Etude de marché de poulets

26 juillet 2021

PLAN

Vue d’ensemble

1. Préparation des données
2. Classification ascendante hiérarchique (1er clustering)
3. Caractérisations des groupes
4. Analyse en Composantes Principales
5. Test de normalité et test de comparaison
6. Choix du sous-groupe du cluster le plus intéressant

Conclusion

Vue d'ensemble

Notre entreprise spécialisée dans la vente de poulets souhaite se développer dans le monde.

Notre objectif est de cibler particulièrement des groupes de pays via une étude approfondie du marché de poulets.

1. Préparation des données

- Importation de données et création de dataframes
- Jointure
- Nettoyage de données

Noms de dataframes et de variables

percent_pop: Évolution de la population entre 2003 et 2013 (%)

ratio_proteine: rapport entre la disponibilité en protéine animale et la disponibilité totale en protéine

proteine_jr: Disponibilité totale en protéine (g/pers/jour)

kcal_jr: Disponibilité totale en kcal (Kcal/pers/jour)

pib: PIB/hbt/an par pays

pop: dataframe de la population

dispo_alim_2003_2013: dataframe disponibilité alimentaire totale de proteine et de kcal

dispo_anim: dataframe disponibilité alimentaire de proteine et de kcal d'origine animale

data: dataframe principal

clusters: dataframe du 1er clustering

cluster_pib: dataframe de la jointure du pib dans le cluster choisi

Importation de données et création de tables

Importation de la population entre 2003 et 2013

pop: dataframe de l'évolution de
la population entre 2003 et 2013

```
population = pd.read_csv("pop_2003_2013_2005_2015.csv", index_col = 0)
population['Valeur'] = population['Valeur'] * 1000
population

#Différence entre les populations de 2003 et de 2013
pop = pd.pivot_table(population, values='Valeur', index=['Zone'],
                     columns='Année', aggfunc=sum).reset_index()
pop['percent_pop'] = (((pop[2013] - pop[2003])/(pop[2003]))*100).round(2)
pop = pop[['Zone', 'percent_pop']]
pop
```

Année	Zone	percent_pop
0	Afghanistan	36.27
1	Afrique du Sud	14.91
2	Albanie	-6.87
3	Algérie	18.21
4	Allemagne	-0.54
...	...	...
234	Îles Salomon	27.81
235	Îles Turques-et-Caïques	38.86
236	Îles Vierges américaines	-2.82
237	Îles Vierges britanniques	33.11
238	Îles Wallis-et-Futuna	-18.15

239 rows × 2 columns

Importation de la table dispo_alim_2003_2013 et dispo_anim

Élément	Zone	kcal_jr	proteine_jr
0	Afghanistan	4163.0	94.69
1	Afrique du Sud	6807.0	107.45
2	Albanie	5169.0	106.87
3	Algérie	6958.0	135.06
4	Allemagne	6744.0	86.10
...	...	...	...
173	Émirats arabes unis	7801.0	139.73
174	Équateur	5182.0	69.52
175	États-Unis d'Amérique	7746.0	92.62
176	Éthiopie	4525.0	106.57
177	Îles Salomon	7622.0	122.99

178 rows × 3 columns

Élément	Zone	kcal_jr_anim	proteine_jr_anim
0	Afghanistan	625.0	33.80
1	Afrique du Sud	1015.0	71.26
2	Albanie	2814.0	161.21
3	Algérie	1012.0	62.25
4	Allemagne	2648.0	156.87
...	...	...	...
173	Émirats arabes unis	1745.0	123.67
174	Équateur	1550.0	89.49
175	États-Unis d'Amérique	2729.0	184.24
176	Éthiopie	376.0	21.78
177	Îles Salomon	390.0	33.03

178 rows × 3 columns

Jointure à gauche des tables dispo_alim_2003_2013, pop et dispo_anim sur la zone

```
df = dispo_alim_2003_2013.merge(dispo_anim, on='Zone', how='left').merge(pop, on ='Zone', how='left')
df['ratio_proteine'] = (df['proteine_jr_anim']/df['proteine_jr']).round(2)
data = df[["Zone","kcal_jr","proteine_jr","percent_pop", "ratio_proteine"]]
data = data[data['Zone'] != "Chine"]
data
```

	Zone	kcal_jr	proteine_jr	percent_pop	ratio_proteine
0	Afghanistan	4163.0	94.69	36.27	0.36
1	Afrique du Sud	6807.0	107.45	14.91	0.66
2	Albanie	5169.0	106.87	-6.87	1.51
3	Algérie	6958.0	135.06	18.21	0.46
4	Allemagne	6744.0	86.10	-0.54	1.82
...	...	...	...	...	...
173	Émirats arabes unis	7801.0	139.73	147.79	0.89
174	Équateur	5182.0	69.52	17.49	1.29
175	États-Unis d'Amérique	7746.0	92.62	9.17	1.99
176	Éthiopie	4525.0	106.57	32.17	0.20
177	Îles Salomon	7622.0	122.99	27.81	0.27

177 rows × 5 columns

Identification des valeurs manquantes

- 5 valeurs manquantes pour percent_pop
- 4 valeurs manquantes pour ratio_proteine

```
data.isnull().sum()

Zone             0
kcal_jr          0
proteine_jr      0
percent_pop      5
ratio_proteine   4
dtype: int64
```

Imputation par la moyenne

```
ratio_mean = df["ratio_proteine"].mean()
percent_pop_mean = df["percent_pop"].mean()

data['ratio_proteine'].fillna(ratio_mean, inplace=True)
data['percent_pop'].fillna(percent_pop_mean, inplace=True)
```

Après l'imputation par la moyenne, il ne reste plus de valeurs manquantes

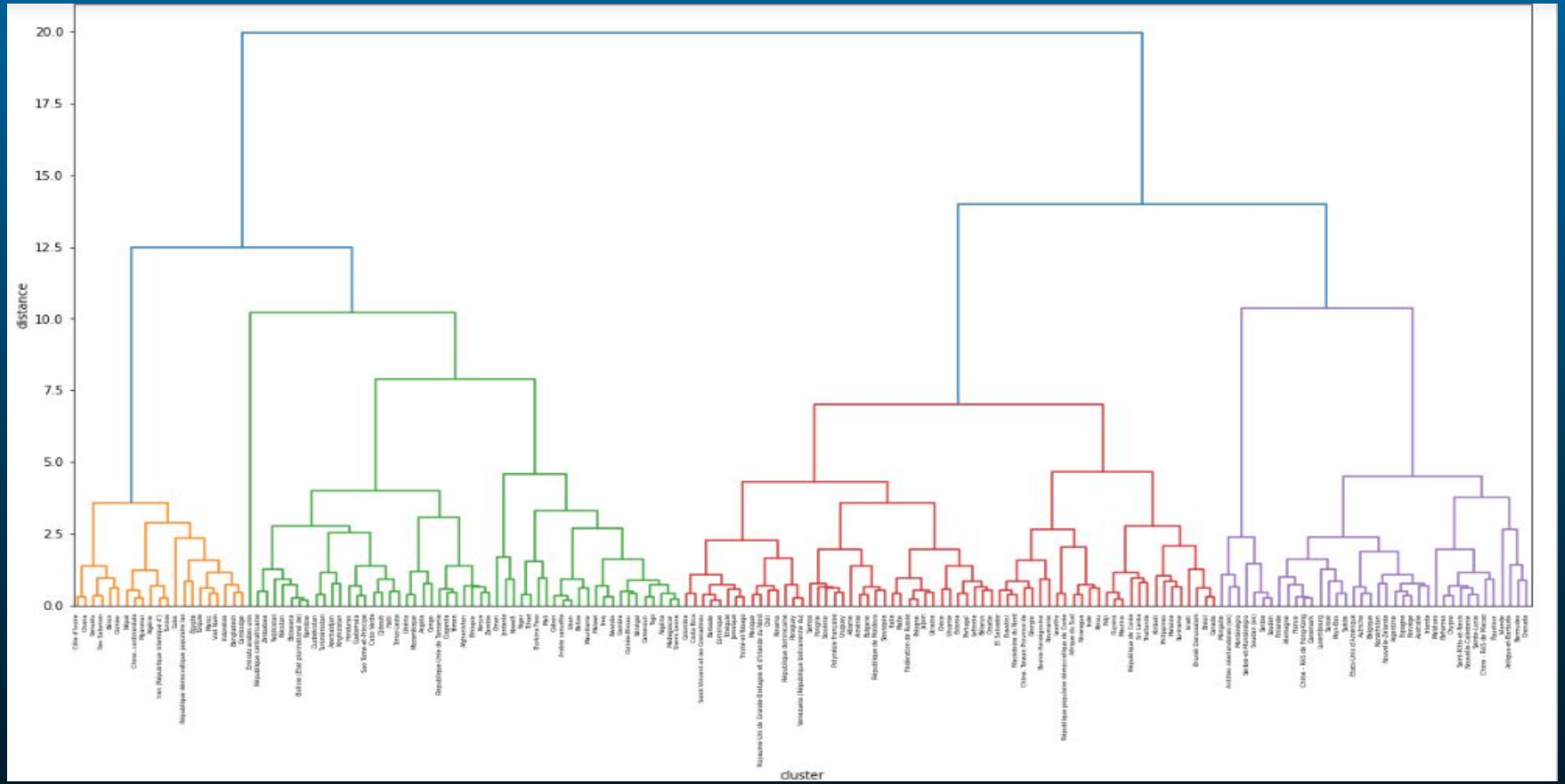
```
data.isnull().sum()
```

```
Zone              0
kcal_jr           0
proteine_jr       0
percent_pop       0
ratio_proteine    0
dtype: int64
```

2. Classification ascendante hierarchique

```
# conversion des nombres entiers en nombres décimaux (float) : necessaire pour le StandardScaler
X1 = X1.astype(np.float64)
names = data.index

# Centrage et Réduction
std_scale = preprocessing.StandardScaler().fit(X1)
X1_scaled = std_scale.transform(X1)

# Clustering hiérarchique
Z1 = linkage(X1_scaled, 'ward')

# Affichage du dendrogramme
plt.figure(figsize=(20,10))
plt.title('classification ascendante hiérarchique Dendrogramme')
plt.ylabel('distance')
plt.xlabel('cluster')
 #treshhold justif

sch.dendrogram(Z1, labels = names, color_threshold=12 )
# color_threshold permet de couper le dendrogramme en groupe à partir de la distance sélectionnée
#Affichage latéral gauche du dendrogramme avec l'option orientation
plt.show()
```

Dendrogramme découpé en 4 clusters



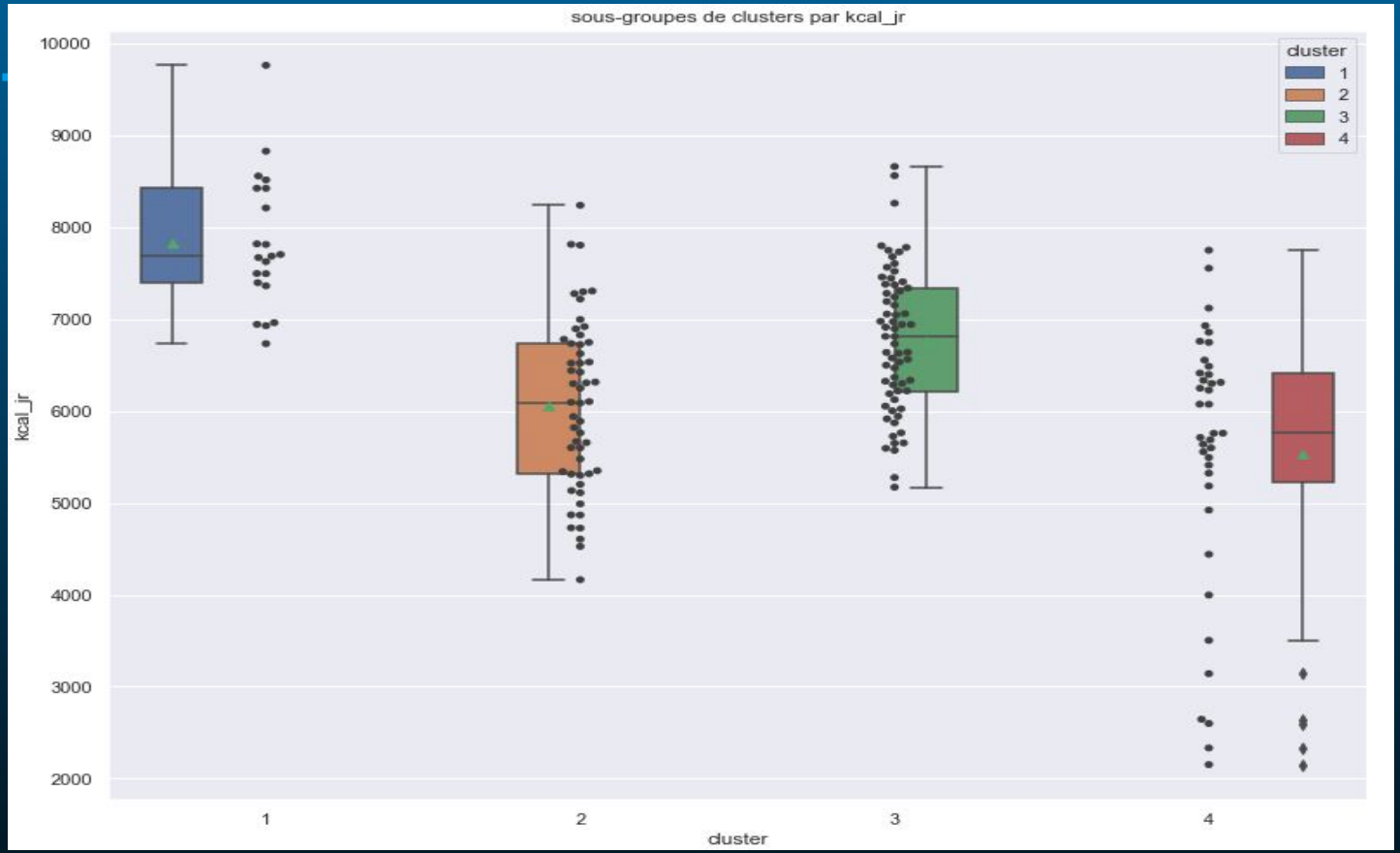
3. Caractérisation des groupes

Découpage du dendrogramme en 4 clusters

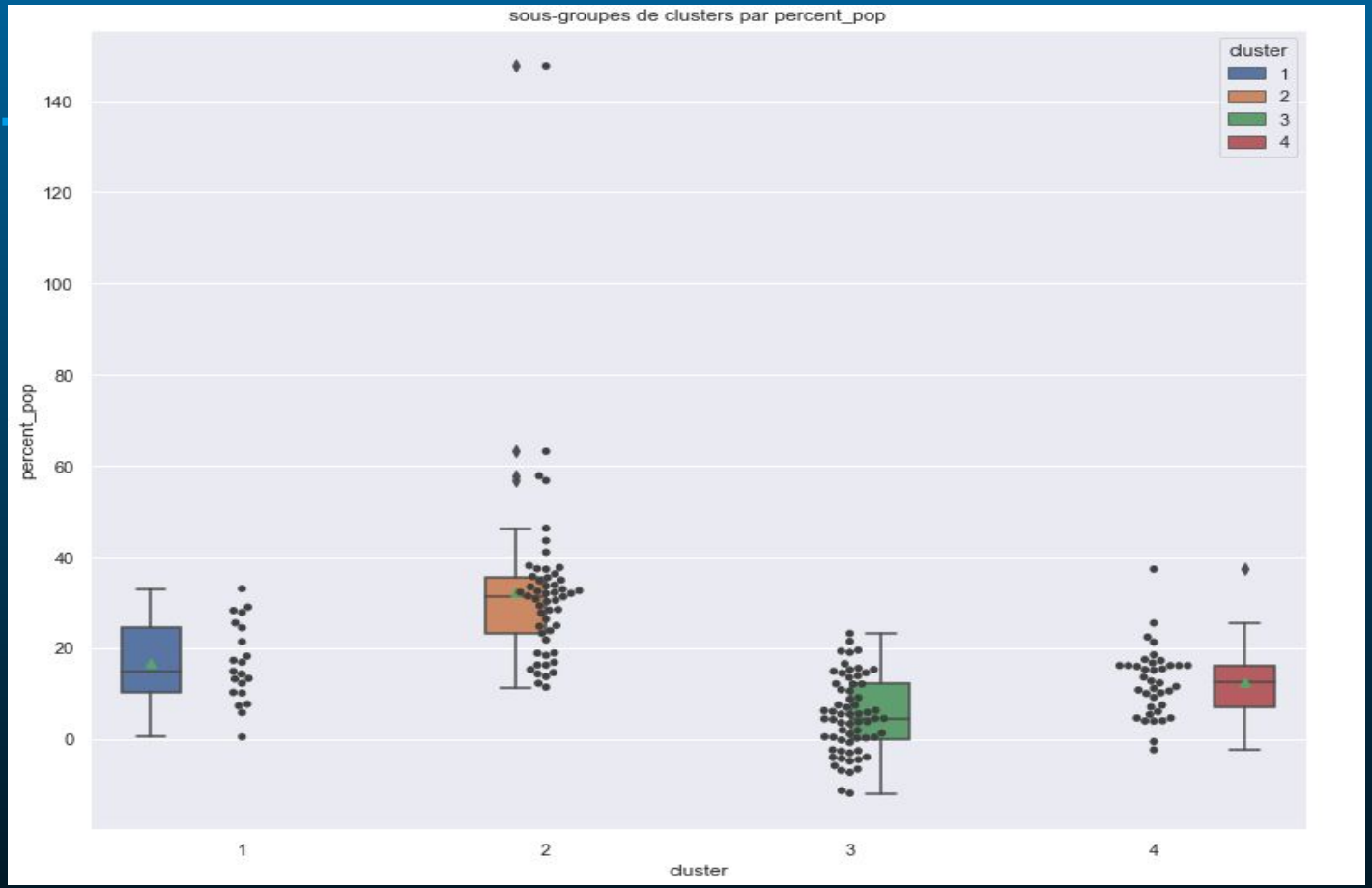
Légendes:

Cluster 1 Cluster 2 Cluster 3 Cluster 4

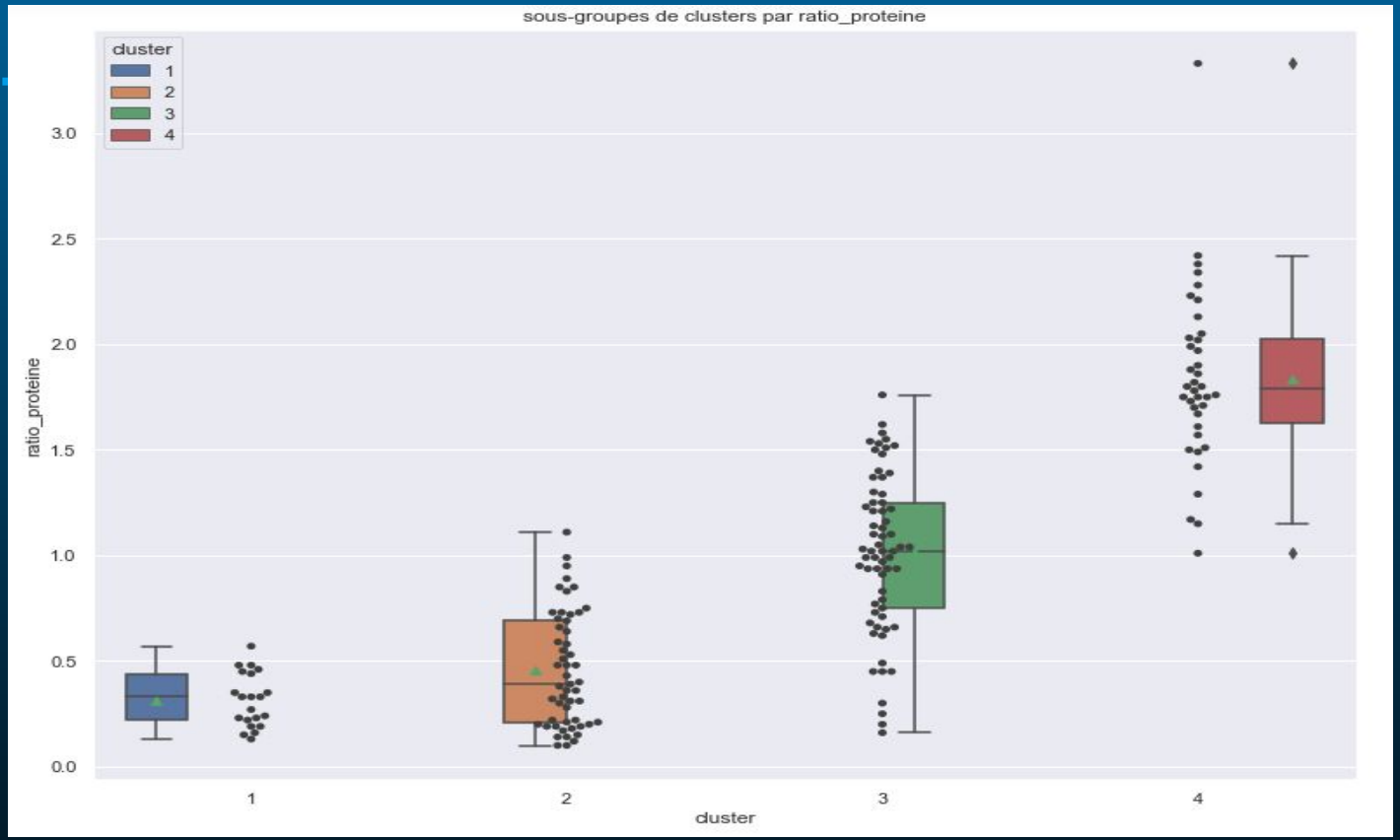
Répartition de kcal_jr selon les clusters



Répartition de proteine_jr selon les clusters



Répartition de ratio_proteine selon les clusters



Choix du cluster 4

Le cluster 1 a une meilleure quantité en kcal/jr et en protéine, cependant la proportion de ratio de protéine d'origine animale est la plus faible.
De ce fait, pour cibler le marché adéquat il nous faudra choisir le meilleur ratio en terme de protéine d'origine animale.
Pour notre cas, nous choisirons le cluster 4 du fait de son excellent ratio.

	cluster	kcal_jr	proteine_jr	percent_pop	ratio_proteine
0	1	7822.476190	145.539048	16.737143	0.313333
1	2	6054.905660	108.437547	32.278679	0.455094
2	3	6762.184615	101.771846	5.122308	1.012263
3	4	5523.710526	74.774211	12.515068	1.835789

Matrice de corrélation de clusters

On note une forte corrélation

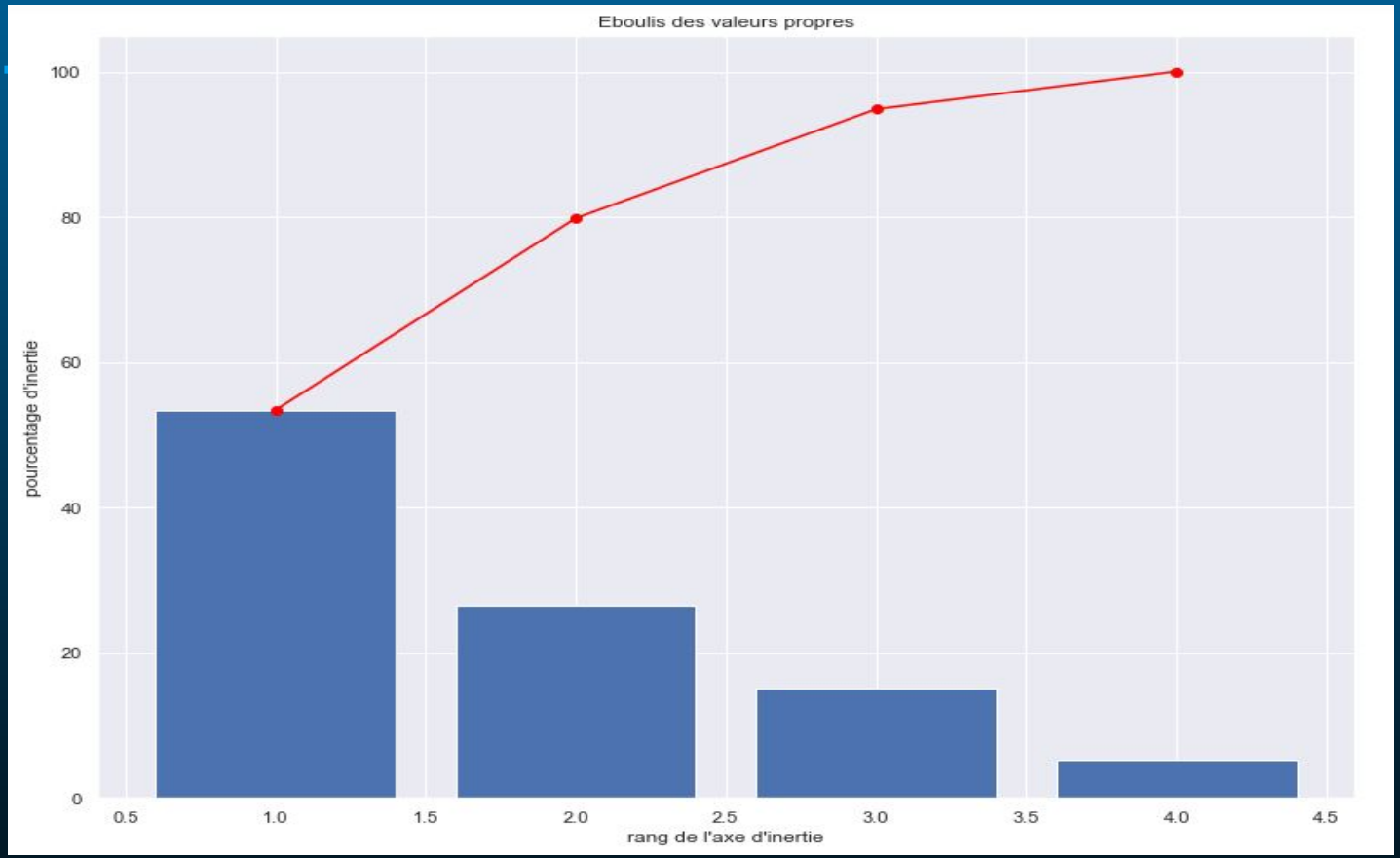
entre kcal_jr et proteine_jr

	kcal_jr	proteine_jr	percent_pop	ratio_proteine
kcal_jr	1.000000	0.645134	0.028898	-0.243950
proteine_jr	0.645134	1.000000	0.199277	-0.635716
percent_pop	0.028898	0.199277	1.000000	-0.344443
ratio_proteine	-0.243950	-0.635716	-0.344443	1.000000

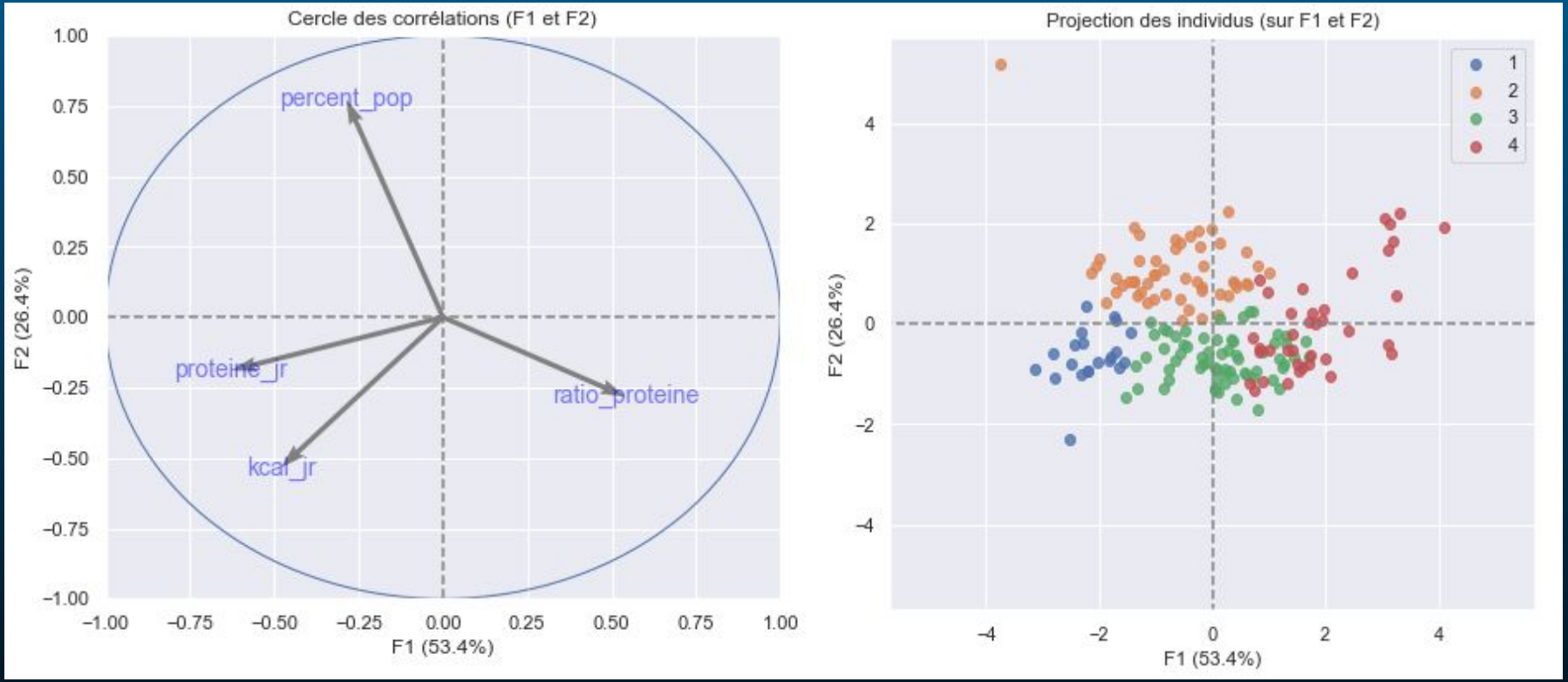
4. Analyse en composantes principales

```
# choix du nombre de composantes à calculer
n_comp = 4

ensemble = [data_acp.loc[cluster_id, "cluster"] for cluster_id in data_acp.index]

# préparation des données pour l'ACP
X3 = data_acp.values
features = data_acp.columns

# Centrage et Réduction
std_scale = preprocessing.StandardScaler().fit(X3)
X3_scaled = std_scale.transform(X3)

# Calcul des composantes principales
pca = decomposition.PCA(n_components=n_comp)
pca.fit(X3_scaled)

# Eboulis des valeurs propres
display_scree_plot(pca)

# Cercle des corrélations
pcs = pca.components_
display_circles(pcs, n_comp, pca, [(0,1),(2,3),(4,5)], labels = np.array(features))
display_circles(pcs, n_comp, pca, [(0,1)], labels = np.array(centroids_cluster.cluster))
```

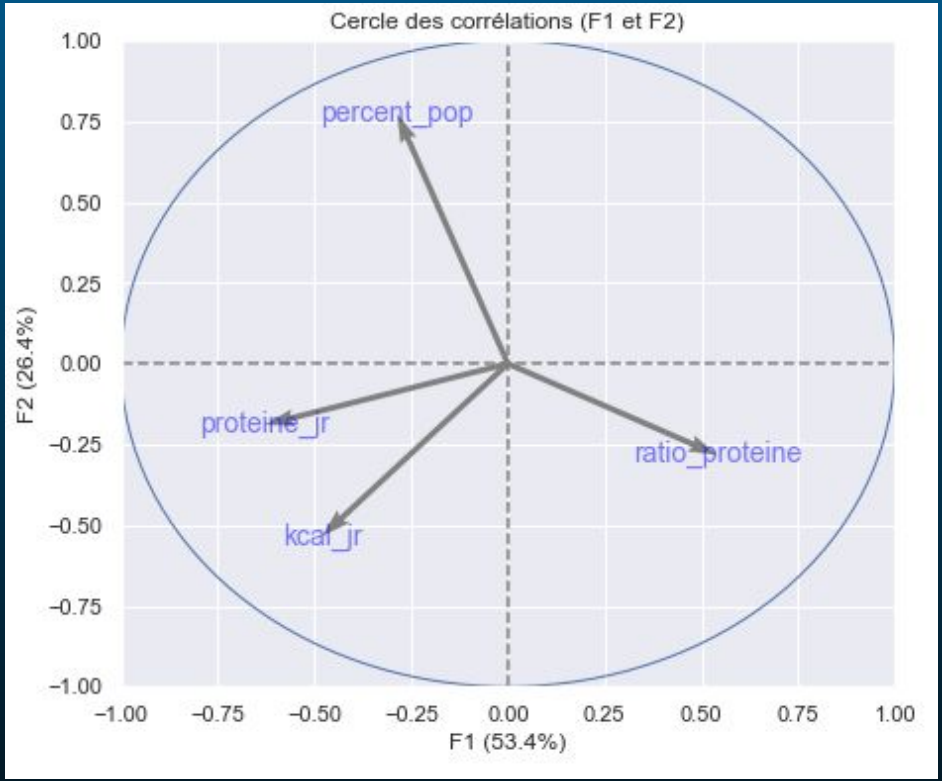
Les axes d'inertie 1 et 2 représentent 80% du pourcentage d'inertie



Cercle de corrélations et projection des individus sur les composantes F1 et F2



Cercle de corrélations et centroïdes des groupes et leurs coordonnées dans F1 et F2



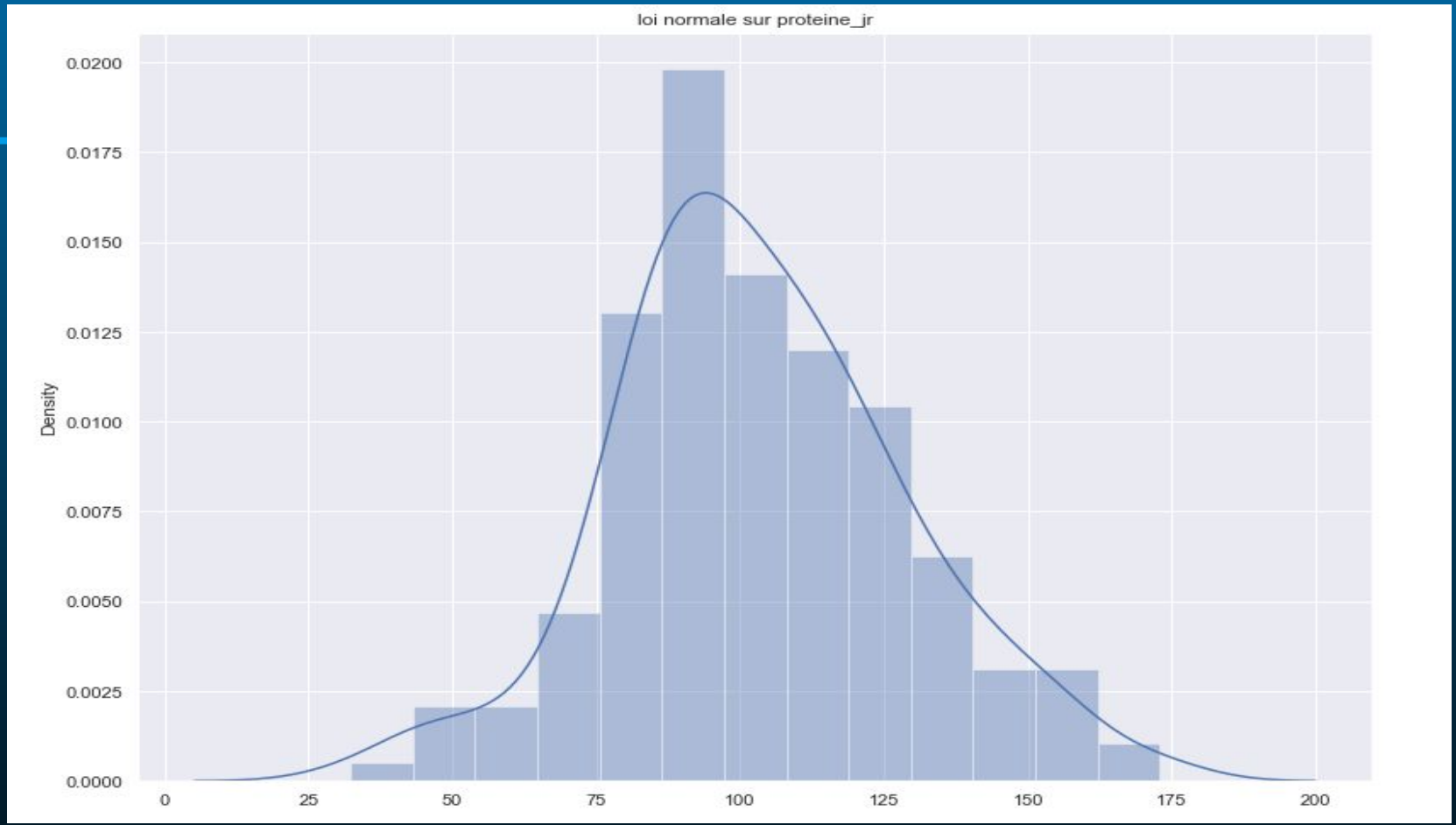
cluster	F1	F2
1	-2.601148	-0.655530
2	-0.670506	1.757512
3	0.546275	-1.136527
4	2.725379	0.034545

5. Tests statistiques

- Test d’adéquation par rapport à la loi normale

- Test de comparaison de 2 populations (dans le cas gaussien)

Application de la courbe de GAUSS sur l'histogramme de proteine_jr



Test de normalité sur le dataframe clusters

H0: La distribution suit la loi normale

H1: La population n'est pas normalement distribuée

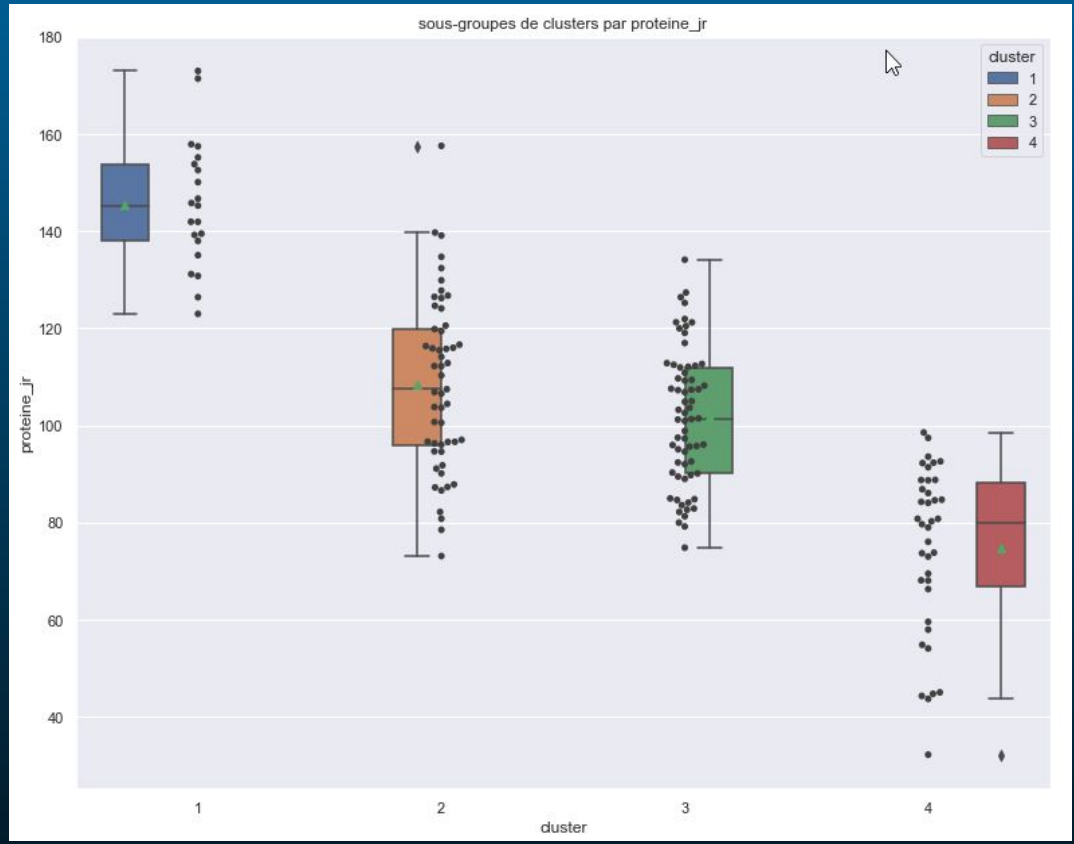
La loi normale est vérifiée pour un test de Shapiro-Wilk avec alpha = 0.05 pour la variable proteine_jr car pvalue = 0.16

On accepte l'hypothèse H0.

	W	pval	normal
kcal_jr	0.962	0.00	False
proteine_jr	0.988	0.16	True
percent_pop	0.814	0.00	False
ratio_proteine	0.936	0.00	False
cluster	0.876	0.00	False

Test de comparaison sur la variable gaussienne

- On fait d'abord le test de student sur les 2 clusters les plus éloignés (cluster 2 et cluster 4)
- Ensuite, On effectue d'abord le test de student sur les 2 clusters les plus proches (cluster 2 et cluster 3)



Test de student sur les 2 clusters les plus éloignés

Test d'égalité des variances

H0: égalité des variances

H1: les variances sont différentes

```
# On teste tout d'abord l'égalité des variances
from scipy.stats import bartlett

scipy.stats.bartlett(comp2,comp4)
```

```
BartlettResult(statistic=0.037317210109214675, pvalue=0.846820641108601)
```

La pvalue = 0.84, on accepte l'égalité des variances pour alpha= 0.05

Test de student sur les 2 clusters les plus éloignés

Test d'égalité des moyennes

H0: égalité des moyennes

H1: les moyennes sont inégales

La pvalue < 0.05, on rejette l'hypothèse H0.

```
# On teste ensuite l'égalité des moyennes:

scipy.stats.ttest_ind(comp1,comp4, equal_var=False)
```

```
Ttest_indResult(statistic=17.566338926043088, pvalue=2.9836219470194515e-23)
```

Test de student sur les 2 clusters les plus proches

Test d'égalité des variances

H0: égalité des variances

H1: les variances sont différentes

La pvalue = 0.063, on accepte l'égalité des variances pour alpha = 0.05 (H0)

```
# On teste tout d'abord l'égalité des variances
from scipy.stats import bartlett

scipy.stats.bartlett(comp2,comp3)
```

```
BartlettResult(statistic=3.4485663071905357, pvalue=0.06330666057828578)
```

Test de student sur les 2 clusters les plus proches

Test d'égalité des moyennes

H0: égalité des moyennes

H1: les moyennes sont inégales

La pvalue < 0.05, on rejette l'hypothèse H0.

```
# On teste ensuite l'égalité des moyennes des clusters les plus proches :

scipy.stats.ttest_ind(comp2,comp3, equal_var=True)
```

```
Ttest_indResult(statistic=2.2833969154117844, pvalue=0.024227567606660245)
```

Les moyennes des clusters sont bien différentes.

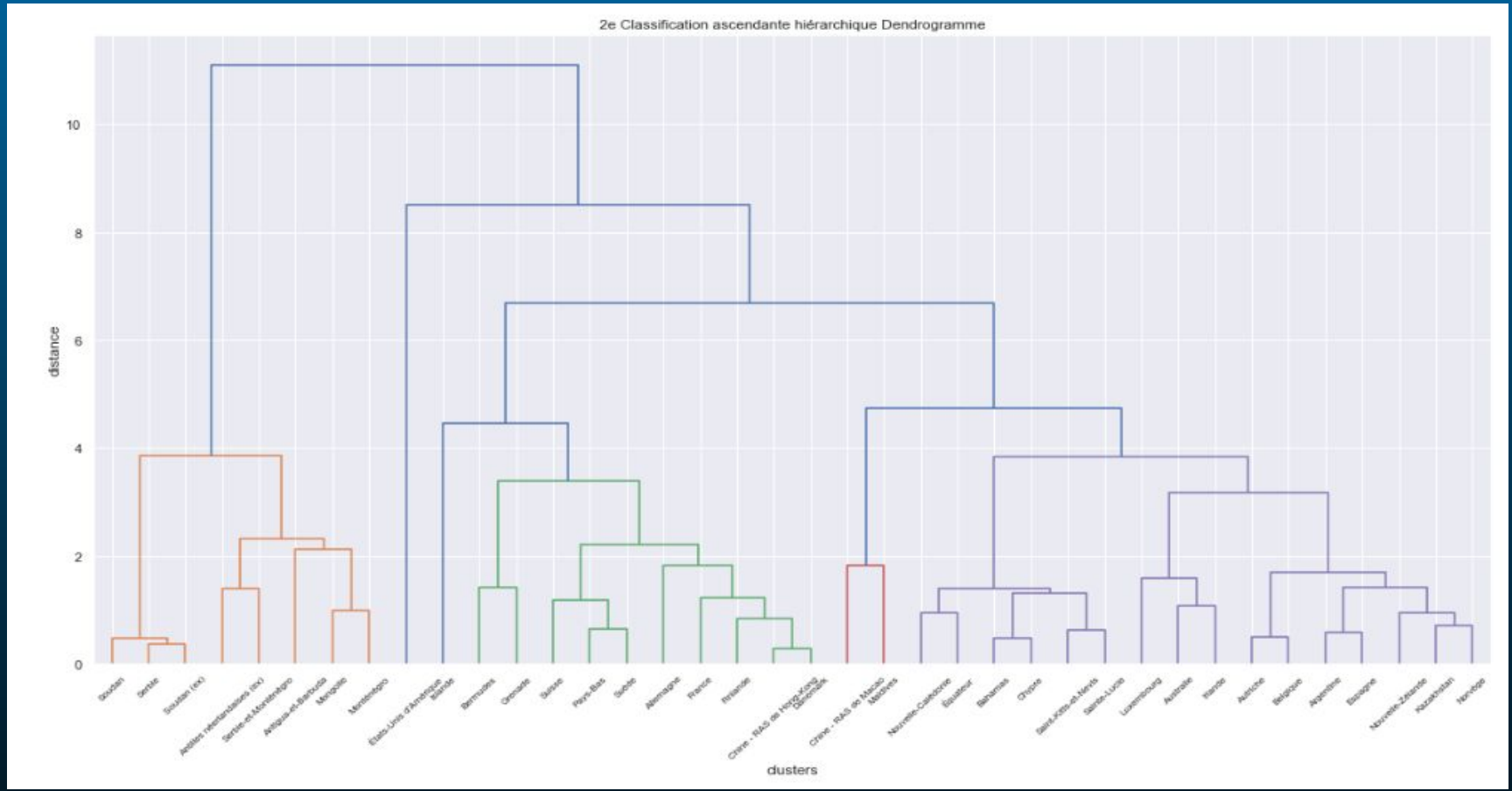
6. Choix du sous-groupe de cluster 4

- Ajout du PIB/hbt
- 2e clustering
- dendrogramme
- Sélection des pays les plus importants en terme de de ratio_proteine et de pib

Ajout du PIB/hbt dans le dataframe de clusters2

	Zone	kcal_jr	proteine_jr	percent_pop	ratio_proteine	cluster	pib
0	Allemagne	6744.0	86.10	-0.54	1.82	4	3.732743e+12
1	Antigua-et-Barbuda	4439.0	54.11	15.40	2.38	4	1.181448e+09
2	Antilles néerlandaises (ex)	3503.0	44.34	22.42	1.75	4	3.660820e+11
3	Argentine	6483.0	80.80	10.79	1.86	4	6.133160e+11
4	Australie	6309.0	84.61	18.50	2.05	4	1.543216e+12

Dendrogramme de cluster_pib avec 6 sous-groupes



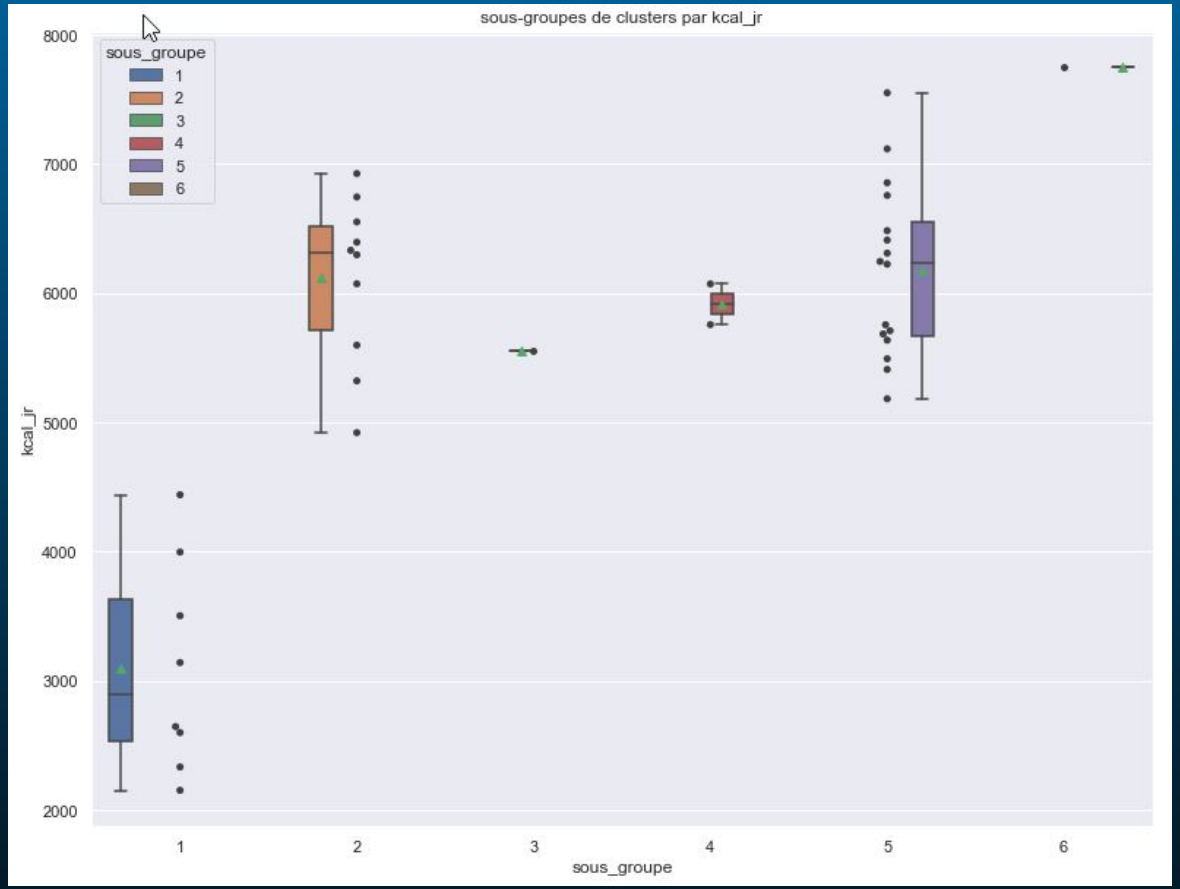
Répartition des variables par sous-groupe

Après le clustering de cluster_pib, nous obtenons 6 sous-groupes.

Nous allons représenter par la boite à moustache la dispersion des variables selon les clusters.

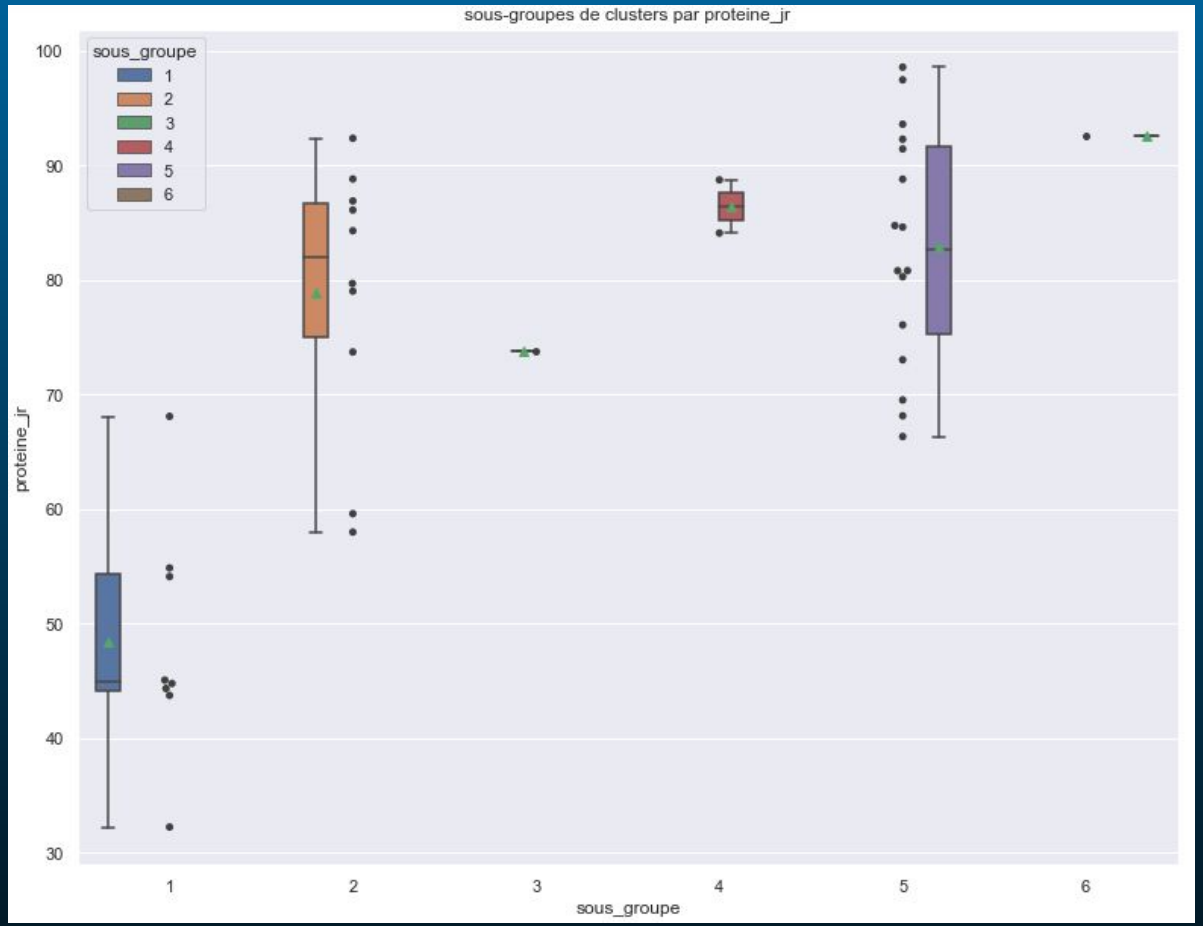
Répartition de kcal_jr selon les sous-groupes

Au moins 50% des populations du sous-groupe 2 consomment 6500 kcal/jr.



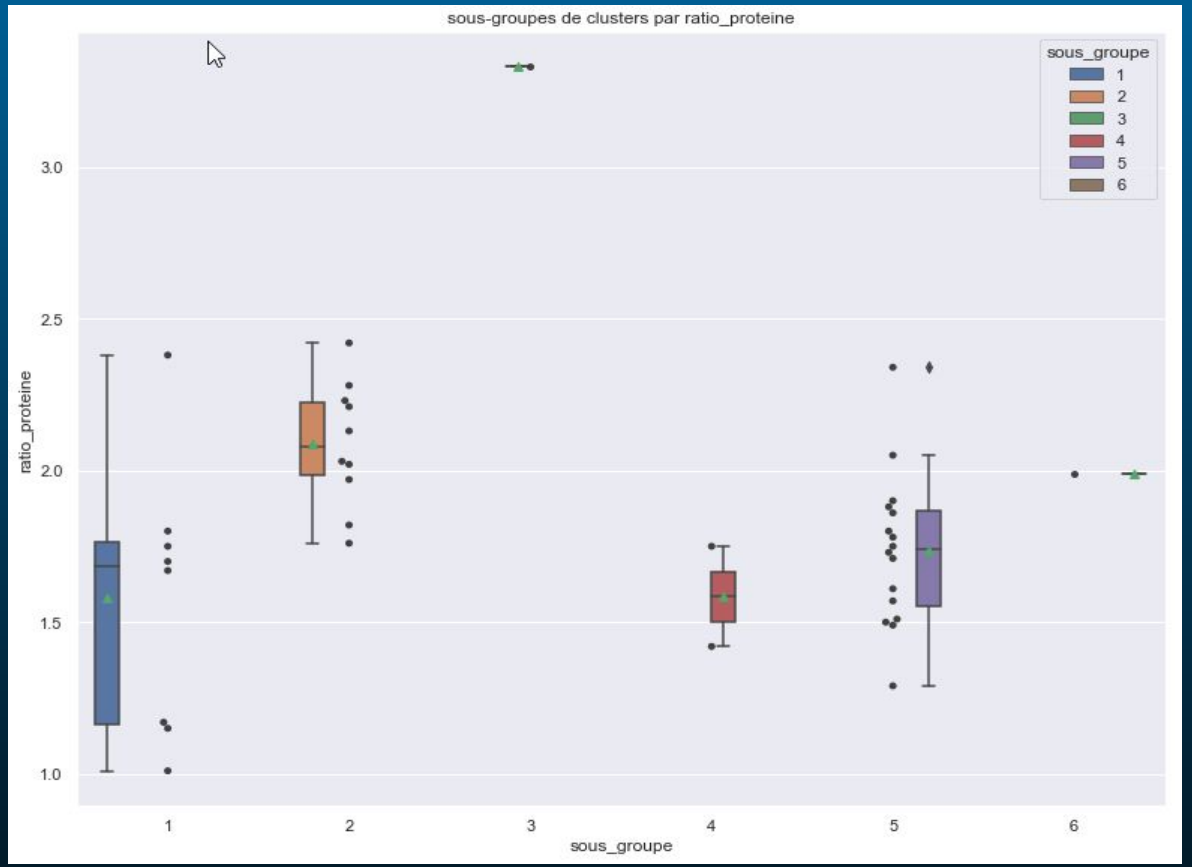
Répartition de proteine_jr selon les sous-groupes

Au moins 50% des populations du sous-groupe 4 consomment 87 g/jr.



Répartition de ratio_proteine selon les sous-groupes

Au moins 50% des populations de ce sous-groupe ont un ratio_proteine égale à 2.1.



Répartition du pib selon les sous-groupes

Le pib/hbt d'au moins 50% des populations du sous-groupe 2 est 0.05*e^(13)
soit 22 120 $/hbt/an

Choix du sous-groupe

Au finish, nous avons choisi les sous-groupes 2 et 6 car possédant les meilleures proportions en terme de ratio et de pib .

	sous_groupe	kcal_jr	proteine_jr	percent_pop	ratio_proteine	pib
0	1	3101.000	48.405000	16.919075	1.578750	1.536002e+11
1	2	6115.200	78.846000	4.424000	2.087000	9.618210e+11
2	3	5554.000	73.830000	13.630000	3.330000	1.603352e+10
3	4	5913.500	86.410000	31.385000	1.585000	2.742369e+10
4	5	6175.875	82.903125	13.150625	1.735625	3.659780e+11
5	6	7746.000	92.620000	9.170000	1.990000	1.678485e+13

Les pays ciblés sont

- Allemagne
- Bermudes
- Chine - RAS de Hong-Kong
- Danemark
- Finlande
- Grenade
- Pays-Bas
- Suisse
- Suède
- États-Unis d'Amérique

Conclusion

L’étude de marché nous a permis de cibler les clusters les plus adéquats en termes de nutrition et de pib avec une démarche scientifique et approfondie en utilisant l’ACP, les tests et les représentations graphiques

Sources

Téléchargement des données de la FAO
http://www.fao.org/faostat/fr/?#data

Cours OpenClassrooms
ACP et Tests

Cours sur l'ACP
http://eric.univ-lyon2.fr/~ricco/tanagra/fichiers/fr_Tanagra_ACP_Python.pdf

Cours ACP Agrocampus
https://www.youtube.com/watch?v=uV5hmpzmWsU

Merci de votre attention

Présenté par Yaya CISSE