

**Parcours de Data Analyst avec**  
**OPENCLASSROOMS**

**Projet 5 : Produisez une étude de marché**

**Cédric PAPIN**

**Mentor: Benjamin Marlé**

**OPENCLASSROOMS**

**18 décembre 2018**

# Sommaire

- ☆ **Présentation du projet**
- ☆ **Librairies**
- ☆ **Fonctions utiles**
- ☆ **Data Set**
- ☆ **Construction et nettoyage du Data Set**
- ☆ **ACP**
- ☆ **Dendrogramme**
- ☆ **Clustering**
- ☆ **Analyse des résultats obtenus**
- ☆ **Tests statistiques**
- ☆ **Test d'adéquation**
- ☆ **Test de comparaison de deux populations**

# Présentation du projet

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Présentation du projet

- ★ Je travaille dans une entreprise d'agroalimentaire spécialisée dans le poulet
- ★ Celle-ci souhaite se développer à l'international
- ★ Je dois donc l'aider à cibler certains pays en travaillant idéalement par groupes
- ★ Nous allons également étudier le régime alimentaire de chaque pays



## Librairies

- ☆ Présentation du projet
- ☆ **Librairies**
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Librairies

- pandas
- numpy
- sklearn
- scipy.cluster.hierarchy
- scipy.stats
- matplotlib.pyplot
- matplotlib.collections

## Fonctions utiles

- ☆ Présentation du projet
- ☆ Librairies
- ☆ **Fonctions utiles**
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Fonctions utiles

1 script notebook contient les lignes de code de toutes nos fonctions utiles

```
def display_circles(pcs, n_comp, pca, axis_ranks, labels=None, label_rotation=0, lims=None):
    for d1, d2 in axis_ranks: # On affiche les 3 premiers plans factoriels, donc les 6 premiÃ"res composantes
        if d2 < n_comp:

            # initialisation de la figure
            fig, ax = plt.subplots(figsize=(7,6))

            # dÃ©termination des limites du graphique
            if lims is not None :
                xmin, xmax, ymin, ymax = lims
            elif pcs.shape[1] < 30 :
                xmin, xmax, ymin, ymax = -1, 1, -1, 1
            else :
                xmin, xmax, ymin, ymax = min(pcs[d1,:]), max(pcs[d1,:]), min(pcs[d2,:]), max(pcs[d2,:])

            # affichage des flÃ"ches
            # s'il y a plus de 30 flÃ"ches, on n'affiche pas le triangle Ã  leur extrÃ©mitÃ©
            if pcs.shape[1] < 30 :
                plt.quiver(np.zeros(pcs.shape[1]), np.zeros(pcs.shape[1]),
                           pcs[d1,:], pcs[d2,:],
                           angles='xy', scale_units='xy', scale=1, color="grey")
                # (voir la doc : https://matplotlib.org/api/\_as\_gen/matplotlib.pyplot.quiver.html)
            else:
                lines = [[[0,0],[x,y]] for x,y in pcs[[d1,d2]].T]
                ax.add_collection(LineCollection(lines, axes=ax, alpha=.1, color='black'))
```

## Data Set

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ **Data Set**
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Data Set

★ Il me revient de sélectionner les données utiles sur le site de la FAO, ce choix se fait en fonction des variables, que je compte obtenir :

- Différence de population entre 2008 et 2013, exprimée en pourcentage
- Proportion de protéines d'origine animale par rapport à la quantité totale de protéines de la disponibilité alimentaire du pays
- Disponibilité alimentaire en protéines par habitant
- Disponibilité alimentaire en calories par habitant
- PIB par habitant
- Quantité de volaille importée par pays

★ Et hop, nous allons construire et nettoyer tout ça ...



# Construction et nettoyage du Data Set

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ **Construction et nettoyage du Data Set**
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Valeurs manquantes

```
In [6]: # recherche des valeurs manquantes au sein de la table 'pop'  
pop.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 232 entries, 0 to 231  
Data columns (total 6 columns):  
Code zone    232 non-null int64  
Zone         232 non-null object  
Produit      232 non-null object  
2008        230 non-null float64  
2013        230 non-null float64  
%evopop     228 non-null float64  
dtypes: float64(3), int64(1), object(2)  
memory usage: 11.0+ KB
```

☆ recherche avec `.info()`

☆ résultat : la table ‘pop’ contient des valeurs manquantes

☆ recherche des individus ayant comme valeur 'NaN' pour la variable '%evopop'

```
In [7]: # recherche des individus ayant comme valeur 'NaN' pour la variable '%evopop'  
pop.loc[np.isnan(pop['%evopop']),:]
```

Out[7]:

	Code zone	Zone	Produit	2008	2013	%evopop
134	151	Antilles néerlandaises (ex)	Population-Estimations	190320.0	NaN	NaN
183	206	Soudan (ex)	Population-Estimations	42218632.0	NaN	NaN
229	276	Soudan	Population-Estimations	NaN	36849918.0	NaN
230	277	Soudan du Sud	Population-Estimations	NaN	11177490.0	NaN

## Valeurs manquantes

- ★ **Recherches supplémentaires sur le web pour conservation ou non de ces individus :**
- ★ depuis 2010, les Antilles néerlandaises ont été dissoutes et rattachées aux Pays-Bas, nous pouvons donc retirer cet individu de notre BDD
- ★ Suite à un référendum en 2011, le Soudan du sud a fait Sécession, donc sur notre intervalle de temps, qui va de 2008 à 2013, nous n'avons pas les données relatives à la population nécessaires, de plus au vu de sa grande instabilité politique, il nous sera difficile de nous y implanter, nous pouvons donc également retirer ce individu de notre BDD

## Valeurs aberrantes

- ★ recherche avec `.sort_values()`
- ★ nous faisons un tri descendant et ascendant
- ★ cette recherche est fait sur chacune des variables

```
In [11]: # recherche des valeurs aberrantes
# on trie la variable 'Code zone' avec la méthode .sort_values()
pop.sort_values(by='Code zone', ascending=True).head()
```

```
Out[11]:
```

	Code zone	Zone	Produit	2008	2013	%evopop
0	1	Arménie	Population-Estimations	2908220.0	2893509.0	-0.505842
1	2	Afghanistan	Population-Estimations	27294031.0	31731688.0	16.258709
2	3	Albanie	Population-Estimations	2991651.0	2918978.0	-2.429194
3	4	Algérie	Population-Estimations	34860715.0	38338562.0	9.976408
4	5	Samoa américaines	Population-Estimations	57030.0	55307.0	-3.021217

## Valeurs en doublons

- ★ 1 : recherche du nombre d'éléments
- ★ 2 : suppression de doublons éventuels
- ★ 3 : recherche du nombre d'éléments après traitement

```
In [24]: # recherche du nombre d'éléments au sein de la table 'pop'  
len(pop)
```

```
Out[24]: 228
```

```
In [25]: # détection et suppression des doublons au sein de la variable 'Pays'  
pop.drop_duplicates(subset='Pays',keep='first',inplace=True)  
len(pop)
```

```
Out[25]: 228
```

## Jointure entière



Rien ne se perd tout se transforme ...

	Code Pays	Pays	%evopop	PIB/hab	%protani	Dispoalim(Kcal/hab/an)	Dispoprot(kg/hab/an)	impvola	expvola
0	1	Arménie	-0.505842	3843.590751	48.034207	1068720.0	32.86460	32000000.0	0.0
1	2	Afghanistan	16.258709	681.033974	20.978541	762850.0	21.26125	48000000.0	NaN
2	3	Albanie	-2.429194	4376.970549	53.329743	1165445.0	40.66830	25000000.0	0.0
3	4	Algérie	9.976408	5471.866638	27.174859	1203040.0	33.56540	3000000.0	0.0
4	6	Andorre	-3.664397	40621.912722	Nan	Nan	Nan	Nan	Nan

## Valeurs manquantes

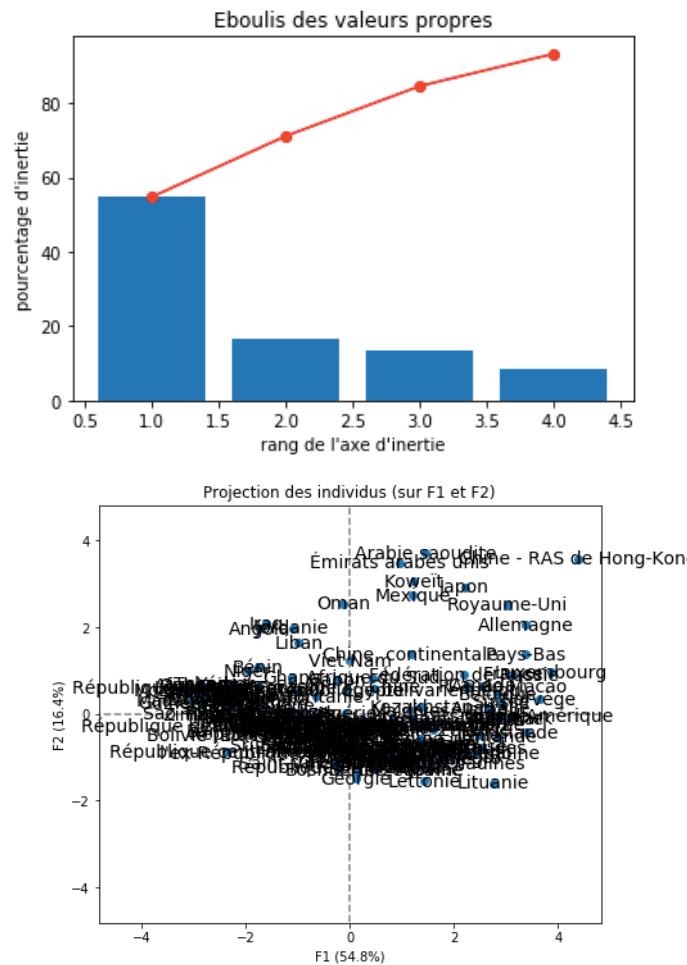
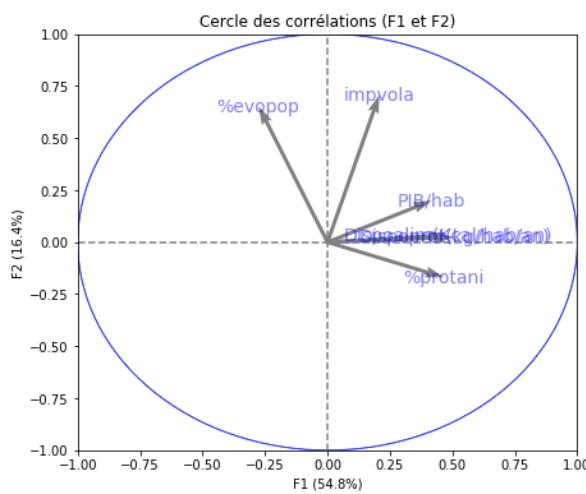
- ★ recherche avec `.info()`
- ★ résultat : la table ‘jointure’ contient des valeurs manquantes
- ★ recherche des individus ayant comme valeur ‘NaN’ sur les différentes variables
- ★ recherches supplémentaires sur le web pour conservation ou non de ces individus
- ★ Suite à ces traitements la table est exportée sous format `.csv`

```
36]: # recherche de valeurs manquantes au sein de la table 'jointure'
jointure.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 207 entries, 0 to 206
Data columns (total 9 columns):
Code Pays          207 non-null int64
Pays              207 non-null object
%evopop           206 non-null float64
PIB/hab           206 non-null float64
%protani          173 non-null float64
Dispoalim(Kcal/hab/an) 173 non-null float64
Dispoprot(kg/hab/an) 173 non-null float64
impvola           172 non-null float64
expvola            146 non-null float64
dtypes: float64(7), int64(1), object(1)
memory usage: 16.2+ KB
```

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

- ★ Eboulis des valeurs propres
- ★ représentation des variables
- ★ représentation des individus

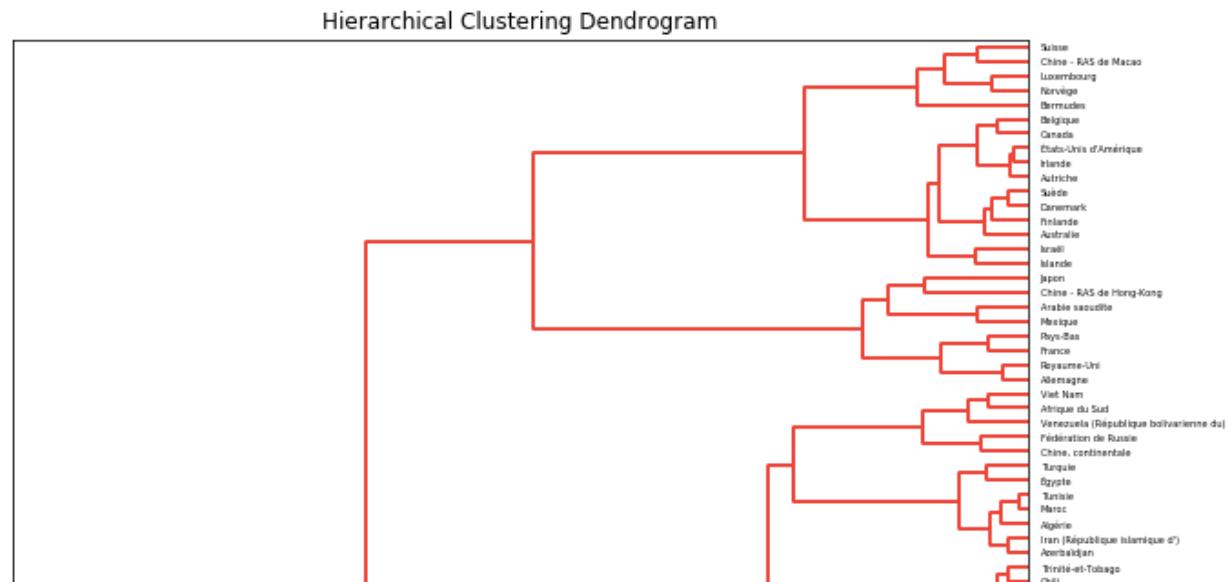


# Dendrogramme

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ **Dendrogramme**
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Dendrogramme

```
3]: # Clustering hiérarchique  
z = linkage(X_scaled, 'ward')  
  
4]: # Affichage du dendrogramme  
plot_dendrogram(z, names)
```



# Clustering

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ **Clustering**
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

# Clustering

```
click to expand output; double click to hide output | en 5 clusters
n_comp = 5
clusters = fcluster(Z, n_comp, criterion='maxclust')

.6]: clusters
.6]: array([3, 2, 3, 3, 2, 3, 3, 5, 5, 3, 3, 2, 5, 2, 2, 3, 3, 2, 3, 3, 3, 3,
   5, 3, 2, 2, 2, 3, 3, 3, 2, 3, 3, 3, 2, 5, 3, 3, 3, 3, 3, 3, 3, 3, 3,
   3, 5, 4, 3, 2, 3, 3, 2, 4, 3, 2, 3, 3, 3, 2, 2, 3, 2, 2, 4, 3, 3,
   5, 2, 2, 3, 2, 5, 5, 3, 2, 3, 3, 4, 1, 3, 2, 2, 3, 1, 3, 2, 1, 2,
   2, 3, 5, 2, 2, 3, 3, 3, 3, 3, 4, 3, 3, 2, 3, 2, 2, 4, 3, 3, 3,
   3, 2, 2, 2, 5, 2, 3, 3, 3, 2, 3, 3, 2, 2, 2, 3, 2, 3, 3, 3, 3, 3,
   2, 4, 2, 2, 3, 3, 3, 3, 2, 2, 5, 5, 3, 2, 3, 2, 3, 1, 3, 3, 1,
   2, 4, 3, 5, 2, 3, 3, 3, 2, 3, 2, 2, 5, 5, 3, 3], dtype=int32)

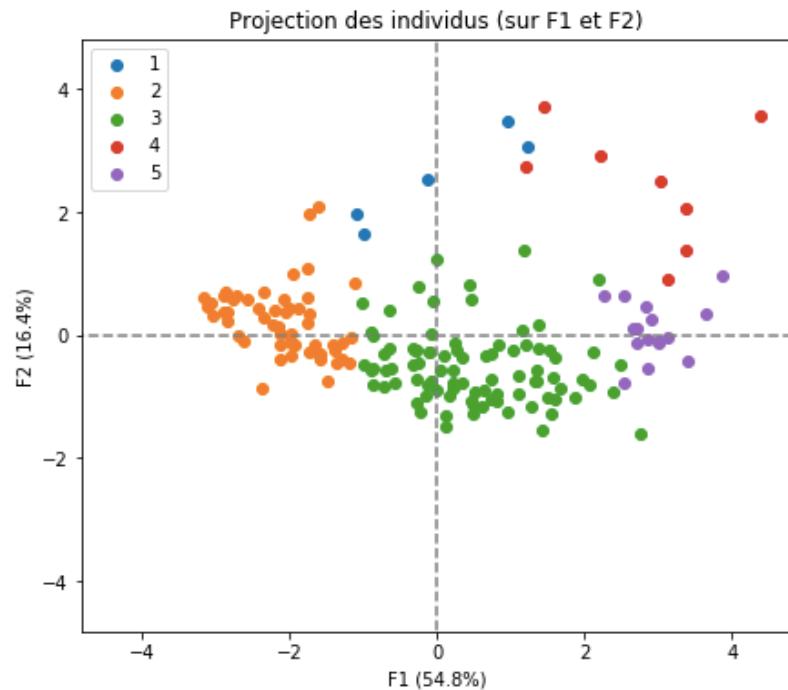
.7]: # jointure des 2 Dataframes
data["clusters"] = clusters
data.head()

.7]:
    %evopop      PIB/hab  %protani Dispoalim(Kcal/hab/an)  Dispoprot(kg/hab/an)      impvola  clusters
    Pays
    Arménie -0.505842  3843.590751  48.034207           1068720.0          32.86460  32000000.0    3
    Afghanistan 16.258709  681.033974  20.978541           762850.0          21.26125  48000000.0    2
    Albanie -2.429194  4376.970549  53.329743           1165445.0          40.66830  25000000.0    3
    Algérie  9.976408  5471.866638  27.174859           1203040.0          33.56540  3000000.0     3
    Angola  19.480850  5258.408672  32.134125           902645.0          20.89990  342000000.0   2
```

# Clustering

## ★ Visualisation des partitions par ACP

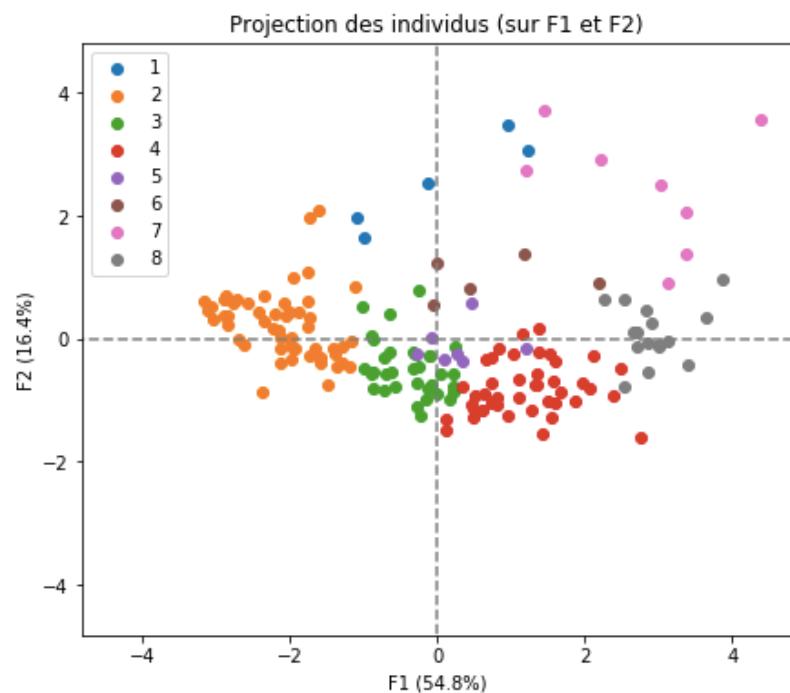
```
display_factorial_planes(X_projected, n_comp, pca, [(0,1),(1,2)], illustrative_var=clusters)
```



## Clustering

- ★ Pour un découpage plus précis, découpe du dendrogramme en 8 clusters
- ★ Visualisation des partitions par ACP

```
display_factorial_planes(X_projected, n_comp, pca, [(0,1),(1,2)], illustrative_var=clusters)
```



## Analyse des résultats obtenus

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ **Analyse des résultats obtenus**
- ☆ Tests statistiques
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

## Analyse des résultats obtenus

### ★ Caractérisation de chacun des groupes obtenus

```
: dataGB = data.groupby(by='clusters').mean()  
dataGB
```

	%evopop	PIB/hab	%protani	Dispoalim(Kcal/hab/an)	Dispoprot(kg/hab/an)	impvola
clusters						
1	31.760946	25139.924007	42.413275	1.174570e+06	33.609930	1.466000e+08
2	14.963904	960.954781	19.234184	8.629006e+05	21.314107	2.466667e+07
3	7.801552	3210.897084	33.566005	9.057046e+05	23.148193	1.555882e+07
4	9.663554	7135.140223	47.548207	9.886481e+05	28.284002	9.070833e+07
5	2.189476	19854.010673	57.413805	1.159542e+06	36.244878	4.413793e+07
6	1.762165	6770.155272	41.388724	1.119805e+06	31.452196	3.452000e+07
7	4.323535	31935.555961	54.225323	1.189848e+06	35.725679	7.624286e+08
8	5.074312	65265.871579	62.156234	1.247805e+06	39.933607	6.735714e+07

## Analyse des résultats obtenus

### ★ Sélection d'un groupe de pays

```
] : # suite à l'analyse des cercles de corrélations :  
# le cluster 1 nous semble très approprié  
  
] : data_1 = data.loc[data["clusters"].isin([1]),:]  
data_1.head()  
  
]  
%evopop PIB/hab %protani Dispoalim(Kcal/hab/an) Dispoprot(kg/hab/an) impvola clusters  
Pays  
Jordanie 29.640905 3992.867103 34.150019 1131500.0 29.05035 84000000.0 1  
Koweït 35.668316 48401.609794 49.501109 1277865.0 39.50760 139000000.0 1  
Liban 28.339618 8721.248317 36.783340 1119090.0 29.09415 17000000.0 1  
Oman 34.522007 21268.756403 51.773371 1147195.0 32.21125 118000000.0 1  
Émirats arabes unis 30.633882 43315.138419 39.858536 1197200.0 38.18630 375000000.0 1
```

## Analyse des résultats obtenus

- ★ **Interprétation de la sélection :**

- ★ - **Jordanie** : PIB/hab le plus faible de la sélection

- ★ - **Koweït** : fin prochaine de l'ère des Hydrocarbures (dont le pétrole), risque de profonds troubles sociaux, industrie rare, pays désertique, le tourisme ne peut pas être développé, la côte étant polluée, conséquence de la guerre du Golfe de 1990-1991

- ★ - **Liban** : victime d'une importante stabilité politique

- ★ - **Oman** : semble être le candidat idéal : population en forte hausse, PIB/hab important, nombreuses importations beaucoup de volailles, économie diversifiée (semi-conducteurs, robotique, infrastructures portuaires, tourisme de luxe), de part sa position géographique, il nous sera possible d'exporter dans les pays de la péninsule arabique, en effet, notre logistique au Moyen-Orient en sera facilitée

# Tests statistiques

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ **Tests statistiques**
- ☆ Test d'adéquation
- ☆ Test de comparaison de deux populations

- ★ **Dans notre partition, nous avons obtenu des groupes distincts, nous allons vérifier qu'ils diffèrent réellement.**
  
- ★ **Pour cela, nous allons effectuer les tests statistiques suivants :**
  - **Un test d'adéquation** : nous devons trouver une variable dont la loi est normale
  - **Un test de comparaison de deux populations** : nous devons choisir 2 clusters

# Tests d'adéquation

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ **Tests d'adéquation**
- ☆ Test de comparaison de deux populations

## Tests d'adéquation

- ★ Parmi notre ensemble de variables, nous en cherchons une dont la loi est normale
- ★ Nous allons utiliser le test de Shapiro-Wilk
- ★ Posons nos hypothèses :
  - $H_0$  suppose qu'un échantillon  $x_1, \dots, x_n$  est issu d'une population normalement distribuée
  - $H_1$  au contraire suppose qu'un échantillon  $x_1, \dots, x_n$  est issu d'une population ne suivant pas une distribution normale
- ★ Il s'agit d'un test bilatéral (nous testons une différence)
- ★ A tout test est associé un risque à dit de première espèce, il s'agit de la probabilité de rejeter l'hypothèse de normalité alors qu'elle est vraie
- ★ Nous nous fixons un niveau d'erreur alpha de 5%

## Interprétation des résultats

```
In [89]: shapiro(bdd['%evopop'])
Out[89]: (0.9437400698661804, 2.735466978265322e-06)

In [90]: shapiro(bdd['PIB/hab'])
Out[90]: (0.6918768882751465, 1.767976798345929e-17)

In [91]: shapiro(bdd['%protani'])
Out[91]: (0.9579640030860901, 5.209694427321665e-05)

In [92]: shapiro(bdd['Dispoalim(Kcal/hab/an)'])
Out[92]: (0.9829756617546082, 0.0345720537006855)

In [93]: shapiro(bdd['Dispoprot(kg/hab/an)'])
Out[93]: (0.9817173480987549, 0.02383829839527607)

In [94]: shapiro(bdd['impvola'])
Out[94]: (0.4882097840309143, 4.598939927887374e-22)
```

- ★ **Sur l'ensemble de nos variables , nous obtenons une p-value inférieure à 5%, ce qui nous permet de rejeter l'hypothèse nulle**
- ★ **Aucune des variables de notre DataFrame ne suit une distribution normale**

# Tests de comparaison de deux populations

- ☆ Présentation du projet
- ☆ Librairies
- ☆ Fonctions utiles
- ☆ Data Set
- ☆ Construction et nettoyage du Data Set
- ☆ ACP
- ☆ Dendrogramme
- ☆ Clustering
- ☆ Analyse des résultats obtenus
- ☆ Tests statistiques
- ☆ Tests d'adéquation
- ☆ **Test de comparaison de deux populations**

## Tests de comparaison de deux populations

- ★ Parmi notre échantillon, nous en sélectionnons les 2 clusters ayant le plus grand nombre d'individus
- ★ Nos variables ne suivant pas une distribution normale, nous testons l'égalité des médianes de la variable 'Dispoalim(Kcal/hab/an)' sur les clusters 3 et 2 à l'aide d'un test de Kolmogorov-Smirnov
- ★ En effet, ce test est non paramétrique, il n'a pas d'hypothèse de normalité
- ★ Posons nos hypothèses :
  - H0 suppose que la médiane soit égale à la médiane de l'autre
  - H1 au contraire suppose que les médianes sont différentes
- ★ Il s'agit d'un test bilatéral (nous testons une différence)
- ★ A tout test est associé un risque a dit de première espèce, il s'agit de la probabilité de rejeter l'hypothèse de normalité alors qu'elle est vraie
- ★ Nous nous fixons un niveau d'erreur alpha de 5%

## Interprétation des résultats

```
]# Test d'égalité des variances sur les clusters 3 et 2 sur la variable 'Dispoalim(Kcal/hab/an)'  
clust_2 = data.loc[data['clusters']==2,'Dispoalim(Kcal/hab/an)']  
clust_3 = data.loc[data['clusters']==3,'Dispoalim(Kcal/hab/an)']  
  
]# Test d'égalité des moyennes sur les clusters 3 et 5 sur la variable 'Dispoalim(Kcal/hab/an)'  
ks_2samp(clust_2, clust_3)  
# Test de comparaison de médianes  
# H0 = médiane de l'un égale médiane de l'autres  
  
]: Ks_2sampResult(statistic=0.630718954248366, pvalue=4.2661702597496815e-08)
```

- ★ Notre hypothèse H0 était que la médiane de l'un soit égale à la médiane de l'autre
- ★ Notre p-value est inférieure à 5%
- ★ Nous rejetons donc l'hypothèse H0, par conséquent les médianes ne sont pas égales
- ★ Les valeurs d'un clusters semblent différentes de celles de l'autre cluster, sur la variable testée

## Remerciements

- OPENCLASSROOMS
- Benjamin Marlé, mon mentor
- Le mentor en charge de ma soutenance pour ce projet

