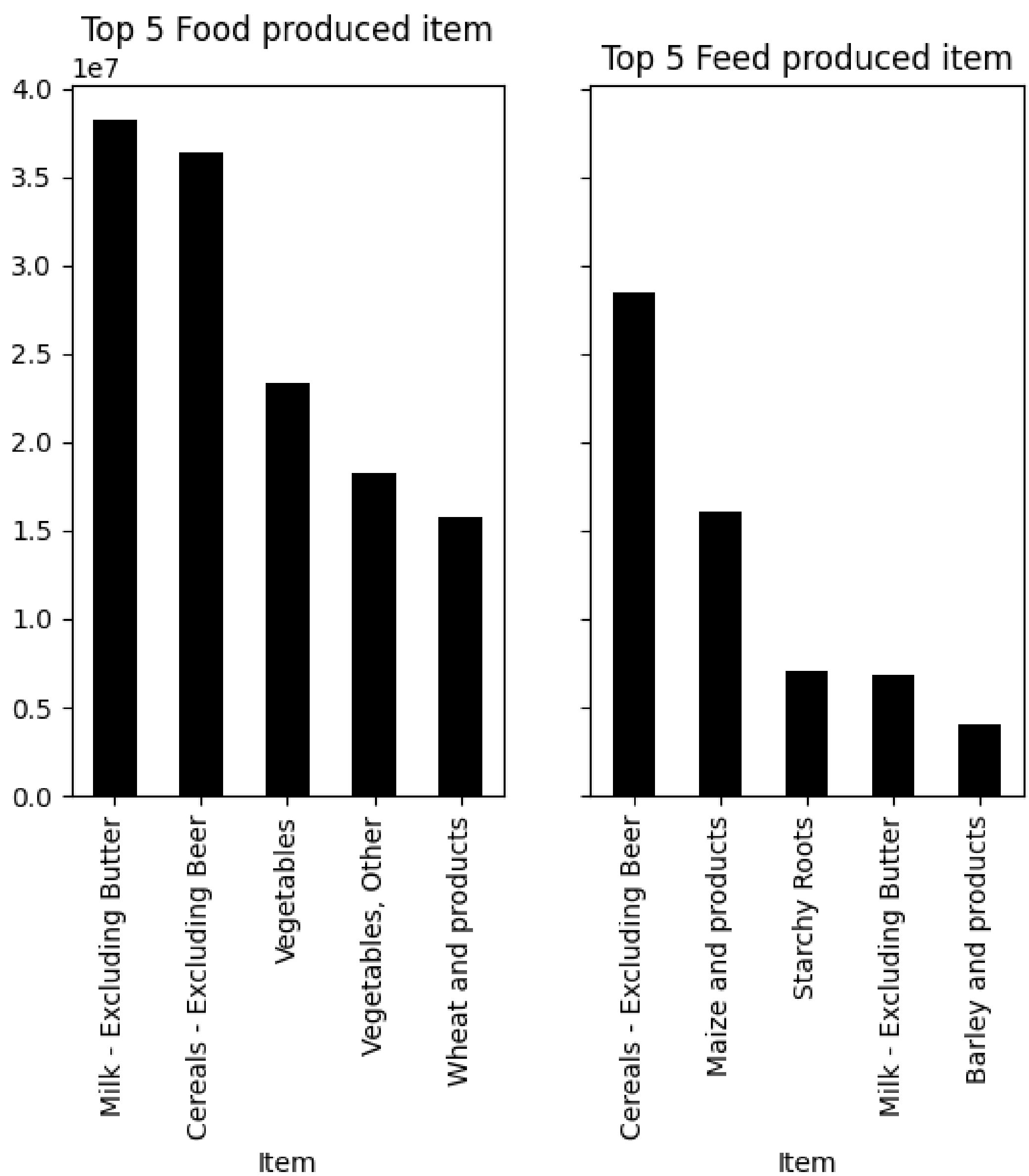
The graph on the left contains only countries producer of Food for humans, is the same as before and this means that Feed production is irrelevant on the total amount. The right graph instead contains only Feed for animals, here is disappeared **India**, probably because the biggest part of Feed is intended for cattle (cows) and those animals are considered sacred there.

RANK OF MOST PRODUCED ITEMS

```
#Rank of most Produced Items
df_item = df.pivot_table(values=year_list,
columns='Element',index=['Item'], aggfunc='sum')
df_item = df_item.T
#FOOD
```

```python
df_food_item = df_item.xs('Food', level=1, axis=0)
df_food_item =
df_food_item.sum(axis=0).sort_values(ascending=False).head()
#FEED
df_feed_item = df_item.xs('Feed', level=1, axis=0)
df_feed_item =
df_feed_item.sum(axis=0).sort_values(ascending=False).head()
#Plot
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True)
df_food_item.plot(kind='bar', title='Top 5 Food produced item',
color='black', ax=ax1)
df_feed_item.plot(kind='bar', title='Top 5 Feed produced item',
color='black' , ax=ax2)
```

```
<Axes: title={'center': 'Top 5 Feed produced item'}, xlabel='Item'>
```



COMMENT:

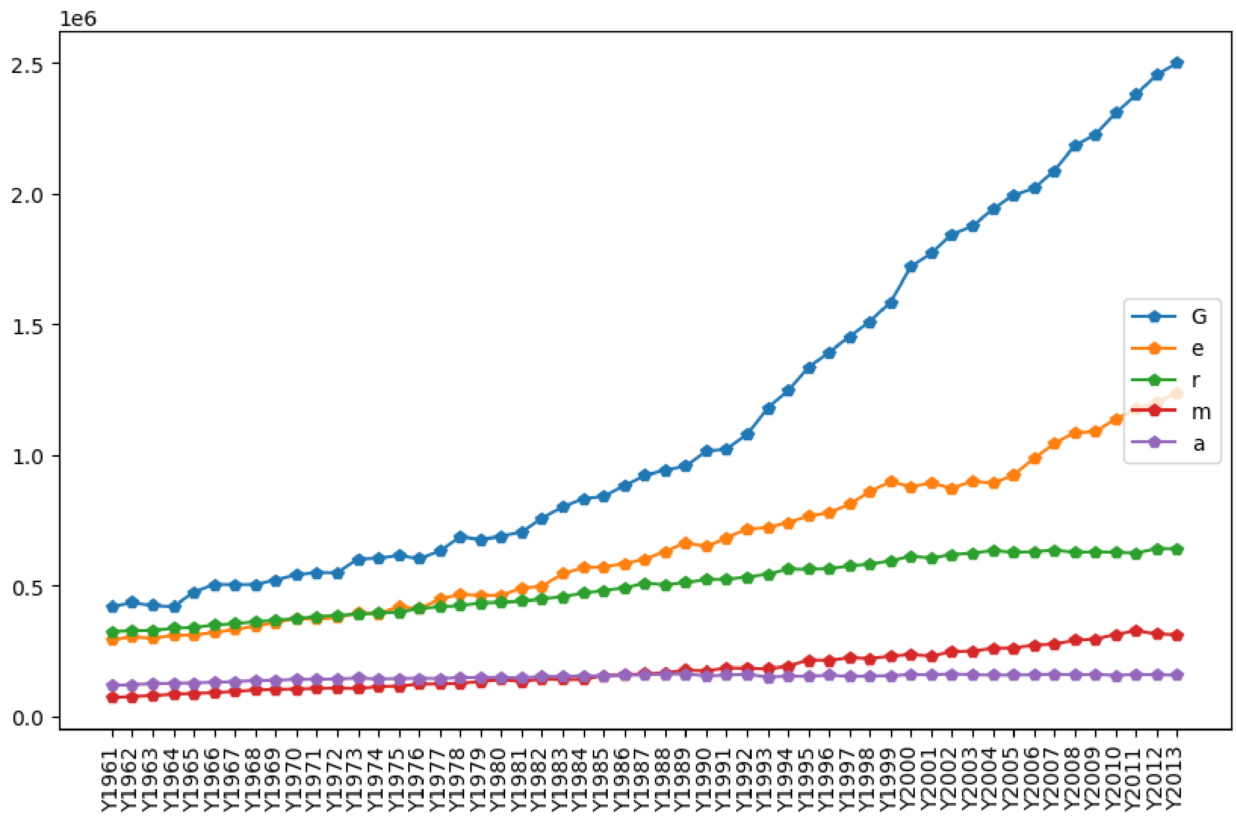
From the calculation appears how Milk is the most produced item in the world from 1960 to 2013, followed by cereals and vegetables, regarding feed intended to animals, Cereals and Maize are the most produced items.

SHOWING DEVELOPING COUNTRIES

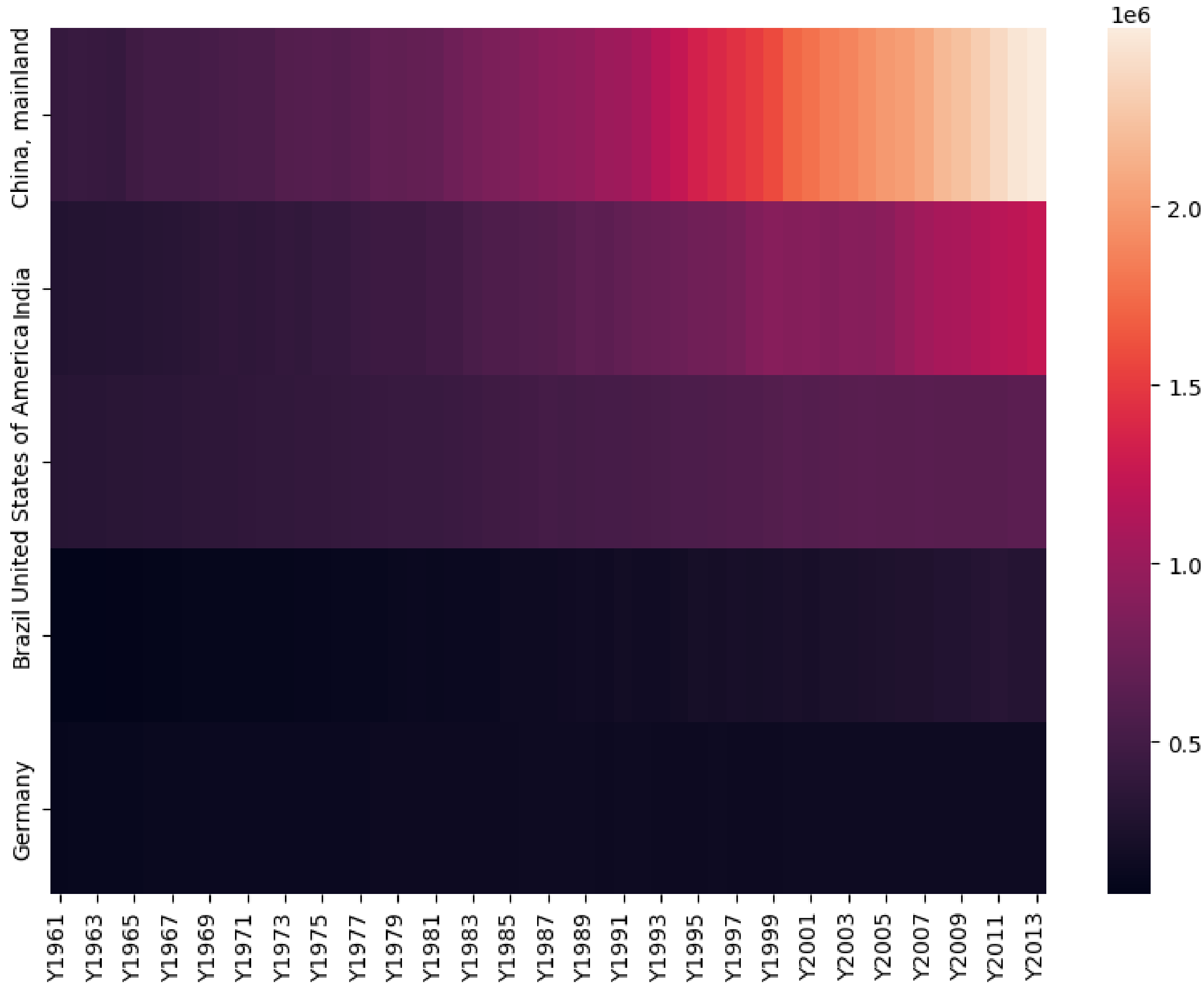
```
df_food_tot
#df_food[i]
```

```
Area
China, mainland              60723839.0
India                        34570279.0
```

```
United States of America    26227192.0
Brazil                       9393856.0
Germany                      7977690.0
dtype: float64

# Visualization of the top 5 producer countries among years
plt.figure(figsize = (10,6))
top_5 = []
for i in df_food_tot.index:
    print(i)
    year = df_food[i]
    top_5.append(year)
    plt.plot(year, marker='p')
    plt.xticks(df_food.index, rotation='vertical')
    plt.legend(i, loc='right')

China, mainland
India
United States of America
Brazil
Germany
```



```
fig, ax = plt.subplots(figsize=(10,7))
sns.heatmap(data=pd.DataFrame(top_5),linewidths=0, ax=ax)
plt.show()
```



CLUSTERING

Clustering is an unsupervised learning method used in order to find new groups from data such that intra-cluster distances are minimized and inter-cluster distances are maximised. It doesn't not require any knowledge about the data, similarly to classification algorithms, clustering algorithms assign (or predict) a number to each data point, indicating which cluster a particular point belongs to. There are two different type of clusering:

- PARTITIONAL
- HIERARCHICAL

Hierarchical clustering requires only a similarity measure, while partitional clustering requires stronger assumptions such as number of clusters and the initial centers. Hierarchical clustering returns a much more meaningful and subjective division of clusters but partitional clustering results in exactly k clusters.

Most common clustering algorithm are: K-Means, Hierarchical, Density-based (DBSCAN).

DATA PREPARATION

Considering that initially not all the countries of the world are included in the dataset (due probably to political problems) and considering that there are some missing values for existing countries, opted to use only information about the last 20 years avalaible (1993-2013) in order also to simplify the amount of data. Remaining missing values are filled with 0 (Belgium, Luxembourg, Montenegro, Serbia, Sudan)

I provide different data structures:

- COUNTRY - POPULATION - SURFACE - Y1993 ------ Y2013
- COUNTRY - POPULATION - SURFACE - TOTAL
- ITEM - Y1993 ------ Y2013 (world wide)
- ITEM - Y1993 ------ Y2013 (for **Italy**)

```
d3 = df.loc[:, 'Y1993':'Y2013'] #take only last 20 years
data1 = new_data.join(d3) #recap: new_data does not contains years
data

d4 = data1.loc[data1['Element'] == 'Food'] #get just food
d5 = d4.drop('Element', axis=1)
d5 = d5.fillna(0) #substitute missing values with 0

year_list = list(d3.iloc[:,:].columns)
d6 = d5.pivot_table(values=year_list, index=['Area'], aggfunc='sum')

italy = d4[d4['Area'] == 'Italy']
italy = italy.pivot_table(values=year_list, index=['Item'],
aggfunc='sum')
italy = pd.DataFrame(italy.to_records())

item = d5.pivot_table(values=year_list, index=['Item'], aggfunc='sum')
item = pd.DataFrame(item.to_records())

d5 = d5.pivot_table(values=year_list, index=['Area', 'Population',
'Surface'], aggfunc='sum')
area = pd.DataFrame(d5.to_records())
d6.loc[:, 'Total'] = d6.sum(axis=1)
d6 = pd.DataFrame(d6.to_records())
d = pd.DataFrame({'Area' : d6['Area'] , 'Total': d6['Total'] ,
'Population': area['Population'], 'Surface': area['Surface']})

d.head()
```

	Area	Total	Population	Surface
0	Afghanistan	326921.0	35530.081	65286.0

```
1              Albania  120172.0    2930.187    2740.0
2              Algeria  896114.0   41318.142  238174.0
3               Angola  352564.0   29784.193  124670.0
4  Antigua and Barbuda    2079.0     102.012      44.0
```

```python
import plotly.graph_objects as geo

data_ = dict(type = 'choropleth',
locations = d['Area'],
locationmode = 'country names',
z = d['Total'],
text = d['Area'],
colorbar = {'title':'Tons of food'})
layout = dict(title = 'Total Production of Food 1993-2013',
geo = dict(showframe = False,
projection = {'type': 'mercator'}))
choromap3 = geo.Figure(data = [data_], layout=layout)
iplot(choromap3)
```

{"config":{"linkText":"Export to plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":[{"colorbar":{"title":{"text":"Tons of food"}},"locationmode":"country names","locations":["Afghanistan","Albania","Algeria","Angola","Antigua and Barbuda","Argentina","Armenia","Australia","Austria","Azerbaijan","Bahamas","Bangladesh","Barbados","Belarus","Belgium","Belize","Benin","Bermuda","Bolivia (Plurinational State of)","Bosnia and Herzegovina","Botswana","Brazil","Brunei Darussalam","Bulgaria","Burkina Faso","Cabo Verde","Cambodia","Cameroon","Canada","Central African Republic","Chad","Chile","China, Hong Kong SAR","China, Macao SAR","China, Taiwan Province of","China, mainland","Colombia","Congo","Costa Rica","Croatia","Cuba","Cyprus","Czechia","Côte d'Ivoire","Democratic People's Republic of Korea","Denmark","Djibouti","Dominica","Dominican Republic","Ecuador","Egypt","El Salvador","Estonia","Eswatini","Ethiopia","Fiji","Finland","France","French Polynesia","Gabon","Gambia","Georgia","Germany","Ghana","Greece","Grenada","Guatemala","Guinea","Guinea-Bissau","Guyana","Haiti","Honduras","Hungary","Iceland","India","Indonesia","Iran (Islamic Republic of)","Iraq","Ireland","Israel","Italy","Jamaica","Japan","Jordan","Kazakhstan","Kenya","Kiribati","Kuwait","Kyrgyzstan","Lao People's Democratic Republic","Latvia","Lebanon","Lesotho","Liberia","Lithuania","Luxembourg","Madagascar","Malawi","Malaysia","Maldives","Mali","Malta","Mauritania","Mauritius","Mexico","Mongolia","Montenegro","Morocco","Mozambique","Myanmar","Namibia","Nepal","Netherlands","New Caledonia","New Zealand","Nicaragua","Niger","Nigeria","North

Macedonia","Norway","Oman","Pakistan","Panama","Paraguay","Peru","Phil
ippines","Poland","Portugal","Republic of Korea","Republic of
Moldova","Romania","Russian Federation","Rwanda","Saint Kitts and
Nevis","Saint Lucia","Saint Vincent and the Grenadines","Samoa","Sao
Tome and Principe","Saudi Arabia","Senegal","Serbia","Sierra
Leone","Slovakia","Slovenia","Solomon Islands","South
Africa","Spain","Sri
Lanka","Sudan","Suriname","Sweden","Switzerland","Tajikistan","Thailan
d","Timor-Leste","Togo","Trinidad and
Tobago","Tunisia","Turkey","Turkmenistan","Uganda","Ukraine","United
Arab Emirates","United Kingdom","United Republic of Tanzania","United
States of America","Uruguay","Uzbekistan","Vanuatu","Venezuela
(Bolivarian Republic of)","Viet
Nam","Yemen","Zambia","Zimbabwe"],"text":
["Afghanistan","Albania","Algeria","Angola","Antigua and
Barbuda","Argentina","Armenia","Australia","Austria","Azerbaijan","Bah
amas","Bangladesh","Barbados","Belarus","Belgium","Belize","Benin","Be
rmuda","Bolivia (Plurinational State of)","Bosnia and
Herzegovina","Botswana","Brazil","Brunei
Darussalam","Bulgaria","Burkina Faso","Cabo
Verde","Cambodia","Cameroon","Canada","Central African
Republic","Chad","Chile","China, Hong Kong SAR","China, Macao
SAR","China, Taiwan Province of","China,
mainland","Colombia","Congo","Costa
Rica","Croatia","Cuba","Cyprus","Czechia","Côte d'Ivoire","Democratic
People's Republic of Korea","Denmark","Djibouti","Dominica","Dominican
Republic","Ecuador","Egypt","El
Salvador","Estonia","Eswatini","Ethiopia","Fiji","Finland","France","F
rench
Polynesia","Gabon","Gambia","Georgia","Germany","Ghana","Greece","Gren
ada","Guatemala","Guinea","Guinea-
Bissau","Guyana","Haiti","Honduras","Hungary","Iceland","India","Indon
esia","Iran (Islamic Republic
of)","Iraq","Ireland","Israel","Italy","Jamaica","Japan","Jordan","Kaz
akhstan","Kenya","Kiribati","Kuwait","Kyrgyzstan","Lao People's
Democratic
Republic","Latvia","Lebanon","Lesotho","Liberia","Lithuania","Luxembou
rg","Madagascar","Malawi","Malaysia","Maldives","Mali","Malta","Maurit
ania","Mauritius","Mexico","Mongolia","Montenegro","Morocco","Mozambiq
ue","Myanmar","Namibia","Nepal","Netherlands","New Caledonia","New
Zealand","Nicaragua","Niger","Nigeria","North
Macedonia","Norway","Oman","Pakistan","Panama","Paraguay","Peru","Phil
ippines","Poland","Portugal","Republic of Korea","Republic of
Moldova","Romania","Russian Federation","Rwanda","Saint Kitts and
Nevis","Saint Lucia","Saint Vincent and the Grenadines","Samoa","Sao
Tome and Principe","Saudi Arabia","Senegal","Serbia","Sierra
Leone","Slovakia","Slovenia","Solomon Islands","South
Africa","Spain","Sri
Lanka","Sudan","Suriname","Sweden","Switzerland","Tajikistan","Thailan

d","Timor-Leste","Togo","Trinidad and
Tobago","Tunisia","Turkey","Turkmenistan","Uganda","Ukraine","United
Arab Emirates","United Kingdom","United Republic of Tanzania","United
States of America","Uruguay","Uzbekistan","Vanuatu","Venezuela
(Bolivarian Republic of)","Viet
Nam","Yemen","Zambia","Zimbabwe"],"type":"choropleth","z":
[326921,120172,896114,352564,2079,1260870,93011,779031,354176,234677,1
0534,1797998,7764,366256,301287,7560,179902,2027,186292,127086,38010,5
363608,9063,232529,290771,10269,185684,438739,1319821,80357,122502,442
651,199523,11460,614834,39003086,1114673,79749,114224,154677,343217,30
030,393672,466487,464102,229300,10936,2888,209343,320778,2255352,12773
9,51901,22362,908083,20265,217594,2499790,7358,40676,21497,120486,3324
458,669110,538562,2511,223753,187570,21351,19244,138918,136890,344326,
11037,19900340,3932518,2123485,482171,188771,262200,2593024,76638,3400
471,126141,505006,701186,2326,77108,146567,110075,81263,136665,32418,5
3157,130558,15243,316546,244860,532251,7819,223695,16499,56113,29633,3
019528,50326,11853,787088,369041,763194,41060,498234,710143,6282,14672
6,82299,223057,3641538,64388,182271,75669,3112101,69708,155647,630532,
1698382,1460041,462108,1477941,108532,870340,4815293,268797,1009,4491,
2781,5658,4321,566755,165602,113940,84660,165403,76559,12120,1054383,1
695650,360730,84646,11149,377793,303248,118220,1477069,15110,98826,306
71,309576,2563733,122484,744769,1608335,147012,2423034,847789,12779748
,103895,683251,6059,682364,1540944,281252,165288,167171]}],"layout":
{"geo":{"projection":
{"type":"mercator"},"showframe":false},"template":{"data":{"bar":
[{"error_x":{"color":"#2a3f5f"},"error_y":
{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}],"
carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
```

```
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF
6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
```

"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"",
"title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Total Production of Food 1993-2013"}}}

DATA FOR CLUSTERING

```
X = pd.DataFrame({'Total': d['Total'], 'Surface' : d['Surface'],
'Population' : d['Population']})
X.head()
```

	Total	Surface	Population
0	326921.0	65286.0	35530.081
1	120172.0	2740.0	2930.187
2	896114.0	238174.0	41318.142
3	352564.0	124670.0	29784.193
4	2079.0	44.0	102.012

Statistical overview of the data

The following statistical measures can be seen for each column using the describe-function of DataFrame of the pandas library:

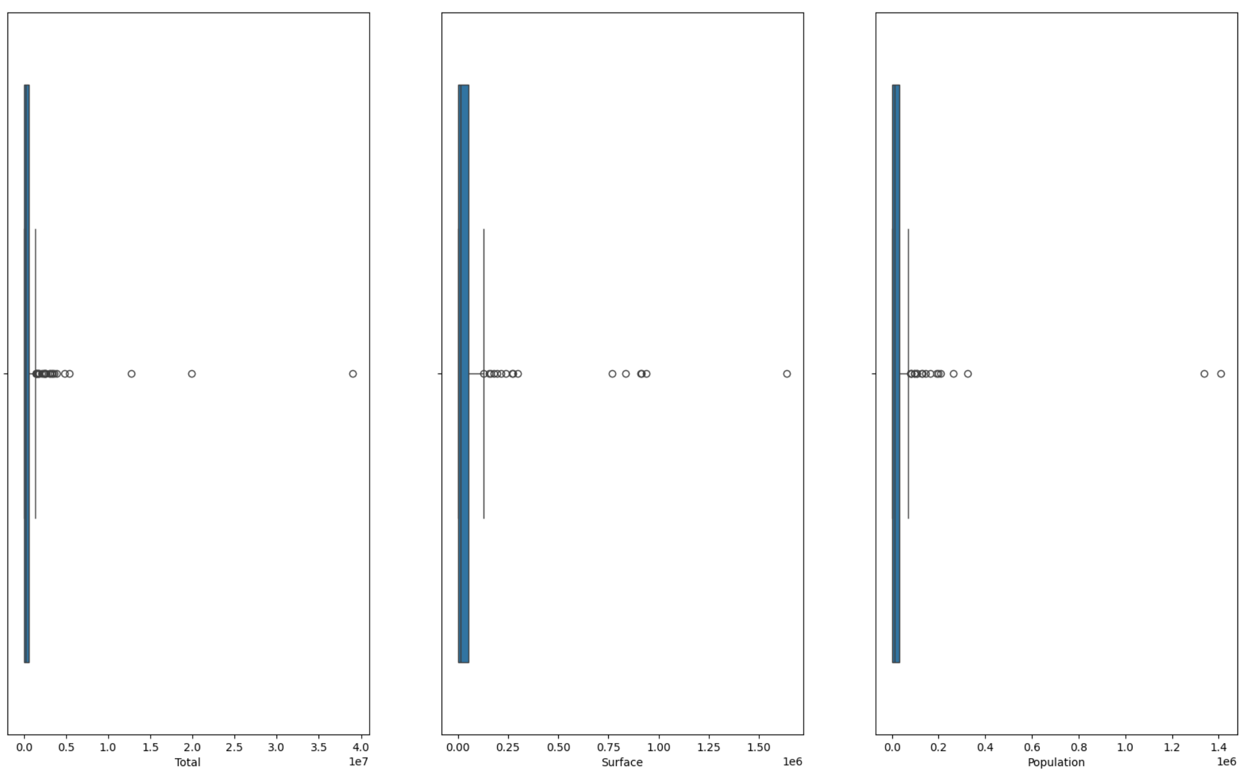
- count: number of samples
- mean: the mean of this attribute among all samples
- std: the standard deviation of this attribute
- min: the minimal value of this attribute
- 25%: the lower percentile
- 50%: the median
- 75%: the upper percentile
- max: the maximal value of this attribute

```
X.describe()
```

	Total	Surface	Population
count	1.740000e+02	1.740000e+02	1.740000e+02
mean	9.536121e+05	6.977175e+04	4.233778e+04
std	3.500840e+06	1.925312e+05	1.516214e+05
min	1.009000e+03	0.000000e+00	5.534500e+01
25%	5.818175e+04	2.802250e+03	2.622920e+03
50%	1.859880e+05	1.295350e+04	9.585345e+03
75%	6.028142e+05	5.237000e+04	3.116425e+04
max	3.900309e+07	1.637687e+06	1.409517e+06

BOXPLOT

```
fig = plt.figure(figsize=(20,26))

ax1 = fig.add_subplot(231)
ax1=sns.boxplot(x='Total',data=X, orient='v')
ax2 = fig.add_subplot(232)
ax2=sns.boxplot(x='Surface',data=X,orient='v')
ax3 = fig.add_subplot(233)
ax3=sns.boxplot(x='Population',data=X, orient='v')

c:\Python311\Lib\site-packages\seaborn\_base.py:1606: UserWarning:

Vertical orientation ignored with only `x` specified.

c:\Python311\Lib\site-packages\seaborn\_base.py:1606: UserWarning:

Vertical orientation ignored with only `x` specified.

c:\Python311\Lib\site-packages\seaborn\_base.py:1606: UserWarning:

Vertical orientation ignored with only `x` specified.
```



COMMENT

From boxplot graphs is possible to see that there are some **outliers** that will surely affect the clustering.

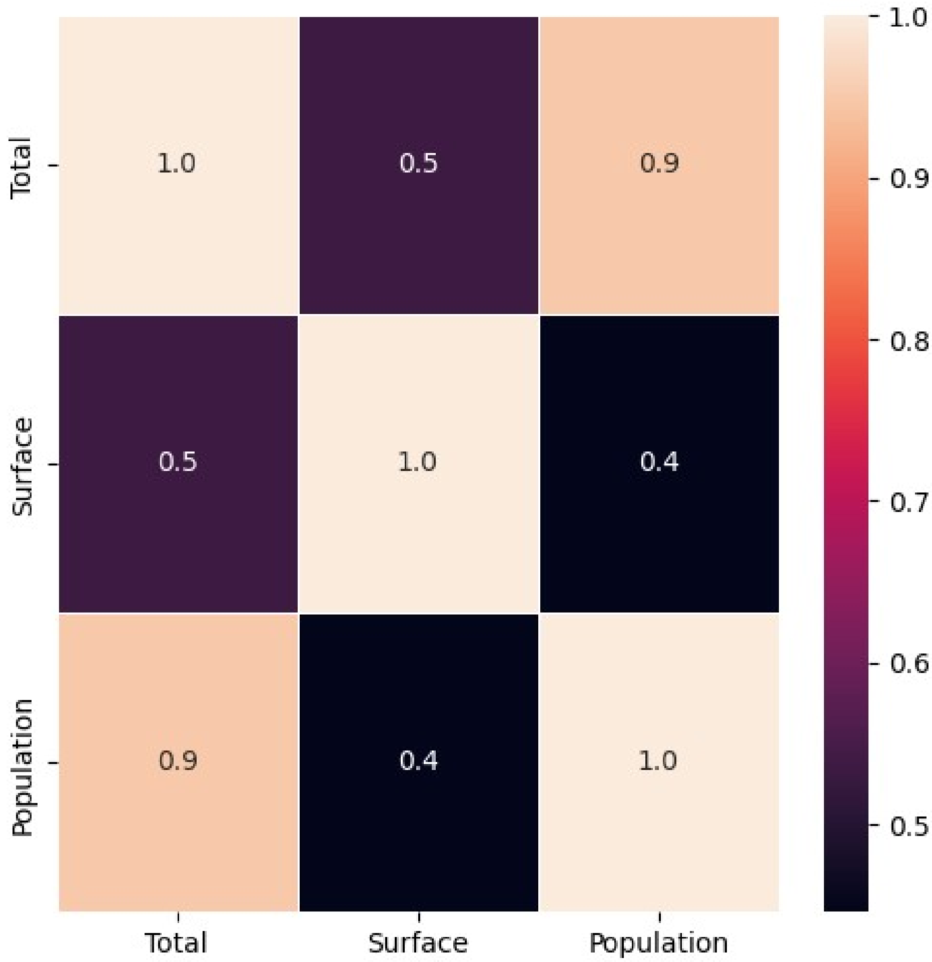
In order to cluster better the Countries a solution could be to cut-off bigger outliers, for example China and India

CORRELATION OF VARIABLES

The correlation matrix is simply a table of correlations. The most common correlation coefficient is Pearson's correlation coefficient, which compares two interval variables or ratio variables, and it's used in this case.

```
f,ax = plt.subplots(figsize=(6, 6))
sns.heatmap(X.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

```
<Axes: >
```



Population and Total sum of productive have a correlation value of ** 0.9 ** This means that countries that are a lot populous produce a lot of food in order to feed them all.

Population and Surface instead have a correlation value of ** 0.4 ** Is reasonable to unserstand why they are not correlated

Total amount of Food and Surface have a correlation value of ** 0.5 ** Not all the surface of a country is cultivable

K-MEANS CLUSTERING

K-Means clustering is one of the simplest and most commonly used clustering algorithms. It tries to find cluster centers that are representative of certain regions of the data. The algorithm alternates between two steps: assigning each data point to the closest cluster center, and then setting each cluster center as the mean of the data points that are assigned to it. The algorithm is finished when the assignment of instances to clusters no longer changes.

It's a partitional complete approach, it requires a metric (how to measure distance) and data needs to be normalized. The most common way to measure distances is with sum of squared error:

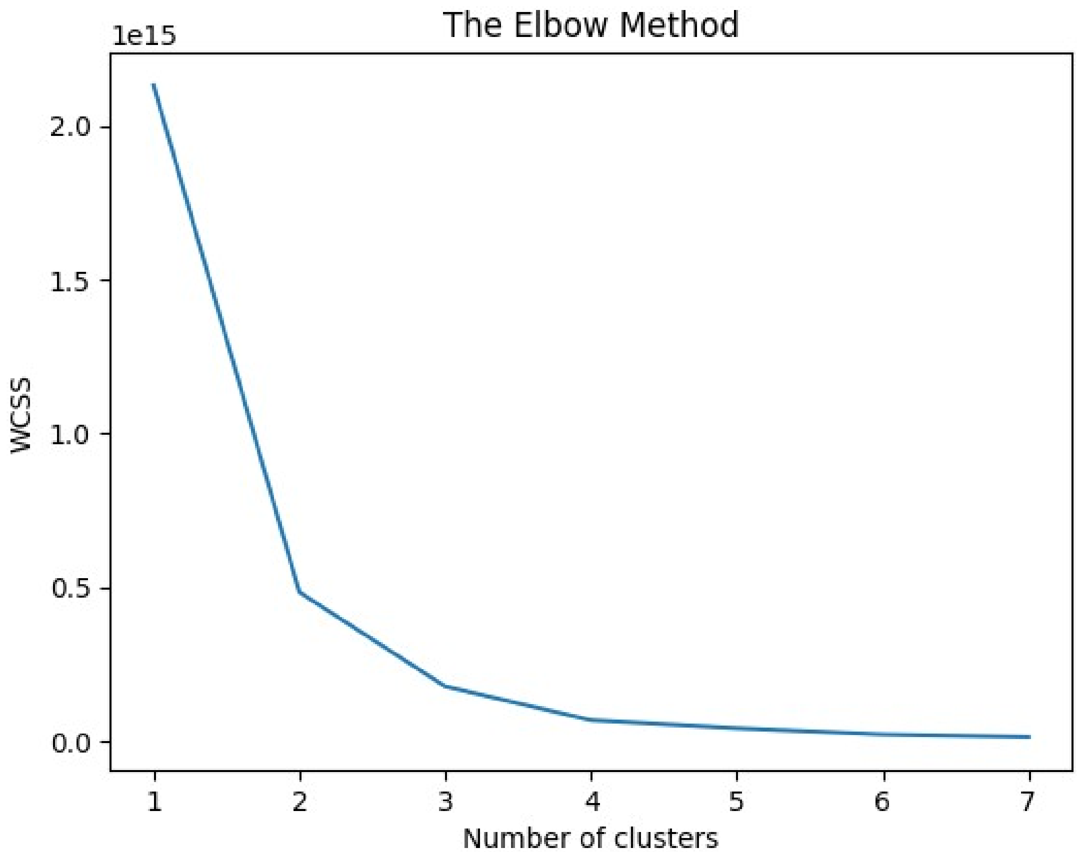
$$SSE = \sum_{i=0}^{n} d(x_i, c_j)^2$$

K-Means has the advantage that it’s pretty fast, as all we’re really doing is computing the distances between points and group centers. It thus has a linear complexity O(n)

ELBOW METHOD

It's possible to find the best value of K on a plot of SSE at varing of number of K, from the graph you choose the value of K for which there is the higher slope in our case K=2

```
wcss = []
for i in range(1,8):
    kmeans = KMeans(n_clusters=i,init='k-means+
+',max_iter=300,n_init=7,random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1,8),wcss)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



K-MEANS

```
def K_Means(X, n):
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    model = KMeans(n)
    model.fit(X)
    clust_labels = model.predict(X)
    cent = model.cluster_centers_
    return (clust_labels, cent)
```

N_CLUSTER = 2

```
clust_labels, cent = K_Means(X, 2)
kmeans = pd.DataFrame(clust_labels)
X.insert((X.shape[1]),'kmeans',kmeans)

c:\Python311\Lib\site-packages\sklearn\cluster\_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4.
Set the value of `n_init` explicitly to suppress the warning
```

```
clust_labels
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
def Plot3dClustering(n, X, type_c):
    data = []
    clusters = []
    colors = ['rgb(228,26,28)','rgb(55,126,184)','rgb(77,175,74)']

    for i in range(n):
        name = i
        color = colors[i]
        x = X[ X[type_c] == i ]['Total']
        y = X[ X[type_c] == i ]['Population']
        z = X[ X[type_c] == i ]['Surface']

        trace = dict(
            name = name,
            x = x, y = y, z = z,
            type = "scatter3d",
            mode = 'markers',
            marker = dict( size=4, color=color, line=dict(width=0) ) )
        data.append( trace )

        cluster = dict(
            color = color,
            opacity = 0.1,
            type = "mesh3d",
            alphahull = 7,
            name = "y",
            x = x, y = y, z = z )
        data.append( cluster )

    layout = dict(
        width=800,
        height=550,
```

```
        autosize=False,
        title='3D Clustering Plot',
        scene=dict(
            xaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                title='Total Production',
                backgroundcolor='rgb(230, 230,230)'
            ),
            yaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                title='Population',
                backgroundcolor='rgb(230, 230,230)'
            ),
            zaxis=dict(
                gridcolor='rgb(255, 255, 255)',
                zerolinecolor='rgb(255, 255, 255)',
                showbackground=True,
                title='Surface Area',
                backgroundcolor='rgb(230, 230,230)'
            ),
            aspectratio = dict( x=1, y=1, z=0.7 ),
            aspectmode = 'manual'
        ),
    )

    fig = dict(data=data, layout=layout)
    iplot(fig, filename='total_surface_population_plot',
validate=False)

Plot3dClustering(n=2, X=X , type_c='kmeans')
```


{"config":{"linkText":"Export to
plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":
[{"marker":{"color":"rgb(228,26,28)","line":
{"width":0},"size":4},"mode":"markers","name":0,"type":"scatter3d","x"
:
[326921,120172,896114,352564,2079,1260870,93011,779031,354176,234677,1
0534,1797998,7764,366256,301287,7560,179902,2027,186292,127086,38010,5
363608,9063,232529,290771,10269,185684,438739,1319821,80357,122502,442
651,199523,11460,614834,1114673,79749,114224,154677,343217,30030,39367
2,466487,464102,229300,10936,2888,209343,320778,2255352,127739,51901,2
2362,908083,20265,217594,2499790,7358,40676,21497,120486,3324458,66911
0,538562,2511,223753,187570,21351,19244,138918,136890,344326,11037,393
2518,2123485,482171,188771,262200,2593024,76638,3400471,126141,505006,
701186,2326,77108,146567,110075,81263,136665,32418,53157,130558,15243,
316546,244860,532251,7819,223695,16499,56113,29633,3019528,50326,11853

,787088,369041,763194,41060,498234,710143,6282,146726,82299,223057,364
1538,64388,182271,75669,3112101,69708,155647,630532,1698382,1460041,46
2108,1477941,108532,870340,4815293,268797,1009,4491,2781,5658,4321,566
755,165602,113940,84660,165403,76559,12120,1054383,1695650,360730,8464
6,11149,377793,303248,118220,1477069,15110,98826,30671,309576,2563733,
122484,744769,1608335,147012,2423034,847789,12779748,103895,683251,605
9,682364,1540944,281252,165288,167171],"y":
[35530.081,2930.187,41318.142,29784.193,102.012,44271.041,2930.45,2445
0.561,8735.453,9827.589,395.361,164669.751,285.719,9468.338,11429.336,
374.681,11175.692,61.349,11051.6,3507.017,2291.661,209288.278,428.697,
7084.571,19193.382,546.388,16005.373,24053.727,36624.199,4659.08,14899
.994,18054.726,7364.883,622.567,23626.456,49065.615,5260.75,4905.769,4
189.353,11484.636,1179.551,10618.303,24294.75,25490.965,5733.551,956.9
85,73.925,10766.998,16624.858,97553.151,6377.853,1309.632,1367.254,104
957.438,905.502,5523.231,64979.548,283.007,2025.137,2100.568,3912.061,
82114.224,28833.629,11159.773,107.825,16913.503,12717.176,1861.283,777
.859,10981.229,9265.067,9721.559,335.025,263991.379,81162.788,38274.61
8,4761.657,8321.57,59359.9,2890.299,127484.45,9702.353,18204.499,49699
.862,116.398,4136.528,6045.117,6858.16,1949.67,6082.357,2233.339,4731.
906,2890.297,583.455,25570.895,18622.104,31624.264,436.33,18541.98,430
.835,4420.184,1265.138,129163.276,3075.647,628.96,35739.58,29668.834,5
3370.609,2533.794,29304.998,17035.938,276.255,4705.818,6217.581,21477.
348,190886.311,2083.16,5305.383,4636.262,197015.955,4098.587,6811.297,
32165.485,104918.09,38170.712,10329.506,50982.212,4051.212,19679.306,1
43989.754,12208.407,55.345,178.844,109.897,196.44,204.327,32938.213,15
850.567,8790.574,7557.212,5447.662,2079.976,611.343,56717.156,46354.32
1,20876.917,40533.33,563.402,9910.701,8476.005,8921.343,69037.513,1296
.311,7797.694,1369.125,11532.127,80745.02,5758.075,42862.958,44222.947
,9400.145,66181.585,57310.019,324459.463,3456.75,31910.641,276.244,319
77.065,95540.8,28250.42,17094.13,16529.904],"z":
[65286,2740,238174,124670,44,273669,2847,768230,8253.1,8265.9,1001,130
17,43,20290.8,3028,2281,11276,5.4,108330,5120,56673,835814,527,10856,2
7360,403,17652,47271,909351,62298,125920,74353.2,105,3.03,3541,110950,
34150,5106,5596,10422,924,7722,31800,12041,4199,2318,75,4831,24836,995
45,2072,4347,1720,100000,1827,30389,54755.7,366,25767,1012,6949,34888,
22754,12890,34,10716,24572,2812,19685,2756,11189,9053,10025,181157,162
876,43412.8,6889,2164,29414,1083,36456,8878,269970,56914,81,1782,19180
,23080,6219,1023,3036,9632,6267.5,243,58180,9428,32855,30,122019,32,10
3070,203,194395,155356,1345,44630,78638,65308,82329,14335,3369,1828,26
331,12034,126670,91077,2522,36512.3,30950,77088,7434,39730,128000,2981
7,30621,9160.52,9742.6,3288,23003,1637687,2467,26,61,39,283,96,214969,
19253,8746,7218,4808.8,2014,2799,121309,50021,6271,0,15600,40734,3951.
6,13996,51089,1487,5439,513,15536,76963,46993,20052,57932,7102,24193,8
8580,914742,17502,42540,1219,88205,31007,52797,74339,38685]},
{"alphahull":7,"color":"rgb(228,26,28)","name":"y","opacity":0.1,"type
":"mesh3d","x":
[326921,120172,896114,352564,2079,1260870,93011,779031,354176,234677,1
0534,1797998,7764,366256,301287,7560,179902,2027,186292,127086,38010,5
363608,9063,232529,290771,10269,185684,438739,1319821,80357,122502,442

651,199523,11460,614834,1114673,79749,114224,154677,343217,30030,39367
2,466487,464102,229300,10936,2888,209343,320778,2255352,127739,51901,2
2362,908083,20265,217594,2499790,7358,40676,21497,120486,3324458,66911
0,538562,2511,223753,187570,21351,19244,138918,136890,344326,11037,393
2518,2123485,482171,188771,262200,2593024,76638,3400471,126141,505006,
701186,2326,77108,146567,110075,81263,136665,32418,53157,130558,15243,
316546,244860,532251,7819,223695,16499,56113,29633,3019528,50326,11853
,787088,369041,763194,41060,498234,710143,6282,146726,82299,223057,364
1538,64388,182271,75669,3112101,69708,155647,630532,1698382,1460041,46
2108,1477941,108532,870340,4815293,268797,1009,4491,2781,5658,4321,566
755,165602,113940,84660,165403,76559,12120,1054383,1695650,360730,8464
6,11149,377793,303248,118220,1477069,15110,98826,30671,309576,2563733,
122484,744769,1608335,147012,2423034,847789,12779748,103895,683251,605
9,682364,1540944,281252,165288,167171],"y":
[35530.081,2930.187,41318.142,29784.193,102.012,44271.041,2930.45,2445
0.561,8735.453,9827.589,395.361,164669.751,285.719,9468.338,11429.336,
374.681,11175.692,61.349,11051.6,3507.017,2291.661,209288.278,428.697,
7084.571,19193.382,546.388,16005.373,24053.727,36624.199,4659.08,14899
.994,18054.726,7364.883,622.567,23626.456,49065.615,5260.75,4905.769,4
189.353,11484.636,1179.551,10618.303,24294.75,25490.965,5733.551,956.9
85,73.925,10766.998,16624.858,97553.151,6377.853,1309.632,1367.254,104
957.438,905.502,5523.231,64979.548,283.007,2025.137,2100.568,3912.061,
82114.224,28833.629,11159.773,107.825,16913.503,12717.176,1861.283,777
.859,10981.229,9265.067,9721.559,335.025,263991.379,81162.788,38274.61
8,4761.657,8321.57,59359.9,2890.299,127484.45,9702.353,18204.499,49699
.862,116.398,4136.528,6045.117,6858.16,1949.67,6082.357,2233.339,4731.
906,2890.297,583.455,25570.895,18622.104,31624.264,436.33,18541.98,430
.835,4420.184,1265.138,129163.276,3075.647,628.96,35739.58,29668.834,5
3370.609,2533.794,29304.998,17035.938,276.255,4705.818,6217.581,21477.
348,190886.311,2083.16,5305.383,4636.262,197015.955,4098.587,6811.297,
32165.485,104918.09,38170.712,10329.506,50982.212,4051.212,19679.306,1
43989.754,12208.407,55.345,178.844,109.897,196.44,204.327,32938.213,15
850.567,8790.574,7557.212,5447.662,2079.976,611.343,56717.156,46354.32
1,20876.917,40533.33,563.402,9910.701,8476.005,8921.343,69037.513,1296
.311,7797.694,1369.125,11532.127,80745.02,5758.075,42862.958,44222.947
,9400.145,66181.585,57310.019,324459.463,3456.75,31910.641,276.244,319
77.065,95540.8,28250.42,17094.13,16529.904],"z":
[65286,2740,238174,124670,44,273669,2847,768230,8253.1,8265.9,1001,130
17,43,20290.8,3028,2281,11276,5.4,108330,5120,56673,835814,527,10856,2
7360,403,17652,47271,909351,62298,125920,74353.2,105,3.03,3541,110950,
34150,5106,5596,10422,924,7722,31800,12041,4199,2318,75,4831,24836,995
45,2072,4347,1720,100000,1827,30389,54755.7,366,25767,1012,6949,34888,
22754,12890,34,10716,24572,2812,19685,2756,11189,9053,10025,181157,162
876,43412.8,6889,2164,29414,1083,36456,8878,269970,56914,81,1782,19180
,23080,6219,1023,3036,9632,6267.5,243,58180,9428,32855,30,122019,32,10
3070,203,194395,155356,1345,44630,78638,65308,82329,14335,3369,1828,26
331,12034,126670,91077,2522,36512.3,30950,77088,7434,39730,128000,2981
7,30621,9160.52,9742.6,3288,23003,1637687,2467,26,61,39,283,96,214969,
19253,8746,7218,4808.8,2014,2799,121309,50021,6271,0,15600,40734,3951.

```
6,13996,51089,1487,5439,513,15536,76963,46993,20052,57932,7102,24193,8
8580,914742,17502,42540,1219,88205,31007,52797,74339,38685]},
{"marker":{"color":"rgb(55,126,184)","line":
{"width":0},"size":4},"mode":"markers","name":1,"type":"scatter3d","x"
:[39003086,19900340],"y":[1409517.397,1339180.127],"z":
[938821.1,297319]},
{"alphahull":7,"color":"rgb(55,126,184)","name":"y","opacity":0.1,"typ
e":"mesh3d","x":[39003086,19900340],"y":[1409517.397,1339180.127],"z":
[938821.1,297319]}],"layout":{"autosize":false,"height":550,"scene":
{"aspectmode":"manual","aspectratio":{"x":1,"y":1,"z":0.7},"xaxis":
{"backgroundcolor":"rgb(230, 230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Total
Production","zerolinecolor":"rgb(255, 255, 255)"},"yaxis":
{"backgroundcolor":"rgb(230, 230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Population","zerolinecolor":"rgb(
255, 255, 255)"},"zaxis":{"backgroundcolor":"rgb(230,
230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Surface
Area","zerolinecolor":"rgb(255, 255, 255)"}},"title":"3D Clustering
Plot","width":800}}
```


```
cluster1 = pd.DataFrame(d[ X['kmeans'] == 1 ]['Area'])
cluster1
```

	Area
35	China, mainland
74	India

COMMENT:

From the analysis China and India results as outlier. They are both big, popolous and producer countries, and they form a single cluster by them self.

```
def Agglomerative(X, n): #number of clusters is not necessary but
Python provides an option of providing the same for easy and simple
use.
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    model = AgglomerativeClustering(n_clusters=n, affinity =
'euclidean', linkage = 'ward')
    clust_labels1 = model.fit_predict(X)
    return (clust_labels1)

clust_labels1 = Agglomerative(X, 2)
agglomerative = pd.DataFrame(clust_labels1)
X.insert((X.shape[1]),'agglomerative',agglomerative)
Plot3dClustering(n=3, X=X, type_c='agglomerative')
```

```
c:\Python311\Lib\site-packages\sklearn\cluster\_agglomerative.py:983:
FutureWarning:
```

```
Attribute `affinity` was deprecated in version 1.2 and will be removed
in 1.4. Use `metric` instead
```

{"config":{"linkText":"Export to plot.ly","plotlyServerURL":"https://plot.ly","showLink":false},"data":[{"marker":{"color":"rgb(228,26,28)","line":{"width":0},"size":4},"mode":"markers","name":0,"type":"scatter3d","x":[326921,120172,896114,352564,2079,1260870,93011,779031,354176,234677,10534,1797998,7764,366256,301287,7560,179902,2027,186292,127086,38010,5363608,9063,232529,290771,10269,185684,438739,1319821,80357,122502,442651,199523,11460,614834,1114673,79749,114224,154677,343217,30030,393672,466487,464102,229300,10936,2888,209343,320778,2255352,127739,51901,22362,908083,20265,217594,2499790,7358,40676,21497,120486,3324458,669110,538562,2511,223753,187570,21351,19244,138918,136890,344326,11037,3932518,2123485,482171,188771,262200,2593024,76638,3400471,126141,505006,701186,2326,77108,146567,110075,81263,136665,32418,53157,130558,15243,316546,244860,532251,7819,223695,16499,56113,29633,3019528,50326,11853,787088,369041,763194,41060,498234,710143,6282,146726,82299,223057,3641538,64388,182271,75669,3112101,69708,155647,630532,1698382,1460041,462108,1477941,108532,870340,4815293,268797,1009,4491,2781,5658,4321,566755,165602,113940,84660,165403,76559,12120,1054383,1695650,360730,84646,11149,377793,303248,118220,1477069,15110,98826,30671,309576,2563733,122484,744769,1608335,147012,2423034,847789,12779748,103895,683251,6059,682364,1540944,281252,165288,167171],"y":[35530.081,2930.187,41318.142,29784.193,102.012,44271.041,2930.45,24450.561,8735.453,9827.589,395.361,164669.751,285.719,9468.338,11429.336,374.681,11175.692,61.349,11051.6,3507.017,2291.661,209288.278,428.697,7084.571,19193.382,546.388,16005.373,24053.727,36624.199,4659.08,14899.994,18054.726,7364.883,622.567,23626.456,49065.615,5260.75,4905.769,4189.353,11484.636,1179.551,10618.303,24294.75,25490.965,5733.551,956.985,73.925,10766.998,16624.858,97553.151,6377.853,1309.632,1367.254,104957.438,905.502,5523.231,64979.548,283.007,2025.137,2100.568,3912.061,82114.224,28833.629,11159.773,107.825,16913.503,12717.176,1861.283,777.859,10981.229,9265.067,9721.559,335.025,263991.379,81162.788,38274.618,4761.657,8321.57,59359.9,2890.299,127484.45,9702.353,18204.499,49699.862,116.398,4136.528,6045.117,6858.16,1949.67,6082.357,2233.339,4731.906,2890.297,583.455,25570.895,18622.104,31624.264,436.33,18541.98,430.835,4420.184,1265.138,129163.276,3075.647,628.96,35739.58,29668.834,53370.609,2533.794,29304.998,17035.938,276.255,4705.818,6217.581,21477.348,190886.311,2083.16,5305.383,4636.262,197015.955,4098.587,6811.297,32165.485,104918.09,38170.712,10329.506,50982.212,4051.212,19679.306,143989.754,12208.407,55.345,178.844,109.897,196.44,204.327,32938.213,15850.567,8790.574,7557.212,5447.662,2079.976,611.343,56717.156,46354.321,20876.917,40533.33,563.402,9910.701,8476.005,8921.343,69037.513,1296.311,7797.694,1369.125,11532.127,80745.02,5758.075,42862.958,44222.947,9400.145,66181.585,57310.019,324459.463,3456.75,31910.641,276.244,31977.065,95540.8,28250.42,17094.13,16529.904],"z":

[65286,2740,238174,124670,44,273669,2847,768230,8253.1,8265.9,1001,130
17,43,20290.8,3028,2281,11276,5.4,108330,5120,56673,835814,527,10856,2
7360,403,17652,47271,909351,62298,125920,74353.2,105,3.03,3541,110950,
34150,5106,5596,10422,924,7722,31800,12041,4199,2318,75,4831,24836,995
45,2072,4347,1720,100000,1827,30389,54755.7,366,25767,1012,6949,34888,
22754,12890,34,10716,24572,2812,19685,2756,11189,9053,10025,181157,162
876,43412.8,6889,2164,29414,1083,36456,8878,269970,56914,81,1782,19180
,23080,6219,1023,3036,9632,6267.5,243,58180,9428,32855,30,122019,32,10
3070,203,194395,155356,1345,44630,78638,65308,82329,14335,3369,1828,26
331,12034,126670,91077,2522,36512.3,30950,77088,7434,39730,128000,2981
7,30621,9160.52,9742.6,3288,23003,1637687,2467,26,61,39,283,96,214969,
19253,8746,7218,4808.8,2014,2799,121309,50021,6271,0,15600,40734,3951.
6,13996,51089,1487,5439,513,15536,76963,46993,20052,57932,7102,24193,8
8580,914742,17502,42540,1219,88205,31007,52797,74339,38685]},
{"alphahull":7,"color":"rgb(228,26,28)","name":"y","opacity":0.1,"type
":"mesh3d","x":
[326921,120172,896114,352564,2079,1260870,93011,779031,354176,234677,1
0534,1797998,7764,366256,301287,7560,179902,2027,186292,127086,38010,5
363608,9063,232529,290771,10269,185684,438739,1319821,80357,122502,442
651,199523,11460,614834,1114673,79749,114224,154677,343217,30030,39367
2,466487,464102,229300,10936,2888,209343,320778,2255352,127739,51901,2
2362,908083,20265,217594,2499790,7358,40676,21497,120486,3324458,66911
0,538562,2511,223753,187570,21351,19244,138918,136890,344326,11037,393
2518,2123485,482171,188771,262200,2593024,76638,3400471,126141,505006,
701186,2326,77108,146567,110075,81263,136665,32418,53157,130558,15243,
316546,244860,532251,7819,223695,16499,56113,29633,3019528,50326,11853
,787088,369041,763194,41060,498234,710143,6282,146726,82299,223057,364
1538,64388,182271,75669,3112101,69708,155647,630532,1698382,1460041,46
2108,1477941,108532,870340,4815293,268797,1009,4491,2781,5658,4321,566
755,165602,113940,84660,165403,76559,12120,1054383,1695650,360730,8464
6,11149,377793,303248,118220,1477069,15110,98826,30671,309576,2563733,
122484,744769,1608335,147012,2423034,847789,12779748,103895,683251,605
9,682364,1540944,281252,165288,167171],"y":
[35530.081,2930.187,41318.142,29784.193,102.012,44271.041,2930.45,2445
0.561,8735.453,9827.589,395.361,164669.751,285.719,9468.338,11429.336,
374.681,11175.692,61.349,11051.6,3507.017,2291.661,209288.278,428.697,
7084.571,19193.382,546.388,16005.373,24053.727,36624.199,4659.08,14899
.994,18054.726,7364.883,622.567,23626.456,49065.615,5260.75,4905.769,4
189.353,11484.636,1179.551,10618.303,24294.75,25490.965,5733.551,956.9
85,73.925,10766.998,16624.858,97553.151,6377.853,1309.632,1367.254,104
957.438,905.502,5523.231,64979.548,283.007,2025.137,2100.568,3912.061,
82114.224,28833.629,11159.773,107.825,16913.503,12717.176,1861.283,777
.859,10981.229,9265.067,9721.559,335.025,263991.379,81162.788,38274.61
8,4761.657,8321.57,59359.9,2890.299,127484.45,9702.353,18204.499,49699
.862,116.398,4136.528,6045.117,6858.16,1949.67,6082.357,2233.339,4731.
906,2890.297,583.455,25570.895,18622.104,31624.264,436.33,18541.98,430
.835,4420.184,1265.138,129163.276,3075.647,628.96,35739.58,29668.834,5
3370.609,2533.794,29304.998,17035.938,276.255,4705.818,6217.581,21477.
348,190886.311,2083.16,5305.383,4636.262,197015.955,4098.587,6811.297,

```
32165.485,104918.09,38170.712,10329.506,50982.212,4051.212,19679.306,1
43989.754,12208.407,55.345,178.844,109.897,196.44,204.327,32938.213,15
850.567,8790.574,7557.212,5447.662,2079.976,611.343,56717.156,46354.32
1,20876.917,40533.33,563.402,9910.701,8476.005,8921.343,69037.513,1296
.311,7797.694,1369.125,11532.127,80745.02,5758.075,42862.958,44222.947
,9400.145,66181.585,57310.019,324459.463,3456.75,31910.641,276.244,319
77.065,95540.8,28250.42,17094.13,16529.904],"z":
[65286,2740,238174,124670,44,273669,2847,768230,8253.1,8265.9,1001,130
17,43,20290.8,3028,2281,11276,5.4,108330,5120,56673,835814,527,10856,2
7360,403,17652,47271,909351,62298,125920,74353.2,105,3.03,3541,110950,
34150,5106,5596,10422,924,7722,31800,12041,4199,2318,75,4831,24836,995
45,2072,4347,1720,100000,1827,30389,54755.7,366,25767,1012,6949,34888,
22754,12890,34,10716,24572,2812,19685,2756,11189,9053,10025,181157,162
876,43412.8,6889,2164,29414,1083,36456,8878,269970,56914,81,1782,19180
,23080,6219,1023,3036,9632,6267.5,243,58180,9428,32855,30,122019,32,10
3070,203,194395,155356,1345,44630,78638,65308,82329,14335,3369,1828,26
331,12034,126670,91077,2522,36512.3,30950,77088,7434,39730,128000,2981
7,30621,9160.52,9742.6,3288,23003,1637687,2467,26,61,39,283,96,214969,
19253,8746,7218,4808.8,2014,2799,121309,50021,6271,0,15600,40734,3951.
6,13996,51089,1487,5439,513,15536,76963,46993,20052,57932,7102,24193,8
8580,914742,17502,42540,1219,88205,31007,52797,74339,38685]},
{"marker":{"color":"rgb(55,126,184)","line":
{"width":0},"size":4},"mode":"markers","name":1,"type":"scatter3d","x"
:[39003086,19900340],"y":[1409517.397,1339180.127],"z":
[938821.1,297319]},
{"alphahull":7,"color":"rgb(55,126,184)","name":"y","opacity":0.1,"typ
e":"mesh3d","x":[39003086,19900340],"y":[1409517.397,1339180.127],"z":
[938821.1,297319]},{"marker":{"color":"rgb(77,175,74)","line":
{"width":0},"size":4},"mode":"markers","name":2,"type":"scatter3d","x"
:[],"y":[],"z":[]},
{"alphahull":7,"color":"rgb(77,175,74)","name":"y","opacity":0.1,"type
":"mesh3d","x":[],"y":[],"z":[]}],"layout":
{"autosize":false,"height":550,"scene":
{"aspectmode":"manual","aspectratio":{"x":1,"y":1,"z":0.7},"xaxis":
{"backgroundcolor":"rgb(230, 230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Total
Production","zerolinecolor":"rgb(255, 255, 255)"},"yaxis":
{"backgroundcolor":"rgb(230, 230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Population","zerolinecolor":"rgb(
255, 255, 255)"},"zaxis":{"backgroundcolor":"rgb(230,
230,230)","gridcolor":"rgb(255, 255,
255)","showbackground":true,"title":"Surface
Area","zerolinecolor":"rgb(255, 255, 255)"}},"title":"3D Clustering
Plot","width":800}}
```


```
cluster1 = pd.DataFrame(d[ X['agglomerative'] == 1 ]['Area'])
cluster1
```

	Area
35	China, mainland
74	India

```
cluster1 = pd.DataFrame(d[ X['agglomerative'] == 1 ]['Area'])
cluster1
```

	Area
35	China, mainland
74	India

```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(25, 15))
plt.title("Food Dendograms")
dend = shc.dendrogram(shc.linkage(X, method='complete'))
```