Étude de marché:
Une entreprise souhaite exporter du poulet

Data source FAO (Food and Agriculture Organization)
http://www.fao.org/faostat/fr/#data\

Rappel du contexte

- Une entreprise française d’agroalimentaire souhaite se développer à l'international. Aucun pays particulier ni aucun continent n'est pour le moment choisi. Tous les pays sont envisageables !
- **Objectif** : Aider à cibler plus précisément des pays spécifiques pour approfondir davantage les études de marché. L'idéal serait de produire des "groupes" de pays, plus ou moins gros, dont on connaît les caractéristiques.
- Dans un premier temps, la stratégie est d'exporter les produits plutôt que de produire sur place, c'est-à-dire dans le(s) nouveau(x) pays ciblé(s).
- **Mission** : Identifier les pays favorables à l'entrée sur le marché du poulet.

Fournir:

1.Un notebook Jupyter contenant l'analyses.

2.Une heatmap avec clusters et les différentes variables (format image).

Pour parvenir à réaliser au mieux cette mission, quelques étapes indispensables seront nécessaires, comme celles-ci :

- Tester la classification ascendante hiérarchique, avec un dendrogramme pour visualisation.
- Ensuite utiliser la méthode des k-means, afin d'affiner l'analyse et comparer les résultats des deux méthodes de clustering .
- Analyser les centroïdes de mes classes.
- Egalement je peut réaliser une ACP afin de visualiser les résultats de mon analyse, comprendre les groupes, les liens entre les variables, les liens entre les individus...

Je vais travailler sur l'élaboration d'un échantillon contenant l'ensemble des pays disponibles, chacun caractérisé par les variables suivantes :

* Disponibilité alimentaire (Kcal/personne/jour)
* Disponibilité de protéines en quantité (g/personne/jour)
* PIB/habitant US
* Exportations Poulet en valeur 1000 USD
* Importation Poulet en valeur 1000 USD
* Réserves en valeur 1000 tetes
* Prix à la Production (USD/tonne)
* Stabilité politique et absence de violence/terrorisme (indice)
* Evolution Population 2017/2018 %

Importer les données, selectionner les élément pertinents pour l'analyse et combiner ces éléments dans une seule DataFrame :

Nous commencons avec un fichier 'DisponibiliteAlimentaire'

```
import pandas as pd
```

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data1 = pd.read_csv('DisponibiliteAlimentaire_2017.csv')
```

```
#Selection des colonnes 'Zone','Élément','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data1=data1[['Zone','Élément','Valeur']]
```

```
# Analyse des modalités de la variable 'Elément' qui semble avoir une importance pour analyse
data1['Élément'].unique()
```

```
]: array(['Production', 'Importations - Quantité', 'Variation de stock',
       'Exportations - Quantité', 'Disponibilité intérieure',
       'Aliments pour animaux', 'Semences', 'Pertes', 'Résidus',
       'Nourriture',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité alimentaire (Kcal/personne/jour)',
       'Disponibilité de protéines en quantité (g/personne/jour)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)',
       'Traitement', 'Autres utilisations (non alimentaire)',
       'Alimentation pour touristes'], dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data1=data1.pivot_table(index='Zone', columns='Élément', values='Valeur' , aggfunc=sum).reset_index()
data1.head()
```

Élément	Zone	Alimentation pour touristes	Aliments pour animaux	Autres utilisations (non alimentaire)	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	Disponibilité intérieure	Exportations - Quantité
0	Afghanistan	NaN	456.0	94.0	1997.0	357.31	30.63	54.09	15139.0	601.0
1	Afrique du Sud	0.0	9371.0	1448.0	2987.0	556.42	81.92	83.36	66840.0	10968.0
2	Albanie	NaN	697.0	177.0	3400.0	1260.49	117.57	119.50	4879.0	156.0

```
# Suppression des colonnes inutiles pour l'analyse
data1.drop(['Production', 'Importations - Quantité', 'Variation de stock',
       'Exportations - Quantité', 'Disponibilité intérieure',
       'Aliments pour animaux', 'Semences', 'Pertes', 'Résidus',
       'Nourriture',
       'Disponibilité alimentaire en quantité (kg/personne/an)',
       'Disponibilité de matière grasse en quantité (g/personne/jour)',
       'Traitement', 'Autres utilisations (non alimentaire)',
       'Alimentation pour touristes'], axis=1, inplace=True)
data1.rename(columns={'Zone':'Pays'}, inplace=True)
data1.head()
```

Élément	Pays	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)
0	Afghanistan	1997.0	54.09
1	Afrique du Sud	2987.0	83.36

De la meme manière avec le fichier PIB

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data2 = pd.read_csv('PIB.csv')
```

```
#Selection des colonnes 'Zone','Élément','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data2=data2[['Zone','Élément','Valeur']]
```

```
# Analyse des modalités de la variable 'Elément' qui semble avoir une importance pour analyse
data2['Élément'].unique()
```

```
array(['Valeur US $', 'Valeur US $ par habitant',
       'Croissance annuelle US$', 'Croissance annuelle US$ par habitant'],
      dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data2=data2.pivot_table(index='Zone', columns='Élément', values='Valeur', aggfunc=sum).reset_index()
data2.head()
```

Élément	Zone	Croissance annuelle US$	Croissance annuelle US$ par habitant	Valeur US $	Valeur US $ par habitant
0	Afghanistan	4.865790	2.227740	1.889635e+04	520.616409
1	Afrique du Sud	17.771989	16.114973	3.490067e+05	6121.876572

```
# Suppression des colonnes inutiles pour l'analyse
data2.drop(['Croissance annuelle US$','Croissance annuelle US$ par habitant','Valeur US $'], axis=1, inplace=True)
data2.rename(columns={'Zone':'Pays', 'Valeur US $ par habitant':'PIB/habitant US $'}, inplace=True)
data2.head()
```

Élément	Pays	PIB/habitant US $
0	Afghanistan	520.616409
1	Afrique du Sud	6121.876572

Nous fusionnons deux DataFrame et poursuivons par le traitement du fichier 'Export/Import'

```
# Jointure entre les 2 datafames (interne en condition left ) pour intégration d'une nouvelle variable
df=pd.merge(data1,data2, on= 'Pays', how='left')
df.head()
```

:

Élément	Pays	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	PIB/habitant US $
0	Afghanistan	1997.0	54.09	520.616409
1	Afrique du Sud	2987.0	83.36	6121.876572
2	Albanie	3400.0	119.50	4514.204908
3	Algérie	3345.0	92.85	4109.696001
4	Allemagne	3559.0	104.07	44651.829102

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data3 = pd.read_csv('exp_imp_valeur_quantiti.csv')
```

```
#Selection des colonnes 'Zone','Élément','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data3= data3[['Zone','Élément','Valeur']]
```

```
# Analyse des modalités de la variable 'Elément' qui semble avoir une importance pour analyse
data3['Élément'].unique()
```

```
: array(['Importations - Quantité', 'Importations - Valeur',
       'Exportations - Quantité', 'Exportations - Valeur'], dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data3=data3.pivot_table(index= 'Zone', columns='Élément', values = 'Valeur', aggfunc= sum).reset_index()
data3.head()
```

Élément	Zone	Exportations - Quantité	Exportations - Valeur	Importations - Quantité	Importations - Valeur
0	Afghanistan	0.0	0.0	18970.0	26227.0
1	Afrique du Sud	8391.0	14777.0	1349.0	9215.0

```
# Suppression des colonnes inutiles pour l'analyse
data3.drop(['Importations - Quantité','Exportations - Quantité'], axis=1, inplace=True)
data3.rename(columns={'Zone':'Pays', 'Importations - Valeur':'Importation Poulet en valeur 1000 US$', 'Exportations - Valeu
data3.head()
```

Élément	Pays	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$
0	Afghanistan	0.0	26227.0
1	Afrique du Sud	14777.0	9215.0

```
# Jointure le datafames 'data3'avec une jointure dataframe précédente , (interne en condition left)
#pour intégration d'une nouvelle variable
df1= pd.merge(df, data3, how='left', left_on=['Pays'], right_on=['Pays'])
df1.head()
```

Élément	Pays	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0

On a fusionné les trois DataFrames et poursuivons par le traitement du fichier 'Reserve'

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data4 = pd.read_csv('reserve.csv')
```

```
#Selection des colonnes 'Zone','Élément','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data4= data4[['Zone','Élément','Valeur']]
```

```
# Analyse des modalités de la variable 'Elément' qui semble avoir une importance pour analyse
data4['Élément'].unique()
```

```
: array(['Réserves'], dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data4=data4.pivot_table(index='Zone', columns='Élément', values='Valeur', aggfunc=sum).reset_index()
```

```
data4.rename(columns={'Zone':'Pays', 'Réserves':'Réserves en valeur 1000 tetes'}, inplace=True)
```

```
# Jointure le datafames 'data4'avec une jointure dataframe précédente , interne en condition left
#pour intégration d'une nouvelle variable
df2= pd.merge(df1, data4, how='left', left_on=['Pays'], right_on=['Pays'])
df2.head()
```

Élément	Pays	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0	13573.0
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0

On a fusionné les quatre DataFrames et poursuivons par le traitement du fichier 'Prix'

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data5 = pd.read_csv('prix.csv')
data5.head()
```

```
#Selection des colonnes 'Zone','Élément','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data5=data5[['Zone','Élément','Valeur']]
```

```
# Analyse des modalités de la variable 'Elément' qui semble avoir une importance pour analyse
data5['Élément'].unique()
```

```
array(['Prix à la Production (USD/tonne)'], dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data5=data5.pivot_table(index='Zone', columns='Élément', values= 'Valeur', aggfunc=sum).reset_index()
```

```
data5.rename(columns={'Zone':'Pays'}, inplace=True)
```

```
# Jointure le datafames 'data5'avec une jointure dataframe précédente , interne en condition left
#pour intégration d'une nouvelle variable
df3= pd.merge(df2, data5, how='left', left_on=['Pays'], right_on=['Pays'])
df3.head()
```

Élément	Pays	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0	13573.0	NaN
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0	NaN

On a fusionné les cinque DataFrames et poursuivons par le traitement du fichier 'Stabilité Politique'

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data6 = pd.read_csv('stab_politique.csv')
```

```
#Selection des colonnes 'Zone','Produit','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data6=data6[['Zone','Produit','Valeur']]
```

```
# Analyse des modalités de la variable 'Produit' qui semble avoir une importance pour analyse
data6['Produit'].unique()
```

```
array(['Stabilité politique et absence de violence/terrorisme (indice)'],
      dtype=object)
```

```
# transformation d'éléments de lignes en colonnes
# Method .pivot_table()  utile pour rendre le dataframe exploitable selon les modalité précédentes
data6=data6.pivot_table(index='Zone', columns='Produit', values='Valeur',aggfunc=sum).reset_index()
data6.head()
```

	Produit	Zone	Stabilité politique et absence de violence/terrorisme (indice)
0	Afghanistan		-2.80
1	Afrique du Sud		-0.28

```
data6.rename(columns={'Zone':'Pays'}, inplace=True)
data6.head()
```

```
# Jointure le datafames 'data6'avec une jointure dataframe précédente , interne en condition left
#pour intégration d'une nouvelle variable
df4=pd.merge(df3,data6, how='left', left_on=['Pays'], right_on=['Pays'])
df4.head()
```

	Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0	13573.0	NaN	-2.80
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0	NaN	-0.28

On a fusionné les six DataFrames et poursuivons par le traitement du fichier 'Population'

```
#Import de fichier csv extraits directement de la base de données du site de la FAO
data7 = pd.read_csv('Population_2000_2018.csv')
```

A partir des données de population, calculer l'évolution sur la période en %. J'ai choisi l'années 2017-2018

```
#Selection des colonnes 'Zone','Année','Valeur', les autres caracteristiques ne sont pas nécessaires dans nos recherches
data7=data7[['Zone','Année','Valeur']]
data7.head()
```

```
# Séléction l'année 2017
pp1=data7[data7["Année"]== 2017]
pp1.head()
```

	Zone	Année	Valeur
17	Afghanistan	2017	36296.113
36	Afrique du Sud	2017	57009.756
55	Albanie	2017	2884.169

```
# Séléction l'année 2018
pp2=data7[data7["Année"]== 2018]
pp2.head()
```

	Zone	Année	Valeur
18	Afghanistan	2018	37171.921
37	Afrique du Sud	2018	57792.518
56	Albanie	2018	2882.740

```
# creation d'une dataframe avec uniquement les années 2017 et 2018
pp=pd.merge(pp1,pp2,left_on= 'Zone', right_on='Zone')
pp.head()
```

	Zone	Année_x	Valeur_x	Année_y	Valeur_y
0	Afghanistan	2017	36296.113	2018	37171.921
1	Afrique du Sud	2017	57009.756	2018	57792.518

```
# Calcul de pourcentage d'évolution de la population entre 2018 et 2017
pp['Evolution Population 2017/2018 %'] = ((pp['Valeur_y']*100/pp['Valeur_x'])-100)
pp.head()
```

	Zone	Année_x	Valeur_x	Année_y	Valeur_y	Evolution Population 2017/2018 %
0	Afghanistan	2017	36296.113	2018	37171.921	2.412953
1	Afrique du Sud	2017	57009.756	2018	57792.518	1.373032

```
# Suppression des colonnes inutiles
pp.drop(['Année_x','Valeur_x','Année_y','Valeur_y'], axis=1, inplace=True)
pp.rename(columns={'Zone':'Pays'}, inplace=True)
pp.head()
```

	Pays	Evolution Population 2017/2018 %
0	Afghanistan	2.412953
1	Afrique du Sud	1.373032

On a fusionné les sept DataFrames et Nettoyer les données

```
# Jointure le datafames 'pp'avec une jointure dataframe précédente , interne en condition left
#pour intégration d'une nouvelle variable
df5=pd.merge(df4,pp, how='left', left_on=['Pays'], right_on=['Pays'])
df5.head()
```

	Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0	13573.0	NaN	-2.80	2.412953
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0	NaN	-0.28	1.373032

Nettoyage des données

```
# détection des valeur 0
df5.isnull().sum()
```

```
5]: Pays                                                    0
    Disponibilité alimentaire (Kcal/personne/jour)          2
    Disponibilité de protéines en quantité (g/personne/jour) 2
```

```
# La fonction .fillna() permet de remplacer les valeurs de NaN  par 0.
df5.fillna(0, inplace=True)
df5.head()
```

```
# verification en doublons
df5.loc[df5[['Pays', 'Disponibilité alimentaire (Kcal/personne/jour)', 'Disponibilité de protéines en quantité (g/personne/
          'Importation Poulet en valeur 1000 US$', 'Réserves en valeur 1000 tetes','Prix à la Production (USD/tonne)',
           'Stabilité politique et absence de violence/terrorisme (indice)','Evolution Population 2017/2018 %']].duplicate
```

Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %

Passons à la partie Analyse

Realise un Clustering Hierarchique

Utilisation du package «scipy»

L'échantillon comporte de 9 variables également 174 pays "maîtrisables" qui permet de commencer par une classification hiérarchique. Algorithme qui a une forte complexité algorithmique en temps et en espace, le clustering hiérarchique est recommandé pour les petits échantillons.

```
#Import des librairies Python
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing
import numpy as np
from sklearn import cluster, metrics
from sklearn import decomposition
```

Extraire les données d'expressions

```
#Transformation en array Numpy
#Indexation selon les pays par la fonction .set_index()
df5.set_index('Pays', inplace=True)
X = df5.values
X.shape
```

```
(174, 9)
```

Appliquer une Normalisation centrée_réduite

- Avent de proceder au Clustering Hierarchique les donnéer dvovent etre normalisées pour ramener les niveau des expression chaque variable a la meme échelle. Si on ne le fais pas les variables avec une fort variance d'expression auront plus de poids dans la classification que les variables avec les variance faible. Une normalisation permet de donner le meme poids à tous le variables.

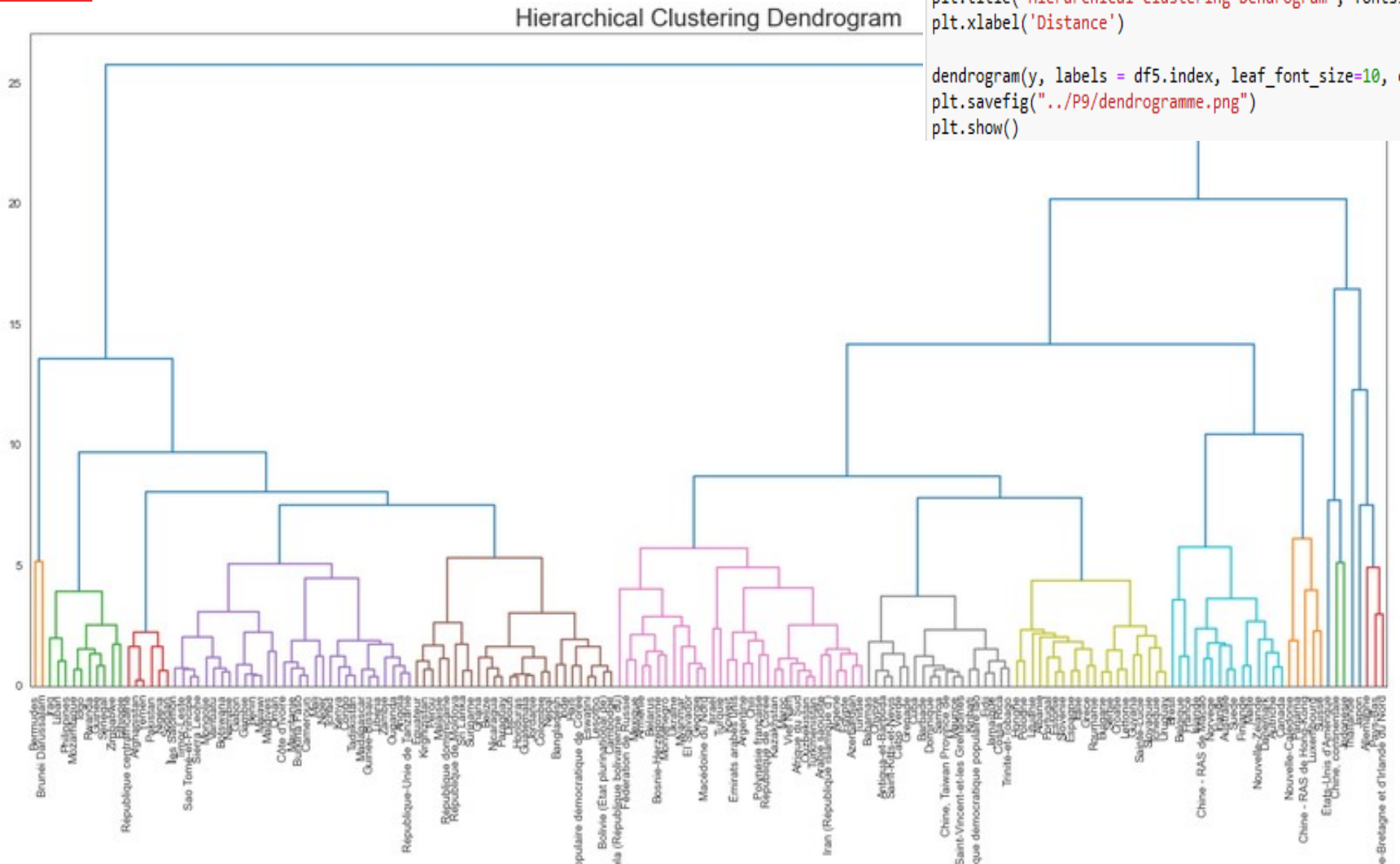
```
#Centrage / réduction des données pour que nos données puissent prendre la même importance
#According to the syntax, we initially create an object of the StandardScaler() function.
#Further, we use fit_transform() along with the assigned object to transform the data and standardize it.
std_scale = preprocessing.StandardScaler().fit(X)
X_scaled = std_scale.transform(X)
```

```
#Clustering hiérarchique: création d'une Matrice des liens selon la Méthode de Ward
y = linkage(X_scaled, method = 'ward', metric='euclidean') # 1.indique quelle notion de dictanse intra-classe, 2. notion de
```

Le Clustering se base uniquement sur le niveaux d'expressionn de variable.Il s'agit d'une methode 'non supervisée'

Premier dendrogramme global

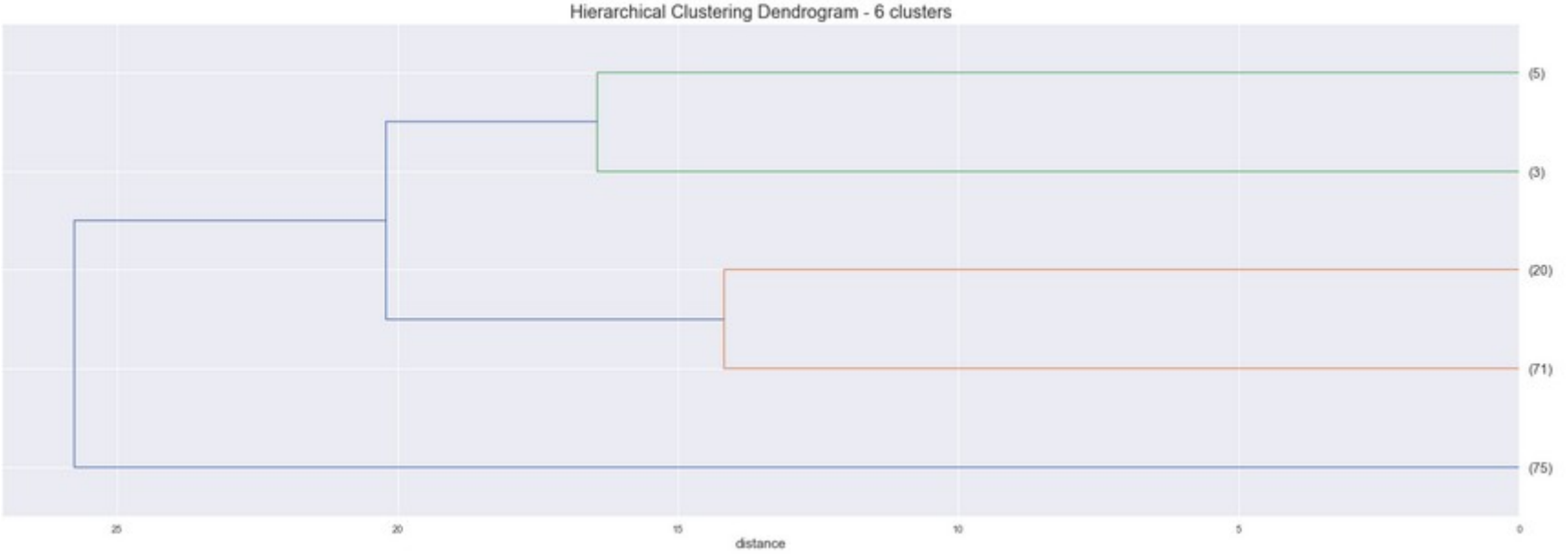
```
#Affichage d'un premier dendrogramme global
fig =plt.figure(figsize=(20,10))
sns.set_style('white')
plt.title('Hierarchical Clustering Dendrogram', fontsize=20)
plt.xlabel('Distance')

dendrogram(y, labels = df5.index, leaf_font_size=10, color_threshold=7, orientation='top')
plt.savefig("../P9/dendrogramme.png")
plt.show()
```



Dendrogramme avec 5 groupes

Le dendrogramme peut être difficile à lire lorsque la matrice d'observation originale est grande. Nous utilisons La troncature pour condenser le dendrogramme et nous l'avons coupé en 5 groupes

```
#Coupage du dendrogramme en 5 groupes pour avoir une première idée du partitionnement
fig = plt.figure(figsize=(30,10))
plt.title('Hierarchical Clustering Dendrogram - 6 clusters', fontsize=20)
plt.xlabel('distance', fontsize=15)

dendrogram(y, labels = df5.index, p=5, truncate_mode='lastp', leaf_font_size=15, orientation='left')
plt.savefig("../P9/dendrogramme2.png")
```



Identifiion 5 group et les integrons dans l'échentillon de départ

```
#Identification des 5 groupes obtenus
groupes_cah = fcluster(y, 5, criterion='maxclust')
print('Numérotation de nos groupes : ' + str(np.unique(groupes_cah)))

#Index trié des groupes
idg = np.argsort(groupes_cah)

#Affichage des pays selon leurs groupes
df_groupes_cah = pd.DataFrame(df5.index[idg], groupes_cah[idg]).reset_index()
df_groupes_cah = df_groupes_cah.rename(columns={'index':'Groupe'})
```

```
Numérotation de nos groupes : [1 2 3 4 5]
```

```
#Intégration des groupes dans notre échantillon de départ représenté par le dataframe "df5"
#Jointure interne nécessaire pour parvenir à agréger nos données
df_groupes_cah = pd.merge(df5, df_groupes_cah, on='Pays')
```

```
#Aperçu des 5 premières lignes
df_groupes_cah.head()
```

	Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %	Group
0	Afghanistan	1997.0	54.09	520.616409	0.0	26227.0	13573.0	0.0	-2.80	2.412953	
1	Afrique du Sud	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0	0.0	-0.28	1.373032	
2	Albanie	3400.0	119.50	4514.204908	0.0	3699.0	7835.0	2469.7	0.38	-0.049546	

```
#Préparation de sous-ensembles permettant de caractériser les groupes un à un
df_groupe1_cah = df_groupes_cah[df_groupes_cah['Groupe'] == 1]
df_groupe2_cah = df_groupes_cah[df_groupes_cah['Groupe'] == 2]
df_groupe3_cah = df_groupes_cah[df_groupes_cah['Groupe'] == 3]
df_groupe4_cah = df_groupes_cah[df_groupes_cah['Groupe'] == 4]
df_groupe5_cah = df_groupes_cah[df_groupes_cah['Groupe'] == 5]
```

On fait une Première Comparaison des moyennes et l'analyse des résultats

```
#Première comparaison des moyennes afin d'identifier le groupe de pays le plus porteur à ce niveau de l'analyse
df_groupes_cah.groupby('Groupe').mean()
```

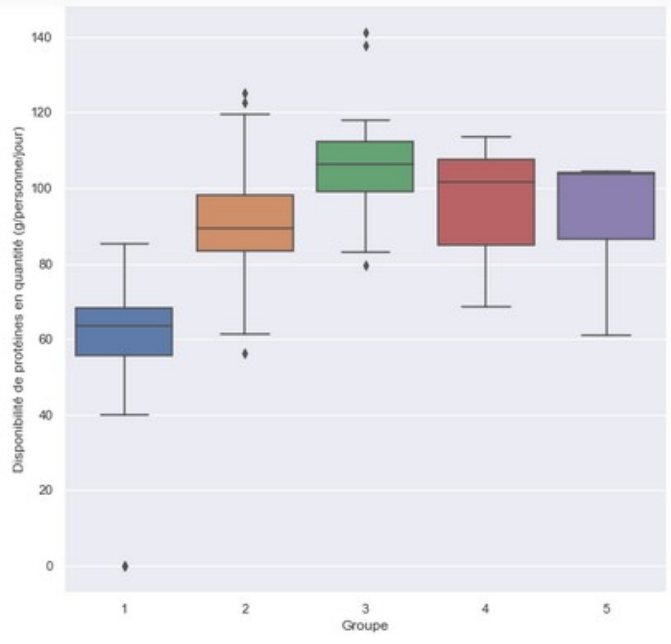
Groupe	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %
1	2412.533333	62.055333	4672.216064	4.062080e+03	5.962533e+03	6.870401e+04	594.708000	-0.521333	2.076631
2	3071.211268	90.729014	12322.389905	2.113166e+04	2.467376e+04	8.589666e+04	530.892958	0.148310	0.494668
3	3351.600000	106.288000	53316.254770	1.225751e+05	1.027587e+05	1.057163e+05	1445.315000	0.891500	0.891253
4	3284.333333	94.560000	24140.793201	5.873510e+05	9.374500e+04	5.812679e+06	1863.600000	-0.070000	0.742005
5	3152.600000	91.966000	35871.211557	1.038104e+06	1.193030e+06	2.042780e+05	1002.640000	0.452000	0.297850

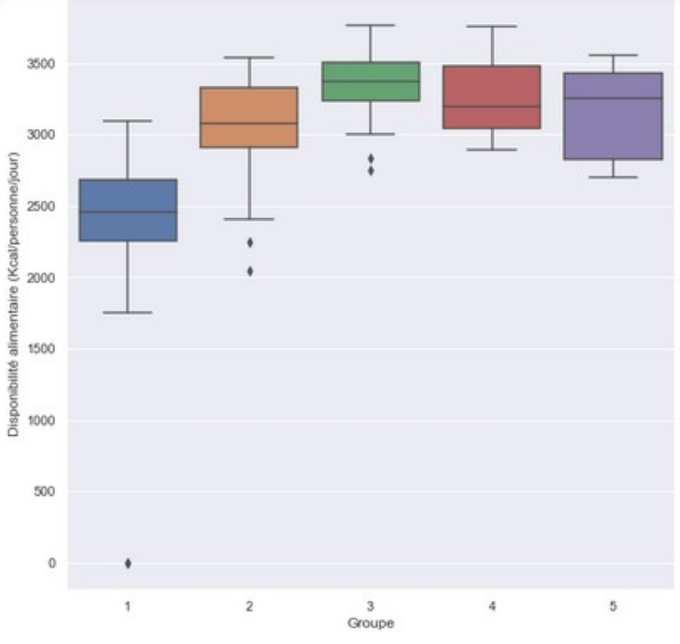
A partir des centoides calculés ci-dessus et comparaison visuelle des groupes (diapo suivant) a mon avis le **Groupe 3** présente un potentiel marché interessant en raison du fait qu'il se distingue des autres groupes avec :

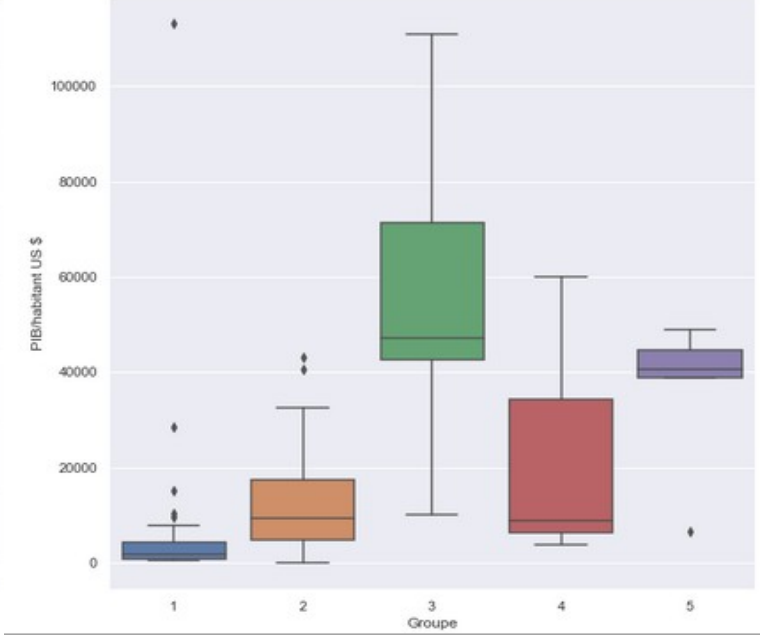
1. Un niveau de PIB élevé,
2. La Disponibilité de protéines en quantité (g/personne/jour) élevé,
3. Un faible niveau de réserves, l'importation et d'exportation de poulets,
4. Un niveau de prix élevé,
5. Ainsi qu'une grande stabilité politique.

Comparaison visuelle des groupes par Boxplot

```
#Comparaison visuelle des groupes par Boxplot, en abscisse les numéros des groupes
plt.figure(figsize=(20, 20))
sns.set(style="darkgrid")
plt.subplot(221)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Disponibilité alimentaire (Kcal/personne/jour)')
plt.subplot(222)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Disponibilité de protéines en quantité (g/personne/jour)')
plt.subplot(223)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='PIB/habitant US $')
plt.subplot(224)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Exportations Poulet en valeur 1000 US$')

plt.show(block=False)
```

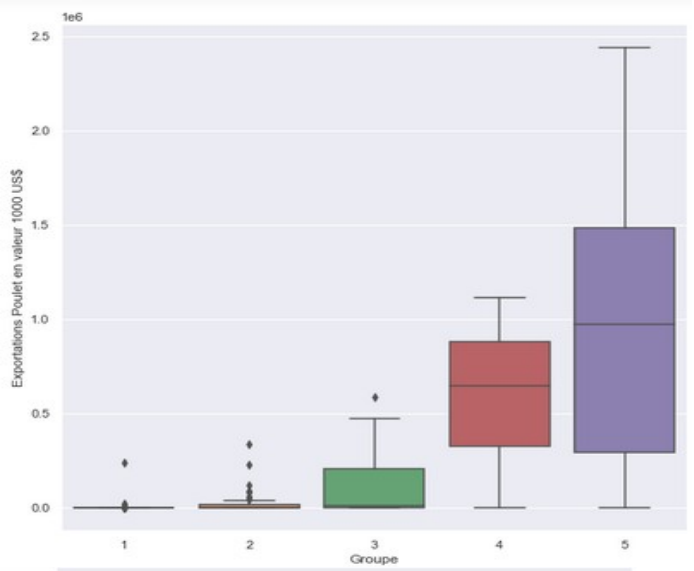


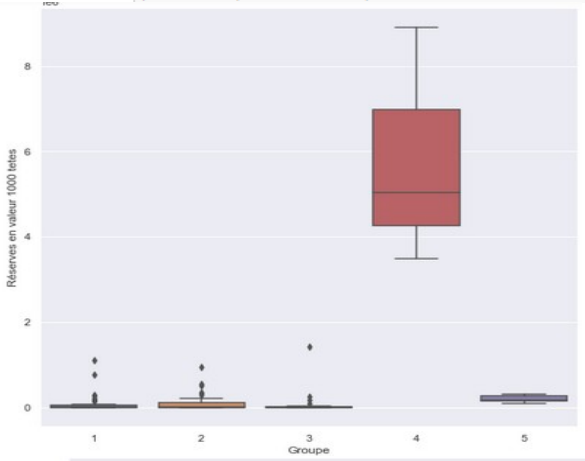


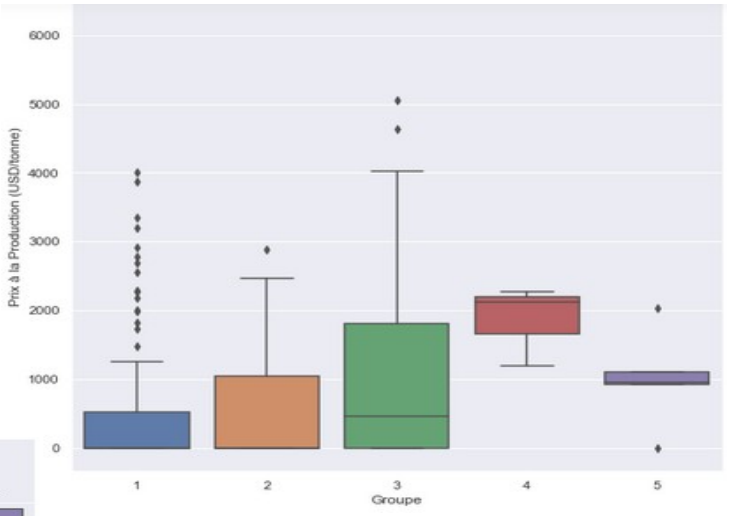


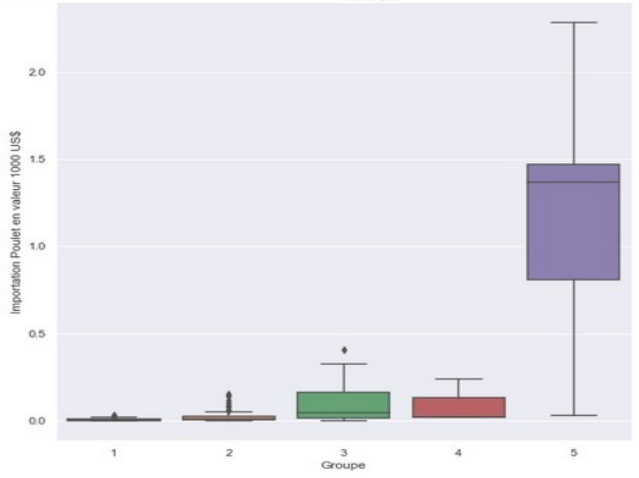
Comparaison visuelle des groupes par Boxplot

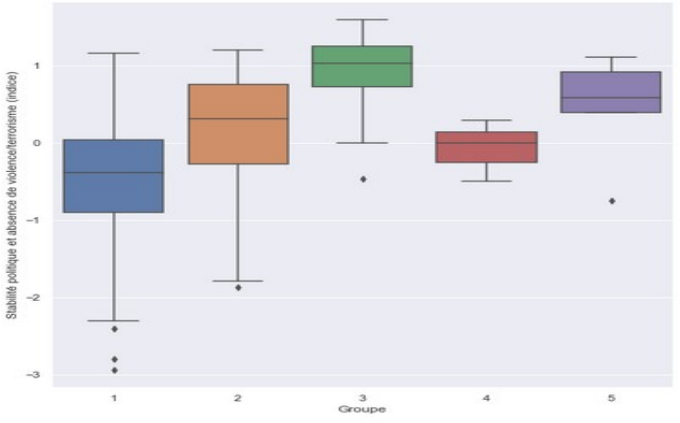
```
#Comparaison visuelle des groupes par Boxplot, en abscisse les numéros des groupes
plt.figure(figsize=(20, 20))
sns.set(style="darkgrid")
plt.subplot(221)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Importation Poulet en valeur 1000 US$')
plt.subplot(222)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Réserves en valeur 1000 tetes')
plt.subplot(223)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Prix à la Production (USD/tonne)')
plt.subplot(224)
sns.boxplot(data=df_groupes_cah, x='Groupe', y='Stabilité politique et absence de violence/terrorisme (indice)')

plt.show(block=False)
```











Liste du group 3 , identifiés comme potentiellement intéressant

```
#Pays du groupe 3 identifiés comme potentiellement intéressant
print(df_groupe3_cah['Pays'].unique())
```

```
['Australie' 'Autriche' 'Belgique' 'Brésil' 'Canada'
 'Chine - RAS de Hong-Kong' 'Chine - RAS de Macao' 'Danemark' 'Finlande'
 'France' 'Irlande' 'Islande' 'Luxembourg' 'Malte' 'Norvège'
 'Nouvelle-Calédonie' 'Nouvelle-Zélande' 'Panama' 'Suisse' 'Suède']
```

```
#Élaboration d'un premier sous-ensemble de groupe 3 validés via la classification hiérarchique
df_cah_subset = df_groupes_cah.query('[3] in Groupe')
df_cah_subset.shape
```

```
(20, 11)
```

```
#Visualisation rapide des premières lignes
df_cah_subset.head()
```

	Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %	Groupe
10	Australie	3307.0	108.01	57628.863849	12264.0	45889.0	95012.0	0.0	0.90	1.275318	3
11	Autriche	3694.0	108.11	47309.051637	120216.0	144215.0	16736.0	1216.7	1.05	0.810519	3
16	Belgique	3770.0	101.35	44025.903247	443931.0	406239.0	26905.0	971.7	0.43	0.546685	3

K-means clustering

Appliquons cette méthode pour pouvoir comparer ce premier choix.

Principe de base : Répartir observation (Pays) en des groupes de façon regreurer oservation similaire et de séparer les observations dissimilairs .

Nous allons maintenant réaliser un clustering K-Means pour plusieurs tailles de clusters, et pour chacune de ces tailles nous allons calculer le coefficient de silhouette, puis l'inertie pour pouvoir ensuite choisir la taille de cluster la plus appropriée. La méthode de coude nous aidera a conclure sur ce point, mais une critique métier reste essentielle pour la pertinence du nombre de clusters en fonction des objectifs de l'entreprise.

- Utilisation du package «scikit-learn»

l'objectif de classification automatique de l'algoritme k-means serait de minimiser l'inertie intra-classes à nombre de classes fixé. en surveillant l'évolution de l'inertie.

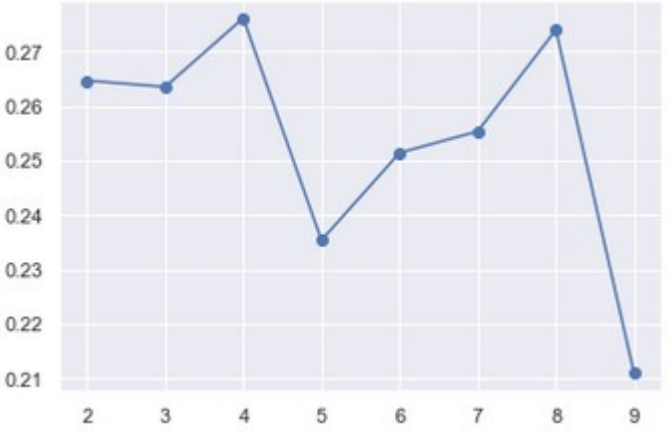
```
#Préparation des données pour le clustering K-Means
#Transformation en array Numpy
X = df5.values
# On le transforme en DataFrame pour pouvoir mieux visualiser nos données :
X = pd.DataFrame(X)
X.head()
```

	0	1	2	3	4	5	6	7	8
0	1997.0	54.09	520.616409	0.0	26227.0	13573.0	0.0	-2.80	2.412953
1	2987.0	83.36	6121.876572	14777.0	9215.0	178634.0	0.0	-0.28	1.373032
2	3400.0	119.50	4514.204908	0.0	3699.0	7835.0	2469.7	0.38	-0.049546

```
#Centrage / réduction des données
std_scale = preprocessing.StandardScaler().fit(X)
X_scaled = std_scale.transform(X)
```

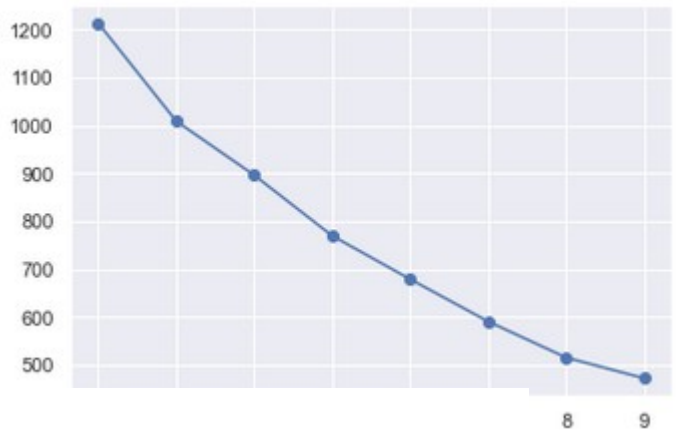
K-means clustering (suit)

```python
#Calcul  de la métrique "silhouette" pour différents nombres de groupes issus de la méthode des centres mobiles
#Liste pour stocker nos coefficients
silhouettes = []

#Boucle itérative de 2 à 10 (clusters) pour tester les possibilités
for k in range(2, 10):
    #Création et ajustement d'un modèle pour chaque k
    cls = cluster.KMeans(n_clusters=k)
    cls.fit(X_scaled)

    #Stockage des coefficients associés
    silh = metrics.silhouette_score(X_scaled, cls.labels_)
    silhouettes.append(silh)

#Visualisation des valeurs de coefficient de silhouette pour chaque nombre de cluster
plt.plot(range(2, 10), silhouettes, marker='o')
plt.show()
```



-> La métrique de l'inertie permet également d'avoir une estimation du clustering le plus optimal, voyons ce que ça donne.

```python
#On crée une liste dans laquelle on stocke les inerties
inerties=[]

#On fait une boucle de 2 à 9 pour tester toutes ces possibiliéts
for k in range(2, 10):
    #pour chaque k, on crée un modèle et on l'ajuste
    km = cluster.KMeans(n_clusters=k)
    km.fit(X_scaled)
    #on stocke l'inertie associée
    inerties.append(km.inertia_)

#Visualisation des valeurs d'inertie pour chaque nombre de cluster
plt.plot(range(2, 10), inerties, marker='o')
plt.show()
```



-> Idéalement, pour optimiser et ne pas perdre trop d'information il serait indiquer de choisir un k = 2.

Maintenant le coefficient de silhouettes expose une possibilité intéressante dans notre contexte métier avec k = 4 Un clustering en 4, puis en 5 permettra également de comparer le partitionnement avec les groupes de la classification hiérarchique. Il est pertinent de comparer les deux méthodes sur le même nombre de clusters.

Passons à la partie visualisation. Pour ce faire, nous allons réaliser une ACP pour la projection des donné

```
#Clustering K-Means en 4 clusters
cls4 = cluster.KMeans(n_clusters=4)
cls4.fit(X_scaled)
```

```
KMeans(n_clusters=4)
```

```
#Clustering K-Means en 5 clusters
cls5 = cluster.KMeans(n_clusters=5)
cls5.fit(X_scaled)
```

```
KMeans(n_clusters=5)
```

```
#Récupération des clusters attribués à chaque individu (classes d'appartenance)
clusters_kmeans4 = cls4.labels_
clusters_kmeans5 = cls5.labels_
```

- Clé de ACP - recherche de la projection pour laquelle l'inertie des points est maximale. Chercher L'axe principale d'inerctie
- L'ACP (*Analyse en Composante Principale*) permettra une visualisation des clusters pays sur le premier plan factoriel (ou plus). Il deviendra alors facile de pouvoir appréhender le "comportement" des différents groupes.

```
#Calcul des composantes principales
pca = decomposition.PCA(svd_solver='full')
pca.fit(X_scaled)
```

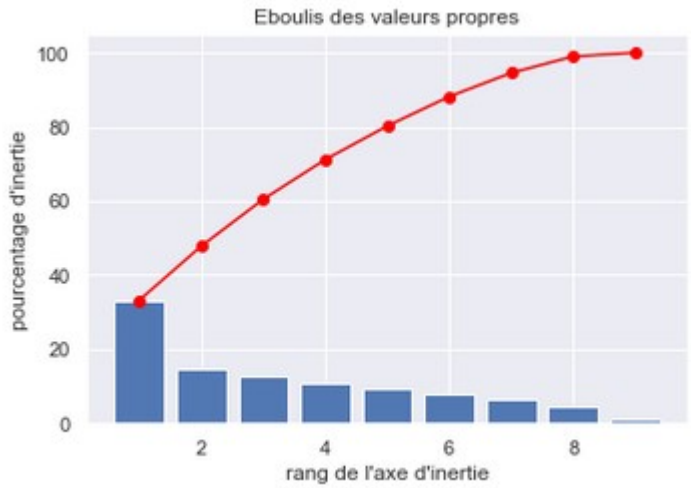
```
PCA(svd_solver='full')
```

```
#Pourcentage de variance expliquée par les composantes principales à l'aide de .explained_variance_ratio_
print(pca.explained_variance_ratio_.cumsum())
```

```
[0.33155484 0.47767839 0.60427845 0.71149336 0.80214978 0.88172722
 0.94551774 0.99013737 1.        ]
```

Plus de 47 % de la variance des données est expliquée par ces deux premières composantes.

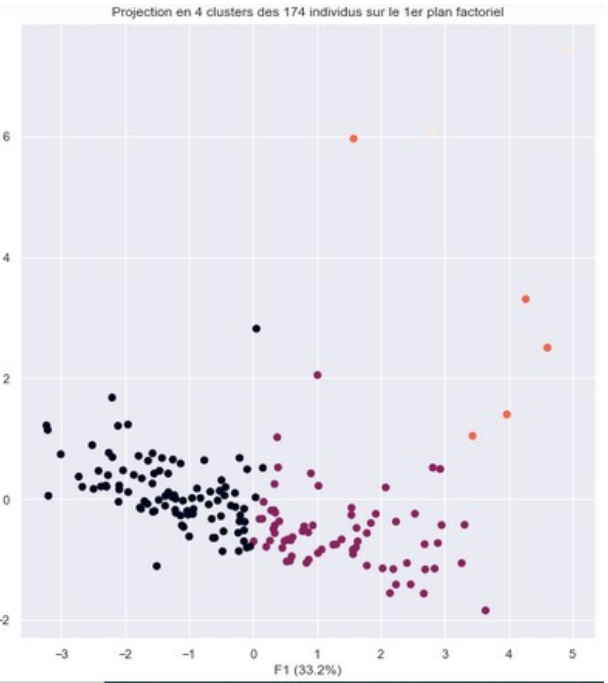
```
#Représentation de la variance expliquée
pca = decomposition.PCA()
pca.fit(X_scaled)
scree = pca.explained_variance_ratio_*100

plt.bar(np.arange(len(scree))+1, scree)
plt.plot(np.arange(len(scree))+1, scree.cumsum(),c="red",marker='o')

plt.xlabel("rang de l'axe d'inertie")
plt.ylabel("pourcentage d'inertie")
plt.title("Eboulis des valeurs propres")
plt.show()
```

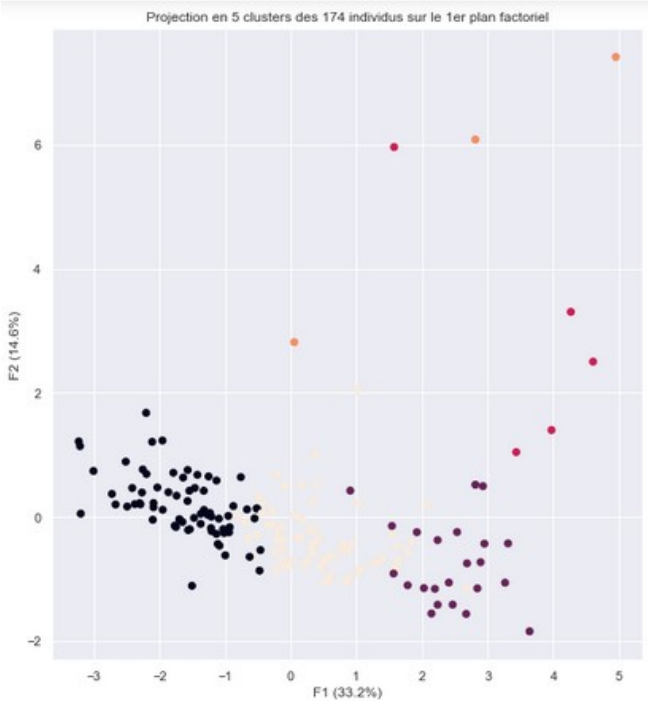


-> La méthode du coude

Visualisation des données projetées sur le premier plan factoriel

```
#Coordonnées factorielles
X_projected = pca.transform(X_scaled)
plt.figure(figsize=(20, 10))

plt.subplot(121)
plt.scatter(X_projected[:, 0], X_projected[:, 1], c=cls4.labels_)
plt.xlabel('F{} ({}%)'.format(1, round(100*pca.explained_variance_ratio_[0],1)))
plt.ylabel('F{} ({}%)'.format(2, round(100*pca.explained_variance_ratio_[1],1)))
plt.title("Projection en 4 clusters des {} individus sur le 1er plan factoriel".format(X_projected.shape[0]))

plt.subplot(122)
plt.scatter(X_projected[:, 0], X_projected[:, 1], c=clusters_kmeans5)
plt.xlabel('F{} ({}%)'.format(1, round(100*pca.explained_variance_ratio_[0],1)))
plt.ylabel('F{} ({}%)'.format(2, round(100*pca.explained_variance_ratio_[1],1)))
plt.title("Projection en 5 clusters des {} individus sur le 1er plan factoriel".format(X_projected.shape[0]))

plt.savefig("../P9/projection_clusters.png")
plt.show()
```





-La projection en **4 clusters** est possible, mais l'analyse sera plus fine en **5 clusters**.

- De plus, la comparaison sera possible avec les 5 groupes identifiés lors du précédent partitionnement

Calcule des Centroides de 5 clusters . Creation une heatmap avec les croisements entre les clusters de pays et les différentes variables

```
#Tableau des Centroïdes 5 clusters dans sa version centrée réduite
#La comparaison est tout de suite simplifiée, les dimensions prenant la même importance!
centroids = cls5.cluster_centers_
pd.DataFrame(centroids, columns=df5.columns)
```

	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %
0	-0.806188	-0.857479	-0.467206	-0.248822	-0.228416	-0.143468	-0.137047	-0.542962	0.842142
1	1.048576	1.330327	1.833120	0.113482	0.193009	-0.167812	0.509656	0.997206	-0.387765
2	0.601268	0.536633	1.029081	3.808222	4.731264	0.025953	0.264420	0.561618	-0.835283
3	0.843447	0.655545	0.461113	2.045261	0.139500	6.835784	1.016748	-0.022442	-0.433291
4	0.383131	0.356024	-0.215540	-0.140564	-0.170784	-0.091544	-0.088390	0.178803	-0.630517

Maintenant, il est nécessaire de caractériser chacun de ces groupes selon nos **9 variables**. La position des **centroïdes** de chacun des groupes indiquera le ou les meilleurs clusters.

```
sns.set_theme()
f,ax = plt.subplots(figsize=(9,6))
clustergrid=sns.heatmap(centroids, annot=True, linewidths=.5, ax=ax)
```

Affichage des pays retenus dans ce clustering

```
#Index trié des clusters
idk = np.argsort(cls5.labels_)

#Affichage des observations selon leurs clusters
df_cls5 = pd.DataFrame(df5.index[idk], cls5.labels_[idk]).reset_index()
df_cls5 = df_cls5.rename(columns={'index':'cluster'})
```

```
#Intégration des clusters dans notre dataframe "df_cls6" par la méthode .mer
#Jointure avec le dataframe initial "df5" selon les pays et condition 'inner
df_cls5 = pd.merge(df5, df_cls5, on='Pays')
df_cls5.head()
```

```
#Affichage des pays retenus dans ce clustering
#Sélection selon indiquateur Exportations Poule et Dispon
print(df_cls5[df_cls5['cluster'] == 3]['Pays'].unique())
print(df_cls5[df_cls5['cluster'] == 1]['Pays'].unique())
```

```
['Chine, continentale' 'Indonésie' "États-Unis d'Amérique"]
['Australie' 'Autriche' 'Belgique' 'Canada' 'Chine - RAS de Hong-Kong'
 'Chine - RAS de Macao' 'Danemark' 'Espagne' 'Finlande' 'France' 'Irlande'
 'Islande' 'Israël' 'Italie' 'Luxembourg' 'Malte' 'Norvège'
 'Nouvelle-Calédonie' 'Nouvelle-Zélande' 'Portugal' 'Suisse' 'Suède'
 'Émirats arabes unis']
```

-> Il est normal de ne pas avoir exactement les mêmes résultats avec la méthode des centres mobiles (K-Means). Le principe est le suivrant, faire varier le nombre de clusters et surveiller l'évolution d'un indicateur de qualité (silhouettes, inerties...), c'est-à-dire l'aptitude des pays à être plus proches de ses congénères du même cluster que les pays des autres clusters.

```
#La correspondance avec les groupes du partitionnement hiérarchique peut-être analysée avec une méthode .crosstab()
pd.crosstab(groupes_cah, cls5.labels_)
```

col_0	0	1	2	3	4
row_0					
1	70	0	0	0	5
2	1	5	0	0	65
3	0	18	0	0	2
4	0	0	0	3	0
5	0	0	5	0	0

-> Sur la méthode hiérarchique le choix a été fait sur les **groupes 3**, avec les K-Means se sont les **clusters 3 et 1**. Les correspondances ci-dessus montrent plusieurs pays communs aux deux approches:

'Australie' 'Autriche' 'Belgique' 'Canada' 'Chine - RAS de Hong-Kong' 'Chine - RAS de Macao' 'Danemark' 'Finlande' 'Irlande' 'Islande' 'Luxembourg' 'Malte' 'Norvège' 'Nouvelle-Calédonie' 'Nouvelle-Zélande' 'Suisse' 'Suède'

On va Approfondir l'analyse des 26 pays

la liste de 26 pays est moyennement importante. On peut l'affiner en échantillon une liste plus sélective.

```
#Création d'un sous-ensemble avec sélection des pays des deux clusters 3 & 1 validés
df_cls_subset = df_cls5.query('[3, 1] in cluster')
df_cls_subset.shape
```

```
(26, 11)
```

```
#Visualisation des premières lignes de notre nouvel échantillon
#Les correspondances des clusters sont bien intégrés…
df_cls_subset.head()
```

```
df_subset= df_cls_subset
del df_subset['cluster']
df_subset.head()
```

	Pays	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %
10	Australie	3307.0	108.01	57628.863849	12264.0	45889.0	95012.0	0.0	0.90	1.275318
11	Autriche	3694.0	108.11	47309.051637	120216.0	144215.0	16736.0	1216.7	1.05	0.810519

```
# 26 Pays sur nos 9 variables, indexation selon les pays de l'échantillon "df_subset"
df_subset.set_index('Pays', inplace=True)
df_subset.shape
```

```
(26, 9)
```

```
#Transformation de l'échantillon en données array numpy
Xs = df_subset.values
```

```
#Centrage / réduction des données
std_scale = preprocessing.StandardScaler().fit(Xs)
Xs_scaled = std_scale.transform(Xs)
```

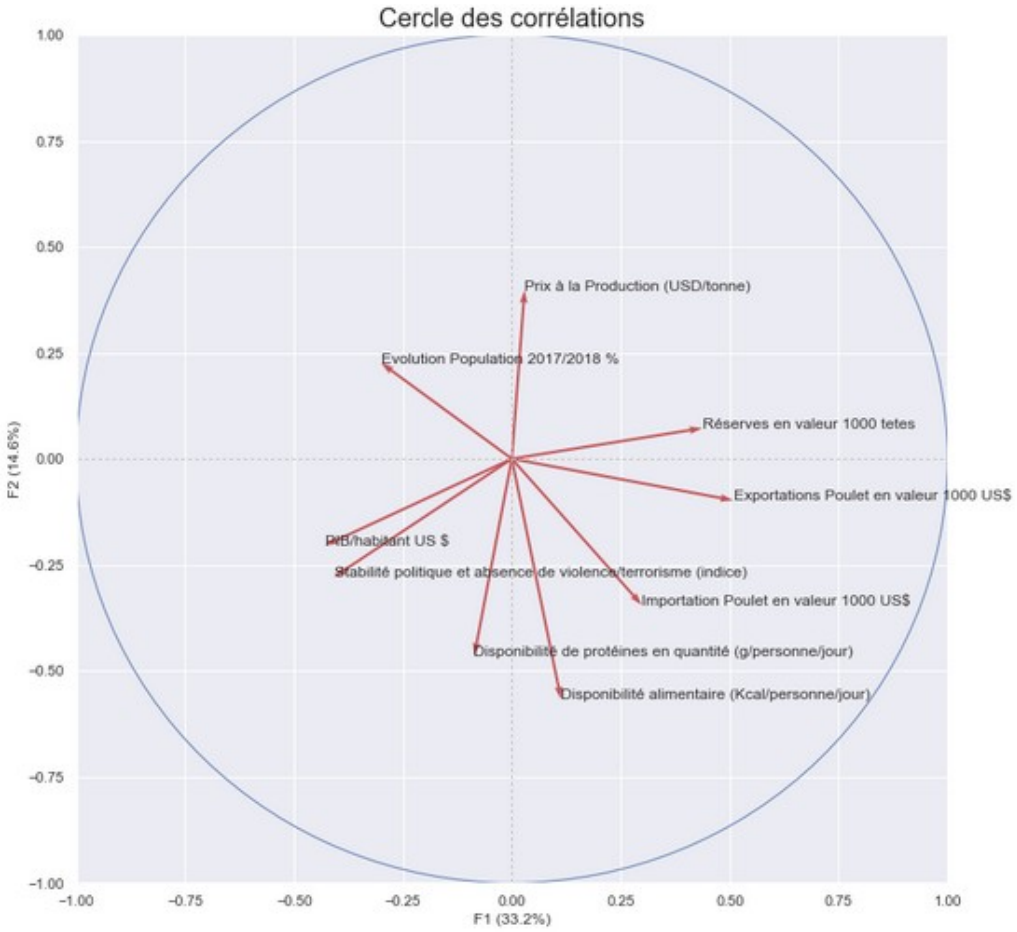
On Projecte ici les variables sur le premier plan factoriel et on créer un Cercle des corrélations de nos variables

Le but est de simplifier l'ensemble des variables en **deux principales variables**, afin de pouvoir identifier des similitudes. On parlera alors de corrélation des variables.

```
#Calcul des composantes principales: instanciation de l'objet pcas
pcas = decomposition.PCA(svd_solver='full')
pcas.fit(Xs_scaled)
```

```
PCA(svd_solver='full')
```

```
#Cercle des corrélations de nos variables
pcs = pcas.components_

fig = plt.subplots(figsize=(12,12))
plt.xlim(-1,1)
plt.ylim(-1,1)

plt.quiver(np.zeros(pcs.shape[1]), np.zeros(pcs.shape[1]),
           pcs[0,:], pcs[1,:],
           angles='xy', scale_units='xy', scale=1, color='r', width= 0.003)

for i, (x, y) in enumerate(zip(pcs[0, :], pcs[1, :])):
    plt.text(x, y, df_subset.columns[i])

circle = plt.Circle((0,0), 1, facecolor='none', edgecolor='b')
plt.gca().add_artist(circle)

#Ajout des axes
plt.plot([-1,1],[0,0],color='silver',linestyle='--',linewidth=1)
plt.plot([0,0],[-1,1],color='silver',linestyle='--',linewidth=1)

plt.title('Cercle des corrélations', fontsize=20)
plt.xlabel('F{} ({}%)'.format(1, round(100*pca.explained_variance_ratio_[0],
plt.ylabel('F{} ({}%)'.format(2, round(100*pca.explained_variance_ratio_[1],

plt.savefig("../P9/cercle_correlation.png")
plt.show()
```



Recherchons des groupes de variables fortement corrélées deux à deux entre elle

l'objectif est de pouvoir les synthétiser par une variable unique F1 et F2 (nos composantes principale

```
#Calcul des composantes principales
#Ici seulement F1 et F2 seront utiles à l'interprétation attendue
Xs_projected = pcas.transform(Xs_scaled)

df_facto = pd.DataFrame(Xs_projected, index=df_subset.index, columns=["F" + str(i+1) for i in range(9)]).iloc[:, :2]
df_facto.head() #Affichage des 5 premières lignes
```

Pays	F1	F2
Australie	-1.162080	0.098186
Autriche	-0.156968	-1.020354
Belgique	1.896492	-1.782392
Canada	0.173701	-0.489355
Chine - RAS de Hong-Kong	-0.248215	-0.520750

```
#Coefficients de la composante principale F1
F1 = pcas.components_[0]
print(F1)
#Coefficients de la composante principale F2
F2 = pcas.components_[1]
print(F2)
```

```
[ 0.11295685 -0.08629645 -0.42895989  0.508285    0.29837728  0.43729603
  0.02922789 -0.40778124 -0.30006136]
[-0.56749885 -0.46426878 -0.20465815 -0.09798902 -0.34481442  0.07217723
  0.39777241 -0.27844932  0.22594092]
```

```
Les coefficients ci-dessus  identifier clairement les variables qui contribuent le plus à F1 et F2 :
 . 0.51 * Exportations Poulet en valeur 1000 US$
 . 0.29 * Importations Poulet en valeur 1000 US$
 . 0.44 * Réserves en valeur 1000 tetes
 . 0.29 * Prix à la Production (USD/tonne)

Et pour F2
 . 0.22 * Evolution Population 2017/2018 %
 . 0.39 * Prix à la Production (USD/tonne)
```

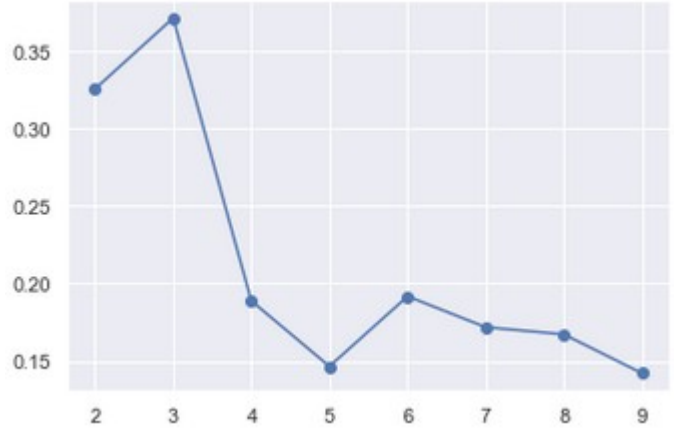
- Sur la **composante F1** la réprésentation des variables se rapproche davantage d'une **dimension Marché** (Réserves, Importation, Exportation).
- Tandis que dans la **composante F2** sont egalement ajoutés Evolution Population 2017/2018 %.

Recherchons des groupes de variables fortement corrélées deux à deux entre elles (suite)

Nos données sont enrichies par une nouvelle dimension, ce qui permettra d'affiner nos choix.

Nous allons appliquer le clustering K-Means à notre nouvel échantillon "df_subset":

```
#Calcul de la métrique "silhouette" pour différents nombres de groupes issus de la méthode des centres mobiles
#Liste pour stocker nos coefficients
silhouettes = []

#Boucle itérative de 2 à 10 pour tester les possibilités
for k in range(2, 10):
    #Création et ajustement d'un modèle pour chaque k
    cls = cluster.KMeans(n_clusters=k)
    cls.fit(Xs_scaled)

    #Stockage des coefficients associés
    silh = metrics.silhouette_score(Xs_scaled, cls.labels_)
    silhouettes.append(silh)

#Visualisation des valeurs de coefficient de silhouette pour chaque nombre de cluster
plt.plot(range(2, 10), silhouettes, marker='o')
plt.show()
```



Un nouveau clustering en 2 clusters semble être une solution adéquate à nos données. Derrière cette aide à la décision du nombre idéal de clusters par rapport aux informations que nous perdrons, il est important de garder à l'esprit notre objectif métier, celui de pouvoir obtenir une liste de pays assez "courte" donc exploitable pour d'autres analyses futures, par exemple en intégrant des dimensions métier propre à notre entreprise.

```
#Nouveau clustering avec k = 2
cls2 = cluster.KMeans(n_clusters=2)
cls2.fit(Xs_scaled)
```

```
KMeans(n_clusters=2)
```

```
#Tableau d'aide à la comparaison des clusters par les centroïdes
centroids2 = cls2.cluster_centers_
df_centroides = pd.DataFrame(centroids2, columns=df_subset.columns)
df_centroides
```

	Disponibilité alimentaire (Kcal/personne /jour)	Disponibilité de protéines en quantité (g/personne /jour)	PIB/habitant US $	Exportations Poulet en valeur 1000 US$	Importation Poulet en valeur 1000 US$	Réserves en valeur 1000 tetes	Prix à la Production (USD/tonne)	Stabilité politique et absence de violence/terrorisme (indice)	Evolution Population 2017/2018 %
0	0.230469	0.211381	0.190210	-0.112410	0.107827	-0.138978	-0.189064	0.194576	-0.021474
1	-1.766928	-1.620586	-1.458276	0.861806	-0.826676	1.065496	1.449493	-1.491751	0.164634

Visualisons et caractérisons 2 groupes par Boxplot

(group 0 , identifiés comme potentiellement intéressant)

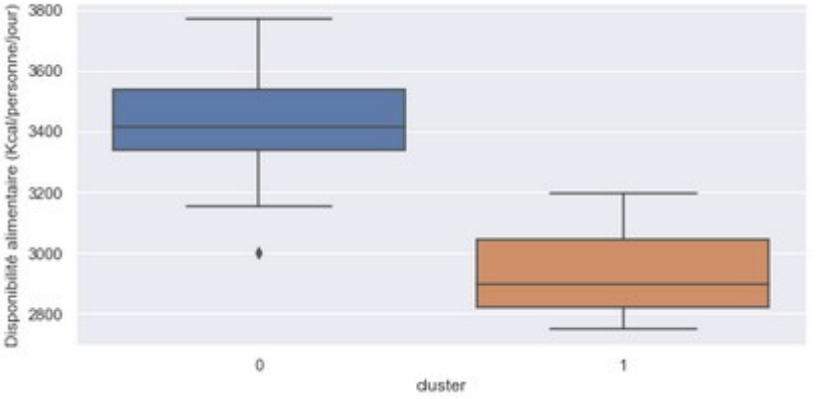
```
#Index trié des clusters
idk = np.argsort(cls2.labels_)

#Affichage des observations selon leurs clusters
df_cls2 = pd.DataFrame(df_subset.index[idk], cls2.labels_[idk]).reset_index()
df_cls2 = df_cls2.rename(columns={'index':'cluster'})
```

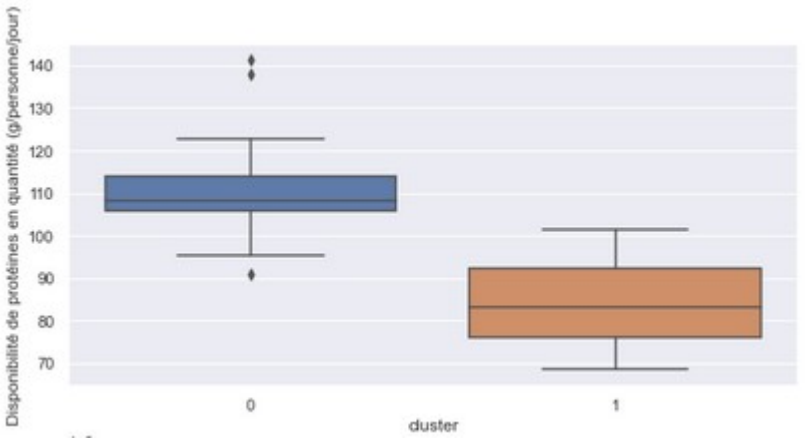
```
#Intégration des numéros de cluster pour chacun des pays restants
#Jointure interne avec le dataframe "df_subset" des 75 pays vu précédemment
df_cls2 = pd.merge(df_subset, df_cls2, on='Pays')
df_cls2.shape
```

```
(26, 11)
```

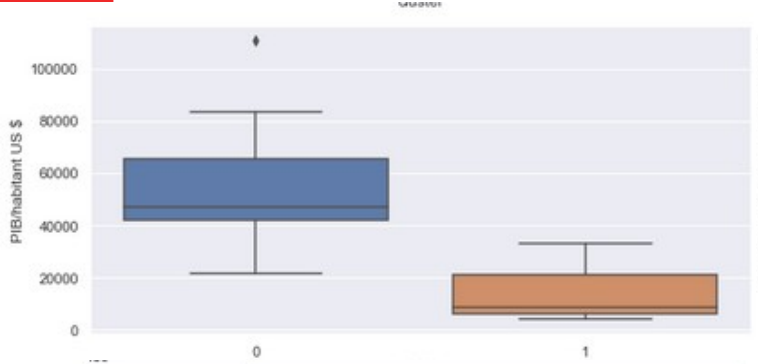
```
#Visu. des premières lignes… l'échantillon semble "propre" et complet avec toutes nos variables
df_cls2.head()
```

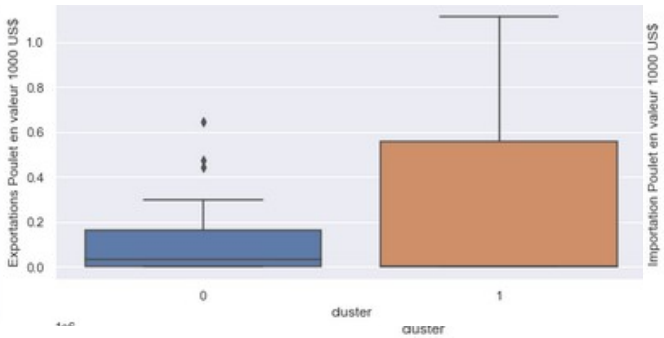
```
#Visualisation et caractérisation des groupes par Boxplot
plt.figure(figsize=(20, 20))

plt.subplot(421)
sns.boxplot(data=df_cls2, x='cluster', y='Disponibilité alimentaire (Kcal/personne/jour)')

plt.subplot(422)
sns.boxplot(data=df_cls2, x='cluster', y='Disponibilité de protéines en quantité (g/personne/jour)')

plt.subplot(423)
sns.boxplot(data=df_cls2, x='cluster', y='PIB/habitant US $')

plt.subplot(424)
sns.boxplot(data=df_cls2, x='cluster', y='Exportations Poulet en valeur 1000 US$')

plt.subplot(425)
sns.boxplot(data=df_cls2, x='cluster', y='Importation Poulet en valeur 1000 US$')

plt.subplot(426)
sns.boxplot(data=df_cls2, x='cluster', y='Réserves en valeur 1000 tetes')

plt.subplot(427)
sns.boxplot(data=df_cls2, x='cluster', y='Prix à la Production (USD/tonne)')

plt.subplot(428)
sns.boxplot(data=df_cls2, x='cluster', y='Stabilité politique et absence de violence/terrorisme (indice)')

plt.show(block=False)
```

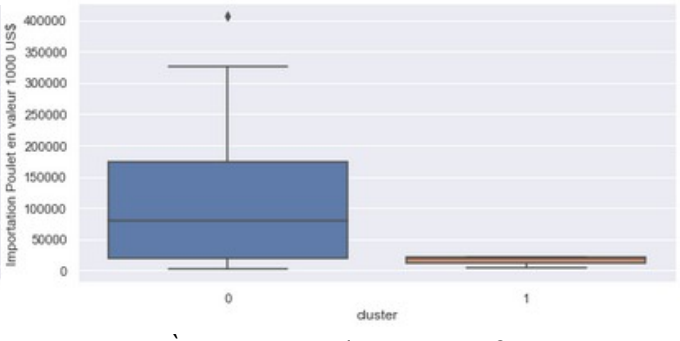


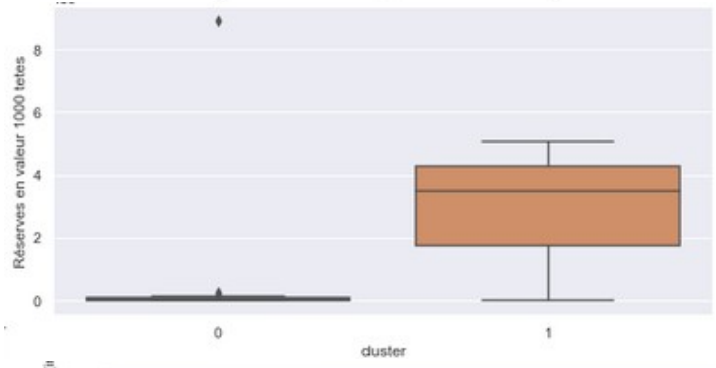


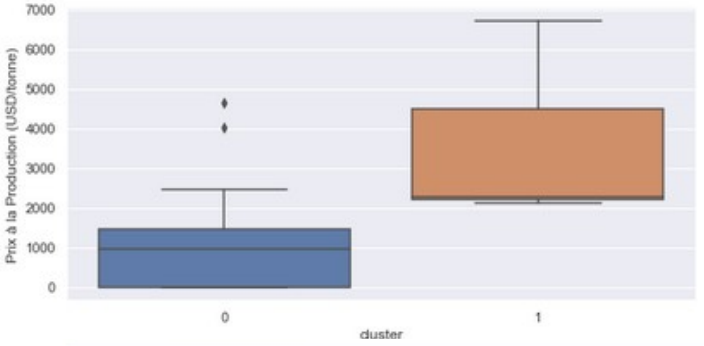
Visualisation et caractérisation des groupes par Boxplot (suite)

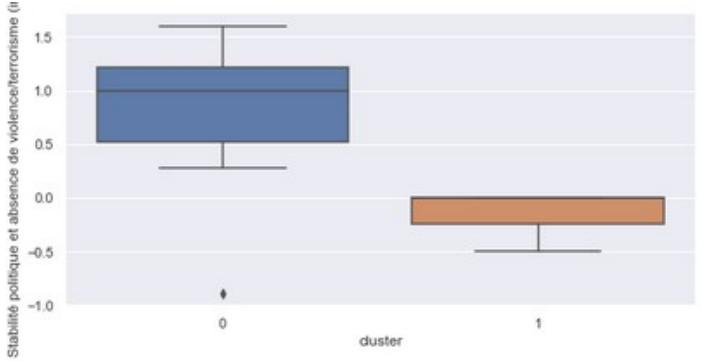


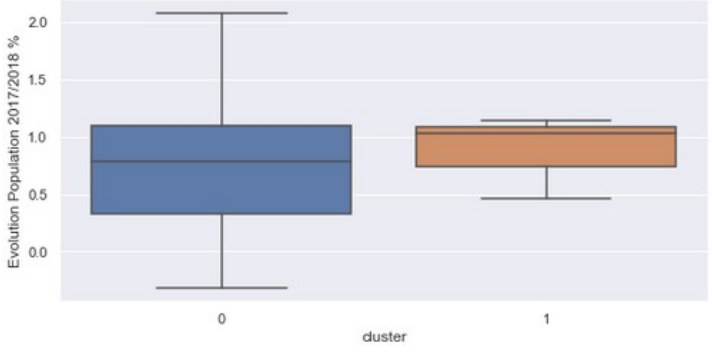












À mon avis, le groupe 0 a un potentiel de marché intéressant car il est différent du groupe 1 avec :

1. Un niveau de PIB élevé,
2. La Disponibilité de protéines en quantité (g/personne/jour) élevé,
3. Un faible niveau de réserves et d'exportation de poulets,
4. Ainsi qu'une grande stabilité politique.

Ci-dessous voyons les clusters plus en détails :

```
#Taille des clusters
print(str(len(df_cls2[df_cls2['cluster'] == 0]['Pays'])) + " pays dans le cluster 0")
print(str(len(df_cls2[df_cls2['cluster'] == 1]['Pays'])) + " pays dans le cluster 1")
```

```
23 pays dans le cluster 0
3 pays dans le cluster 1
```

```
#Pays du cluster 0
df_cls2[df_cls2['cluster'] == 0]['Pays'].unique()
```

```
array(['Australie', 'Autriche', 'Belgique', 'Canada',
       'Chine - RAS de Hong-Kong', 'Chine - RAS de Macao', 'Danemark',
       'Espagne', 'Finlande', 'France', 'Irlande', 'Islande', 'Israël',
       'Italie', 'Luxembourg', 'Malte', 'Norvège', 'Nouvelle-Zélande',
       'Portugal', 'Suisse', 'Suède', 'Émirats arabes unis',
       "États-Unis d'Amérique"], dtype=object)
```

```
#Pays du cluster 1
df_cls2[df_cls2['cluster'] == 1]['Pays'].unique()
```

```
array(['Chine, continentale', 'Indonésie', 'Nouvelle-Calédonie'],
      dtype=object)
```

Représentation de nos 2 clusters sur le premier plan factoriel (F1, F2) :

```
#Pourcentage de la variance expliquée
pcas.explained_variance_ratio_.cumsum()
```

```
l]: array([0.27994192, 0.54326377, 0.678878  , 0.77782931, 0.85657154,
           0.92726918, 0.95852802, 0.98171444, 1.        ])
```

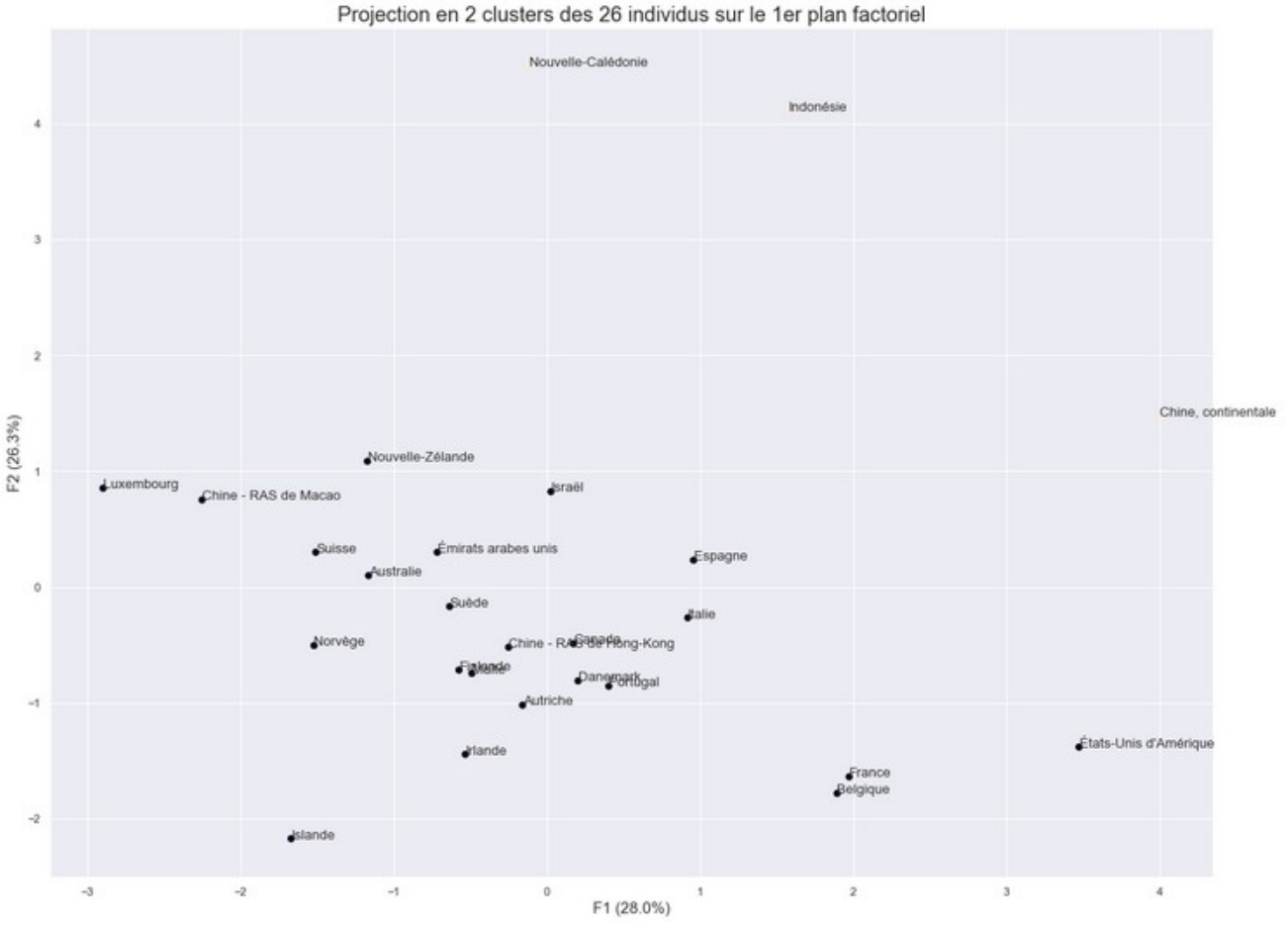
-> **54%** de la variance est expliquée avec les deux premières composantes principales.

-> Le choix sera fait selon la stratégie de l'entreprise

Par exemple, si l'objectif est de cibler les pays au plus haut PIB/habitant . Ou de faibles niveaux d'exportation / d'importation ou de réserves (voir ci-dessous)

Projection des 2 clusters sur le premier plan factoriel (F1, F2) :

```
#Projection des 2 clusters sur le premier plan factoriel (F1, F2)
#Coordonnées factorielles
Xs_projected = pcas.transform(Xs_scaled)
plt.figure(figsize=(20, 15))

plt.scatter(Xs_projected[:, 0], Xs_projected[:, 1], c=cls2.labels_)
for i,(x,y) in enumerate(Xs_projected[:,[0,1]]):
                    plt.text(x, y, df_subset.index[i], fontsize='13')

plt.xlabel('F{} ({}%)'.format(1, round(100*pcas.explained_variance_ratio_[0],1)),
plt.ylabel('F{} ({}%)'.format(2, round(100*pcas.explained_variance_ratio_[1],1)),
plt.title("Projection en 2 clusters des {} individus sur le 1er plan factoriel".f

plt.show()
```



->Cette projection est très intéressante car elle nous donne une vision du potentiel des pays.
->Par exemple, les pays situés à droite ont de meilleurs "Réservation" et "Exportation". Les pays situés plus haut seront d'avantage intéressant sur un aspect "Prix". (comme les variable sur la cercle de correlaiton)

->On le sait, étudier les axes d'inertie des individus est équivalent à étudier les axes principaux d'inertie des variables !
Il nous dit que, quand on analyse un échantillon, étudier des individus ou étudier des variables, c'est en fait étudier deux facettes d'une même chose!

Contribution des pays dans l'inertie totale :

Regardons les différentes contributions des pays à l'inertie totale.

```
#Affichage des 26 pays les plus contributeurs
di = np.sum(Xs_scaled**2,axis=1)
ctr_indiv_inertie = pd.DataFrame(di, index=df_subset.index, columns=['d_i']).sort_values(by='d_i', ascending=False)
ctr_indiv_inertie[:26]
```

Pays	
États-Unis d'Amérique	25.488845
Chine, continentale	25.333917
Nouvelle-Calédonie	23.920269
Indonésie	22.780082
Luxembourg	17.372553
Israël	12.425164
Belgique	12.270836
Chine - RAS de Hong-Kong	10.916857
Islande	10.792056
Chine - RAS de Macao	9.036796
France	8.815983
Portugal	6.615066
Irlande	5.569162
Nouvelle-Zélande	5.505630
Suisse	4.871846
Malte	4.435066
Italie	3.948460
Norvège	3.702171
Espagne	3.409207
Finlande	3.405036
Émirats arabes unis	3.132721
Australie	2.450217
Canada	2.264707
Autriche	2.136644
Suède	1.771846
Danemark	1.628862

--> Les premiers pays intéressants dont la contribution dans l'inertie totale est "maximale" sont : Luxembourg, Belgique,Israël, Islande,Irlande, Suisse.
→ Ce choix a également été fait en tenant compte de la proximité géographique des cibles.
→ On remarquera que ces pays du **cluster 0**.

Aussi grand contibution dans l'inertie total ont États-Unis d'Amérique, Calédonie, Chine, Indanesie , mais ils sont loin.

Merci pour votre attention !

Data source FAO (Food and Agriculture Organization)
http://www.fao.org/faostat/fr/#data\