P11_Poulet_Analyse

August 30, 2025

ÉTUDE DE MARCHÉ SUR LE POULET - ANALYSE DES DONNÉES

1 OBJECTIFS DE CE NOTEBOOK

- Présenter les tendances mondiales du marché de la viande de poulet sur la période 2010-2023 ;
- Réaliser une Analyse en Composantes Principales pour déterminer ce qui "distingue" le mieux les pays cibles sur les données moyennées (de façon arithmétique) au cours du temps ;
- Benchmarker les pays cibles selon les variables d'intérêt identifiés grâce à un algorithme de clustering.

Partie 1 - Importation des bibliothèques Python et chargement des fichiers

[1]:
```
#Importation de la librairie Pandas
import pandas as pd
import numpy as np
```

[2]:
```
#Importation de librairies spécifiques
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import plotly.express as px
import seaborn as sb
import statsmodels.api as sm

from scipy.spatial.distance import squareform as sqf
from scipy import stats as st
from matplotlib import colormaps as cm
```

[3]:
```
# Fonctions pour ACP
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

[4]:
```
# Fonctions pour clustering
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.cluster import KMeans
```

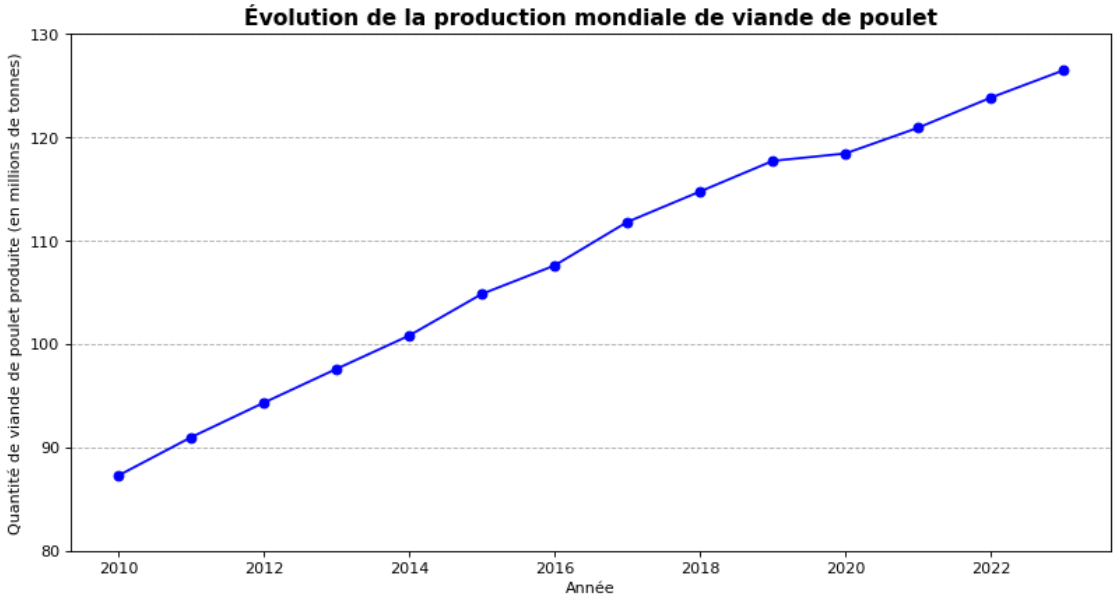
[5]:
```
# Séries temporelles brutes avec valeurs manquantes
df_world = pd.read_csv("world_data.csv", sep=',')
# Séries temporelles détaillées (feature engineering) et sans valeur manquante
df_dtlt = pd.read_csv("time_data.csv", sep=',')
# Données moyennées complètes sur les pays cibles
df_work = pd.read_csv("work_data.csv", sep=',')
```

Partie 2 - Quelques chiffres sur le marché de la viande de poulet

2.1 - Évolution des volumes mondiaux produits, exportés, importés

[6]:
```
# Évolution de la production mondiale de viande de poulet
df_plot = df_world.groupby("Année")["prod"].sum().reset_index().
 ↪sort_values("Année", ascending=True)
display(df_plot)
fig = plt.figure(figsize=(12,6), dpi=80)
# Graphique X-Y avec points reliés
plt.plot(df_plot['Année'],df_plot['prod']/1e6,'o-',color='blue')
plt.xlabel("Année", fontsize=10)
plt.ylabel("Quantité de viande de poulet produite (en millions de tonnes)",␣
 ↪fontsize=10)
#Resserrage de ma fenêtre pour mettre en valeur l'évolution
plt.ylim([80,130])
#plt.yticks(np.linspace(520,550,num=7))
#Traits de repère
plt.grid(visible=True, axis='y', linestyle='--')
plt.title("Évolution de la production mondiale de viande de␣
 ↪poulet",fontsize=14, fontweight='heavy')
plt.savefig("Production_poulet.png")
plt.show()
```

	Année	prod
0	2010	8.725923e+07
1	2011	9.096430e+07
2	2012	9.430685e+07
3	2013	9.757971e+07
4	2014	1.008051e+08
5	2015	1.048329e+08
6	2016	1.075863e+08
7	2017	1.117978e+08
8	2018	1.147447e+08
9	2019	1.177071e+08
10	2020	1.184262e+08
11	2021	1.209177e+08
12	2022	1.238304e+08
13	2023	1.264769e+08

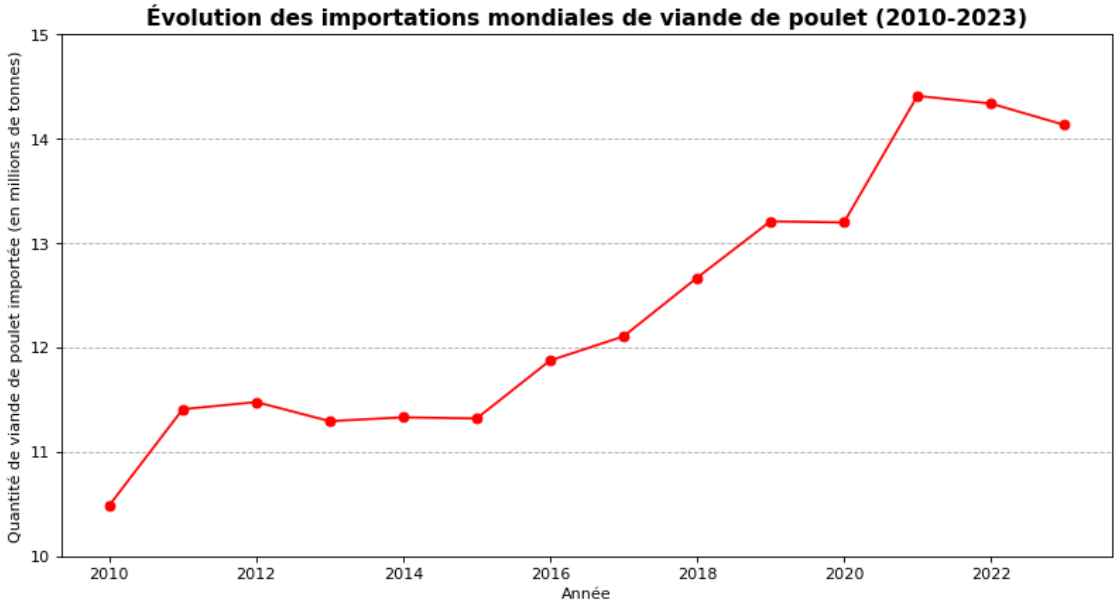


Croissance régulière avec environ +3 millions de tonnes de viande de poulet supplémentaires produits chaque année, sauf en 2020 où il y a eu creux, probablement dû à la pandémie de CoViD-19. Augmentation de la production de +25% en 10 ans, soit environ 2,2 % de croissance chaque année.

[7]:
```
# Évolution des volumes d'importation mondiaux de viande de poulet
df_plot = df_world.groupby("Année")["imp"].sum().reset_index().
 ↪sort_values("Année", ascending=True)
display(df_plot)
fig = plt.figure(figsize=(12,6), dpi=80)
# Graphique X-Y avec points reliés
plt.plot(df_plot['Année'],df_plot['imp']/1e6,'o-',color='red')
plt.xlabel("Année", fontsize=10)
plt.ylabel("Quantité de viande de poulet importée (en millions de tonnes)",␣
 ↪fontsize=10)
#Resserrage de ma fenêtre pour mettre en valeur l'évolution
plt.ylim([10,15])
#plt.yticks(np.linspace(520,550,num=7))
#Traits de repère
plt.grid(visible=True, axis='y', linestyle='--')
plt.title("Évolution des importations mondiales de viande de poulet␣
 ↪(2010-2023)",fontsize=14, fontweight='heavy')
plt.savefig("Importations_poulet.png")
plt.show()
```

	Année	imp
0	2010	10486428.00
1	2011	11407722.00
2	2012	11476420.10

3	2013	11293584.00
4	2014	11330122.89
5	2015	11318925.81
6	2016	11874453.38
7	2017	12105680.85
8	2018	12666462.12
9	2019	13208066.35
10	2020	13196634.56
11	2021	14408898.20
12	2022	14336185.63
13	2023	14131333.32



Baisse en 2013 puis stagnation jusqu'en 2015, augmentation de 2016 à 2019, stagnation en 2020, rebond en 2021, restagnation en 2022.

[8]:
```
df_plot = df_world.groupby("Année")[["imp","prod"]].sum().reset_index().
  ↪sort_values("Année", ascending=True)
df_plot["%prod"] = round(100*df_plot["imp"]/df_plot["prod"],2)
display(df_plot)
```

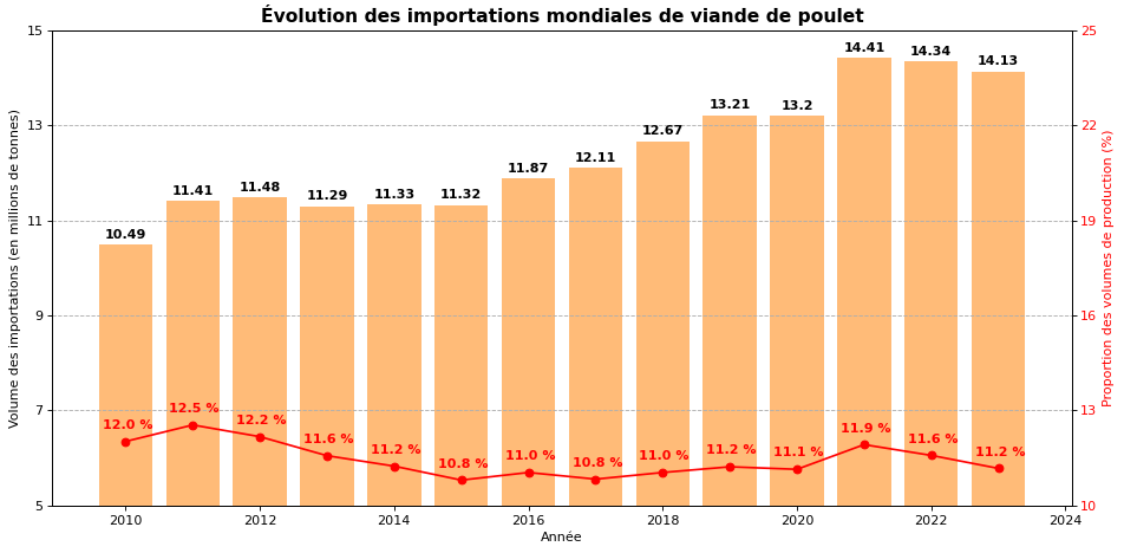
	Année	imp	prod	%prod
0	2010	10486428.00	8.725923e+07	12.02
1	2011	11407722.00	9.096430e+07	12.54
2	2012	11476420.10	9.430685e+07	12.17
3	2013	11293584.00	9.757971e+07	11.57
4	2014	11330122.89	1.008051e+08	11.24
5	2015	11318925.81	1.048329e+08	10.80
6	2016	11874453.38	1.075863e+08	11.04

```
7    2017  12105680.85  1.117978e+08  10.83
8    2018  12666462.12  1.147447e+08  11.04
9    2019  13208066.35  1.177071e+08  11.22
10   2020  13196634.56  1.184262e+08  11.14
11   2021  14408898.20  1.209177e+08  11.92
12   2022  14336185.63  1.238304e+08  11.58
13   2023  14131333.32  1.264769e+08  11.17
```

[9]:
```
df_plot = df_world.groupby("Année")[["imp","prod","exp","Totale","Disponibilité␣
 ↪- Viande de Volailles",\
                                    "Disponibilité - Viandes"]].sum().
 ↪reset_index().sort_values("Année", ascending=True)
df_plot["imp%prod"] = round(100*df_plot["imp"]/df_plot["prod"],2)
display(df_plot)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['imp']/1e6, color=cm['tab20'](3))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['imp'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Volume des importations (en millions de tonnes)", fontsize=10)
ax1.set_ylim([5,15])
ax1.set_yticks(np.linspace(5,15,6))
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['imp%prod'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['imp%prod'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['imp%prod'][i]),
                 xytext=(df_plot['Année'][i]-0.35,df_plot['imp%prod'][i]+0.4),
                 textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion des volumes de production (%)", color='red')
ax2.set_ylim([10,25])
ax2.set_yticks(np.linspace(10,25,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution des importations mondiales de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped
```

```
plt.savefig("Imp-prod_combined.png")
plt.show()
```

```
    Année          imp          prod          exp        Totale  \
0    2010  10486428.00  8.725923e+07  11654015.00  7.019945e+09
1    2011  11407722.00  9.096430e+07  12436540.40  7.109087e+09
2    2012  11476420.10  9.430685e+07  12618431.00  7.199350e+09
3    2013  11293584.00  9.757971e+07  12739594.00  7.289930e+09
4    2014  11330122.89  1.008051e+08  13178789.90  7.379761e+09
5    2015  11318925.81  1.048329e+08  12820538.66  7.468666e+09
6    2016  11874453.38  1.075863e+08  13328140.41  7.556747e+09
7    2017  12105680.85  1.117978e+08  13894228.96  7.643804e+09
8    2018  12666462.12  1.147447e+08  14339086.72  7.728090e+09
9    2019  13208066.35  1.177071e+08  14648443.57  7.809496e+09
10   2020  13196634.56  1.184262e+08  14553485.89  7.885209e+09
11   2021  14408898.20  1.209177e+08  14948942.29  7.952668e+09
12   2022  14336185.63  1.238304e+08  15405711.97  8.019652e+09
13   2023  14131333.32  1.264769e+08  15271128.61  8.089999e+09

    Disponibilité - Viande de Volailles  Disponibilité - Viandes  imp%prod
0                               97733000.0              289506000.0     12.02
1                              101744000.0              291966000.0     12.54
2                              103905000.0              297114000.0     12.17
3                              107101000.0              303163000.0     11.57
4                              108649000.0              308754000.0     11.24
5                              112033000.0              315036000.0     10.80
6                              115551000.0              320406000.0     11.04
7                              117731000.0              324572000.0     10.83
8                              122571000.0              332087000.0     11.04
9                              128941000.0              337082000.0     11.22
10                             133369000.0              332891000.0     11.14
11                             133570000.0              344933000.0     11.92
12                             136975000.0              356170000.0     11.58
13                                     0.0                      0.0     11.17
```



[10]:
```
df_plot["dispo"] = df_plot["prod"] + df_plot["imp"] - df_plot["exp"]

df_plot["imp%dispo"] = round(100*df_plot["imp"]/df_plot["dispo"],2)
display(df_plot)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['imp']/1e6, color=cm['tab20'](3))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['imp'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Volume des importations (en millions de tonnes)", fontsize=10)
ax1.set_ylim([5,15])
ax1.set_yticks(np.linspace(5,15,6))
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['imp%dispo'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['imp%dispo'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['imp%dispo'][i]),
                 xytext=(df_plot['Année'][i]-0.35,df_plot['imp%dispo'][i]+0.4),
                 textcoords='data', color='red', fontweight='heavy')
```

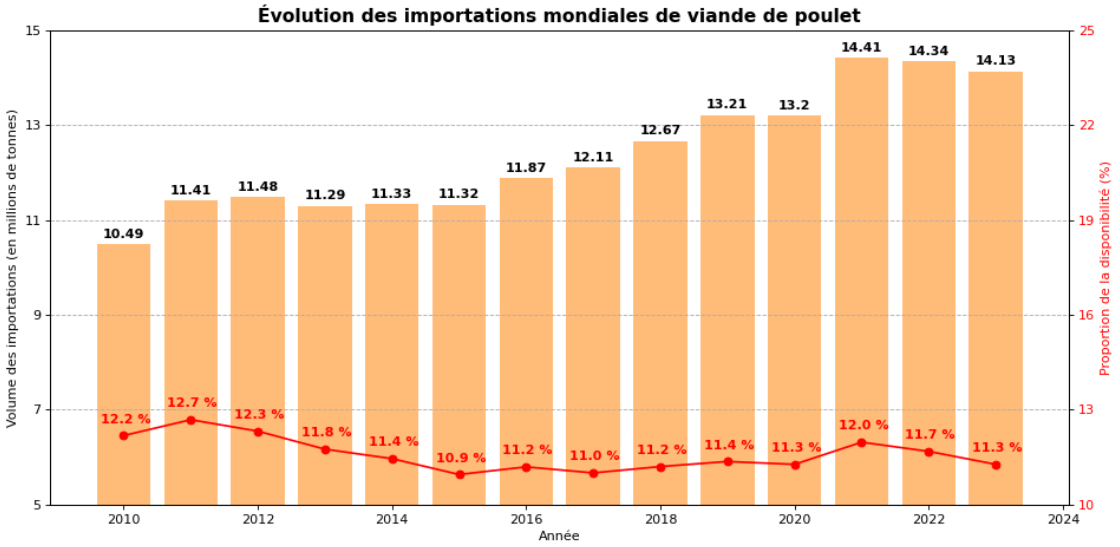
```
ax2.set_ylabel("Proportion de la disponibilité (%)", color='red')
ax2.set_ylim([10,25])
ax2.set_yticks(np.linspace(10,25,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution des importations mondiales de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("Imp-dispo_combined.png")
plt.show()
```

	Année	imp	prod	exp	Totale
0	2010	10486428.00	8.725923e+07	11654015.00	7.019945e+09
1	2011	11407722.00	9.096430e+07	12436540.40	7.109087e+09
2	2012	11476420.10	9.430685e+07	12618431.00	7.199350e+09
3	2013	11293584.00	9.757971e+07	12739594.00	7.289930e+09
4	2014	11330122.89	1.008051e+08	13178789.90	7.379761e+09
5	2015	11318925.81	1.048329e+08	12820538.66	7.468666e+09
6	2016	11874453.38	1.075863e+08	13328140.41	7.556747e+09
7	2017	12105680.85	1.117978e+08	13894228.96	7.643804e+09
8	2018	12666462.12	1.147447e+08	14339086.72	7.728090e+09
9	2019	13208066.35	1.177071e+08	14648443.57	7.809496e+09
10	2020	13196634.56	1.184262e+08	14553485.89	7.885209e+09
11	2021	14408898.20	1.209177e+08	14948942.29	7.952668e+09
12	2022	14336185.63	1.238304e+08	15405711.97	8.019652e+09
13	2023	14131333.32	1.264769e+08	15271128.61	8.089999e+09

	Disponibilité - Viande de Volailles	Disponibilité - Viandes	imp%prod
0	97733000.0	289506000.0	12.02
1	101744000.0	291966000.0	12.54
2	103905000.0	297114000.0	12.17
3	107101000.0	303163000.0	11.57
4	108649000.0	308754000.0	11.24
5	112033000.0	315036000.0	10.80
6	115551000.0	320406000.0	11.04
7	117731000.0	324572000.0	10.83
8	122571000.0	332087000.0	11.04
9	128941000.0	337082000.0	11.22
10	133369000.0	332891000.0	11.14
11	133570000.0	344933000.0	11.92
12	136975000.0	356170000.0	11.58
13	0.0	0.0	11.17

	dispo	imp%dispo
0	8.609164e+07	12.18
1	8.993548e+07	12.68
2	9.316483e+07	12.32

3	9.613370e+07	11.75
4	9.895642e+07	11.45
5	1.033313e+08	10.95
6	1.061326e+08	11.19
7	1.100093e+08	11.00
8	1.130721e+08	11.20
9	1.162668e+08	11.36
10	1.170694e+08	11.27
11	1.203776e+08	11.97
12	1.227609e+08	11.68
13	1.253371e+08	11.27



[11]:
```
df_plot["exp%prod"] = round(100*df_plot["exp"]/df_plot["prod"],2)
display(df_plot)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['exp']/1e6, color=cm['tab20'](9))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['exp'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Volume des exportations (en millions de tonnes)", fontsize=10)
ax1.set_ylim([8,18])
ax1.set_yticks(np.linspace(8,18,6))
plt.grid(visible=True, axis='y', linestyle='--')
```

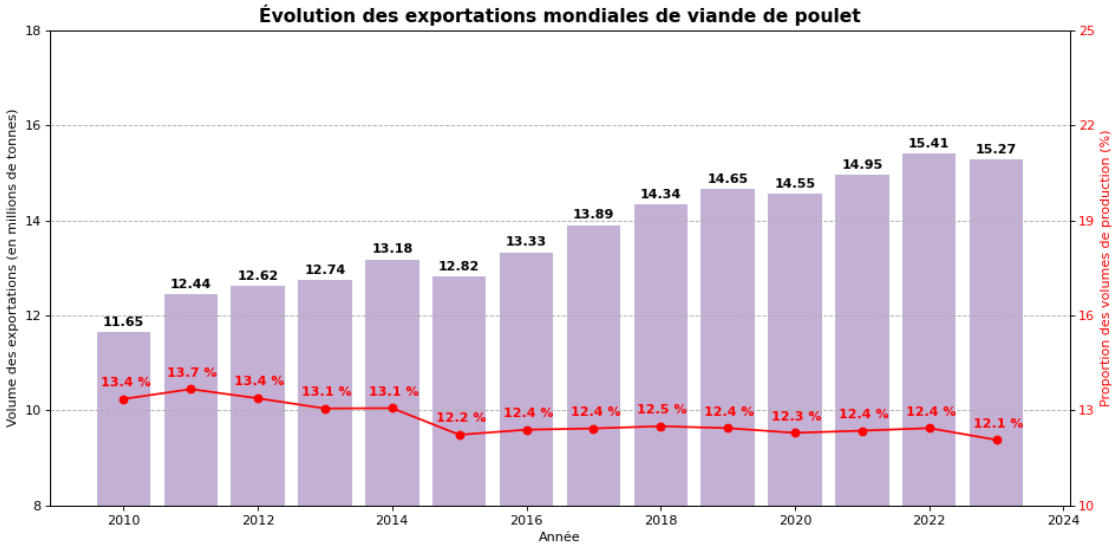
```
ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['exp%prod'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['exp%prod'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['exp%prod'][i]),
                 xytext=(df_plot['Année'][i]-0.35,df_plot['exp%prod'][i]+0.4),
                 textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion des volumes de production (%)", color='red')
ax2.set_ylim([10,25])
ax2.set_yticks(np.linspace(10,25,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution des exportations mondiales de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("exp-prod_combined.png")
plt.show()
```

	Année	imp	prod	exp	Totale	\
0	2010	10486428.00	8.725923e+07	11654015.00	7.019945e+09	
1	2011	11407722.00	9.096430e+07	12436540.40	7.109087e+09	
2	2012	11476420.10	9.430685e+07	12618431.00	7.199350e+09	
3	2013	11293584.00	9.757971e+07	12739594.00	7.289930e+09	
4	2014	11330122.89	1.008051e+08	13178789.90	7.379761e+09	
5	2015	11318925.81	1.048329e+08	12820538.66	7.468666e+09	
6	2016	11874453.38	1.075863e+08	13328140.41	7.556747e+09	
7	2017	12105680.85	1.117978e+08	13894228.96	7.643804e+09	
8	2018	12666462.12	1.147447e+08	14339086.72	7.728090e+09	
9	2019	13208066.35	1.177071e+08	14648443.57	7.809496e+09	
10	2020	13196634.56	1.184262e+08	14553485.89	7.885209e+09	
11	2021	14408898.20	1.209177e+08	14948942.29	7.952668e+09	
12	2022	14336185.63	1.238304e+08	15405711.97	8.019652e+09	
13	2023	14131333.32	1.264769e+08	15271128.61	8.089999e+09	

	Disponibilité - Viande de Volailles	Disponibilité - Viandes	imp%prod	\
0	97733000.0	289506000.0	12.02	
1	101744000.0	291966000.0	12.54	
2	103905000.0	297114000.0	12.17	
3	107101000.0	303163000.0	11.57	
4	108649000.0	308754000.0	11.24	
5	112033000.0	315036000.0	10.80	
6	115551000.0	320406000.0	11.04	
7	117731000.0	324572000.0	10.83	
8	122571000.0	332087000.0	11.04	
9	128941000.0	337082000.0	11.22	

10		133369000.0	332891000.0	11.14
11		133570000.0	344933000.0	11.92
12		136975000.0	356170000.0	11.58
13		0.0	0.0	11.17

	dispo	imp%dispo	exp%prod
0	8.609164e+07	12.18	13.36
1	8.993548e+07	12.68	13.67
2	9.316483e+07	12.32	13.38
3	9.613370e+07	11.75	13.06
4	9.895642e+07	11.45	13.07
5	1.033313e+08	10.95	12.23
6	1.061326e+08	11.19	12.39
7	1.100093e+08	11.00	12.43
8	1.130721e+08	11.20	12.50
9	1.162668e+08	11.36	12.44
10	1.170694e+08	11.27	12.29
11	1.203776e+08	11.97	12.36
12	1.227609e+08	11.68	12.44
13	1.253371e+08	11.27	12.07



Les exportations totales de viande de poulet représentent environ 12.5% du volume de la production annuelle. Concrètement, cela signifie qu'en moyenne sur 8 tonnes de viande de poulet produites par un pays, 1 tonne est exportée. Cette proportion est remarquablement stable à partir de 2015. On remarque que les exportations déclarées sont supérieures aux importations déclarées : cela signifie qu'en réalité une part de la marchandise est perdue dans la chaîne logistique (pourrissement par exemple).

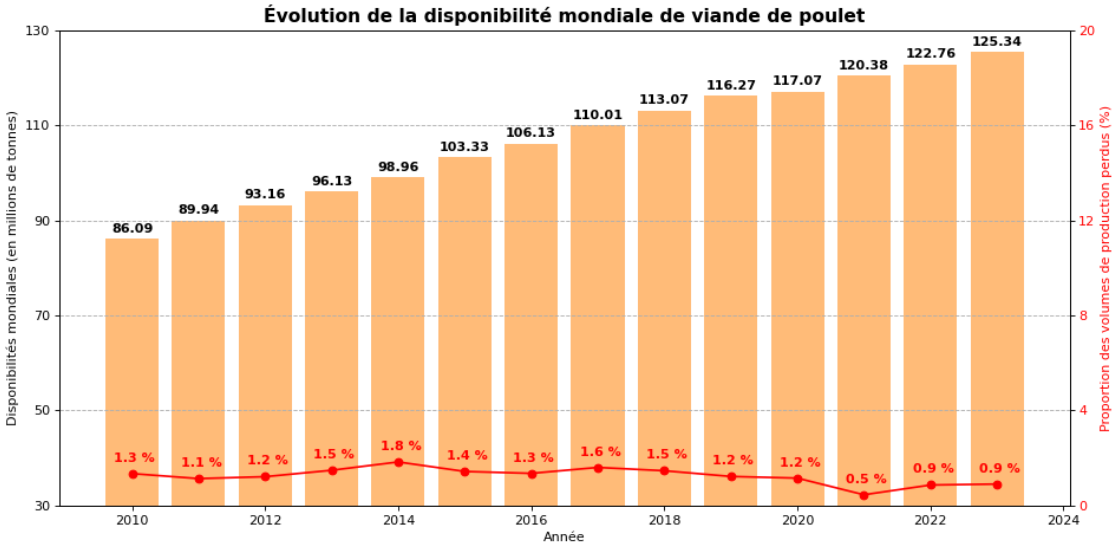
[12]:
```
df_plot["disp%prod"] = round(100*df_plot["dispo"]/df_plot["prod"],2)
df_plot["%pertes"] = 100 - df_plot["disp%prod"]
display(df_plot)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['dispo']/1e6, color=cm['tab20'](3))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['dispo'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Disponibilités mondiales (en millions de tonnes)", fontsize=10)
ax1.set_ylim([30,130])
ax1.set_yticks(np.linspace(30,130,6))
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['%pertes'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['%pertes'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['%pertes'][i]),
                 xytext=(df_plot['Année'][i]-0.27,df_plot['%pertes'][i]+0.5),
                 textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion des volumes de production perdus (%)", color='red')
ax2.set_ylim([0,20])
ax2.set_yticks(np.linspace(0,20,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution de la disponibilité mondiale de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("Prod-pertes_combined.png")
plt.show()
```

	Année	imp	prod	exp	Totale	\
0	2010	10486428.00	8.725923e+07	11654015.00	7.019945e+09	
1	2011	11407722.00	9.096430e+07	12436540.40	7.109087e+09	
2	2012	11476420.10	9.430685e+07	12618431.00	7.199350e+09	
3	2013	11293584.00	9.757971e+07	12739594.00	7.289930e+09	
4	2014	11330122.89	1.008051e+08	13178789.90	7.379761e+09	
5	2015	11318925.81	1.048329e+08	12820538.66	7.468666e+09	
6	2016	11874453.38	1.075863e+08	13328140.41	7.556747e+09	
7	2017	12105680.85	1.117978e+08	13894228.96	7.643804e+09	

8	2018	12666462.12	1.147447e+08	14339086.72	7.728090e+09
9	2019	13208066.35	1.177071e+08	14648443.57	7.809496e+09
10	2020	13196634.56	1.184262e+08	14553485.89	7.885209e+09
11	2021	14408898.20	1.209177e+08	14948942.29	7.952668e+09
12	2022	14336185.63	1.238304e+08	15405711.97	8.019652e+09
13	2023	14131333.32	1.264769e+08	15271128.61	8.089999e+09

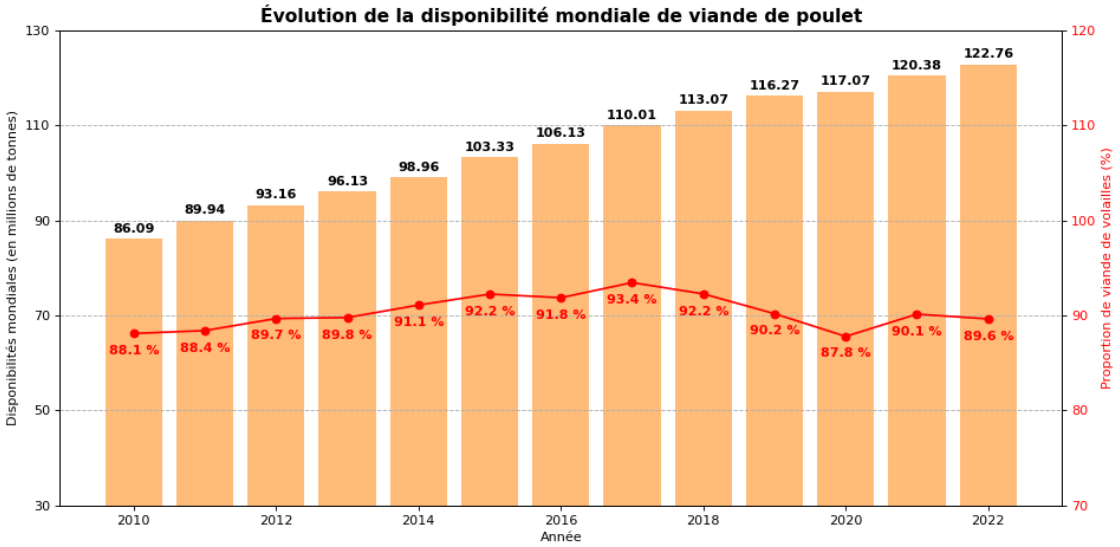
	Disponibilité - Viande de Volailles	Disponibilité - Viandes	imp%prod	\
0	97733000.0	289506000.0	12.02	
1	101744000.0	291966000.0	12.54	
2	103905000.0	297114000.0	12.17	
3	107101000.0	303163000.0	11.57	
4	108649000.0	308754000.0	11.24	
5	112033000.0	315036000.0	10.80	
6	115551000.0	320406000.0	11.04	
7	117731000.0	324572000.0	10.83	
8	122571000.0	332087000.0	11.04	
9	128941000.0	337082000.0	11.22	
10	133369000.0	332891000.0	11.14	
11	133570000.0	344933000.0	11.92	
12	136975000.0	356170000.0	11.58	
13	0.0	0.0	11.17	

	dispo	imp%dispo	exp%prod	disp%prod	%pertes
0	8.609164e+07	12.18	13.36	98.66	1.34
1	8.993548e+07	12.68	13.67	98.87	1.13
2	9.316483e+07	12.32	13.38	98.79	1.21
3	9.613370e+07	11.75	13.06	98.52	1.48
4	9.895642e+07	11.45	13.07	98.17	1.83
5	1.033313e+08	10.95	12.23	98.57	1.43
6	1.061326e+08	11.19	12.39	98.65	1.35
7	1.100093e+08	11.00	12.43	98.40	1.60
8	1.130721e+08	11.20	12.50	98.54	1.46
9	1.162668e+08	11.36	12.44	98.78	1.22
10	1.170694e+08	11.27	12.29	98.85	1.15
11	1.203776e+08	11.97	12.36	99.55	0.45
12	1.227609e+08	11.68	12.44	99.14	0.86
13	1.253371e+08	11.27	12.07	99.10	0.90



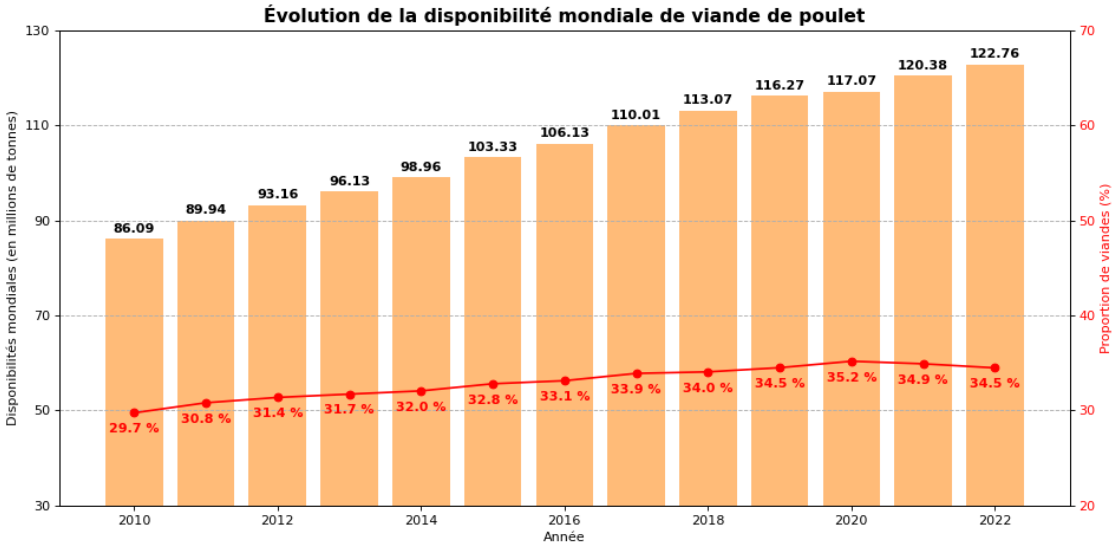
[13]:
```
df_plot = df_plot.loc[df_plot["Année"]<2023].copy()

df_plot["%VVolailles"] = round(100*df_plot["dispo"]/df_plot["Disponibilité -␣
 ↪Viande de Volailles"],2)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['dispo']/1e6, color=cm['tab20'](3))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['dispo'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Disponibilités mondiales (en millions de tonnes)", fontsize=10)
ax1.set_ylim([30,130])
ax1.set_yticks(np.linspace(30,130,6))
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['%VVolailles'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['%VVolailles'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['%VVolailles'][i]),
                 xytext=(df_plot['Année'][i]-0.35,df_plot['%VVolailles'][i]-2.
 ↪2),
```

```
                textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion de viande de volailles (%)", color='red')
ax2.set_ylim([70,120])
ax2.set_yticks(np.linspace(70,120,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution de la disponibilité mondiale de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("dispo_VVolailles_combined.png")
plt.show()
```



[14]:
```
df_plot["%Viandes"] = round(100*df_plot["dispo"]/df_plot["Disponibilité -␣
 ↪Viandes"],2)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['dispo']/1e6, color=cm['tab20'](3))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['dispo'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Disponibilités mondiales (en millions de tonnes)", fontsize=10)
ax1.set_ylim([30,130])
ax1.set_yticks(np.linspace(30,130,6))
```

```
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['%Viandes'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['%Viandes'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['%Viandes'][i]),
                 xytext=(df_plot['Année'][i]-0.35,df_plot['%Viandes'][i]-2.1),
                 textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion de viandes (%)", color='red')
ax2.set_ylim([20,70])
ax2.set_yticks(np.linspace(20,70,6))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution de la disponibilité mondiale de viande de poulet",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("Poulet_Viandes_combined.png")
plt.show()
```



[15]:
```
df_plot["dispo/hab"] = round(df_plot["dispo"]*1e3/df_plot["Totale"],2)
display(df_plot)

fig = plt.figure(figsize=(12,6), dpi=80)
# Graphique X-Y avec points reliés
```

```
plt.plot(df_plot['Année'],df_plot['dispo/hab'],'o-',color='red')
plt.xlabel("Année", fontsize=10)
plt.ylabel("Disponibilité en viande de poulet (en kg/an/pers)", fontsize=10)
#Resserrage de ma fenêtre pour mettre en valeur l'évolution
plt.ylim([12,16])
#plt.yticks(np.linspace(520,550,num=7))
#Traits de repère
plt.grid(visible=True, axis='y', linestyle='--')
plt.title("Évolution de la disponibilité mondiale en viande de poulet par␣
 ↪personne",fontsize=14, fontweight='heavy')
plt.savefig("Dispo-per-hab.png")
plt.show()
```

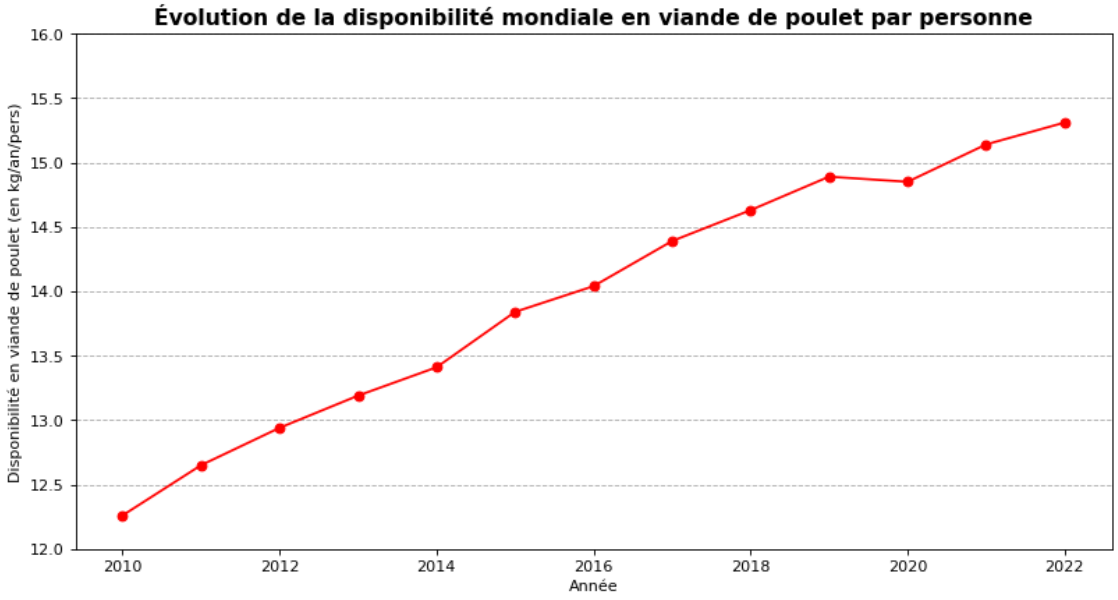
	Année	imp	prod	exp	Totale
0	2010	10486428.00	8.725923e+07	11654015.00	7.019945e+09
1	2011	11407722.00	9.096430e+07	12436540.40	7.109087e+09
2	2012	11476420.10	9.430685e+07	12618431.00	7.199350e+09
3	2013	11293584.00	9.757971e+07	12739594.00	7.289930e+09
4	2014	11330122.89	1.008051e+08	13178789.90	7.379761e+09
5	2015	11318925.81	1.048329e+08	12820538.66	7.468666e+09
6	2016	11874453.38	1.075863e+08	13328140.41	7.556747e+09
7	2017	12105680.85	1.117978e+08	13894228.96	7.643804e+09
8	2018	12666462.12	1.147447e+08	14339086.72	7.728090e+09
9	2019	13208066.35	1.177071e+08	14648443.57	7.809496e+09
10	2020	13196634.56	1.184262e+08	14553485.89	7.885209e+09
11	2021	14408898.20	1.209177e+08	14948942.29	7.952668e+09
12	2022	14336185.63	1.238304e+08	15405711.97	8.019652e+09

	Disponibilité - Viande de Volailles	Disponibilité - Viandes	imp%prod
0	97733000.0	289506000.0	12.02
1	101744000.0	291966000.0	12.54
2	103905000.0	297114000.0	12.17
3	107101000.0	303163000.0	11.57
4	108649000.0	308754000.0	11.24
5	112033000.0	315036000.0	10.80
6	115551000.0	320406000.0	11.04
7	117731000.0	324572000.0	10.83
8	122571000.0	332087000.0	11.04
9	128941000.0	337082000.0	11.22
10	133369000.0	332891000.0	11.14
11	133570000.0	344933000.0	11.92
12	136975000.0	356170000.0	11.58

	dispo	imp%dispo	exp%prod	disp%prod	%pertes	%VVolailles
0	8.609164e+07	12.18	13.36	98.66	1.34	88.09
1	8.993548e+07	12.68	13.67	98.87	1.13	88.39
2	9.316483e+07	12.32	13.38	98.79	1.21	89.66
3	9.613370e+07	11.75	13.06	98.52	1.48	89.76

4	9.895642e+07	11.45	13.07	98.17	1.83	91.08
5	1.033313e+08	10.95	12.23	98.57	1.43	92.23
6	1.061326e+08	11.19	12.39	98.65	1.35	91.85
7	1.100093e+08	11.00	12.43	98.40	1.60	93.44
8	1.130721e+08	11.20	12.50	98.54	1.46	92.25
9	1.162668e+08	11.36	12.44	98.78	1.22	90.17
10	1.170694e+08	11.27	12.29	98.85	1.15	87.78
11	1.203776e+08	11.97	12.36	99.55	0.45	90.12
12	1.227609e+08	11.68	12.44	99.14	0.86	89.62

	%Viandes	dispo/hab
0	29.74	12.26
1	30.80	12.65
2	31.36	12.94
3	31.71	13.19
4	32.05	13.41
5	32.80	13.84
6	33.12	14.04
7	33.89	14.39
8	34.05	14.63
9	34.49	14.89
10	35.17	14.85
11	34.90	15.14
12	34.47	15.31

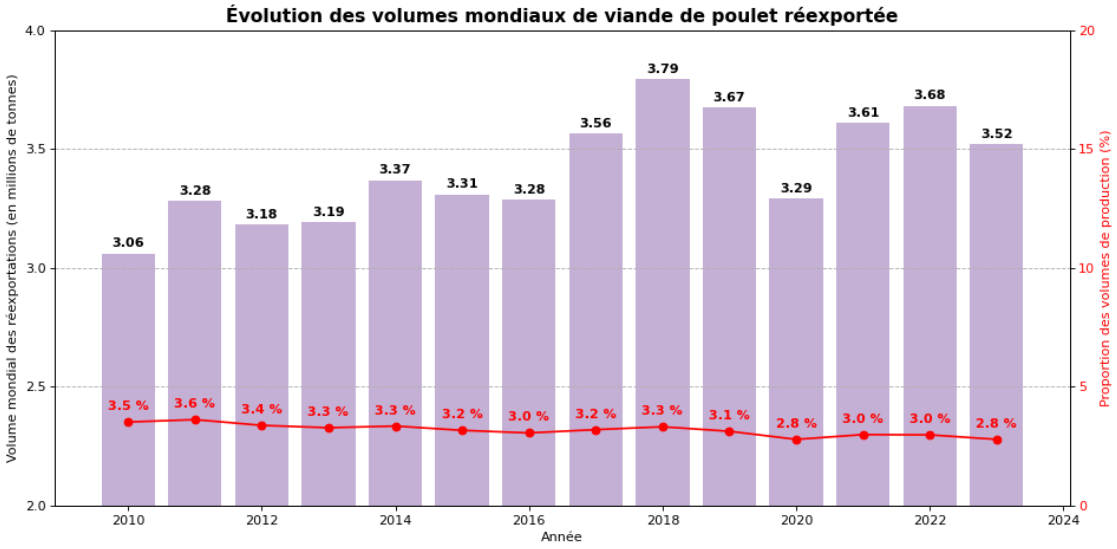


Croissance moyenne d'environ 1,8% / an, soit environ 250 g/an supplémentaire chaque année. En rapportant cela au nombre de jours par an, cela représente 6 jours de consommation supplémentaires.

[16]:
```
# On peut aussi évaluer la quantité de marchandise réexportée
df_world["reexp"] = df_world[['exp','imp']].min(axis=1)
df_plot = df_world.groupby("Année")[["prod","reexp"]].sum().reset_index().
 ↪sort_values("Année", ascending=True)
df_plot["reexp%prod"] = round(100*df_plot["reexp"]/df_plot["prod"],2)

display(df_plot)

fig, ax1 = plt.subplots(figsize=(12,6), dpi=80)

p=ax1.bar(df_plot['Année'],df_plot['reexp']/1e6, color=cm['tab20'](9))

plt.bar_label(p, label_type='edge',
              labels=round((df_plot['reexp'])/1e6,2), padding=3.0,
              fontsize=10, fontweight='bold')

ax1.set_xlabel("Année", fontsize=10)
ax1.set_ylabel("Volume mondial des réexportations (en millions de tonnes)",␣
 ↪fontsize=10)
ax1.set_ylim([2,4])
ax1.set_yticks(np.linspace(2,4,5))
plt.grid(visible=True, axis='y', linestyle='--')

ax2 = ax1.twinx()  # crée un nouveau jeu d'axes avec le même axe x que ax1

ax2.plot(df_plot['Année'],df_plot['reexp%prod'],'-o',color='red')

for i in range(df_plot.shape[0]):
    ax2.annotate('%.1f' %df_plot['reexp%prod'][i] + ' %',
                 xy=(df_plot['Année'][i],df_plot['reexp%prod'][i]),
                 xytext=(df_plot['Année'][i]-0.31,df_plot['reexp%prod'][i]+0.
 ↪50),
                 textcoords='data', color='red', fontweight='heavy')

ax2.set_ylabel("Proportion des volumes de production (%)", color='red')
ax2.set_ylim([0,20])
ax2.set_yticks(np.linspace(0,20,5))
ax2.tick_params(axis='y', labelcolor='red')
plt.title("Évolution des volumes mondiaux de viande de poulet réexportée",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout()  # otherwise the right y-label is slightly clipped

plt.savefig("reexp-prod_combined.png")
plt.show()
```

	Année	prod	reexp	reexp%prod
0	2010	8.725923e+07	3060302.00	3.51

1	2011	9.096430e+07	3281233.00	3.61
2	2012	9.430685e+07	3182307.00	3.37
3	2013	9.757971e+07	3191176.00	3.27
4	2014	1.008051e+08	3367899.09	3.34
5	2015	1.048329e+08	3308220.06	3.16
6	2016	1.075863e+08	3284778.96	3.05
7	2017	1.117978e+08	3563820.95	3.19
8	2018	1.147447e+08	3793756.76	3.31
9	2019	1.177071e+08	3673699.19	3.12
10	2020	1.184262e+08	3292873.09	2.78
11	2021	1.209177e+08	3609008.88	2.98
12	2022	1.238304e+08	3681433.00	2.97
13	2023	1.264769e+08	3518466.37	2.78

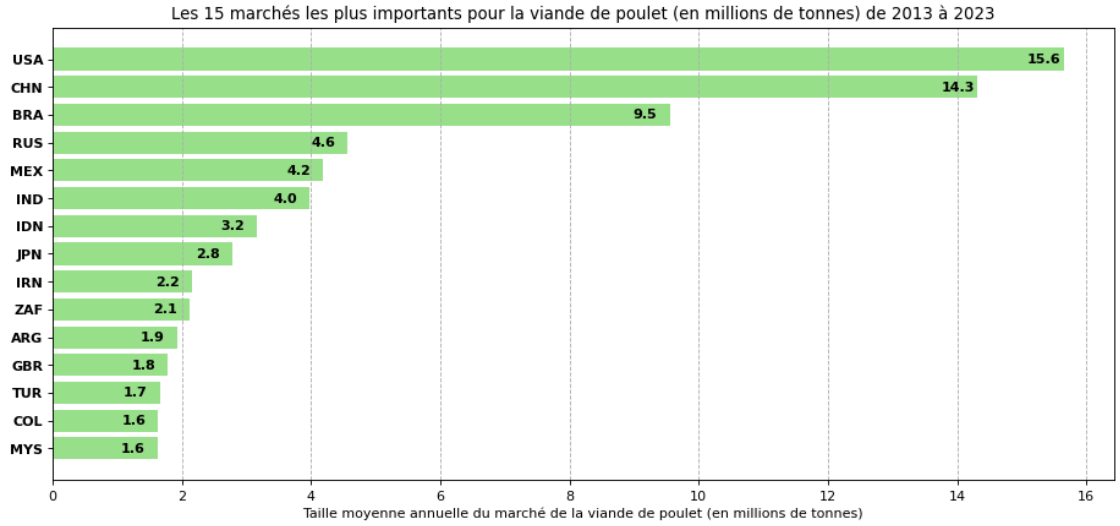


2.2 - Pays avec la plus grande disponibilité

[17]:
```
df_world["dispo"] = df_world["imp"] - df_world["exp"] + df_world["prod"]
df_plot = df_world.loc[df_world["Année"]>2012]
df_plotavg = df_plot.groupby(by=["ISO3","Zone"])["dispo"].mean().reset_index().
  ↪sort_values("dispo", ascending=False).head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['dispo']/1e6,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](5))
plt.bar_label(p, label_type='edge', fmt='%.1f', padding=-27.0, fontsize=10,␣
  ↪fontweight='bold')
```

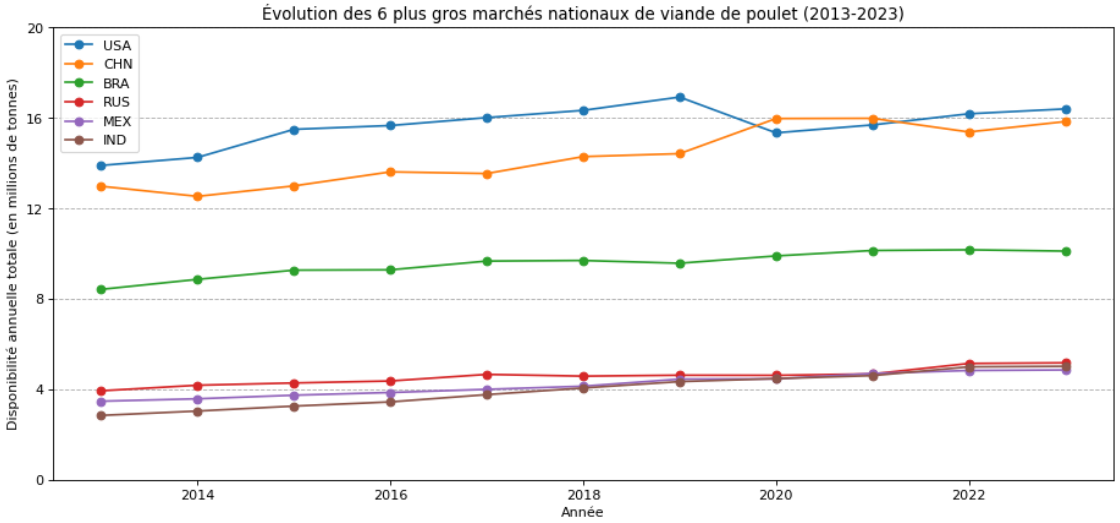
```
plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel('Taille moyenne annuelle du marché de la viande de poulet (en␣
 ↪millions de tonnes)')
plt.yticks(fontweight='bold')
plt.title("Les 15 marchés les plus importants pour la viande de poulet (en␣
 ↪millions de tonnes) de 2013 à 2023")
plt.savefig("Top15_dispo.png")

plt.show()
```

	ISO3	Zone	dispo
218	USA	États-Unis d'Amérique	1.564719e+07
38	CHN	Chine (continentale)	1.431571e+07
28	BRA	Brésil	9.549333e+06
176	RUS	Fédération de Russie	4.564941e+06
131	MEX	Mexique	4.187794e+06
96	IND	Inde	3.983027e+06
94	IDN	Indonésie	3.168203e+06
105	JPN	Japon	2.781075e+06
98	IRN	Iran (République islamique d')	2.154058e+06
230	ZAF	Afrique du Sud	2.116256e+06
7	ARG	Argentine	1.932672e+06
73	GBR	Royaume-Uni de Grande-Bretagne et d'Irlande du…	1.785543e+06
211	TUR	Türkiye	1.660044e+06
44	COL	Colombie	1.634536e+06
146	MYS	Malaisie	1.626998e+06



Attention toutefois, l'Inde (IND) et l'Indonésie (IDN) ne sont pas des pays importateurs : leur marché n'est donc pas accessible.

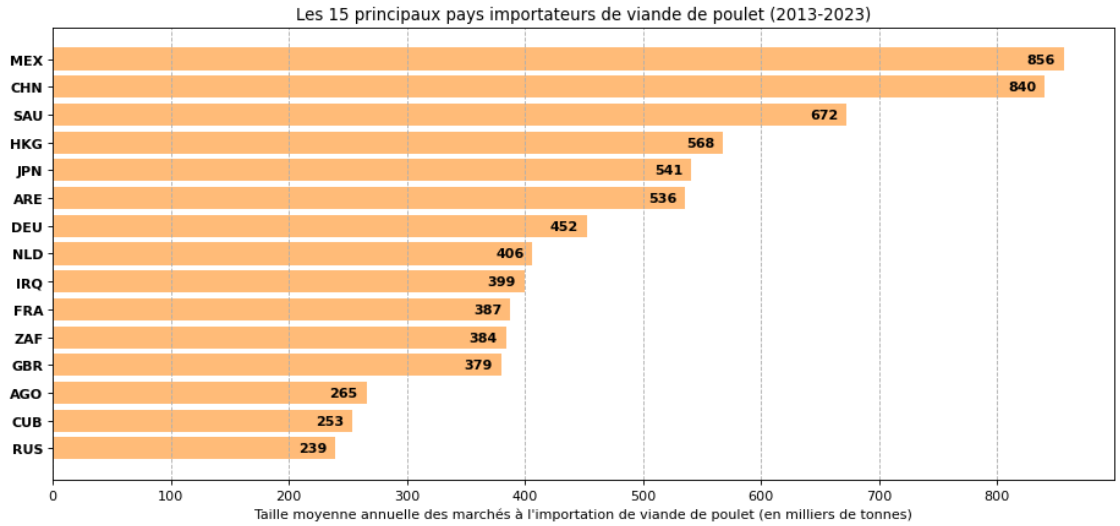
[18]:
```
# Évolution au cours du temps des 5 plus gros marchés
lz = list((df_plotavg.head(6))["ISO3"])
df_plotz = df_plot[df_plot['ISO3'].isin(lz)]

# Graphique à courbes multiples pour suivre l'évolution de l'aide apportées aux␣
 ↪différents pays

fig = plt.figure(figsize=(14,6), dpi=80)

# Boucle sur les pays
for i in range(len(lz)):
    z = lz[i]
    plt.plot(df_plotz.loc[df_plotz['ISO3']==z,'Année'],
             df_plotz.loc[df_plotz['ISO3']==z,'dispo']/1e6,
             'o-',color=cm['tab10'](i))

plt.grid(visible=True, axis='y', linestyle='--')
plt.xlabel('Année')
plt.yticks(np.linspace(0,20,6))
plt.ylabel('Disponibilité annuelle totale (en millions de tonnes)')
plt.legend(lz)
plt.title("Évolution des 6 plus gros marchés nationaux de viande de poulet␣
 ↪(2013-2023)")
plt.savefig("Top6Dispo_Evo.png")

plt.show()
```



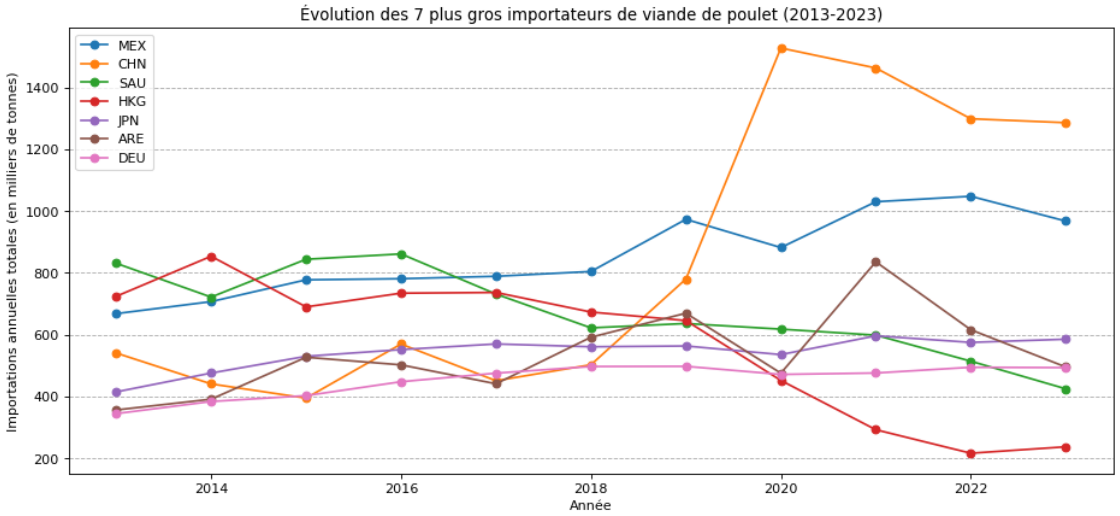
2.3 - Principaux pays importateurs

[19]:
```
df_plotavg = df_plot.groupby(by=["ISO3","Zone"])["imp"].mean().reset_index().
 ↪sort_values("imp", ascending=False).head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['imp']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](3))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-27.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Taille moyenne annuelle des marchés à l'importation de viande de␣
 ↪poulet (en milliers de tonnes)")
plt.yticks(fontweight='bold')
plt.title("Les 15 principaux pays importateurs de viande de poulet (2013-2023)")
plt.savefig("Top15_imp.png")

plt.show()
```

	ISO3	Zone	imp
131	MEX	Mexique	856599.909091
38	CHN	Chine (continentale)	840851.567273
178	SAU	Arabie saoudite	672511.840000
89	HKG	Chine - RAS de Hong-Kong	568167.512727
105	JPN	Japon	541214.275455
6	ARE	Émirats arabes unis	536015.670000
53	DEU	Allemagne	452590.406364
154	NLD	Pays-Bas (Royaume des)	406365.299091
99	IRQ	Iraq	399469.313636
69	FRA	France	387470.482727
230	ZAF	Afrique du Sud	384106.888182
73	GBR	Royaume-Uni de Grande-Bretagne et d'Irlande du…	379821.650000
2	AGO	Angola	265843.717273
48	CUB	Cuba	253975.006364
176	RUS	Fédération de Russie	239799.517273



[20]:
```
# Évolution au cours du temps des 7 plus gros marchés
lz = list((df_plotavg.head(7))["ISO3"])
df_plotz = df_plot[df_plot['ISO3'].isin(lz)]

# Graphique à courbes multiples pour suivre l'évolution de l'aide apportées aux␣
 ↪différents pays

fig = plt.figure(figsize=(14,6), dpi=80)

# Boucle sur les pays
for i in range(len(lz)):
    z = lz[i]
    plt.plot(df_plotz.loc[df_plotz['ISO3']==z,'Année'],
             df_plotz.loc[df_plotz['ISO3']==z,'imp']/1e3,
             'o-',color=cm['tab10'](i))

plt.grid(visible=True, axis='y', linestyle='--')
plt.xlabel('Année')
plt.ylabel('Importations annuelles totales (en milliers de tonnes)')
plt.legend(lz)
plt.title("Évolution des "+ str(len(lz)) +" plus gros importateurs de viande de␣
 ↪poulet (2013-2023)")
plt.savefig("Top7Imp_evo.png")

plt.show()
```

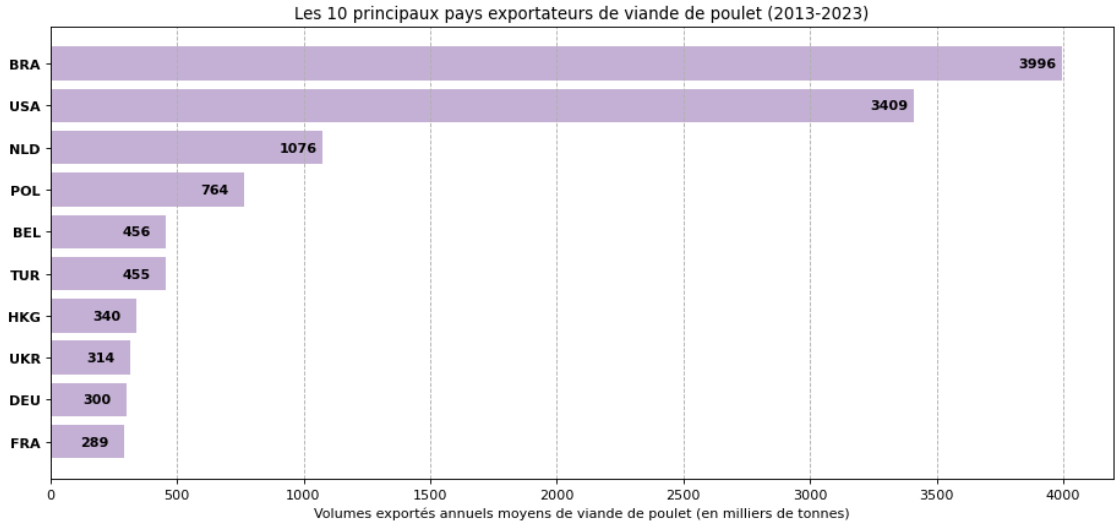


2.4 - Principaux pays exportateurs

[21]:
```
df_plotavg = df_plot.groupby(by=["ISO3","Zone"])["exp"].mean().reset_index().
 ↪sort_values("exp", ascending=False).head(10)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['exp']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](9))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-32.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Volumes exportés annuels moyens de viande de poulet (en milliers de␣
 ↪tonnes)")
plt.yticks(fontweight='bold')
plt.title("Les 10 principaux pays exportateurs de viande de poulet (2013-2023)")
plt.savefig("Top10_exp.png")

plt.show()
```

	ISO3	Zone	exp
28	BRA	Brésil	3.996504e+06
218	USA	États-Unis d'Amérique	3.409222e+06
154	NLD	Pays-Bas (Royaume des)	1.076731e+06
166	POL	Pologne	7.643988e+05
15	BEL	Belgique	4.568880e+05
211	TUR	Türkiye	4.558297e+05

```
89    HKG  Chine - RAS de Hong-Kong  3.400280e+05
216   UKR                   Ukraine  3.141351e+05
53    DEU                 Allemagne  3.007098e+05
69    FRA                    France  2.899821e+05
```



2.5 - Pays les plus intéressants en termes de croissance des importations (2013-2023)

On reprend notre fonction qui calculait les taux de croissance pour qu'elle calcule les variations annuelles moyennes.

[22]:
```
def VarRA_multicol(df, colonnes, debut=2013, fin=2023):
    """
    Remarque importante : cette fonction a vocation à s'appliquer à des␣
 ↪variables ayant des valeurs positives ou nulles
    """
    resultats = {}
    dfb = df.loc[(df["Année"]>=debut)&(df["Année"]<=fin)].copy()
    nm = (debut+fin)/2
    dfb["wgt"] = dfb["Année"] - nm
    dfb["wgt2"] = dfb["wgt"] ** 2
    # On suppose qu'on a les données de toues les années pour toutes les␣
 ↪variables
    # Dénominateur du calcul de a
    swgt2 = sum(dfb["wgt2"].unique())
    for col in colonnes:

        dfb["wgt_"+col] = (dfb[col]) * (dfb["wgt"])
        # agrégation des valeurs pour calculs : calcul du numérateur de a du␣
 ↪modèle
```

```
        wsum_dfb = (dfb[["ISO3", "wgt_"+col]]).groupby("ISO3")["wgt_"+col].
↪sum().reset_index().rename(columns={"wgt_"+col: "ra"})

        # calcul de a du modèle en divisant par le dénominateur
        wsum_dfb["ra_"+col] = round(wsum_dfb["ra"] / swgt2, 2)

        resultats["ra_"+col] = wsum_dfb[["ISO3", "ra_"+col]].
↪set_index("ISO3")["ra_"+col]

    # EN vertu de la remarque préliminaire, si la valeur moyenne est nulle,␣
↪toutes les valeurs sont nulles
    return (pd.DataFrame(resultats).fillna(0.0))
```

[23]:
```
df_ra = VarRA_multicol(df=df_dtlt,␣
↪colonnes=["imp","exp","prod","dispo","pop","dispo/hab"], debut=2013,␣
↪fin=2023).reset_index()
display(df_ra)
```

	ISO3	ra_imp	ra_exp	ra_prod	ra_dispo	ra_pop	ra_dispo/hab
0	AFG	-6696.12	10.02	839.72	-5866.42	1994825.91	-0.25
1	AGO	-12918.77	0.91	5617.73	-7301.96	2120350.55	-0.94
2	ALB	4153.05	4.91	217.50	4365.64	-18112.78	1.63
3	ARE	48999.28	16812.38	-149.98	32036.92	479892.45	0.75
4	ARG	1043.02	-34966.51	79886.90	115896.43	601613.42	2.04
..	…	…	…	…	…	…	
182	WSM	688.81	14.28	-26.28	648.25	3820.31	1.74
183	YEM	5856.22	-6.46	7216.08	13078.77	2007095.44	-0.13
184	ZAF	-5550.53	-3229.17	54523.38	52202.03	1685415.22	-0.12
185	ZMB	6797.66	1939.15	1090.30	5948.80	1066024.53	0.13
186	ZWE	-531.09	0.00	9407.50	8876.40	465245.87	0.39

[187 rows x 7 columns]

La disponibilité totale d'un pays (=consommation) peut dépendre du niveau de la consommation individuelle (chaque individu d'un pays consomme plus) ou de la population totale. Il faut récupérer les valeurs moyennes des variables pop et dispo/hab pour calculer convenablement les composantes de la variation de la disponibilité.

[24]:
```
df_vm = df_dtlt.loc[df_dtlt["Année"]>2012]
df_vm = (df_vm[["ISO3","pop","dispo/hab", "imp", "exp"]].
↪groupby("ISO3")[["pop","dispo/hab","imp","exp"]].mean().reset_index()).
↪round(2)
df_ra = df_ra.merge(df_vm, on="ISO3", how='inner')
display(df_ra)
```

	ISO3	ra_imp	ra_exp	ra_prod	ra_dispo	ra_pop	ra_dispo/hab	\
0	AFG	-6696.12	10.02	839.72	-5866.42	1994825.91	-0.25	
1	AGO	-12918.77	0.91	5617.73	-7301.96	2120350.55	-0.94	

```
2    ALB   4153.05      4.91    217.50    4365.64   -18112.78          1.63
3    ARE  48999.28  16812.38   -149.98   32036.92   479892.45          0.75
4    ARG   1043.02 -34966.51  79886.90  115896.43   601613.42          2.04
..   …       …       …        …         …          …                  …
182  WSM    688.81     14.28    -26.28     648.25     3820.31          1.74
183  YEM   5856.22     -6.46   7216.08   13078.77  2007095.44         -0.13
184  ZAF  -5550.53  -3229.17  54523.38   52202.03  1685415.22         -0.12
185  ZMB   6797.66   1939.15   1090.30    5948.80  1066024.53          0.13
186  ZWE   -531.09      0.00   9407.50    8876.40   465245.87          0.39

              pop  dispo/hab        imp        exp
0    36758062.91       1.59   30146.73      52.98
1    31358489.00      10.06  265843.72      72.78
2     2876923.27      13.42   24500.61       4.91
3     9259546.36      58.23  536015.67   47777.40
4    44395603.91      43.46    4318.53  209064.85
..           …         …         …         …
182    207637.27      77.46   15739.14      34.87
183  34180357.18       8.35  100922.62      15.84
184  58886231.00      35.95  384106.89   51249.95
185  18006950.64       3.40   17087.18    3800.64
186  15097522.91       6.48    5285.46       0.00

[187 rows x 11 columns]
```

Par abus de langage, on va désigner *ra_vpop* comme étant la variation de disponibilité due à la variation de population (à consommation par tête constante) et *ra_vdispo/hab* la variation de disponibilité due à la variation de la consommation individuelle (à population totale constante).

[25]:
```
# On divise par 1000 car la disponibilité par personne est donnée en kg
df_ra["ra_vpop"] = round(df_ra["ra_pop"] * df_ra["dispo/hab"] / 1000, 2)
df_ra["ra_vdispo/hab"] = round(df_ra["ra_dispo/hab"] * df_ra["pop"] /1000, 2)
# Ne pas oublier le résidu, qui est du second ordre
df_ra["res"] = round(df_ra["ra_pop"] * df_ra["ra_dispo/hab"] / 1000, 2)
display(df_ra)
```

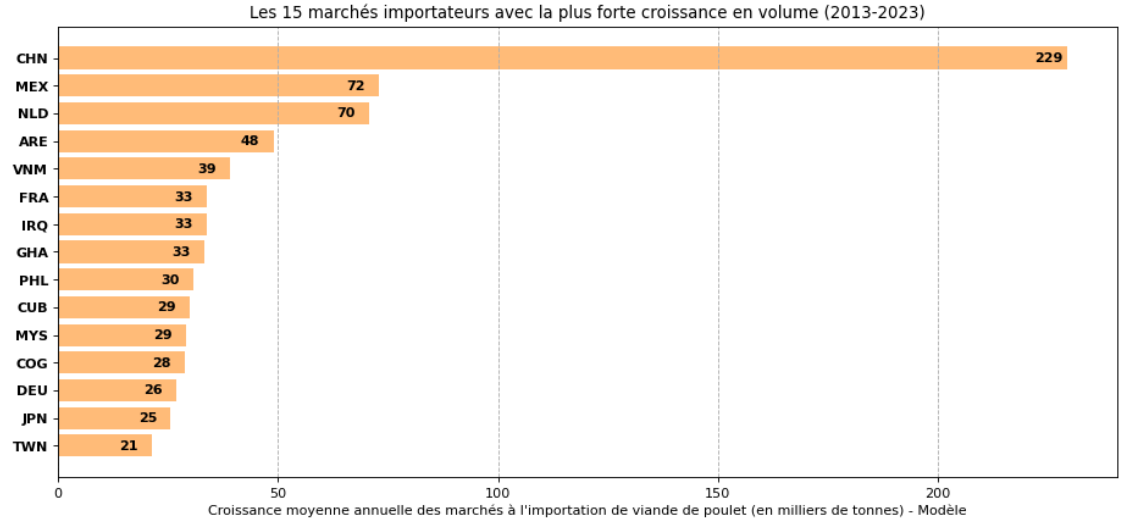
```
    ISO3    ra_imp    ra_exp   ra_prod   ra_dispo      ra_pop  ra_dispo/hab  \
0    AFG  -6696.12     10.02    839.72   -5866.42  1994825.91         -0.25
1    AGO -12918.77      0.91   5617.73   -7301.96  2120350.55         -0.94
2    ALB   4153.05      4.91    217.50    4365.64   -18112.78          1.63
3    ARE  48999.28  16812.38   -149.98   32036.92   479892.45          0.75
4    ARG   1043.02 -34966.51  79886.90  115896.43   601613.42          2.04
..   …       …       …        …         …          …                  …
182  WSM    688.81     14.28    -26.28     648.25     3820.31          1.74
183  YEM   5856.22     -6.46   7216.08   13078.77  2007095.44         -0.13
184  ZAF  -5550.53  -3229.17  54523.38   52202.03  1685415.22         -0.12
185  ZMB   6797.66   1939.15   1090.30    5948.80  1066024.53          0.13
186  ZWE   -531.09      0.00   9407.50    8876.40   465245.87          0.39
```

```
              pop  dispo/hab        imp        exp   ra_vpop  ra_vdispo/hab  \
0    36758062.91       1.59   30146.73      52.98   3171.77       -9189.52
1    31358489.00      10.06  265843.72      72.78  21330.73      -29476.98
2     2876923.27      13.42   24500.61       4.91   -243.07        4689.38
3     9259546.36      58.23  536015.67   47777.40  27944.14        6944.66
4    44395603.91      43.46    4318.53  209064.85  26146.12       90567.03
..           …          …          …          …         …              …
182    207637.27      77.46   15739.14      34.87    295.92         361.29
183  34180357.18       8.35  100922.62      15.84  16759.25       -4443.45
184  58886231.00      35.95  384106.89   51249.95  60590.68       -7066.35
185  18006950.64       3.40   17087.18    3800.64   3624.48        2340.90
186  15097522.91       6.48    5285.46       0.00   3014.79        5888.03

         res
0    -498.71
1   -1993.13
2     -29.52
3     359.92
4    1227.29
..       …
182     6.65
183  -260.92
184  -202.25
185   138.58
186   181.45

[187 rows x 14 columns]
```

[26]:
```
df_plotavg = df_ra[["ISO3","ra_imp"]].sort_values("ra_imp", ascending=False).
 ↪head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['ra_imp']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](3))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-24.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Croissance moyenne annuelle des marchés à l'importation de viande␣
 ↪de poulet (en milliers de tonnes) - Modèle")
plt.yticks(fontweight='bold')
plt.title("Les 15 marchés importateurs avec la plus forte croissance en volume␣
 ↪(2013-2023)")
```

```
plt.show()
```

	ISO3	ra_imp
31	CHN	229251.50
107	MEX	72829.52
123	NLD	70716.71
3	ARE	48999.28
180	VNM	39117.37
56	FRA	33915.62
78	IRQ	33819.02
60	GHA	33171.09
132	PHL	30887.75
40	CUB	29964.47
118	MYS	29062.78
35	COG	28898.67
43	DEU	26857.08
84	JPN	25627.01
171	TWN	21344.35



Ce résultat est déjà très intéressant en soi. Mais il faut garder à l'esprit que les pays peuvent en même temps importer et exporter de la viande de poulet, c'est ce qu'on avait vu à la fin du paragraphe 2.1 : environ 3% des volumes de la production mondiale annuelle de viande de poulet est réexportée. Si on veut s'inscrire dans une démarche responsable, il faut pénaliser les pays exportent de la viande de poulet. On introduit un ratio de pertinence :

$$r = 1 - \frac{min(imp, exp)}{imp}$$

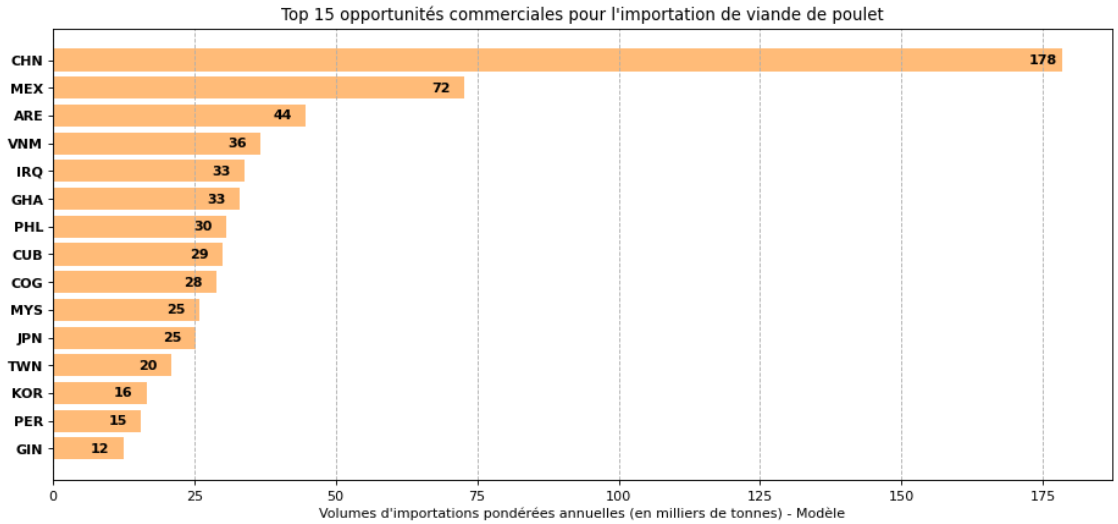
Si imp=0 le ratio de pertinence sera aussi égal à 0.

[27]:
```
df_ra["per"] = round(1 - (df_ra[["imp","exp"]]).min(axis=1)/df_ra["imp"],3).
  ↪fillna(0.0)
df_ra["score"] = round(df_ra["ra_imp"] * df_ra["per"],1)
display(df_ra)
```

```
    ISO3    ra_imp    ra_exp   ra_prod   ra_dispo      ra_pop  ra_dispo/hab  \
0    AFG  -6696.12     10.02    839.72   -5866.42  1994825.91         -0.25
1    AGO -12918.77      0.91   5617.73   -7301.96  2120350.55         -0.94
2    ALB   4153.05      4.91    217.50    4365.64   -18112.78          1.63
3    ARE  48999.28  16812.38   -149.98   32036.92   479892.45          0.75
4    ARG   1043.02 -34966.51  79886.90  115896.43   601613.42          2.04
..   …       …       …       …         …          …
182  WSM    688.81     14.28    -26.28     648.25     3820.31          1.74
183  YEM   5856.22     -6.46   7216.08   13078.77  2007095.44         -0.13
184  ZAF  -5550.53  -3229.17  54523.38   52202.03  1685415.22         -0.12
185  ZMB   6797.66   1939.15   1090.30    5948.80  1066024.53          0.13
186  ZWE   -531.09      0.00   9407.50    8876.40   465245.87          0.39

             pop  dispo/hab        imp        exp   ra_vpop  ra_vdispo/hab  \
0    36758062.91       1.59   30146.73      52.98   3171.77       -9189.52
1    31358489.00      10.06  265843.72      72.78  21330.73      -29476.98
2     2876923.27      13.42   24500.61       4.91   -243.07        4689.38
3     9259546.36      58.23  536015.67   47777.40  27944.14        6944.66
4    44395603.91      43.46    4318.53  209064.85  26146.12       90567.03
..           …       …       …         …       …            …
182    207637.27      77.46   15739.14      34.87    295.92         361.29
183  34180357.18       8.35  100922.62      15.84  16759.25       -4443.45
184  58886231.00      35.95  384106.89   51249.95  60590.68       -7066.35
185  18006950.64       3.40   17087.18    3800.64   3624.48        2340.90
186  15097522.91       6.48    5285.46       0.00   3014.79        5888.03

         res    per    score
0    -498.71  0.998  -6682.7
1   -1993.13  1.000 -12918.8
2     -29.52  1.000   4153.0
3     359.92  0.911  44638.3
4    1227.29  0.000      0.0
..       …    …       …
182     6.65  0.998    687.4
183  -260.92  1.000   5856.2
184  -202.25  0.867  -4812.3
185   138.58  0.778   5288.6
186   181.45  1.000   -531.1

[187 rows x 16 columns]
```

[28]:
```
df_plotavg = df_ra[["ISO3","score"]].sort_values("score", ascending=False).
 ↪head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['score']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](3))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-24.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Volumes d'importations pondérées annuelles (en milliers de tonnes)␣
 ↪- Modèle")
plt.yticks(fontweight='bold')
plt.title("Top 15 opportunités commerciales pour l'importation de viande de␣
 ↪poulet")

plt.show()
```

	ISO3	score
31	CHN	178357.7
107	MEX	72611.0
3	ARE	44638.3
180	VNM	36770.3
78	IRQ	33785.2
60	GHA	33104.7
132	PHL	30609.8
40	CUB	29964.5
35	COG	28869.8
118	MYS	25894.9
84	JPN	25242.6
171	TWN	20938.8
91	KOR	16531.8
131	PER	15546.9
61	GIN	12410.2

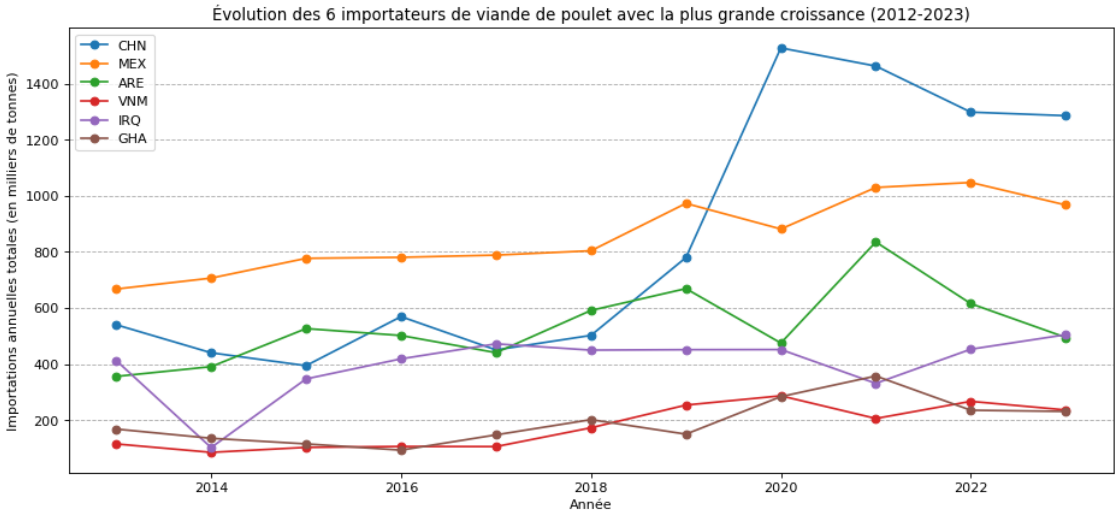


[29]:
```
# Évolution au cours du temps des 7 plus gros marchés
lz = list((df_plotavg.head(6))["ISO3"])
df_plotz = df_dtlt[df_dtlt['ISO3'].isin(lz)]
df_plotz = df_plotz.loc[df_plotz["Année"]>2012].copy()

display(df_plotz[df_plotz["Année"]==2013][["ISO3","Zone","Année","imp"]])

display(df_plotz[df_plotz["Année"]==2023][["ISO3","Zone","Année","imp"]])

# Graphique à courbes multiples pour suivre l'évolution de l'aide apportées aux␣
 ↪différents pays

fig = plt.figure(figsize=(14,6), dpi=80)

# Boucle sur les pays
for i in range(len(lz)):
    z = lz[i]
    plt.plot(df_plotz.loc[df_plotz['ISO3']==z,'Année'],
             df_plotz.loc[df_plotz['ISO3']==z,'imp']/1e3,
             'o-',color=cm['tab10'](i))

plt.grid(visible=True, axis='y', linestyle='--')
plt.xlabel('Année')
plt.ylabel('Importations annuelles totales (en milliers de tonnes)')
plt.legend(lz)
plt.title("Évolution des "+ str(len(lz)) +" importateurs de viande de poulet␣
 ↪avec la plus grande croissance (2012-2023)")
plt.savefig("Top7Croissance_evo.png")
```

```
plt.show()
```

	ISO3	Zone	Année	imp
37	ARE	Émirats arabes unis	2013	355880.0
373	CHN	Chine (continentale)	2013	540156.0
721	GHA	Ghana	2013	168235.0
937	IRQ	Iraq	2013	412851.0
1285	MEX	Mexique	2013	667549.0
2161	VNM	Viet Nam	2013	115187.0

	ISO3	Zone	Année	imp
47	ARE	Émirats arabes unis	2023	494893.11
383	CHN	Chine (continentale)	2023	1285570.36
731	GHA	Ghana	2023	231024.48
947	IRQ	Iraq	2023	504732.02
1295	MEX	Mexique	2023	967112.39
2171	VNM	Viet Nam	2023	236321.76



Les pays qui arrivent en tête du lassement ont vu d'importantes variations de leurs importations sur les années 2019 à 2023. Il semble pertinent de restreindre l'intervalle de temps pour notre scoring des pays.

2.6 - Pays les plus intéressants en termes de croissance des importations (2017-2023)

[30]:
```
df_ra = VarRA_multicol(df=df_dtlt,␣
↪colonnes=["imp","exp","prod","dispo","pop","dispo/hab"], debut=2017,␣
↪fin=2023).reset_index()
display(df_ra)
```

	ISO3	ra_imp	ra_exp	ra_prod	ra_dispo	ra_pop	ra_dispo/hab
0	AFG	-6288.90	-43.91	169.97	-6075.02	1936669.64	-0.22
1	AGO	-15100.91	-56.67	7312.99	-7731.25	2169839.00	-0.84

```
2    ALB  11211.02     11.57   1051.23   12250.68   -30598.71          4.46
3    ARE  26984.90  23581.44    632.93    4036.39   458921.07         -2.04
4    ARG    589.14 -21858.70  79040.43  101488.27   399527.21          1.85
..   …        …        …         …         …          …
182  WSM    367.66     12.29     -3.40     351.98     3792.93          0.27
183  YEM  15764.76    -21.15  12388.64   28174.56  2085988.86          0.32
184  ZAF -72087.50  -2568.09  79782.36   10262.95  1869800.29         -0.93
185  ZMB   8353.39   1734.52   1242.96    7861.83  1092602.93          0.21
186  ZWE    765.03      0.00   6548.35    7313.38   512862.14          0.25

[187 rows x 7 columns]
```

[31]:
```
df_vm = df_dtlt.loc[df_dtlt["Année"]>2016]
df_vm = (df_vm[["ISO3","pop","dispo/hab", "imp", "exp"]].
 ↪groupby("ISO3")[["pop","dispo/hab","imp","exp"]].mean().reset_index()).
 ↪round(2)
df_ra = df_ra.merge(df_vm, on="ISO3", how='inner')
display(df_ra)
```

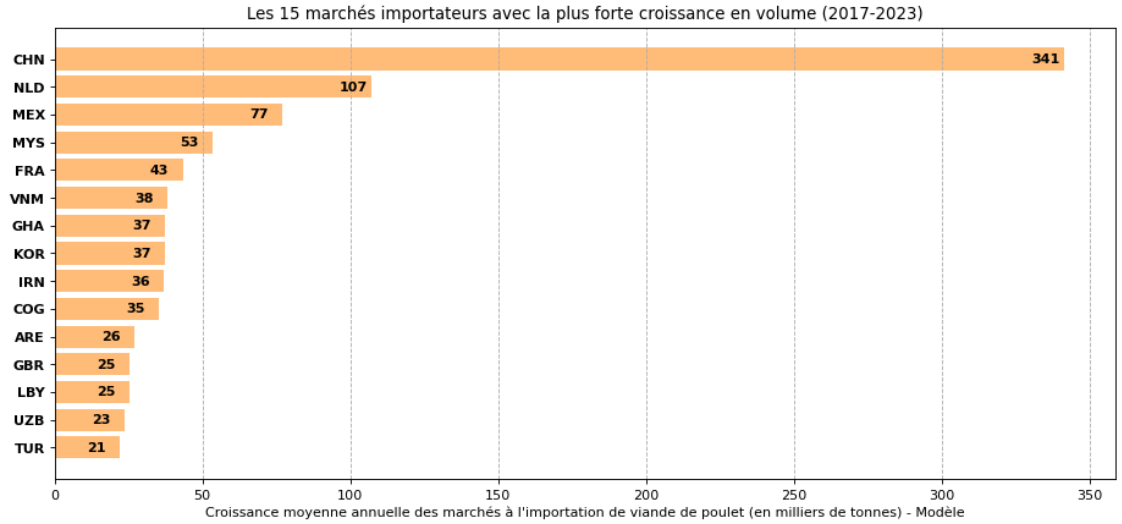
```
    ISO3    ra_imp    ra_exp   ra_prod   ra_dispo      ra_pop  ra_dispo/hab  \
0    AFG  -6288.90    -43.91    169.97   -6075.02  1936669.64         -0.22
1    AGO -15100.91    -56.67   7312.99   -7731.25  2169839.00         -0.84
2    ALB  11211.02     11.57   1051.23   12250.68   -30598.71          4.46
3    ARE  26984.90  23581.44    632.93    4036.39   458921.07         -2.04
4    ARG    589.14 -21858.70  79040.43  101488.27   399527.21          1.85
..   …        …        …         …         …          …
182  WSM    367.66     12.29     -3.40     351.98     3792.93          0.27
183  YEM  15764.76    -21.15  12388.64   28174.56  2085988.86          0.32
184  ZAF -72087.50  -2568.09  79782.36   10262.95  1869800.29         -0.93
185  ZMB   8353.39   1734.52   1242.96    7861.83  1092602.93          0.21
186  ZWE    765.03      0.00   6548.35    7313.38   512862.14          0.25

              pop  dispo/hab        imp        exp
0    38770155.57       1.34   23377.28      83.26
1    33468017.43       9.28  259283.30      87.90
2     2862619.29      14.20   26413.99       7.71
3     9725826.14      59.45  588752.62   63464.80
4    45052541.57      45.39    5451.87  175425.32
..           …         …         …         …
182    211489.14      79.21   16443.85      51.66
183  36168039.86       8.13  105296.54      14.10
184  60498832.14      36.09  396442.12   46806.16
185  19066947.57       3.48   23333.90    5930.76
186  15550325.43       6.94    4157.33       0.00

[187 rows x 11 columns]
```

[32]:
```
# On divise par 1000 car la disponibilité par personne est donnée en kg
df_ra["ra_vpop"] = round(df_ra["ra_pop"] * df_ra["dispo/hab"] / 1000, 2)
df_ra["ra_vdispo/hab"] = round(df_ra["ra_dispo/hab"] * df_ra["pop"] /1000, 2)
# Ne pas oublier le résidu, qui est du second ordre
df_ra["res"] = round(df_ra["ra_pop"] * df_ra["ra_dispo/hab"] / 1000, 2)
display(df_ra)
```

```
    ISO3    ra_imp    ra_exp   ra_prod   ra_dispo      ra_pop  ra_dispo/hab  \
0    AFG  -6288.90    -43.91    169.97   -6075.02  1936669.64         -0.22
1    AGO -15100.91    -56.67   7312.99   -7731.25  2169839.00         -0.84
2    ALB  11211.02     11.57   1051.23   12250.68   -30598.71          4.46
3    ARE  26984.90  23581.44    632.93    4036.39   458921.07         -2.04
4    ARG    589.14 -21858.70  79040.43  101488.27   399527.21          1.85
..   …       …        …        …         …          …                  …
182  WSM    367.66     12.29     -3.40     351.98     3792.93          0.27
183  YEM  15764.76    -21.15  12388.64   28174.56  2085988.86          0.32
184  ZAF -72087.50  -2568.09  79782.36   10262.95  1869800.29         -0.93
185  ZMB   8353.39   1734.52   1242.96    7861.83  1092602.93          0.21
186  ZWE    765.03      0.00   6548.35    7313.38   512862.14          0.25

              pop  dispo/hab        imp        exp   ra_vpop  ra_vdispo/hab  \
0    38770155.57       1.34   23377.28      83.26   2595.14       -8529.43
1    33468017.43       9.28  259283.30      87.90  20136.11      -28113.13
2     2862619.29      14.20   26413.99       7.71   -434.50       12767.28
3     9725826.14      59.45  588752.62   63464.80  27282.86      -19840.69
4    45052541.57      45.39    5451.87  175425.32  18134.54       83347.20
..           …          …        …         …       …              …
182    211489.14      79.21   16443.85      51.66    300.44          57.10
183  36168039.86       8.13  105296.54      14.10  16959.09       11573.77
184  60498832.14      36.09  396442.12   46806.16  67481.09      -56263.91
185  19066947.57       3.48   23333.90    5930.76   3802.26        4004.06
186  15550325.43       6.94    4157.33       0.00   3559.26        3887.58

        res
0    -426.07
1   -1822.66
2    -136.47
3    -936.20
4     739.13
..       …
182     1.02
183   667.52
184 -1738.91
185   229.45
186   128.22

[187 rows x 14 columns]
```

[33]:
```
df_plotavg = df_ra[["ISO3","ra_imp"]].sort_values("ra_imp", ascending=False).
 ↪head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['ra_imp']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](3))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-24.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Croissance moyenne annuelle des marchés à l'importation de viande␣
 ↪de poulet (en milliers de tonnes) - Modèle")
plt.yticks(fontweight='bold')
plt.title("Les 15 marchés importateurs avec la plus forte croissance en volume␣
 ↪(2017-2023)")

plt.show()
```

	ISO3	ra_imp
31	CHN	341451.86
123	NLD	107274.59
107	MEX	77126.89
118	MYS	53348.08
56	FRA	43370.71
180	VNM	38041.45
60	GHA	37502.98
91	KOR	37421.45
77	IRN	36866.64
35	COG	35088.71
3	ARE	26984.90
58	GBR	25425.61
96	LBY	25213.86
177	UZB	23757.10
169	TUR	21903.08



[34]:
```
df_ra["per"] = round(1 - (df_ra[["imp","exp"]]).min(axis=1)/df_ra["imp"],3).
↪fillna(0.0)
df_ra["score"] = round(df_ra["ra_imp"] * df_ra["per"],1)
df_ra["score"] = 0.5*(df_ra["score"] + abs(df_ra["score"]))
display(df_ra)
```

	ISO3	ra_imp	ra_exp	ra_prod	ra_dispo	ra_pop	ra_dispo/hab	\
0	AFG	-6288.90	-43.91	169.97	-6075.02	1936669.64	-0.22	
1	AGO	-15100.91	-56.67	7312.99	-7731.25	2169839.00	-0.84	
2	ALB	11211.02	11.57	1051.23	12250.68	-30598.71	4.46	
3	ARE	26984.90	23581.44	632.93	4036.39	458921.07	-2.04	
4	ARG	589.14	-21858.70	79040.43	101488.27	399527.21	1.85	
..	…	…	…	…	…	…	…	
182	WSM	367.66	12.29	-3.40	351.98	3792.93	0.27	
183	YEM	15764.76	-21.15	12388.64	28174.56	2085988.86	0.32	
184	ZAF	-72087.50	-2568.09	79782.36	10262.95	1869800.29	-0.93	
185	ZMB	8353.39	1734.52	1242.96	7861.83	1092602.93	0.21	
186	ZWE	765.03	0.00	6548.35	7313.38	512862.14	0.25	

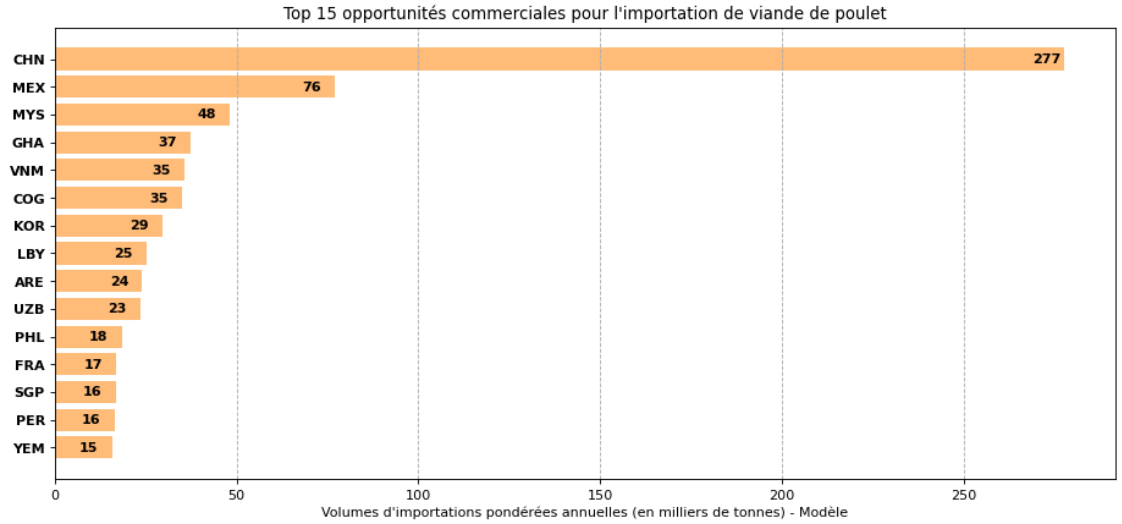
	pop	dispo/hab	imp	exp	ra_vpop	ra_vdispo/hab	\
0	38770155.57	1.34	23377.28	83.26	2595.14	-8529.43	
1	33468017.43	9.28	259283.30	87.90	20136.11	-28113.13	
2	2862619.29	14.20	26413.99	7.71	-434.50	12767.28	
3	9725826.14	59.45	588752.62	63464.80	27282.86	-19840.69	
4	45052541.57	45.39	5451.87	175425.32	18134.54	83347.20	
..	…	…	…	…	…	…	
182	211489.14	79.21	16443.85	51.66	300.44	57.10	
183	36168039.86	8.13	105296.54	14.10	16959.09	11573.77	
184	60498832.14	36.09	396442.12	46806.16	67481.09	-56263.91	
185	19066947.57	3.48	23333.90	5930.76	3802.26	4004.06	

```
186  15550325.43       6.94    4157.33       0.00    3559.26        3887.58

          res    per    score
0     -426.07  0.996      0.0
1    -1822.66  1.000      0.0
2     -136.47  1.000  11211.0
3     -936.20  0.892  24070.5
4      739.13  0.000      0.0
..        …    …       …
182      1.02  0.997    366.6
183    667.52  1.000  15764.8
184 -1738.91  0.882      0.0
185    229.45  0.746   6231.6
186    128.22  1.000    765.0

[187 rows x 16 columns]
```

[35]:
```
df_plotavg = df_ra[["ISO3","score"]].sort_values("score", ascending=False).
 ↪head(15)
display(df_plotavg)

fig = plt.figure(figsize=(14,6), dpi=80)

p=plt.barh(df_plotavg.shape[0]-np.arange(df_plotavg.shape[0]),
           df_plotavg['score']/1e3,
           tick_label=df_plotavg['ISO3'], color=cm['tab20'](3))
plt.bar_label(p, label_type='edge', fmt='%d', padding=-24.0, fontsize=10,␣
 ↪fontweight='bold')

plt.grid(visible=True, axis='x', linestyle='--')
plt.xlabel("Volumes d'importations pondérées annuelles (en milliers de tonnes)␣
 ↪- Modèle")
plt.yticks(fontweight='bold')
plt.title("Top 15 opportunités commerciales pour l'importation de viande de␣
 ↪poulet")

plt.show()
```

```
    ISO3     score
31   CHN  277600.4
107  MEX   76972.6
118  MYS   48173.3
60   GHA   37503.0
180  VNM   35720.9
35   COG   35018.5
91   KOR   29562.9
96   LBY   25213.9
3    ARE   24070.5
```

```
177  UZB   23614.6
132  PHL   18476.4
56   FRA   17001.3
145  SGP   16794.5
131  PER   16496.4
183  YEM   15764.8
```

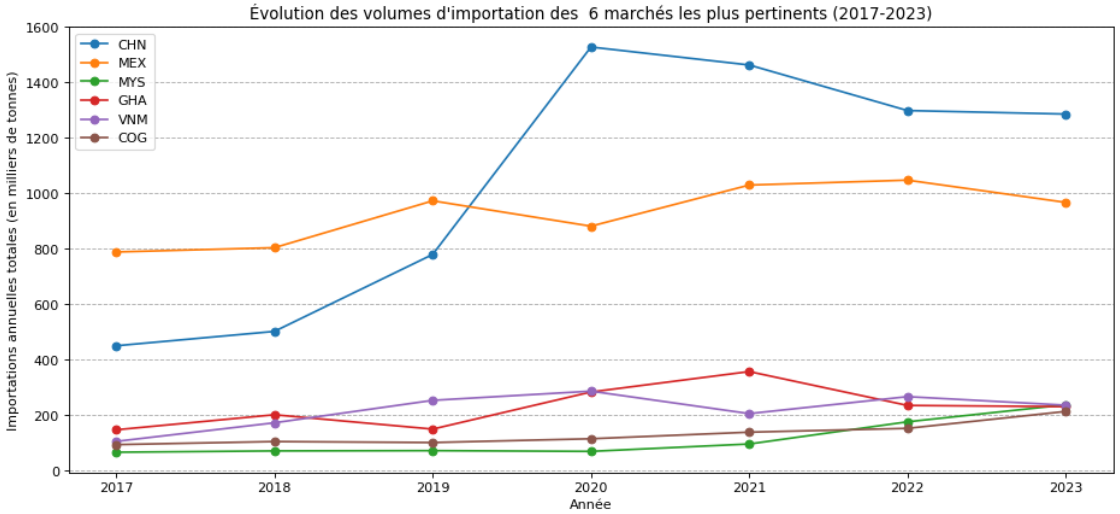


[36]:
```
# Évolution au cours du temps des 7 plus gros marchés
lz = list((df_plotavg.head(6))["ISO3"])
df_plotz = df_dtlt[df_dtlt['ISO3'].isin(lz)]
df_plotz = df_plotz.loc[df_plotz["Année"]>2016].copy()

display(df_plotz[df_plotz["Année"]==2017][["ISO3","Zone","Année","imp"]])

display(df_plotz[df_plotz["Année"]==2023][["ISO3","Zone","Année","imp"]])

# Graphique à courbes multiples pour suivre l'évolution de l'aide apportées aux␣
 ↪différents pays

fig = plt.figure(figsize=(14,6), dpi=80)

# Boucle sur les pays
for i in range(len(lz)):
    z = lz[i]
    plt.plot(df_plotz.loc[df_plotz['ISO3']==z,'Année'],
             df_plotz.loc[df_plotz['ISO3']==z,'imp']/1e3,
             'o-',color=cm['tab10'](i))

plt.grid(visible=True, axis='y', linestyle='--')
```

```
plt.xlabel('Année')
plt.ylabel('Importations annuelles totales (en milliers de tonnes)')
plt.legend(lz)
plt.title("Évolution des volumes d'importation des  "+ str(len(lz)) +" marchés
 ↪les plus pertinents (2017-2023)")
plt.savefig("Top7Croissance_evo.png")

plt.show()
```

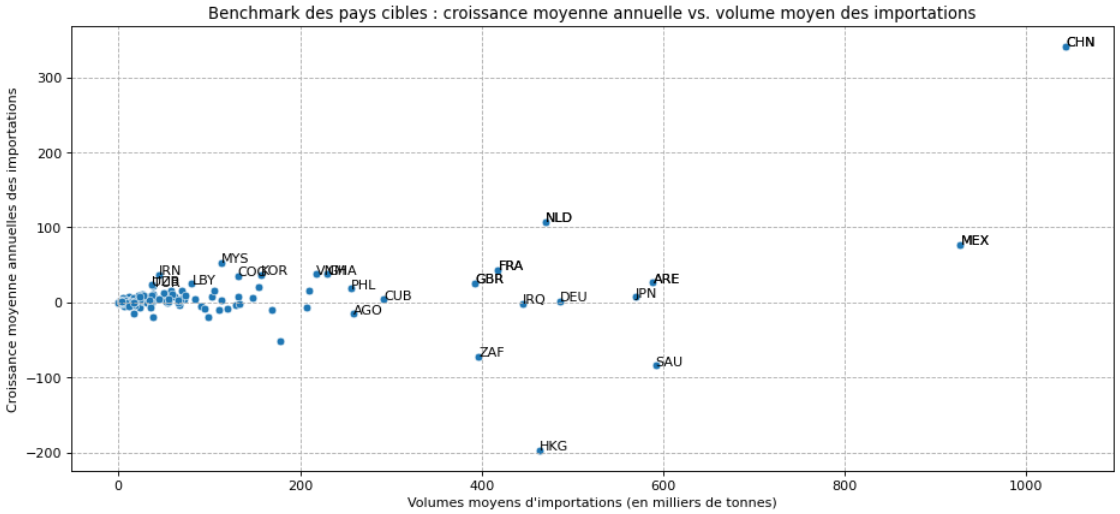
	ISO3	Zone	Année	imp
377	CHN	Chine (continentale)	2017	450411.21
425	COG	Congo	2017	94170.10
725	GHA	Ghana	2017	147537.69
1289	MEX	Mexique	2017	788451.84
1421	MYS	Malaisie	2017	66697.04
2165	VNM	Viet Nam	2017	105695.06

	ISO3	Zone	Année	imp
383	CHN	Chine (continentale)	2023	1285570.36
431	COG	Congo	2023	213739.85
731	GHA	Ghana	2023	231024.48
1295	MEX	Mexique	2023	967112.39
1427	MYS	Malaisie	2023	237671.44
2171	VNM	Viet Nam	2023	236321.76



[37]:
```
# Scatterplot Croissance annuelle moyenne vs. volume moyen d'importations

df_plot = df_ra[["ISO3","imp","ra_imp"]].copy()
df_plot["imp"] /= 1e3
df_plot["ra_imp"] /= 1e3
```

```
fig = plt.figure(figsize=(14,6), dpi=80)

sb.scatterplot(data=df_plot,
               x='imp',
               y='ra_imp')

plt.xlabel("Volumes moyens d'importations (en milliers de tonnes)")
plt.ylabel("Croissance moyenne annuelles des importations")
plt.grid(visible=True, axis='both', linestyle='--')
plt.title("Benchmark des pays cibles : croissance moyenne annuelle vs. volume␣
 ↪moyen des importations")

df_plotMA = df_plot.sort_values('imp', ascending=False, ignore_index=True).
 ↪head(15)
df_plotRA = df_plot.sort_values('ra_imp', ascending=False, ignore_index=True).
 ↪head(15)

for i in range(df_plotMA.shape[0]):
    plt.annotate(df_plotMA['ISO3'][i],
                 xy=(df_plotMA['imp'][i],df_plotMA['ra_imp'][i]),
                 textcoords='data')

for i in range(df_plotRA.shape[0]):
    plt.annotate(df_plotRA['ISO3'][i],
                 xy=(df_plotRA['imp'][i],df_plotRA['ra_imp'][i]),
                 textcoords='data')

plt.show()
```



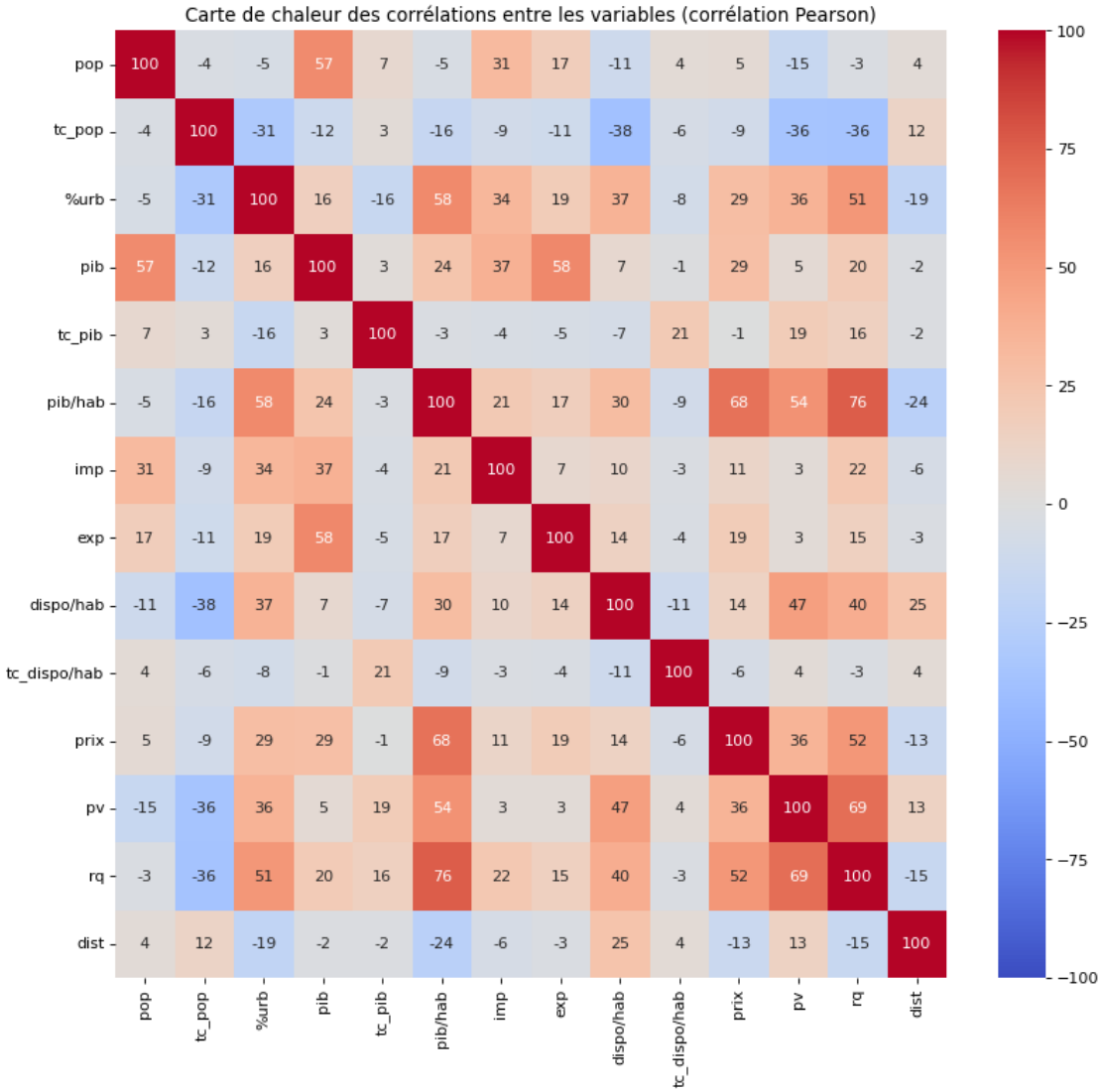
Partie 3 - Analyse en Composantes Principales

3.1 - Éboulis des valeurs propres

[38]:
```
display(list(df_work.columns))
```

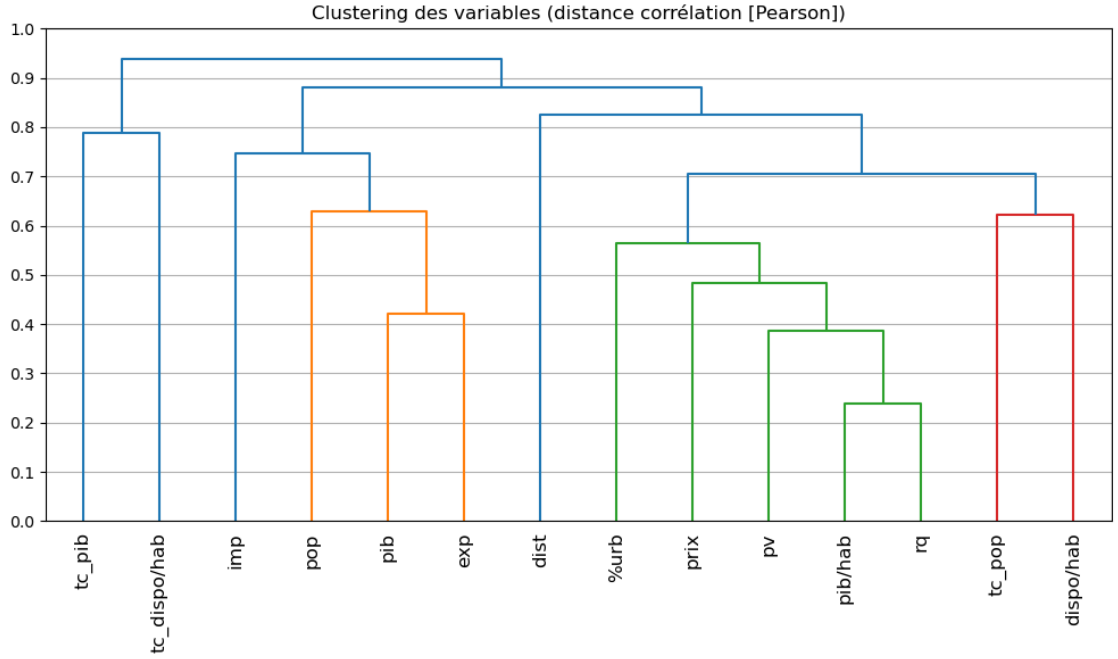
```
['Unnamed: 0',
 'ISO3',
 'Zone',
 'pop',
 'tc_pop',
 '%urb',
 'pib',
 'tc_pib',
 'pib/hab',
 'imp',
 'exp',
 'dispo/hab',
 'tc_dispo/hab',
 'prix',
 'pv',
 'rq',
 'dist']
```

[39]:
```
fig = plt.figure(figsize=(12,11), dpi=80)

sb.heatmap(100*df_work[list(df_work.columns)[3:]].corr(method='pearson'),\
           vmin=-100,vmax=100,cmap='coolwarm', fmt='.0f', annot=True)
#plt.xticks(rotation=45)
plt.title("Carte de chaleur des corrélations entre les variables (corrélation␣
 ↪Pearson)")
#plt.savefig("Heatmap.png")
plt.show()
```



[40]:
```
vcorr = (df_work[list(df_work.columns)[3:]].corr(method='pearson')).abs()

# Distances entre variables
distance = 1 - vcorr
cond = sqf(distance, checks=False)

# Clustering hiérarchique des variables
Z = linkage(cond, method='average')

# visualiser le dendrogramme
plt.figure(figsize=(10, 6))
dendrogram(Z, labels=list(df_work.columns)[3:], leaf_rotation=90)
plt.yticks(np.linspace(0,1,11))
```

```
plt.grid(axis='y')
plt.title("Clustering des variables (distance corrélation [Pearson])")
plt.tight_layout()
plt.savefig("Dendrogramme_variables.png")
plt.show()
```



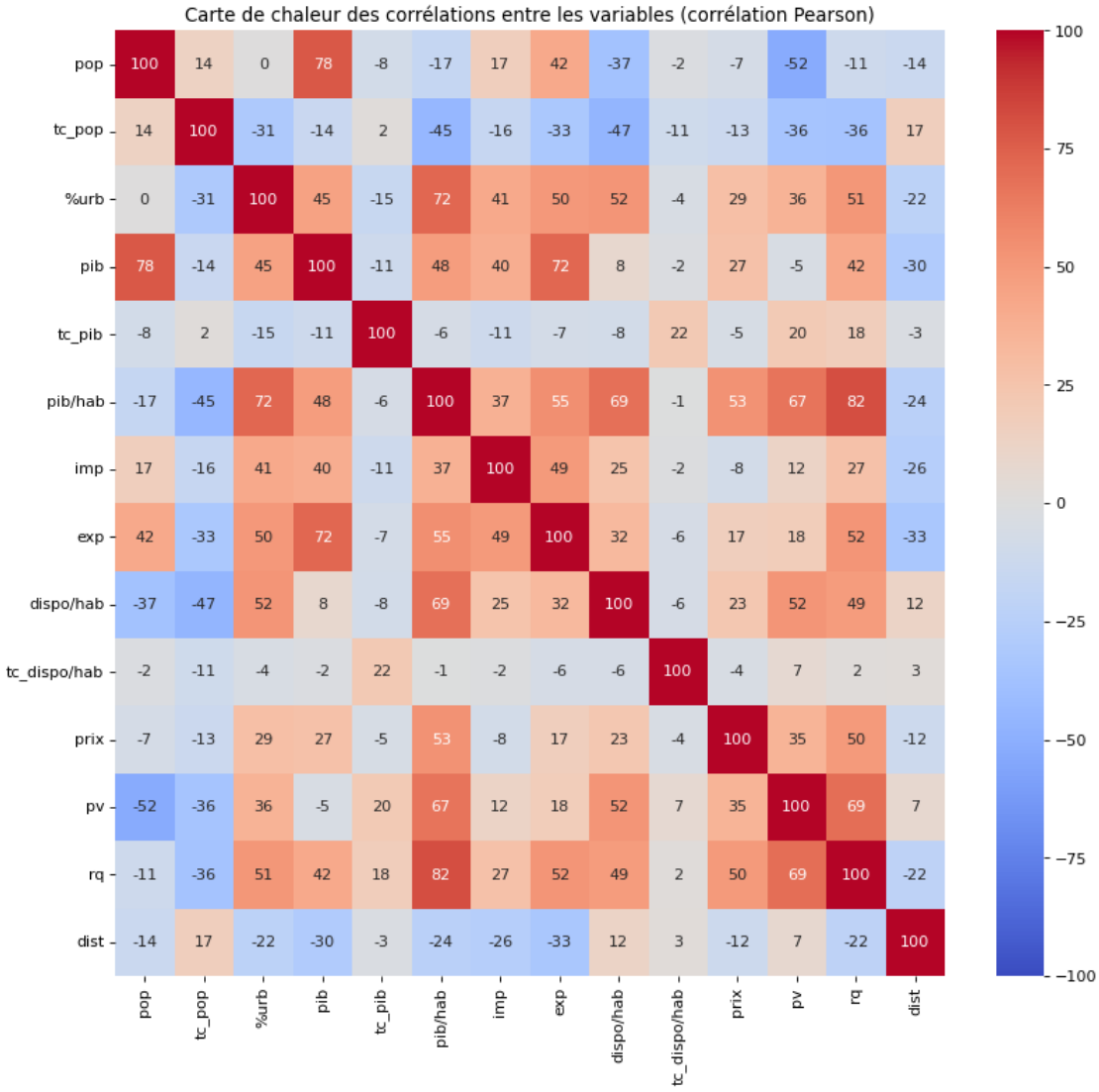
[41]:
```
#Lvar = list(df_work.columns)[3:]
#data = df_work[Lvar].copy()
# On effectue les transformations logarithmiques pour certaines variables
#data["pop"] = data["pop"].apply(lambda x: np.log(1+x))
#data["pib"] = data["pib"].apply(lambda x: np.log(1+x))
#data["pib/hab"] = data["pib/hab"].apply(lambda x: np.log(1+x))
#data["imp"] = data["imp"].apply(lambda x: np.log(1+x))
#data["exp"] = data["exp"].apply(lambda x: np.log(1+x))
#display(data.round(3))
```

[42]:
```
Lvar = list(df_work.columns)[3:]
data = df_work[Lvar].copy()
#On applique les transformations logarithmiques pour avoir les distributions␣
 ↪les plus proches possibles de lois normales
data["tc_pib"] = data["tc_pib"].apply(lambda x: np.sign(x)*np.log(1+abs(x)/10))
data["tc_dispo/hab"] = data["tc_dispo/hab"].apply(lambda x: np.sign(x)*np.
 ↪log(1+abs(x)/10))
data["dist"] = data["dist"].apply(lambda x: np.log(1+x/1e4))
data["prix"] = data["prix"].apply(lambda x: np.log(1+x))
```

```
data["pib/hab"] = data["pib/hab"].apply(lambda x: np.log(1+x))
data["dispo/hab"] = data["dispo/hab"].apply(lambda x: np.log(1+x/10))
data["imp"] = data["imp"].apply(lambda x: np.log(1+x/100))
data["pop"] = data["pop"].apply(lambda x: np.log(1+x/1e5))
data["pib"] = data["pib"].apply(lambda x: np.log(1+x))
data["exp"] = data["exp"].apply(lambda x: np.log(1+20*x))
display(data.round(3))
```

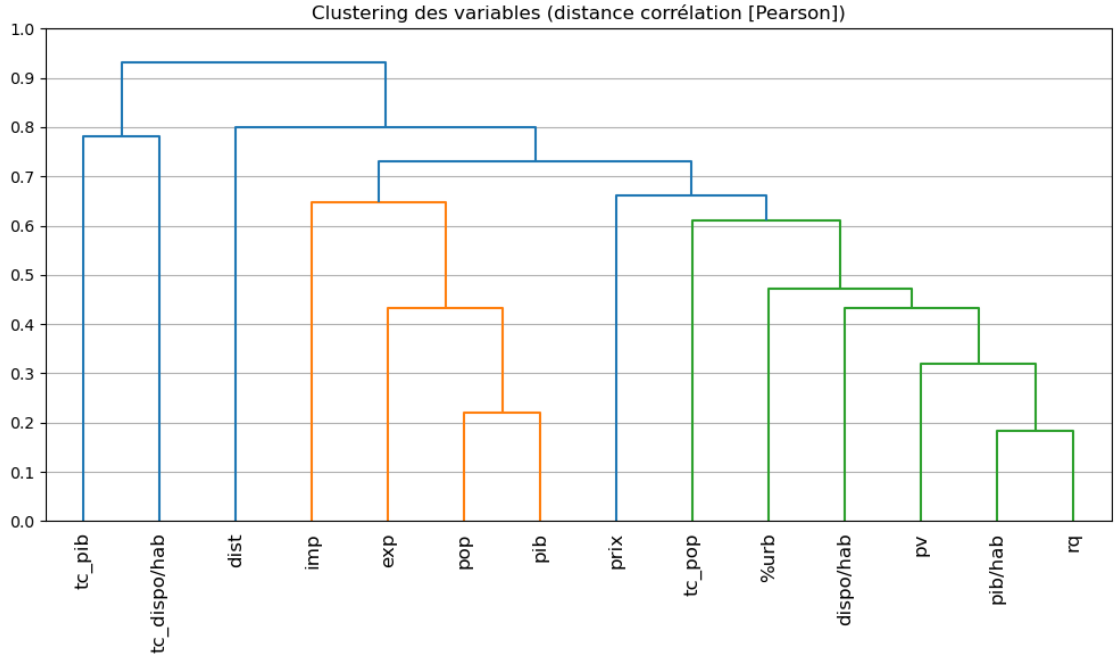
```
       pop  tc_pop   %urb     pib  tc_pib  pib/hab    imp     exp  dispo/hab  \
0    5.910    5.43  25.26  23.620  -0.376    6.216  5.712   6.967      0.148
1    5.751    6.76  64.48  25.360  -0.748    8.130  7.886   7.284      0.696
2    3.393   -0.63  61.43  23.448   0.798    8.579  5.505   4.597      0.851
3    4.539    5.18  89.57  26.753   0.377   10.713  8.587  13.770      1.920
4    6.098    1.36  92.41  27.050  -0.175    9.443  3.788  15.246      1.676
..     …      …     …      …       …       …      …      …          …
182  1.124    1.84  17.45  20.585   0.252    8.341  5.065   6.549      2.169
183  5.837    5.87  31.04  23.662  -1.334    6.366  6.918   5.761      0.607
184  6.380    2.86  64.58  26.661   0.110    8.771  8.254  13.840      1.525
185  5.199    5.92  42.75  23.913   0.078    7.216  5.147  11.239      0.293
186  5.024    3.08  36.22  23.847   0.585    7.313  3.986   0.000      0.500

     tc_dispo/hab   prix    pv    rq   dist
0          -0.945  7.148 -2.60 -1.23  0.444
1          -0.660  7.899 -0.45 -0.85  0.501
2           0.795  6.782  0.24  0.22  0.149
3           0.121  7.367  0.68  0.99  0.422
4           0.385  7.385  0.00 -0.58  0.745
..            …    …   …   …    …
182         0.203  6.849  1.12 -0.20  0.956
183        -0.145  7.224 -2.68 -1.53  0.426
184        -0.032  6.625 -0.34  0.05  0.660
185         0.324  7.256  0.11 -0.55  0.566
186         0.471  6.489 -0.80 -1.57  0.585

[187 rows x 14 columns]
```

[43]:
```
fig = plt.figure(figsize=(12,11), dpi=80)

sb.heatmap(100*data.corr(method='pearson'),\
           vmin=-100,vmax=100,cmap='coolwarm', fmt='.0f', annot=True)
#plt.xticks(rotation=45)
plt.title("Carte de chaleur des corrélations entre les variables (corrélation␣
 ↪Pearson)")
#plt.savefig("Heatmap.png")
plt.show()
```



[44]:
```
vcorr = (data.corr(method='pearson')).abs()

# Distances entre variables
distance = 1 - vcorr
cond = sqf(distance, checks=False)

# Clustering hiérarchique des variables
Z = linkage(cond, method='average')

# visualiser le dendrogramme
plt.figure(figsize=(10, 6))
dendrogram(Z, labels=list(data.columns), leaf_rotation=90)
plt.yticks(np.linspace(0,1,11))
```

```
plt.grid(axis='y')
plt.title("Clustering des variables (distance corrélation [Pearson])")
plt.tight_layout()
plt.savefig("Dendrogramme_variables.png")
plt.show()
```



[45]:
```
#Scaling
scaler = StandardScaler(with_std=True)
scaler

#Fitting
scaler.fit(data)

#Transformation
data_scaled = scaler.transform(data)
data_scaled

idx = ["mean", "std"]

df_scaled = pd.DataFrame(data_scaled)
pd.DataFrame(data_scaled).describe().round(2).loc[idx, :]
```

[45]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
mean	0.0	0.0	-0.0	0.0	0.0	0.0	-0.0	0.0	0.0	-0.0	0.0	0.0	0.0	0.0
std	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

[46]:
```
display(df_scaled.describe().round(3))
```

```
              0        1        2        3        4        5        6        7   \
count   187.000  187.000  187.000  187.000  187.000  187.000  187.000  187.000
mean      0.000    0.000   -0.000    0.000    0.000    0.000   -0.000    0.000
std       1.003    1.003    1.003    1.003    1.003    1.003    1.003    1.003
min      -2.314   -2.246   -2.052   -2.973   -3.822   -2.150   -2.022   -1.729
25%      -0.591   -0.772   -0.808   -0.541   -0.389   -0.787   -0.603   -0.761
50%       0.082   -0.021    0.042   -0.015    0.220    0.026    0.167    0.065
75%       0.704    0.763    0.828    0.749    0.657    0.775    0.754    0.814
max       2.774    2.059    1.934    2.701    2.222    2.093    1.891    2.064

              8        9        10       11       12       13
count   187.000  187.000  187.000  187.000  187.000  187.000
mean      0.000   -0.000    0.000    0.000    0.000    0.000
std       1.003    1.003    1.003    1.003    1.003    1.003
min      -1.688   -3.048   -2.616   -2.769   -2.316   -1.793
25%      -0.886   -0.444   -0.773   -0.593   -0.695   -0.717
50%       0.096    0.083    0.024    0.073   -0.081   -0.021
75%       0.748    0.586    0.761    0.839    0.679    0.753
max       2.322    2.504    2.725    1.636    2.265    2.624
```

[47]:
```
n = len(Lvar)
for i in range(n):
    print("\nVariable "+Lvar[i])
    display(df_work.loc[abs(df_scaled[i])>3][["ISO3","Zone",Lvar[i]]])
```

```
Variable pop
```

```
Empty DataFrame
Columns: [ISO3, Zone, pop]
Index: []
```

```
Variable tc_pop
```

```
Empty DataFrame
Columns: [ISO3, Zone, tc_pop]
Index: []
```

```
Variable %urb
```

```
Empty DataFrame
Columns: [ISO3, Zone, %urb]
Index: []
```

```
Variable pib
```

```
Empty DataFrame
Columns: [ISO3, Zone, pib]
Index: []
```

```
Variable tc_pib
```

	ISO3	Zone	tc_pib
143	SDN	Soudan	-20.07
151	SSD	Soudan du Sud	-26.38
179	VEN	Venezuela (République bolivarienne du)	-26.53
183	YEM	Yémen	-27.96

```
Variable pib/hab
```

```
Empty DataFrame
Columns: [ISO3, Zone, pib/hab]
Index: []
```

```
Variable imp
```

```
Empty DataFrame
Columns: [ISO3, Zone, imp]
Index: []
```

```
Variable exp
```

```
Empty DataFrame
Columns: [ISO3, Zone, exp]
Index: []
```

```
Variable dispo/hab
```

```
Empty DataFrame
Columns: [ISO3, Zone, dispo/hab]
Index: []
```

```
Variable tc_dispo/hab
```

	ISO3	Zone	tc_dispo/hab
99	LSO	Lesotho	-25.76

```
Variable prix
```

```
Empty DataFrame
Columns: [ISO3, Zone, prix]
Index: []
```

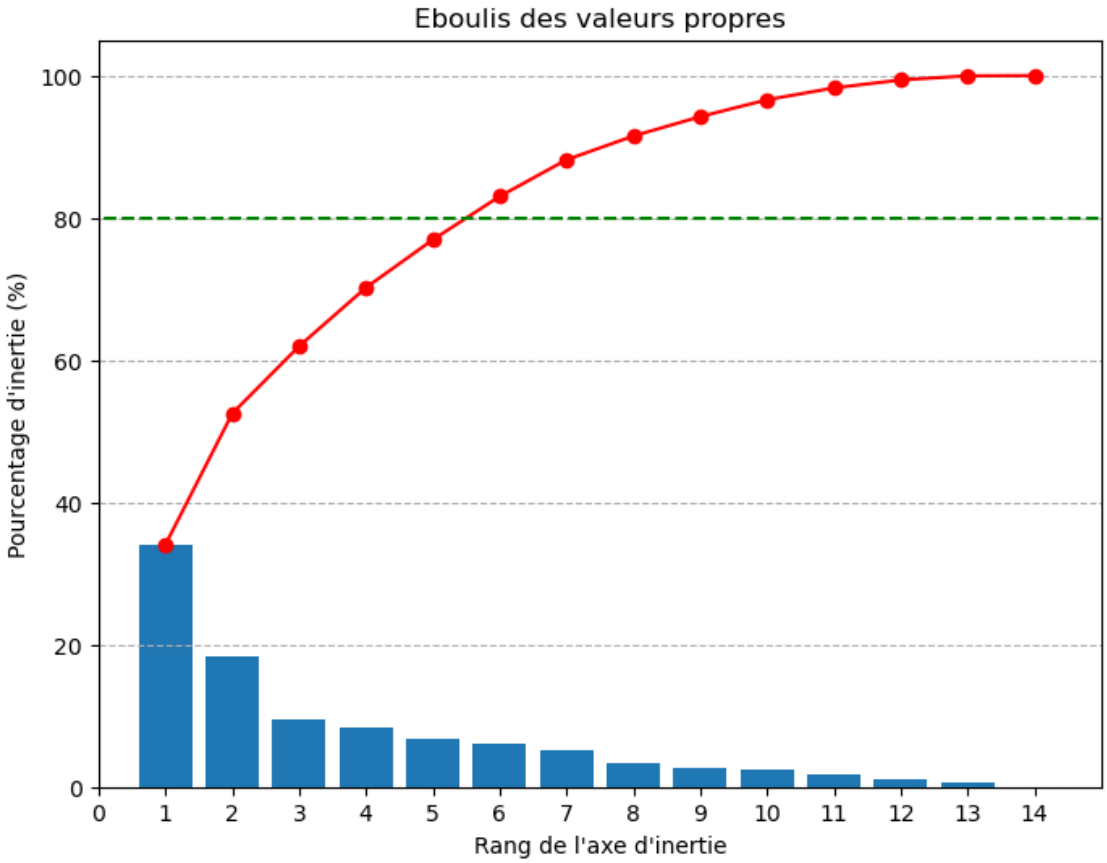
```
Variable pv

Empty DataFrame
Columns: [ISO3, Zone, pv]
Index: []


Variable rq

Empty DataFrame
Columns: [ISO3, Zone, rq]
Index: []


Variable dist

Empty DataFrame
Columns: [ISO3, Zone, dist]
Index: []
```

[48]:
```
pca = PCA(n_components=n)
pca.fit(data_scaled)
```

[48]:
```
PCA(n_components=14)
```

[49]:
```
var = pca.explained_variance_ratio_
cumsum_var = np.cumsum(var)
display(cumsum_var)
d_list = range(1, n+1)
list(d_list)

#Éboulis des valeurs propres
fig, ax = plt.subplots(figsize=(8, 6))
plt.bar(d_list, 100*var)
plt.plot(d_list, 100*cumsum_var,c="red",marker='o')
plt.xlabel("Rang de l'axe d'inertie")
plt.ylabel("Pourcentage d'inertie (%)")
plt.title("Eboulis des valeurs propres")
plt.xticks(np.linspace(0,n,n+1))
plt.plot(np.linspace(-1,16,50), [80]*50, linestyle='--', color='green')
plt.grid(axis='y', linestyle='--')
plt.xlim([0,15])
plt.savefig("Eboulis_VP.png")
plt.show(block=False)
```

```
array([0.34053572, 0.52499669, 0.61953363, 0.70195681, 0.76927529,
       0.83017067, 0.88169909, 0.9150556 , 0.94234165, 0.96603592,
       0.98270948, 0.99386074, 0.99970816, 1.        ])
```



En conservant les 6 premières valeurs propres (3 premiers plans factoriels), on conserve environ 83% de l'information initiale, et on conserve tous les axes d'inertie qui contribuent individuellement à au moins 6% de la variance du jeu de données centré réduit.

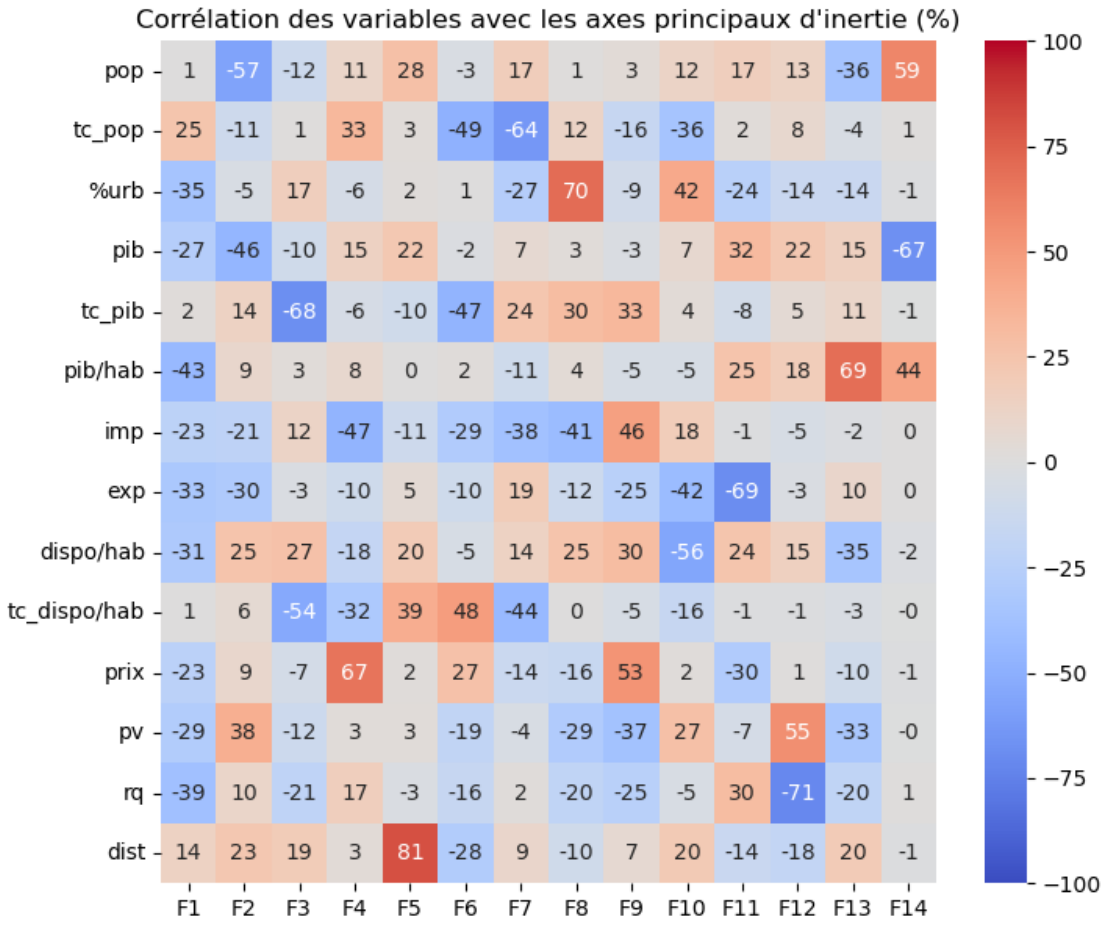
3.2 - Axes principaux

[50]:
```
pcs = pca.components_
df_pcs = pd.DataFrame(pcs)
df_pcs.columns = data.columns
df_pcs.index = [f"F{i}" for i in d_list]
display(df_pcs.round(2))
```

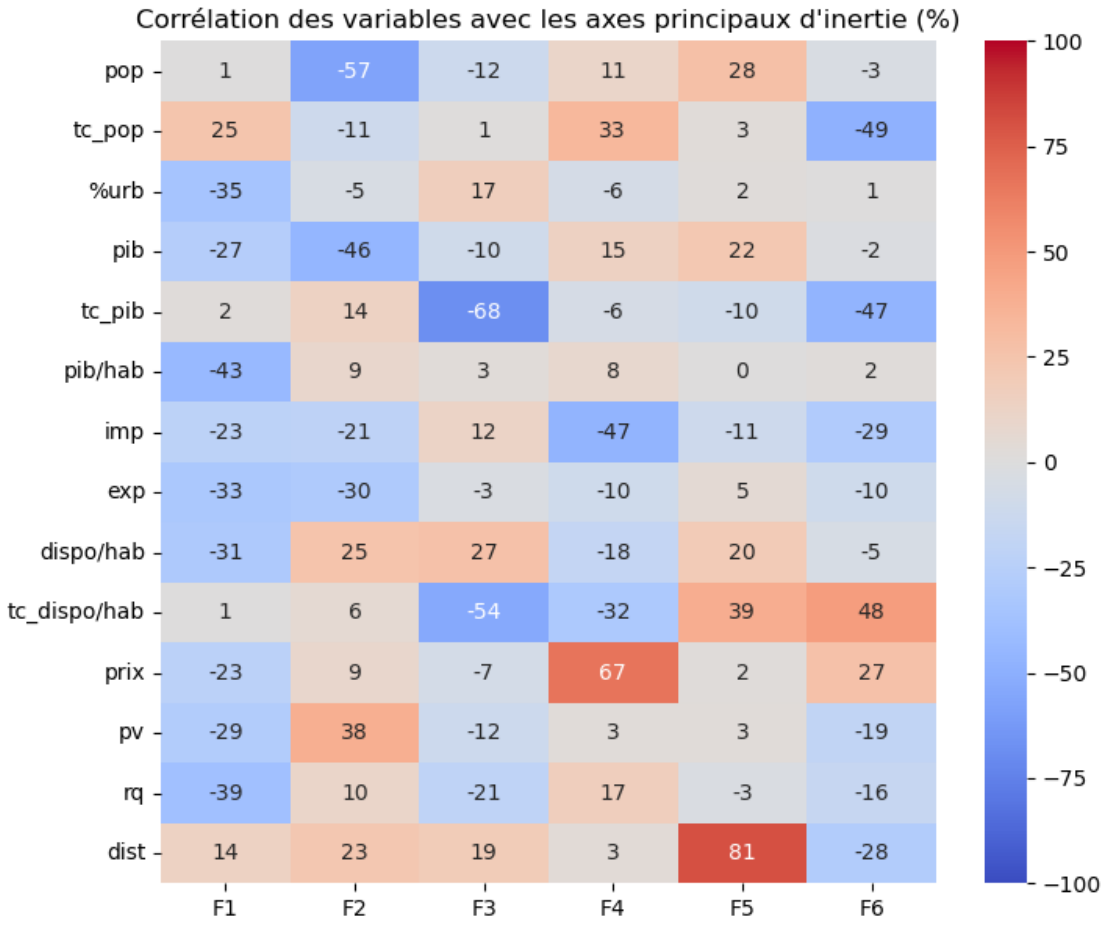
```
      pop  tc_pop  %urb   pib  tc_pib  pib/hab   imp   exp  dispo/hab  \
F1   0.01    0.25 -0.35 -0.27    0.02    -0.43 -0.23 -0.33      -0.31
F2  -0.57   -0.11 -0.05 -0.46    0.14     0.09 -0.21 -0.30       0.25
F3  -0.12    0.01  0.17 -0.10   -0.68     0.03  0.12 -0.03       0.27
F4   0.11    0.33 -0.06  0.15   -0.06     0.08 -0.47 -0.10      -0.18
F5   0.28    0.03  0.02  0.22   -0.10     0.00 -0.11  0.05       0.20
F6  -0.03   -0.49  0.01 -0.02   -0.47     0.02 -0.29 -0.10      -0.05
F7   0.17   -0.64 -0.27  0.07    0.24    -0.11 -0.38  0.19       0.14
F8   0.01    0.12  0.70  0.03    0.30     0.04 -0.41 -0.12       0.25
```

```
F9   0.03   -0.16 -0.09 -0.03    0.33    -0.05  0.46 -0.25       0.30
F10  0.12   -0.36  0.42  0.07    0.04    -0.05  0.18 -0.42      -0.56
F11  0.17    0.02 -0.24  0.32   -0.08     0.25 -0.01 -0.69       0.24
F12  0.13    0.08 -0.14  0.22    0.05     0.18 -0.05 -0.03       0.15
F13 -0.36   -0.04 -0.14  0.15    0.11     0.69 -0.02  0.10      -0.35
F14  0.59    0.01 -0.01 -0.67   -0.01     0.44  0.00  0.00      -0.02

     tc_dispo/hab  prix    pv    rq  dist
F1           0.01 -0.23 -0.29 -0.39  0.14
F2           0.06  0.09  0.38  0.10  0.23
F3          -0.54 -0.07 -0.12 -0.21  0.19
F4          -0.32  0.67  0.03  0.17  0.03
F5           0.39  0.02  0.03 -0.03  0.81
F6           0.48  0.27 -0.19 -0.16 -0.28
F7          -0.44 -0.14 -0.04  0.02  0.09
F8           0.00 -0.16 -0.29 -0.20 -0.10
F9          -0.05  0.53 -0.37 -0.25  0.07
F10         -0.16  0.02  0.27 -0.05  0.20
F11         -0.01 -0.30 -0.07  0.30 -0.14
F12         -0.01  0.01  0.55 -0.71 -0.18
F13         -0.03 -0.10 -0.33 -0.20  0.20
F14         -0.00 -0.01 -0.00  0.01 -0.01
```

[51]:
```
fig, ax = plt.subplots(figsize=(8, 7))
sb.heatmap(100*df_pcs.T, vmin=-100, vmax=100, annot=True, cmap="coolwarm",␣
 ↪fmt=".0f")
plt.title("Corrélation des variables avec les axes principaux d'inertie (%)")
plt.savefig("Correlation_inertia_axes.png")
```



[52]:
```
fig, ax = plt.subplots(figsize=(8, 7))
sb.heatmap(100*df_pcs.iloc[0:6].T, vmin=-100, vmax=100, annot=True,␣
↪cmap="coolwarm", fmt=".0f")
plt.title("Corrélation des variables avec les axes principaux d'inertie (%)")
plt.savefig("Correlation_inertia_axes6.png")
```



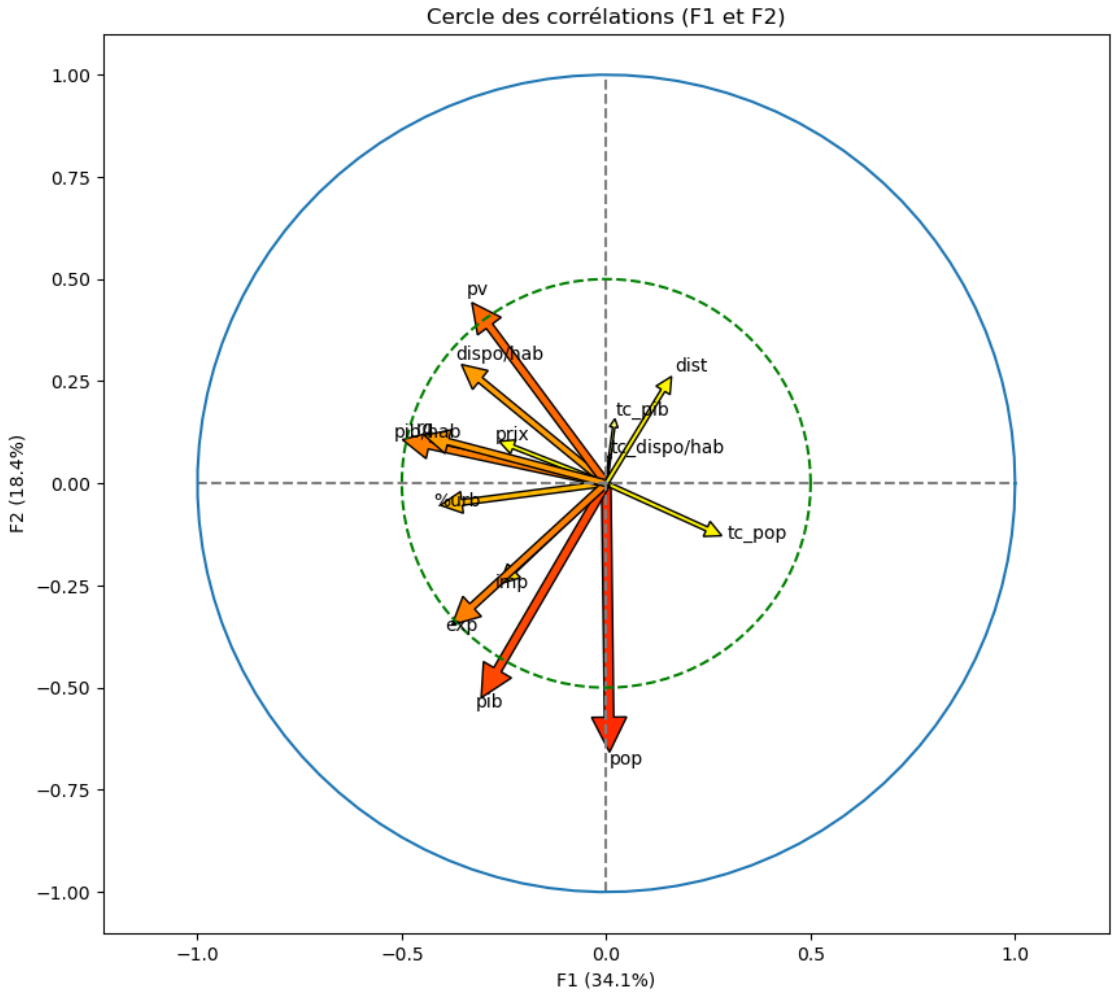
[53]:
```
######

def correlation_graph(pca,
                      x_y,
                      features,
                      fig_name) :
    """Affiche le graphe des correlations

    Positional arguments :
    -------------------------------
    pca : sklearn.decomposition.PCA : notre objet PCA qui a été fit
    x_y : list ou tuple : le couple x,y des plans à afficher, exemple [0,1]␣
 ↪pour F1, F2
    features : list ou tuple : la liste des features (ie des dimensions) à␣
 ↪représenter
    fig_name : nom du fichier image pour le graphique généré
    """
```

```
    # Extrait x et y
    x,y=x_y

    # Taille de l'image (en inches)
    fig, ax = plt.subplots(figsize=(10, 9))

    # Pour chaque composante :
    for i in range(0, pca.components_.shape[1]):

        v_norm = np.sqrt((pca.components_[x, i])**2 + (pca.components_[y,␣
↪i])**2)
        print("Norme vecteur",features[i],"dans cercle des corrélations :
↪",round(v_norm, 3))

        # Les flèches
        ax.arrow(0,0,
                pca.components_[x, i],
                pca.components_[y, i],
                head_width=0.15*v_norm,
                head_length=0.15*v_norm,
                width=0.04*v_norm,
                color=cm["hot"](1-v_norm), ec='black')

        # Les labels
        plt.text(pca.components_[x, i] + 0.2*pca.components_[x, i],
                pca.components_[y, i] + 0.2*pca.components_[y, i],
                features[i])

    # Affichage des lignes horizontales et verticales
    plt.plot([-1, 1], [0, 0], color='grey', ls='--')
    plt.plot([0, 0], [-1, 1], color='grey', ls='--')

    # Nom des axes, avec le pourcentage d'inertie expliqué
    plt.xlabel('F{} ({}%)'.format(x+1, round(100*pca.
↪explained_variance_ratio_[x],1)))
    plt.ylabel('F{} ({}%)'.format(y+1, round(100*pca.
↪explained_variance_ratio_[y],1)))

    # J'ai copié collé le code sans le lire
    plt.title("Cercle des corrélations (F{} et F{})".format(x+1, y+1))

    # Le cercle
    an = np.linspace(0, 2 * np.pi, 100)
    plt.plot(np.cos(an), np.sin(an))
    plt.plot(0.5*np.cos(an), 0.5*np.sin(an), linestyle='--', color='green')
```

```
    # Axes et display
    plt.axis('equal')

    # Enregistrement figure
    plt.savefig(fig_name)

    plt.show(block=False)

######
```

[54]:
```
correlation_graph(pca, x_y=(0,1), features=list(df_pcs.columns),␣
↪fig_name="Factoriel_1st.png")
```

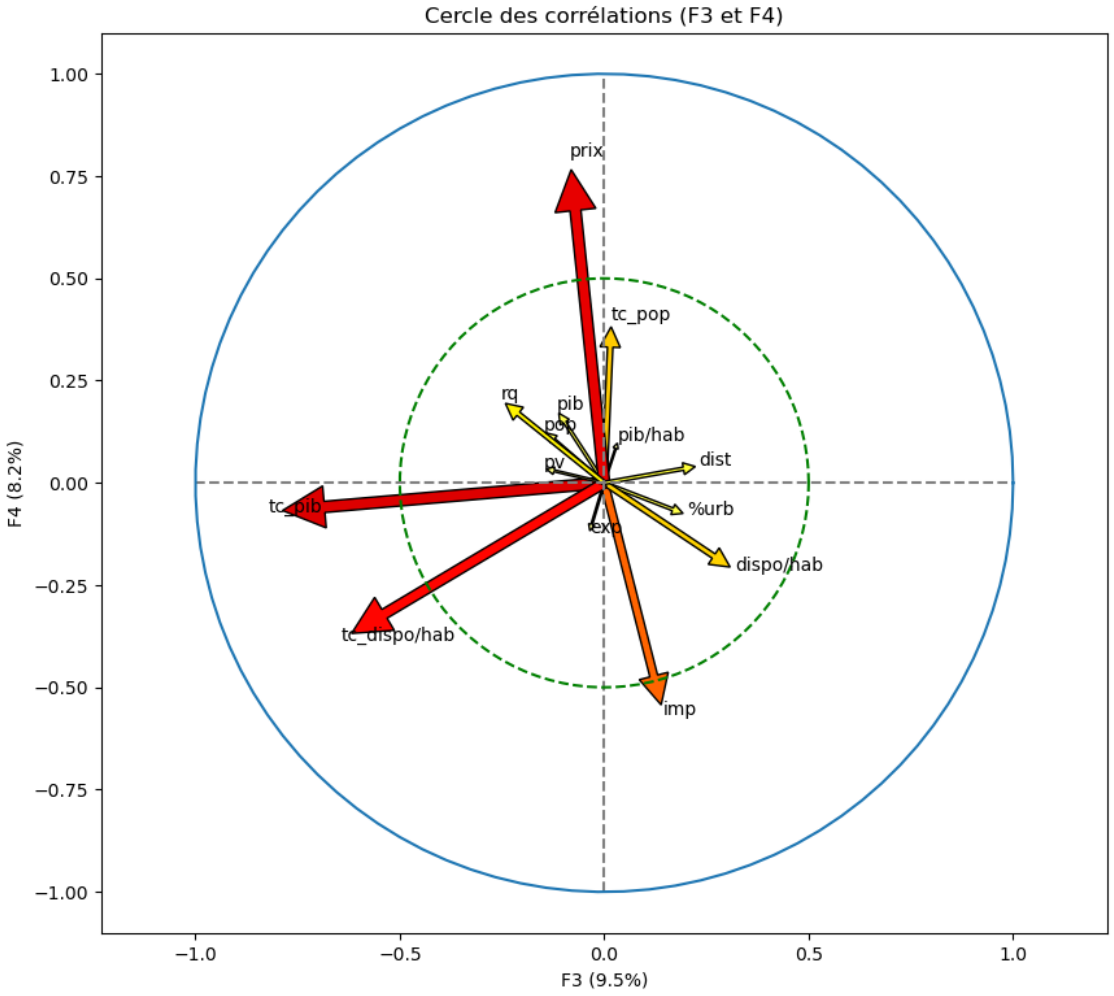
```
Norme vecteur pop dans cercle des corrélations : 0.572
Norme vecteur tc_pop dans cercle des corrélations : 0.27
Norme vecteur %urb dans cercle des corrélations : 0.356
Norme vecteur pib dans cercle des corrélations : 0.528
Norme vecteur tc_pib dans cercle des corrélations : 0.139
Norme vecteur pib/hab dans cercle des corrélations : 0.443
Norme vecteur imp dans cercle des corrélations : 0.311
Norme vecteur exp dans cercle des corrélations : 0.445
Norme vecteur dispo/hab dans cercle des corrélations : 0.399
Norme vecteur tc_dispo/hab dans cercle des corrélations : 0.063
Norme vecteur prix dans cercle des corrélations : 0.245
Norme vecteur pv dans cercle des corrélations : 0.479
Norme vecteur rq dans cercle des corrélations : 0.4
Norme vecteur dist dans cercle des corrélations : 0.267
```



[55]:
```
correlation_graph(pca, x_y=(2,3), features=list(df_pcs.columns),␣
↪fig_name="factoriel_2nd.png")
```

```
Norme vecteur pop dans cercle des corrélations : 0.162
Norme vecteur tc_pop dans cercle des corrélations : 0.332
Norme vecteur %urb dans cercle des corrélations : 0.18
Norme vecteur pib dans cercle des corrélations : 0.177
Norme vecteur tc_pib dans cercle des corrélations : 0.687
Norme vecteur pib/hab dans cercle des corrélations : 0.089
Norme vecteur imp dans cercle des corrélations : 0.487
Norme vecteur exp dans cercle des corrélations : 0.106
Norme vecteur dispo/hab dans cercle des corrélations : 0.322
Norme vecteur tc_dispo/hab dans cercle des corrélations : 0.626
Norme vecteur prix dans cercle des corrélations : 0.67
Norme vecteur pv dans cercle des corrélations : 0.127
Norme vecteur rq dans cercle des corrélations : 0.27
```

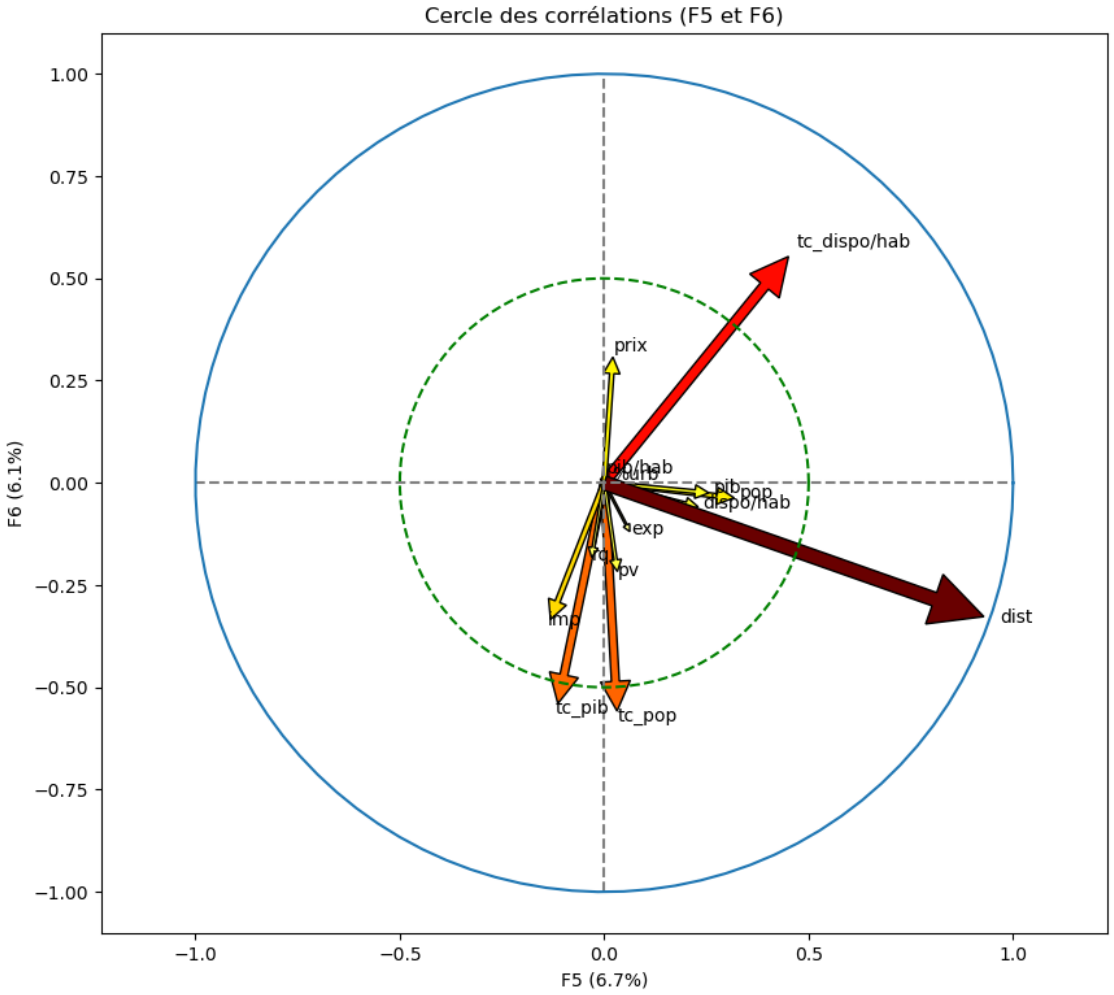
```
Norme vecteur dist dans cercle des corrélations : 0.197
```



[56]:
```
correlation_graph(pca, x_y=(4,5), features=list(df_pcs.columns),␣
↪fig_name="factoriel_3rd.png")
```

```
Norme vecteur pop dans cercle des corrélations : 0.278
Norme vecteur tc_pop dans cercle des corrélations : 0.486
Norme vecteur %urb dans cercle des corrélations : 0.018
Norme vecteur pib dans cercle des corrélations : 0.223
Norme vecteur tc_pib dans cercle des corrélations : 0.48
Norme vecteur pib/hab dans cercle des corrélations : 0.021
Norme vecteur imp dans cercle des corrélations : 0.312
Norme vecteur exp dans cercle des corrélations : 0.117
Norme vecteur dispo/hab dans cercle des corrélations : 0.206
Norme vecteur tc_dispo/hab dans cercle des corrélations : 0.621
Norme vecteur prix dans cercle des corrélations : 0.268
Norme vecteur pv dans cercle des corrélations : 0.191
```

```
Norme vecteur rq dans cercle des corrélations : 0.16
Norme vecteur dist dans cercle des corrélations : 0.856
```



3.3 - Projection du nuage de points des pays dans le sous-espace des axes principaux d'inertie

[57]:
```
data_proj = pd.DataFrame(pca.transform(data_scaled), columns=["F"+str(i+1) for
↪i in range(df_pcs.shape[0])])
# On ne conserve que les 6 premières composantes
data_projr = data_proj.iloc[:,:6]
display(data_projr)
```

	F1	F2	F3	F4	F5	F6
0	3.369691	-2.462420	2.366826	0.871522	-1.067739	-0.470384
1	0.423112	-1.716547	2.830295	1.715752	-0.202786	-0.483922
2	-0.055927	0.703596	-1.321501	-1.766448	-1.152531	0.672394
3	-3.018967	-0.460972	0.609357	-0.199513	0.278353	-1.612130
4	-1.634342	-0.919605	1.324582	-0.003787	2.108809	0.560164

```
..         …         …         …         …         …         …
182  0.783387  3.295666  1.297701 -1.362977  1.182258 -0.931044
183  3.056081 -2.558989  3.346755  0.241144 -0.200593  1.158555
184 -0.838738 -1.656736  1.264758 -1.459882  1.085064 -1.383716
185  1.470314 -0.850923  0.119856  0.433334  0.366158 -0.563950
186  3.127805 -0.164400 -0.266938 -1.097688  0.336000 -0.140417

[187 rows x 6 columns]
```

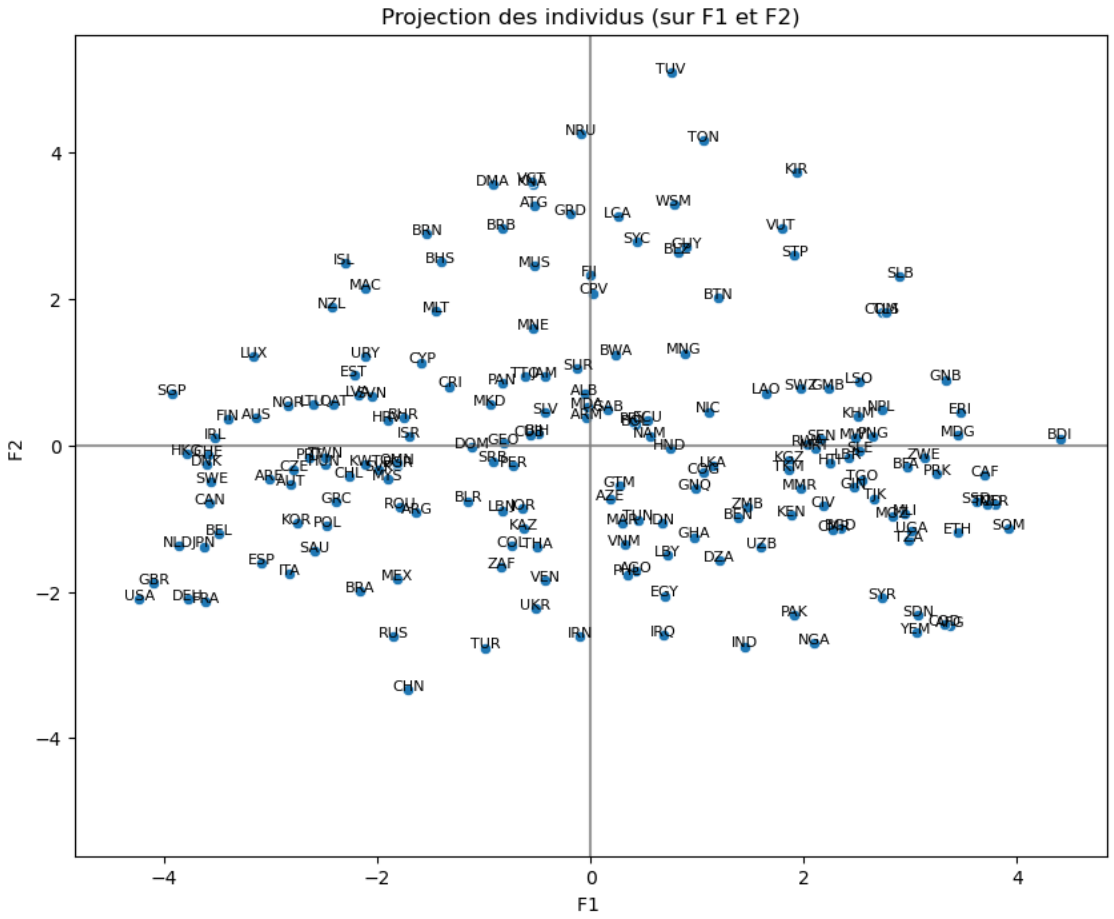
[58]:
```python
def display_factorial_planes(   X_projected,
                                x_y,
                                pca=None,
                                labels = None,
                                clusters=None,
                                palette=cm['rainbow'],
                                alpha=1,
                                figsize=[10,8],
                                marker="o",
                                name=""):
    """
    Affiche la projection des individus

    Positional arguments :
    -----------------------------------
    X_projected : np.array, pd.DataFrame, list of list : la matrice des points␣
↪projetés
    x_y : list ou tuple : le couple x,y des plans à afficher, exemple [0,1]␣
↪pour F1, F2

    Optional arguments :
    -----------------------------------
    pca : sklearn.decomposition.PCA : un objet PCA qui a été fit, cela nous␣
↪permettra d'afficher la variance de chaque composante, default = None
    labels : list ou tuple : les labels des individus à projeter, default = None
    clusters : list ou tuple : la liste des clusters auquel appartient chaque␣
↪individu, default = None
    palette : colormap : palette de couleurs pour afficher les points selon␣
↪leur cluster, default = cm['rainbow']
    alpha : float in [0,1] : paramètre de transparence, 0=100% transparent,␣
↪1=0% transparent, default = 1
    figsize : list ou tuple : couple width, height qui définit la taille de la␣
↪figure en inches, default = [10,8]
    marker : str : le type de marker utilisé pour représenter les individus,␣
↪points croix etc etc, default = ""
    name : str : nom du fichier sous lequel sera enregistré le graphique,␣
↪default = None
    """
```

```
    # Transforme X_projected en np.array
    X_ = np.array(X_projected)

    # On définit la forme de la figure si elle n'a pas été donnée
    if not figsize:
        figsize = (7,6)

    # On gère les labels
    if  labels is None :
        labels = []
    try :
        len(labels)
    except Exception as e :
        raise e

    # On vérifie la variable axis
    if not len(x_y) ==2 :
        raise AttributeError("2 axes sont demandés")
    if max(x_y )>= X_.shape[1] :
        raise AttributeError("la variable axis n'est pas bonne")

    # on définit x et y
    x, y = x_y

    # Initialisation de la figure
    fig, ax = plt.subplots(1, 1, figsize=figsize)

    # On vérifie s'il y a des clusters ou non
    c = None if clusters is None else clusters

    # Les points
    # plt.scatter(   X_[:, x], X_[:, y], alpha=alpha,
    #                     c=c, cmap="Set1", marker=marker)
    if len(labels) : # SI des labels sont affichés, pas de couleur de contour
        sb.scatterplot(data=None, x=X_[:, x], y=X_[:, y],\
                       hue=c, marker=marker, palette=palette)
    else :
        sb.scatterplot(data=None, x=X_[:, x], y=X_[:, y],\
                       hue=c, marker=marker, palette=palette,␣
↪edgecolors='black')

    # Si la variable pca a été fournie, on peut calculer le % de variance de␣
↪chaque axe
    if pca :
        v1 = str(round(100*pca.explained_variance_ratio_[x]))  + " %"
        v2 = str(round(100*pca.explained_variance_ratio_[y]))  + " %"
```

```
    else :
        v1=v2= ''

    # Nom des axes, avec le pourcentage d'inertie expliqué
    ax.set_xlabel(f'F{x+1} {v1}')
    ax.set_ylabel(f'F{y+1} {v2}')

    # Valeur x max et y max
    x_max = np.abs(X_[:, x]).max() *1.1
    y_max = np.abs(X_[:, y]).max() *1.1

    # On borne x et y
    ax.set_xlim(left=-x_max, right=x_max)
    ax.set_ylim(bottom= -y_max, top=y_max)

    # Affichage des lignes horizontales et verticales
    plt.plot([-x_max, x_max], [0, 0], color='grey', alpha=0.8)
    plt.plot([0,0], [-y_max, y_max], color='grey', alpha=0.8)

    # Affichage des labels des points
    if len(labels) :
        # Boucle for pour étiqueter les points
        for i,(_x,_y) in enumerate(X_[:,[x,y]]):
            plt.text(_x, _y+0.05, labels[i], fontsize='8',␣
 ↪ha='center',va='center')

    # Titre et display
    plt.title(f"Projection des individus (sur F{x+1} et F{y+1})")
    if len(name) :
        plt.savefig(name)
    plt.show()
```

[59]:
```
x_y = [0,1]
display_factorial_planes(X_projected=data_proj, x_y=x_y,␣
 ↪labels=list(df_work["ISO3"]), name="Pays_Fact1st.png")
```

```
C:\Users\nicol\AppData\Local\Temp\ipykernel_31564\1868735426.py:65: UserWarning:
Ignoring `palette` because no `hue` variable has been assigned.
  sb.scatterplot(data=None, x=X_[:, x], y=X_[:, y],\
```

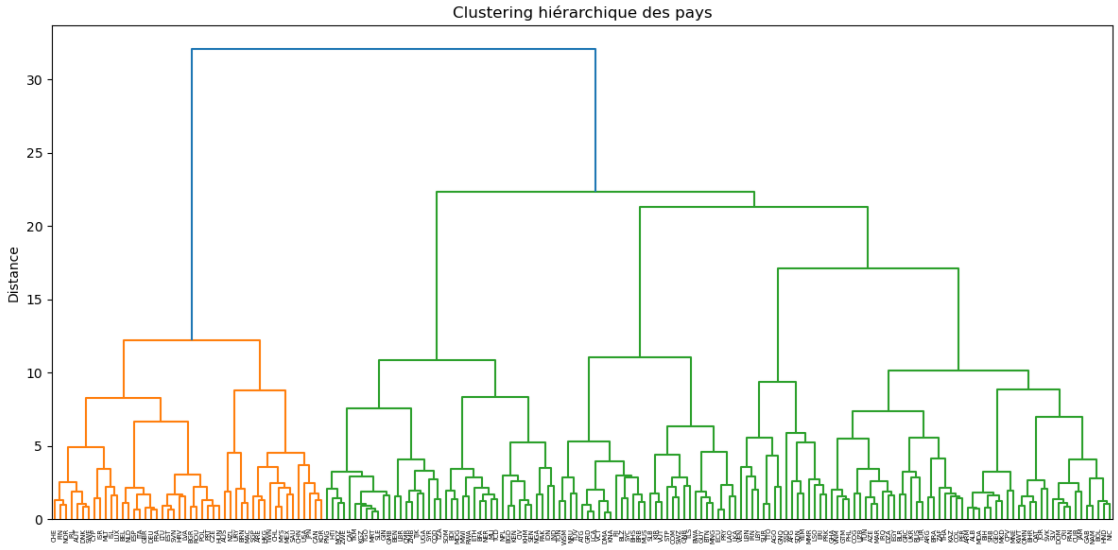


Partie 4 - Clustering

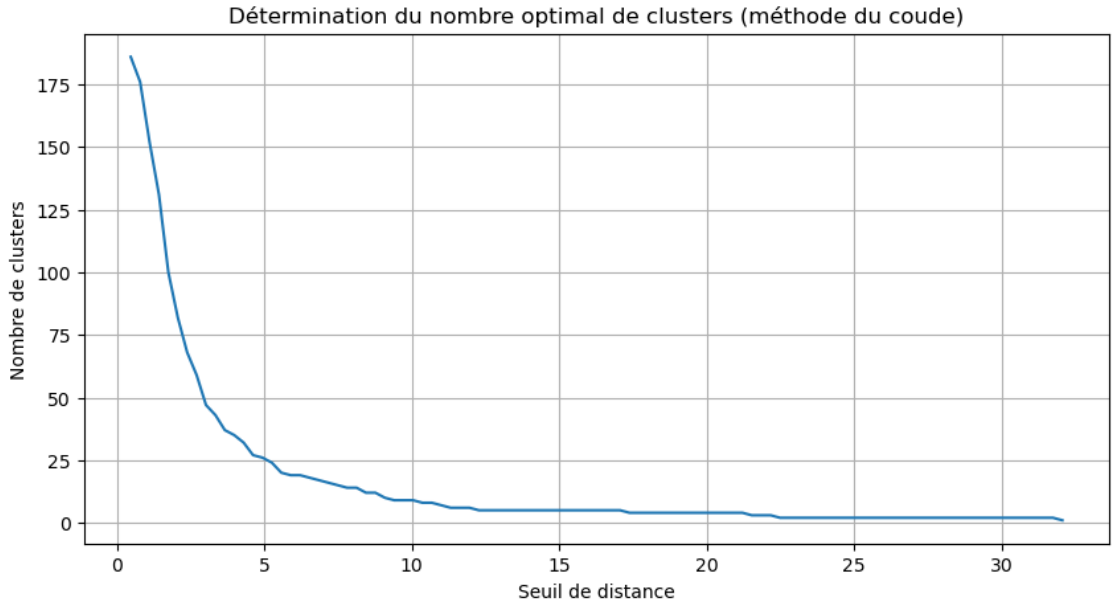
4.1 - Clustering Ascendant Hiérarchique (CAH)

Étant donné qu'on a besoin de 6 dimensions pour décrire correctement nos données, une visualisation graphique des clusters ne sera pas possible.

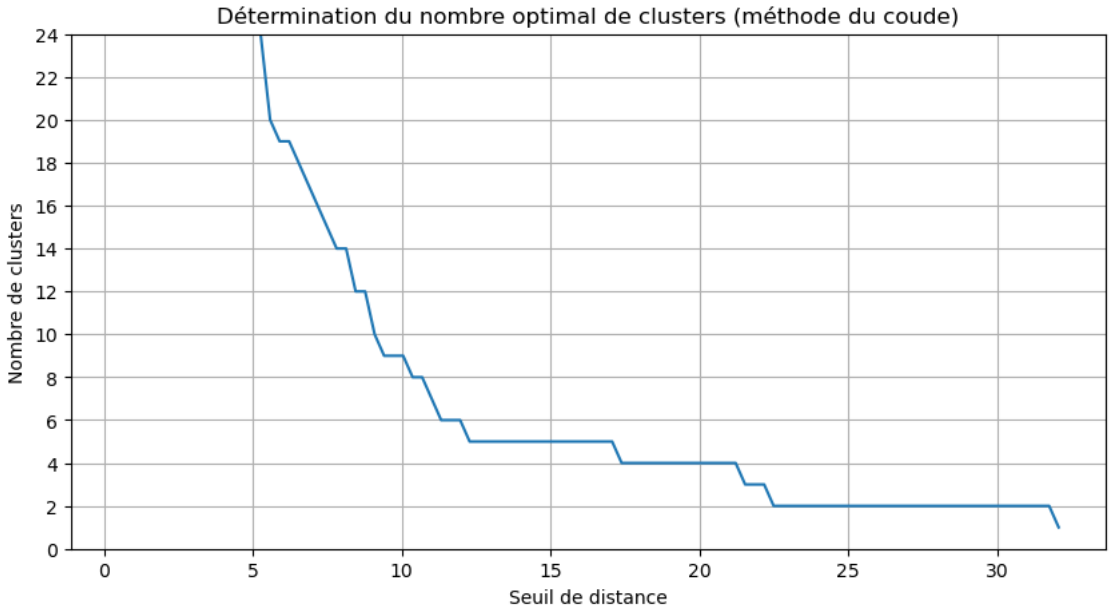
[60]:
```
# Calcul des distances
distance_data = pdist(data_projr, metric='euclidean')
Z = linkage(distance_data, method='ward')

plt.figure(figsize=(12, 6))
dendrogram(Z, labels=list(df_work["ISO3"]), leaf_rotation=90)
plt.title("Clustering hiérarchique des pays")
plt.ylabel("Distance")
plt.tight_layout()
plt.savefig("dendrogramme_pays.png")
plt.show()
```



[61]:
```
distances = Z[:, 2]
# Tri des distances (optionnel mais plus clair pour visualiser)
thresholds = np.linspace(distances.min(), distances.max(), 100)
n_clusters = [len(np.unique(fcluster(Z, t, criterion='distance'))) for t in␣
 ↪thresholds]

# Tracé
plt.figure(figsize=(10, 5))
plt.plot(thresholds, n_clusters)
plt.xlabel("Seuil de distance")
plt.ylabel("Nombre de clusters")
plt.title("Détermination du nombre optimal de clusters (méthode du coude)")
plt.grid(True)
plt.show()
```



[62]:
```
# Tracé
plt.figure(figsize=(10, 5))
plt.plot(thresholds, n_clusters)
plt.ylim([0,20])
plt.yticks(np.linspace(0,24,13))
plt.xlabel("Seuil de distance")
plt.ylabel("Nombre de clusters")
plt.title("Détermination du nombre optimal de clusters (méthode du coude)")
plt.grid(True)
plt.savefig("Coude_nb-clusters.png")
plt.show()
```



[63]:
```
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import fcluster

# Tester plusieurs k
for k in range(2, 40):
    labels = fcluster(Z, k, criterion='maxclust')
    score = silhouette_score(data_projr, labels)
    print(f"Silhouette score pour {k} clusters : {score:.3f}")
```

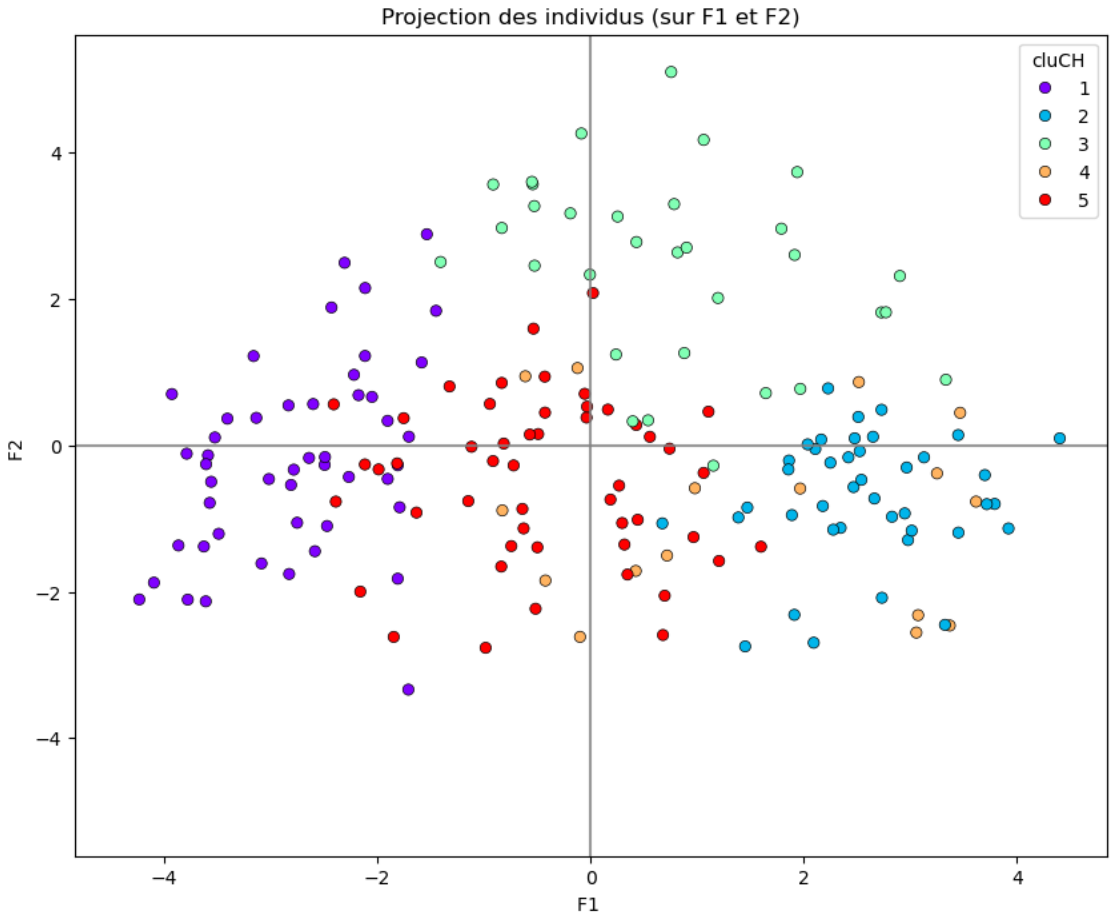
```
Silhouette score pour 2 clusters : 0.221
Silhouette score pour 3 clusters : 0.159
Silhouette score pour 4 clusters : 0.188
Silhouette score pour 5 clusters : 0.192
Silhouette score pour 6 clusters : 0.195
Silhouette score pour 7 clusters : 0.169
Silhouette score pour 8 clusters : 0.170
Silhouette score pour 9 clusters : 0.170
Silhouette score pour 10 clusters : 0.179
Silhouette score pour 11 clusters : 0.178
Silhouette score pour 12 clusters : 0.182
Silhouette score pour 13 clusters : 0.184
Silhouette score pour 14 clusters : 0.187
Silhouette score pour 15 clusters : 0.186
Silhouette score pour 16 clusters : 0.193
Silhouette score pour 17 clusters : 0.195
Silhouette score pour 18 clusters : 0.197
Silhouette score pour 19 clusters : 0.204
```

```
Silhouette score pour 20 clusters : 0.205
Silhouette score pour 21 clusters : 0.209
Silhouette score pour 22 clusters : 0.214
Silhouette score pour 23 clusters : 0.222
Silhouette score pour 24 clusters : 0.218
Silhouette score pour 25 clusters : 0.217
Silhouette score pour 26 clusters : 0.222
Silhouette score pour 27 clusters : 0.226
Silhouette score pour 28 clusters : 0.229
Silhouette score pour 29 clusters : 0.229
Silhouette score pour 30 clusters : 0.231
Silhouette score pour 31 clusters : 0.233
Silhouette score pour 32 clusters : 0.236
Silhouette score pour 33 clusters : 0.237
Silhouette score pour 34 clusters : 0.237
Silhouette score pour 35 clusters : 0.241
Silhouette score pour 36 clusters : 0.241
Silhouette score pour 37 clusters : 0.242
Silhouette score pour 38 clusters : 0.244
Silhouette score pour 39 clusters : 0.244
```

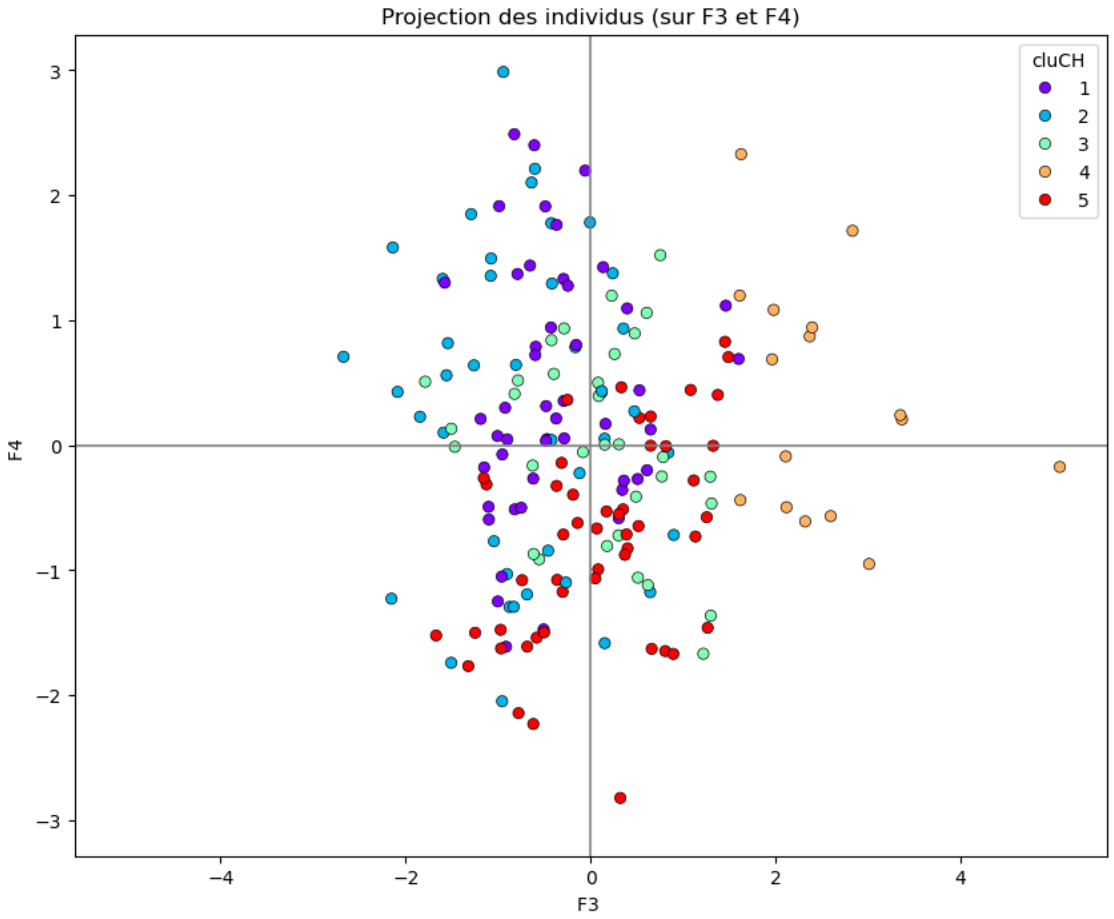
Il semble y avoir un bon compromis (score de silhouette/nombre de clusters) pour 5 clusters.

[64]:
```
#On récupère 8 clusters
clusters = fcluster(Z, 5, criterion='maxclust')
data_proj['cluCH'] = clusters
```

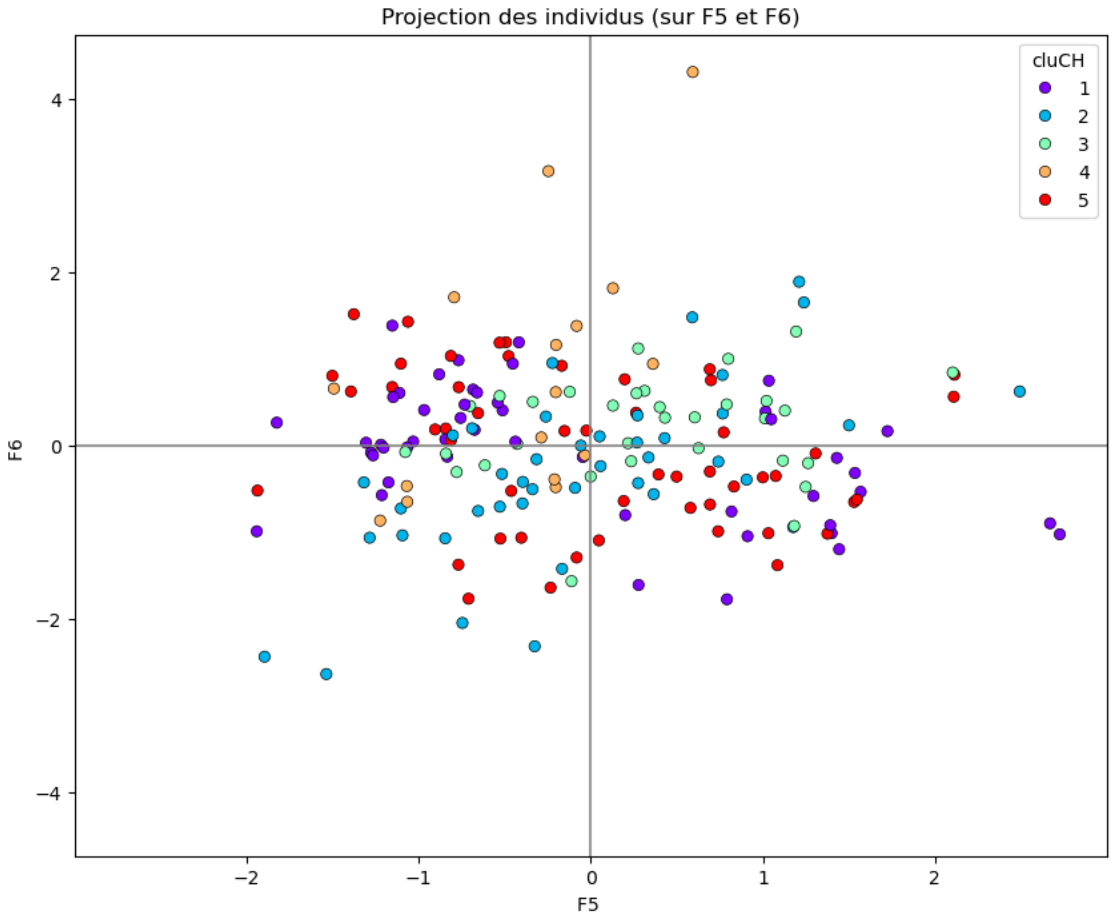
[65]:
```
display_factorial_planes(X_projected=data_proj, x_y=[0,1],␣
 ↪clusters=data_proj['cluCH'], name="ClustersCAH_1st.png")
```



[66]:
```
display_factorial_planes(X_projected=data_proj, x_y=[2,3],␣
↪clusters=data_proj['cluCH'], name="ClustersCAH_2nd.png")
```



[67]:
```
display_factorial_planes(X_projected=data_proj, x_y=[4,5],␣
↪clusters=data_proj['cluCH'])
```



[68]:
```
df_work["cluCH"]=clusters
display(df_work.groupby("cluCH")["Zone"].count())
```

```
cluCH
1    48
2    41
3    32
4    16
5    50
Name: Zone, dtype: int64
```

[69]:
```
for i in df_work["cluCH"].unique():
    df_worki = df_work.loc[df_work["cluCH"]==i]
    display(df_worki[["cluCH","ISO3","Zone"]])
    display(df_worki.describe())
```

	cluCH	ISO3	Zone
0	4	AFG	Afghanistan
1	4	AGO	Angola

```
50       4  ERI                                      Érythrée
64       4  GNQ                            Guinée équatoriale
77       4  IRN                Iran (République islamique d')
94       4  LBN                                        Liban
96       4  LBY                                        Libye
99       4  LSO                                      Lesotho
111      4  MMR                                      Myanmar
135      4  PRK  République populaire démocratique de Corée
143      4  SDN                                        Soudan
151      4  SSD                                  Soudan du Sud
153      4  SUR                                      Suriname
167      4  TTO                              Trinité-et-Tobago
179      4  VEN        Venezuela (République bolivarienne du)
183      4  YEM                                        Yémen
```

	Unnamed: 0	pop	tc_pop	%urb	pib	\
count	16.000000	1.600000e+01	16.000000	16.000000	1.600000e+01	
mean	106.437500	2.327489e+07	2.639375	54.221875	6.720813e+10	
std	57.584684	2.396016e+07	2.745208	23.085981	1.078486e+11	
min	0.000000	5.995486e+05	-1.860000	23.600000	2.147574e+09	
25%	73.750000	2.966584e+06	1.192500	31.460000	1.192514e+10	
50%	105.000000	1.837137e+07	2.350000	60.040000	2.212410e+10	
75%	151.500000	3.482478e+07	5.532500	66.357500	6.305317e+10	
max	183.000000	8.571424e+07	6.760000	97.150000	4.137097e+11	

	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	16.000000	16.000000	16.000000	16.000000	16.000000	
mean	-10.413125	4436.552500	42303.170625	3086.321875	16.006250	
std	10.141472	4687.494848	68162.427489	10266.708875	15.654664	
min	-27.960000	499.650000	30.270000	0.000000	0.440000	
25%	-15.285000	659.420000	1741.797500	2.070000	3.232500	
50%	-8.335000	2414.520000	18841.410000	34.540000	11.120000	
75%	-2.062500	7143.502500	42792.450000	514.512500	23.125000	
max	3.670000	17386.600000	265843.720000	41362.090000	51.570000	

	tc_dispo/hab	prix	pv	rq	dist	cluCH
count	16.00000	16.000000	16.000000	16.000000	16.00000	16.0
mean	-6.93375	1485.193750	-1.205625	-1.356250	5993.99375	4.0
std	12.38008	517.034538	0.994049	0.683773	2046.55523	0.0
min	-25.76000	960.300000	-2.680000	-2.370000	2007.70000	4.0
25%	-13.04500	1172.000000	-2.125000	-1.927500	4898.00000	4.0
50%	-7.85500	1367.500000	-1.270000	-1.490000	5608.55000	4.0
75%	0.62500	1525.000000	-0.392500	-0.805000	7323.70000	4.0
max	22.90000	2693.000000	0.240000	-0.040000	9054.10000	4.0

```
     cluCH ISO3                      Zone
2        5  ALB                   Albanie
4        5  ARG                 Argentine
5        5  ARM                   Arménie
```

9	5	AZE	Azerbaïdjan
16	5	BHR	Bahreïn
18	5	BIH	Bosnie-Herzégovine
19	5	BLR	Bélarus
21	5	BOL	Bolivie (État plurinational de)
22	5	BRA	Brésil
35	5	COG	Congo
36	5	COL	Colombie
38	5	CPV	Cabo Verde
39	5	CRI	Costa Rica
40	5	CUB	Cuba
46	5	DOM	République dominicaine
47	5	DZA	Algérie
49	5	EGY	Égypte
57	5	GAB	Gabon
59	5	GEO	Géorgie
60	5	GHA	Ghana
65	5	GRC	Grèce
67	5	GTM	Guatemala
70	5	HND	Honduras
78	5	IRQ	Iraq
82	5	JAM	Jamaïque
83	5	JOR	Jordanie
85	5	KAZ	Kazakhstan
92	5	KWT	Koweït
104	5	MAR	Maroc
105	5	MDA	République de Moldova
108	5	MKD	Macédoine du Nord
112	5	MNE	Monténégro
119	5	NAM	Namibie
122	5	NIC	Nicaragua
128	5	OMN	Oman
130	5	PAN	Panama
131	5	PER	Pérou
132	5	PHL	Philippines
138	5	QAT	Qatar
140	5	RUS	Fédération de Russie
148	5	SLV	El Salvador
150	5	SRB	Serbie
154	5	SVK	Slovaquie
162	5	THA	Thaïlande
168	5	TUN	Tunisie
169	5	TUR	Türkiye
174	5	UKR	Ukraine
177	5	UZB	Ouzbékistan
180	5	VNM	Viet Nam
184	5	ZAF	Afrique du Sud

	Unnamed: 0	pop	tc_pop	%urb	pib	\
count	50.000000	5.000000e+01	50.000000	50.000000	5.000000e+01	
mean	87.580000	2.872504e+07	1.874800	68.018600	2.049796e+11	
std	55.739588	4.188206e+07	2.343367	15.229427	3.829295e+11	
min	2.000000	5.148465e+05	-2.810000	36.130000	2.081184e+09	
25%	39.250000	3.885810e+06	0.165000	55.355000	2.151964e+10	
50%	82.500000	1.010160e+07	2.155000	67.815000	6.922151e+10	
75%	131.750000	3.915159e+07	3.615000	79.210000	2.080496e+11	
max	184.000000	2.055801e+08	6.260000	100.090000	1.970029e+12	

	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	50.00000	50.000000	50.000000	5.000000e+01	50.000000	
mean	5.69980	9122.092400	72265.932600	1.149670e+05	24.761800	
std	4.99842	10910.433735	93444.333849	5.675159e+05	12.768585	
min	-4.29000	2106.780000	5.170000	2.410000e+00	3.730000	
25%	1.58250	3904.582500	11374.705000	1.951400e+02	14.717500	
50%	5.89500	5897.045000	32521.840000	1.244350e+03	21.180000	
75%	8.76750	9846.080000	99211.162500	1.132862e+04	32.887500	
max	17.17000	70276.490000	399469.310000	3.996504e+06	52.010000	

	tc_dispo/hab	prix	pv	rq	dist	cluCH
count	50.000000	50.000000	50.00000	50.000000	50.000000	50.0
mean	4.912000	1241.164000	-0.27240	-0.144800	5413.536000	5.0
std	6.507103	403.175588	0.63404	0.573924	3202.410281	0.0
min	-9.930000	502.500000	-2.38000	-1.430000	1094.600000	5.0
25%	0.832500	976.875000	-0.51750	-0.457500	2137.975000	5.0
50%	4.690000	1111.000000	-0.32500	-0.005000	4912.650000	5.0
75%	7.185000	1489.500000	0.08000	0.210000	8784.575000	5.0
max	24.910000	2287.000000	0.86000	0.920000	11072.200000	5.0

	cluCH	ISO3	Zone
3	1	ARE	Émirats arabes unis
7	1	AUS	Australie
8	1	AUT	Autriche
11	1	BEL	Belgique
15	1	BGR	Bulgarie
24	1	BRN	Brunéi Darussalam
28	1	CAN	Canada
29	1	CHE	Suisse
30	1	CHL	Chili
31	1	CHN	Chine (continentale)
41	1	CYP	Chypre
42	1	CZE	Tchéquie
43	1	DEU	Allemagne
45	1	DNK	Danemark
51	1	ESP	Espagne
52	1	EST	Estonie
54	1	FIN	Finlande

```
56       1  FRA                                                France
58       1  GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…
69       1  HKG                            Chine - RAS de Hong-Kong
71       1  HRV                                             Croatie
73       1  HUN                                             Hongrie
76       1  IRL                                             Irlande
79       1  ISL                                             Islande
80       1  ISR                                              Israël
81       1  ITA                                              Italie
84       1  JPN                                               Japon
91       1  KOR                                 République de Corée
100      1  LTU                                           Lituanie
101      1  LUX                                        Luxembourg
102      1  LVA                                           Lettonie
103      1  MAC                                 Chine - RAS de Macao
107      1  MEX                                           Mexique
110      1  MLT                                             Malte
118      1  MYS                                           Malaisie
123      1  NLD                               Pays-Bas (Royaume des)
124      1  NOR                                           Norvège
127      1  NZL                                 Nouvelle-Zélande
134      1  POL                                           Pologne
136      1  PRT                                          Portugal
139      1  ROU                                          Roumanie
142      1  SAU                                   Arabie saoudite
145      1  SGP                                         Singapour
155      1  SVN                                          Slovénie
156      1  SWE                                             Suède
171      1  TWN                                   Chine - Taïwan
175      1  URY                                          Uruguay
176      1  USA                                 États-Unis d'Amérique

       Unnamed: 0           pop     tc_pop        %urb           pib  \
count   48.000000  4.800000e+01  48.000000   48.000000  4.800000e+01
mean    82.833333  5.762156e+07   1.282500   78.896458  1.449925e+12
std     49.283519  2.066420e+08   1.617065   12.661556  3.601530e+12
min      3.000000  3.529255e+05  -1.890000   54.130000  1.422809e+10
25%     42.750000  4.659692e+06   0.350000   69.207500  7.109680e+10
50%     79.500000  9.518880e+06   1.035000   80.705000  3.841705e+11
75%    123.250000  3.390131e+07   2.297500   87.712500  1.027867e+12
max    176.000000  1.411122e+09   5.180000  103.100000  2.110548e+13

          tc_pib        pib/hab            imp           exp  dispo/hab  \
count  48.000000      48.000000      48.000000  4.800000e+01  48.000000
mean    6.409167   39804.142917  159853.651458  1.724619e+05  28.386458
std     4.769171   23872.661171  228427.099150  5.184787e+05  12.830398
min    -9.300000    9709.000000       0.000000  0.000000e+00  10.140000
25%     3.937500   20090.080000   13868.977500  5.838305e+03  18.702500
```

```
50%     5.405000   36212.205000   53385.385000  2.376283e+04  26.925000
75%     9.357500   50774.867500  162317.150000  9.627045e+04  34.157500
max    16.940000  118168.180000  856599.910000  3.409222e+06  62.160000
```

```
       tc_dispo/hab         prix         pv         rq          dist  cluCH
count     48.000000    48.000000  48.000000  48.000000     48.000000   48.0
mean       3.303958  2284.652083   0.663542   1.196458   4514.302083    1.0
std        2.883525   806.856338   0.531383   0.552751   4799.723237    0.0
min       -3.520000   933.300000  -1.030000  -0.300000    262.400000    1.0
25%        1.592500  1751.000000   0.427500   0.852500   1049.800000    1.0
50%        3.090000  2088.500000   0.765000   1.210000   1809.900000    1.0
75%        4.547500  2558.250000   1.002500   1.677500   9037.825000    1.0
max       10.640000  4717.000000   1.450000   2.180000  19263.900000    1.0
```

```
     cluCH ISO3                                Zone
6        3  ATG                 Antigua-et-Barbuda
17       3  BHS                            Bahamas
20       3  BLZ                             Belize
23       3  BRB                            Barbade
25       3  BTN                            Bhoutan
26       3  BWA                           Botswana
37       3  COM                            Comores
44       3  DMA                          Dominique
48       3  ECU                           Équateur
55       3  FJI                              Fidji
63       3  GNB                      Guinée-Bissau
66       3  GRD                            Grenade
68       3  GUY                             Guyana
89       3  KIR                           Kiribati
90       3  KNA               Saint-Kitts-et-Nevis
93       3  LAO  République démocratique populaire lao
97       3  LCA                       Sainte-Lucie
98       3  LKA                          Sri Lanka
113      3  MNG                           Mongolie
116      3  MUS                            Maurice
126      3  NRU                              Nauru
137      3  PRY                           Paraguay
146      3  SLB                         Îles Salomon
152      3  STP               Sao Tomé-et-Principe
157      3  SWZ                           Eswatini
158      3  SYC                          Seychelles
165      3  TLS                        Timor-Leste
166      3  TON                              Tonga
170      3  TUV                             Tuvalu
178      3  VCT        Saint-Vincent-et-les Grenadines
181      3  VUT                            Vanuatu
182      3  WSM                              Samoa
```

```
       Unnamed: 0           pop     tc_pop       %urb           pib  \
```

count	32.000000	3.200000e+01	32.000000	32.000000	3.200000e+01
mean	97.250000	2.204013e+06	1.932188	45.524687	1.079715e+10
std	57.173195	4.905311e+06	1.981753	21.452477	2.362652e+10
min	6.000000	1.058909e+04	-2.330000	17.450000	4.957169e+07
25%	47.000000	1.154623e+05	0.537500	28.180000	8.930121e+08
50%	95.000000	3.842084e+05	2.015000	41.695000	1.796940e+09
75%	153.250000	1.278242e+06	3.522500	64.092500	8.216360e+09
max	182.000000	2.221518e+07	5.850000	98.960000	1.047675e+11

	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	32.000000	32.000000	32.000000	32.000000	32.000000	
mean	6.845625	7703.003750	4107.773750	226.327813	33.964687	
std	5.950787	6745.045851	4284.385554	893.733068	26.882902	
min	0.060000	697.710000	3.870000	0.000000	2.070000	
25%	3.397500	3262.347500	549.822500	0.007500	8.010000	
50%	6.005000	5720.375000	2939.310000	4.365000	32.900000	
75%	7.870000	9615.235000	6178.352500	26.330000	52.130000	
max	30.940000	30537.680000	15739.140000	4892.910000	92.210000	

	tc_dispo/hab	prix	pv	rq	dist	cluCH
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.0
mean	6.690312	1621.312500	0.577812	-0.245000	9819.978125	3.0
std	7.061814	403.875834	0.522529	0.561995	3770.245841	0.0
min	-4.340000	785.500000	-0.550000	-1.260000	4440.600000	3.0
25%	1.607500	1375.750000	0.170000	-0.585000	7037.475000	3.0
50%	5.440000	1668.500000	0.790000	-0.285000	8524.400000	3.0
75%	10.430000	1870.250000	0.952500	0.120000	13439.475000	3.0
max	22.610000	2357.000000	1.270000	1.090000	16941.400000	3.0

	cluCH	ISO3	Zone
10	2	BDI	Burundi
12	2	BEN	Bénin
13	2	BFA	Burkina Faso
14	2	BGD	Bangladesh
27	2	CAF	République centrafricaine
32	2	CIV	Côte d'Ivoire
33	2	CMR	Cameroun
34	2	COD	République démocratique du Congo
53	2	ETH	Éthiopie
61	2	GIN	Guinée
62	2	GMB	Gambie
72	2	HTI	Haïti
74	2	IDN	Indonésie
75	2	IND	Inde
86	2	KEN	Kenya
87	2	KGZ	Kirghizistan
88	2	KHM	Cambodge
95	2	LBR	Libéria

```
106      2  MDG                          Madagascar
109      2  MLI                                Mali
114      2  MOZ                          Mozambique
115      2  MRT                          Mauritanie
117      2  MWI                              Malawi
120      2  NER                               Niger
121      2  NGA                             Nigéria
125      2  NPL                               Népal
129      2  PAK                            Pakistan
133      2  PNG          Papouasie-Nouvelle-Guinée
141      2  RWA                             Rwanda
144      2  SEN                             Sénégal
147      2  SLE                        Sierra Leone
149      2  SOM                              Somalie
159      2  SYR          République arabe syrienne
160      2  TCD                               Tchad
161      2  TGO                                Togo
163      2  TJK                         Tadjikistan
164      2  TKM                        Turkménistan
172      2  TZA        République-Unie de Tanzanie
173      2  UGA                             Ouganda
185      2  ZMB                              Zambie
186      2  ZWE                            Zimbabwe

       Unnamed: 0           pop     tc_pop       %urb           pib  \
count   41.000000  4.100000e+01  41.000000  41.000000  4.100000e+01
mean   102.951220  7.442665e+07   4.658049  36.407073  1.383901e+11
std     53.266758  2.170029e+08   1.512611  12.803377  4.467195e+11
min     10.000000  2.400124e+06   1.660000  11.900000  1.720094e+09
25%     62.000000  1.094864e+07   4.080000  26.670000  9.805267e+09
50%    114.000000  1.800695e+07   4.920000  37.240000  1.642475e+10
75%    147.000000  2.914337e+07   5.600000  45.380000  4.464754e+10
max    186.000000  1.371249e+09   7.190000  56.130000  2.683277e+12

          tc_pib      pib/hab           imp          exp  dispo/hab  \
count  41.000000    41.000000     41.000000    41.000000  41.000000
mean    9.979512  1345.633902  13942.784634   452.703902   4.059512
std     4.942439  1108.666387  22232.108308  1200.886410   2.912681
min    -4.660000   274.730000      4.600000     0.000000   0.470000
25%     7.950000   756.290000    181.400000     0.700000   1.470000
50%    10.030000  1038.470000   3803.150000     7.870000   3.400000
75%    12.930000  1574.060000  20903.350000   122.680000   6.260000
max    21.350000  6677.140000  94154.900000  5133.530000  11.680000

       tc_dispo/hab         prix         pv         rq          dist  cluCH
count     41.000000    41.000000  41.000000  41.000000     41.000000   41.0
mean       5.640732  1273.829268  -0.925854  -0.820732   6185.585366    2.0
std       11.458384   603.707235   0.742506   0.483132   2242.677229    0.0
```

```
min      -19.910000   473.700000  -2.750000  -2.060000   3280.700000    2.0
25%       -1.010000   808.200000  -1.340000  -0.960000   4722.100000    2.0
50%        5.670000  1076.000000  -0.790000  -0.750000   5582.500000    2.0
75%       14.130000  1570.000000  -0.390000  -0.550000   7244.000000    2.0
max       36.580000  2687.000000   0.110000   0.070000  14565.800000    2.0
```

[70]:
```
px.choropleth(df_work, locations="ISO3", locationmode="ISO-3",\
              color_continuous_scale=px.colors.sequential.Rainbow,␣
 ↪color='cluCH',\
              width=600, height=400)
```

4.2 - Clustering par l'algorithme K-Means

[71]:
```
inertias = []
n_clust=5
kmeans = KMeans(n_clusters=n_clust, random_state=42)
kmeans.fit(data_projr)
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
```

[71]:
```
KMeans(n_clusters=5, random_state=42)
```

[72]:
```
clusters = kmeans.fit_predict(data_projr)
pd.Series.value_counts(clusters).sort_index()
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
```

[72]:
```
0    34
1    28
2    49
3    46
4    30
Name: count, dtype: int64
```

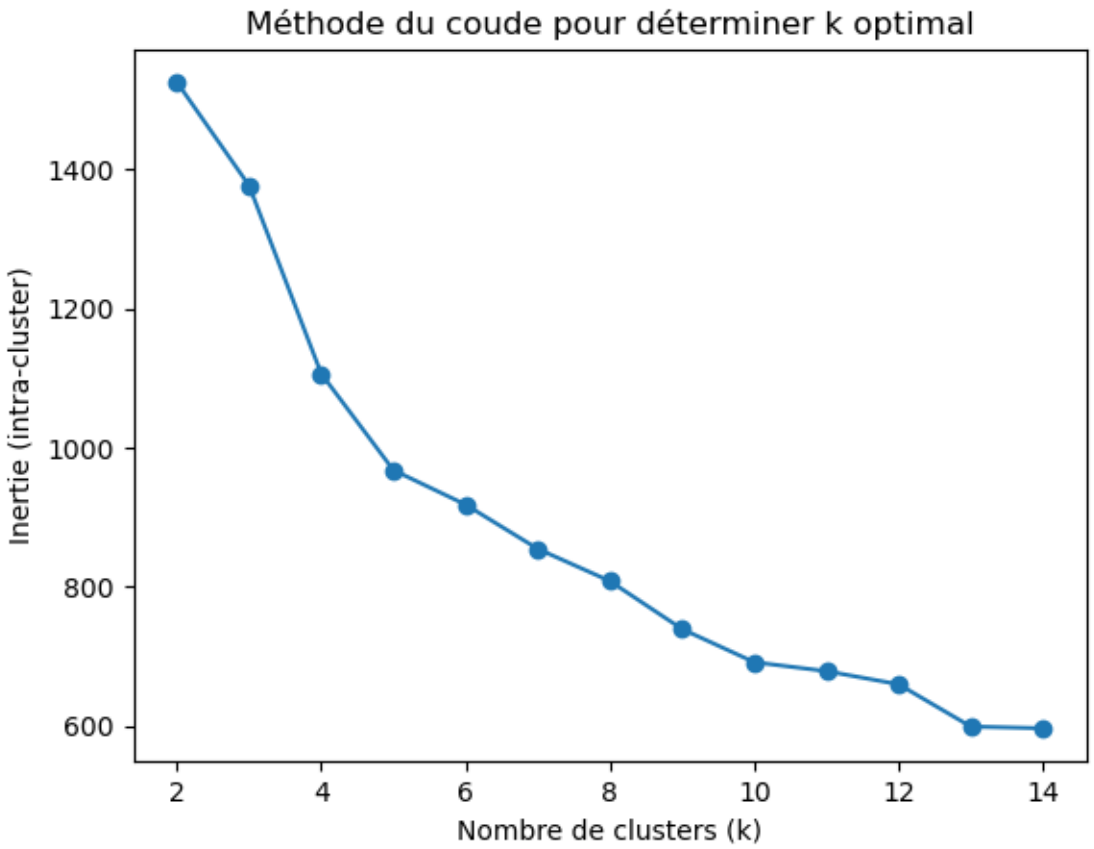
La répartition du nombre de pays par cluster est différente de ce que nous avions obtenu par clustering hiérarchique.

[73]:
```
inertias = []
k_range = range(2, 15)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=808)
    kmeans.fit(data_projr)
    inertias.append(kmeans.inertia_)

plt.plot(k_range, inertias, marker='o')
plt.xlabel('Nombre de clusters (k)')
plt.ylabel('Inertie (intra-cluster)')
plt.title('Méthode du coude pour déterminer k optimal')
plt.show()
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
```

```
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
```

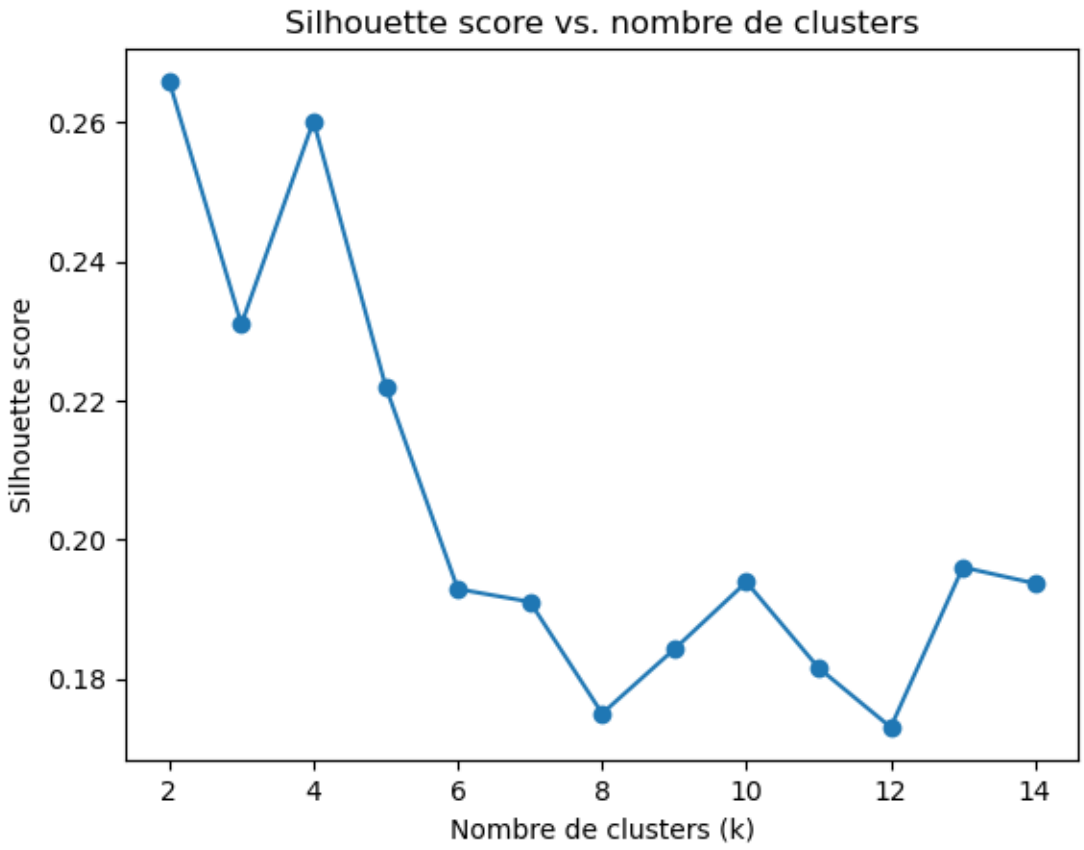


[74]:
```
silhouette_scores = []

for k in range(2, 15):
    kmeans = KMeans(n_clusters=k, random_state=808)
    labels = kmeans.fit_predict(data_projr)
    score = silhouette_score(data_projr, labels)
    silhouette_scores.append(score)

plt.plot(range(2, 15), silhouette_scores, marker='o')
plt.xlabel('Nombre de clusters (k)')
```

```python
plt.ylabel('Silhouette score')
plt.title('Silhouette score vs. nombre de clusters')
plt.show()
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:
```

```
KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
```



L'algorithme K-Means suggère un nombre de clusters autour de 5

[75]:
```
inertias = []
n_clust=5
kmeans = KMeans(n_clusters=n_clust, random_state=808)
kmeans.fit(data_projr)
clusters = kmeans.fit_predict(data_projr)
pd.Series.value_counts(clusters).sort_index()
```

```
C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\nicol\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1446:
UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
```

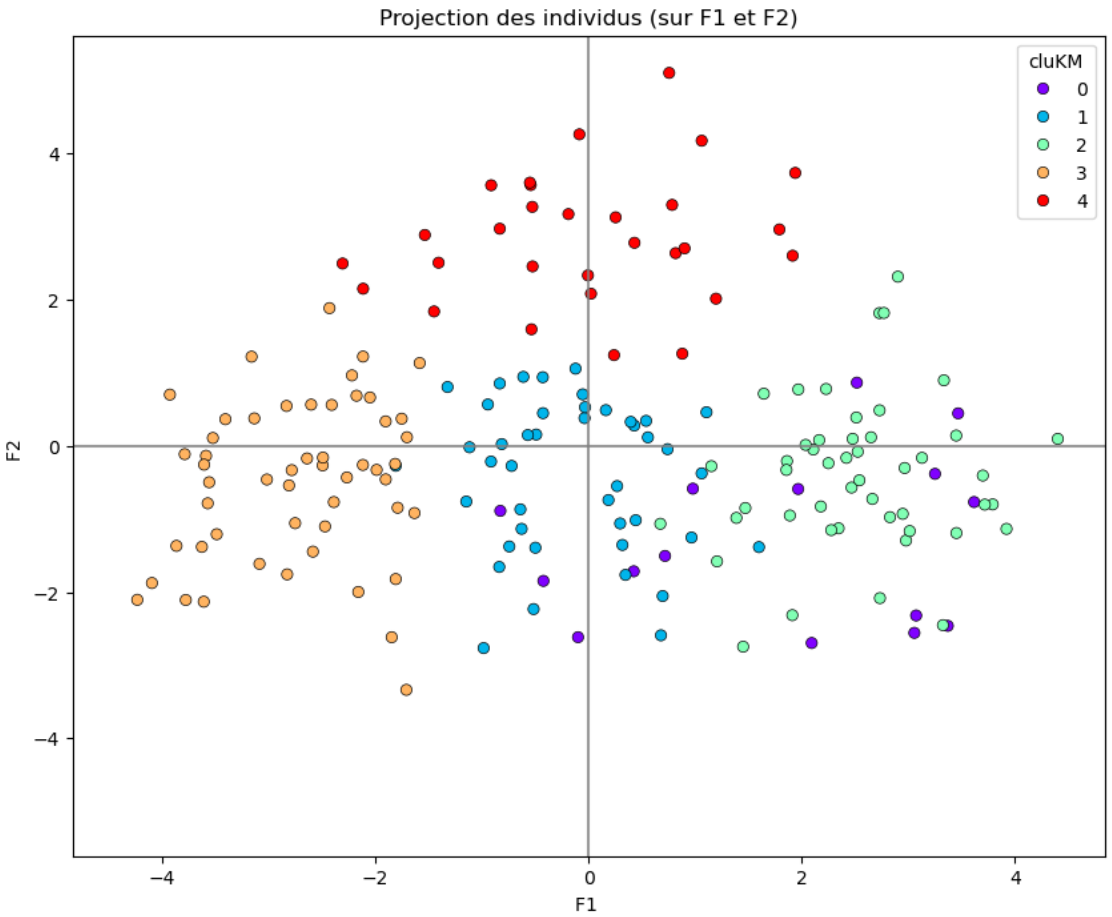
```
variable OMP_NUM_THREADS=1.
```

[75]:
```
0    15
1    43
2    48
3    52
4    29
Name: count, dtype: int64
```

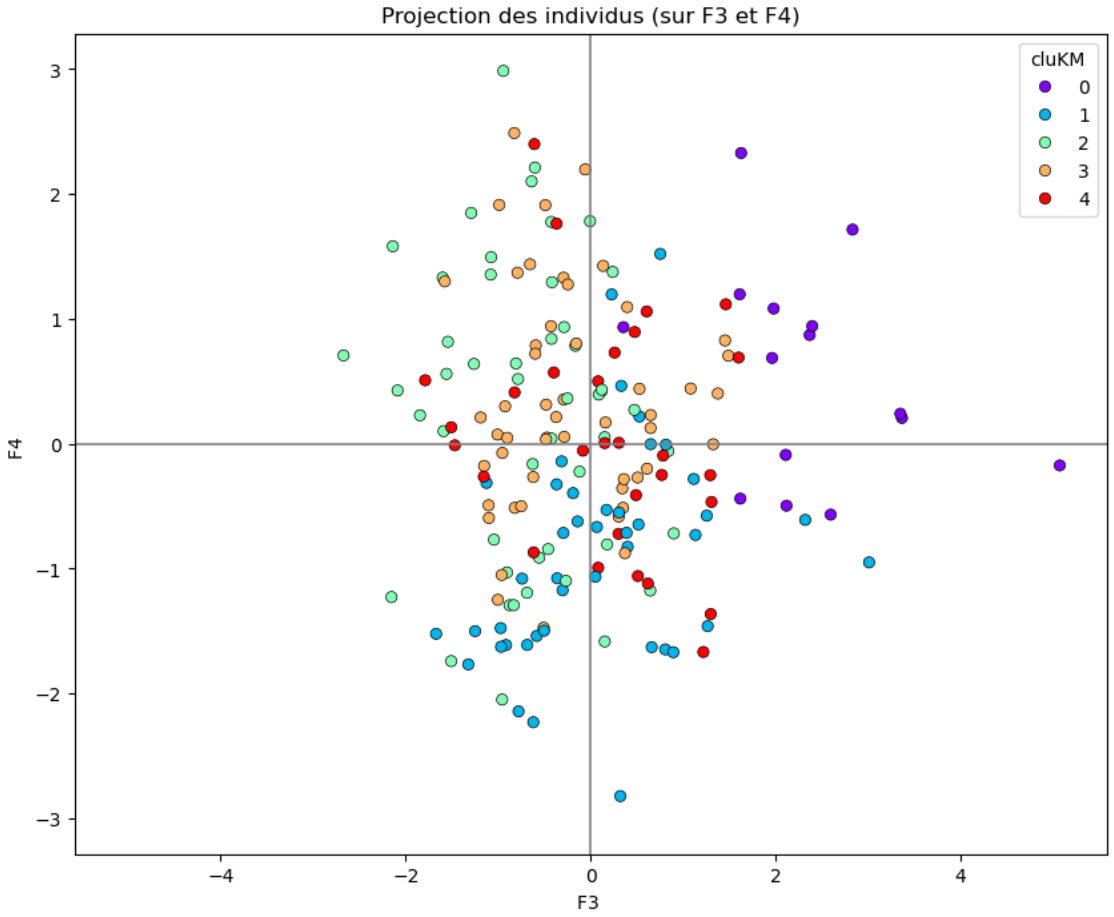
[76]:
```
df_work["cluKM"] = clusters
```

[77]:
```
px.choropleth(df_work, locations="ISO3", locationmode="ISO-3",\
              color_continuous_scale=px.colors.sequential.Rainbow,␣
 ↪color='cluKM',\
              width=600, height=400)
```

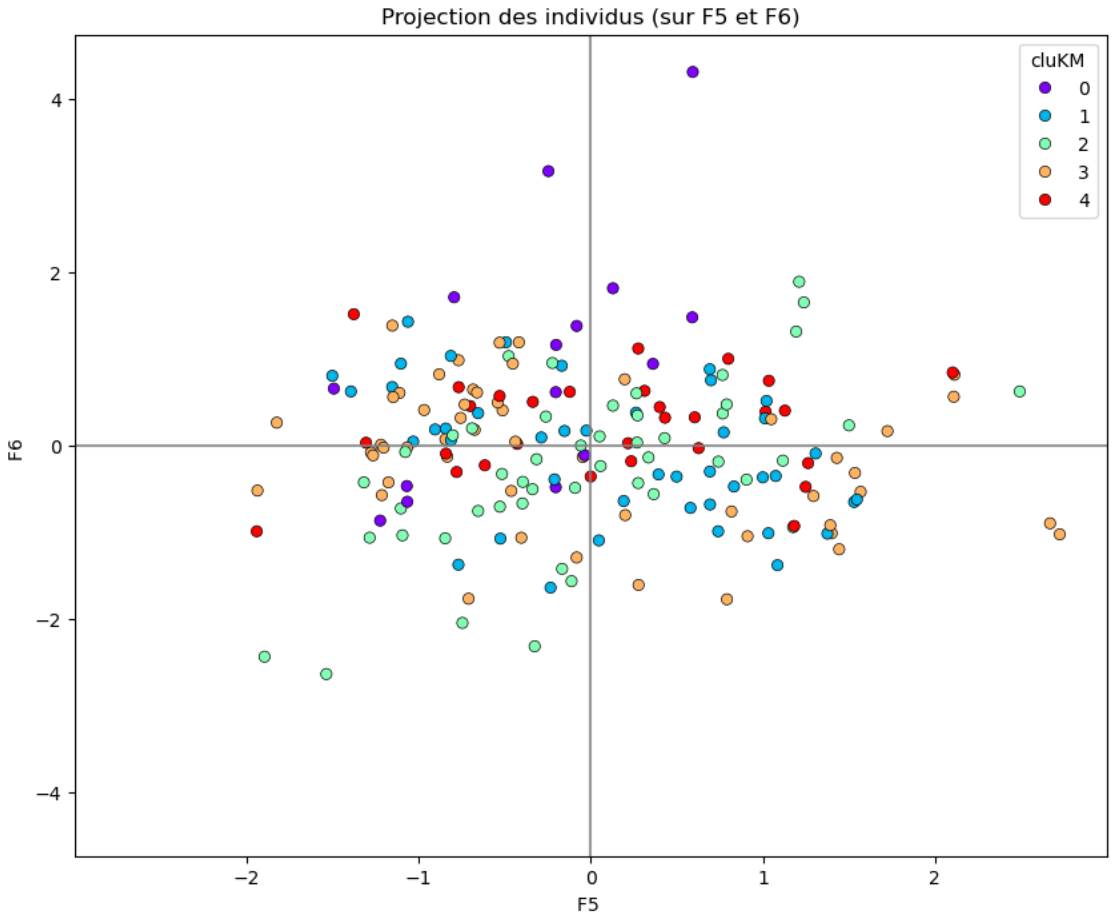
[78]:
```
data_proj['cluKM'] = clusters
display_factorial_planes(X_projected=data_proj, x_y=[0,1],␣
 ↪clusters=data_proj['cluKM'])
```



[79]:
```
display_factorial_planes(X_projected=data_proj, x_y=[2,3],␣
↪clusters=data_proj['cluKM'])
```



[80]:
```
display_factorial_planes(X_projected=data_proj, x_y=[4,5],␣
↪clusters=data_proj['cluKM'])
```



[81]:
```
for i in df_work["cluKM"].unique():
    df_worki = df_work.loc[df_work["cluKM"]==i]
    display(df_worki[["cluKM","ISO3","Zone"]])
    display(df_worki.describe())
```

	cluKM	ISO3	Zone
0	0	AFG	Afghanistan
1	0	AGO	Angola
50	0	ERI	Érythrée
64	0	GNQ	Guinée équatoriale
77	0	IRN	Iran (République islamique d')
94	0	LBN	Liban
96	0	LBY	Libye
99	0	LSO	Lesotho
111	0	MMR	Myanmar
121	0	NGA	Nigéria
135	0	PRK	République populaire démocratique de Corée
143	0	SDN	Soudan
151	0	SSD	Soudan du Sud

```
179      0  VEN      Venezuela (République bolivarienne du)
183      0  YEM                                      Yémen

       Unnamed: 0           pop     tc_pop       %urb           pib  \
count   15.000000  1.500000e+01  15.000000  15.000000  1.500000e+01
mean   100.266667  3.833344e+07   2.898667  53.544000  9.997066e+10
std     55.770533  5.149327e+07   2.847891  23.806593  1.468941e+11
min      0.000000  1.607591e+06  -1.860000  23.600000  2.147574e+09
25%     70.500000  6.404910e+06   1.155000  31.320000  1.522499e+10
50%     99.000000  2.945571e+07   2.480000  58.860000  4.174515e+10
75%    139.000000  4.040943e+07   5.635000  68.055000  8.635549e+10
max    183.000000  2.046637e+08   6.760000  97.150000  4.537867e+11

          tc_pib      pib/hab            imp           exp  dispo/hab  \
count  15.000000    15.000000      15.000000     15.000000  15.000000
mean  -10.709333  3253.844667   42864.737333   3139.752667  10.768667
std    10.368083  3150.804028   70745.196999  10627.178994  10.042980
min   -27.960000   499.650000      30.270000      0.000000   0.440000
25%   -16.880000   653.110000    1630.425000      1.260000   1.620000
50%    -8.390000  1435.090000    9160.140000     15.840000   8.350000
75%    -2.885000  5896.595000   54585.200000     62.880000  19.690000
max     3.670000  8894.240000  265843.720000  41362.090000  29.620000

       tc_dispo/hab         prix         pv         rq         dist  \
count      15.00000    15.000000  15.000000  15.000000    15.000000
mean       -6.16000  1556.726667  -1.446000  -1.460667  5746.913333
std        13.39039   492.388326   0.858951   0.587762  2079.911884
min       -25.76000   982.900000  -2.680000  -2.370000  2007.700000
25%       -13.94000  1300.500000  -2.180000  -1.965000  4667.250000
50%        -9.35000  1376.000000  -1.440000  -1.530000  5317.300000
75%         2.23000  1526.000000  -0.690000  -0.995000  7064.600000
max        22.90000  2693.000000  -0.170000  -0.540000  9054.100000

          cluCH  cluKM
count  15.000000   15.0
mean    3.866667    0.0
std     0.516398    0.0
min     2.000000    0.0
25%     4.000000    0.0
50%     4.000000    0.0
75%     4.000000    0.0
max     4.000000    0.0

     cluKM ISO3                          Zone
2        1  ALB                       Albanie
5        1  ARM                       Arménie
9        1  AZE                    Azerbaïdjan
15       1  BGR                       Bulgarie
18       1  BIH             Bosnie-Herzégovine
```

```
19       1  BLR                          Bélarus
21       1  BOL  Bolivie (État plurinational de)
35       1  COG                            Congo
36       1  COL                         Colombie
39       1  CRI                       Costa Rica
40       1  CUB                             Cuba
46       1  DOM          République dominicaine
48       1  ECU                         Équateur
49       1  EGY                           Égypte
57       1  GAB                             Gabon
59       1  GEO                          Géorgie
60       1  GHA                             Ghana
67       1  GTM                       Guatemala
70       1  HND                        Honduras
78       1  IRQ                            Iraq
82       1  JAM                        Jamaïque
83       1  JOR                        Jordanie
85       1  KAZ                       Kazakhstan
104      1  MAR                            Maroc
105      1  MDA           République de Moldova
108      1  MKD               Macédoine du Nord
119      1  NAM                         Namibie
122      1  NIC                       Nicaragua
130      1  PAN                          Panama
131      1  PER                           Pérou
132      1  PHL                     Philippines
137      1  PRY                        Paraguay
148      1  SLV                     El Salvador
150      1  SRB                           Serbie
153      1  SUR                        Suriname
162      1  THA                       Thaïlande
167      1  TTO               Trinité-et-Tobago
168      1  TUN                         Tunisie
169      1  TUR                         Türkiye
174      1  UKR                         Ukraine
177      1  UZB                      Ouzbékistan
180      1  VNM                        Viet Nam
184      1  ZAF                 Afrique du Sud

       Unnamed: 0           pop     tc_pop       %urb           pib  \
count   43.000000  4.300000e+01  43.000000  43.000000  4.300000e+01
mean    91.697674  2.327066e+07   1.762558  63.550000  1.213613e+11
std     57.403557  2.961828e+07   2.247872  12.172175  1.668035e+11
min      2.000000  5.995486e+05  -2.810000  36.130000  4.193954e+09
25%     43.000000  3.975570e+06   0.200000  54.710000  1.972999e+10
50%     83.000000  9.991382e+06   2.190000  63.400000  6.260971e+10
75%    142.500000  3.217293e+07   3.000000  71.780000  1.347359e+11
max    184.000000  1.091200e+08   6.020000  86.800000  8.722975e+11
```

	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	43.000000	43.000000	43.000000	43.000000	43.000000	
mean	6.019535	6090.316744	68678.858372	31624.191163	24.011395	
std	5.563173	3380.762607	95801.361486	93069.085803	12.560833	
min	-8.280000	2106.780000	3.870000	4.910000	3.730000	
25%	1.435000	3518.565000	11923.920000	182.260000	14.735000	
50%	6.940000	5411.210000	29078.140000	1082.160000	20.120000	
75%	9.715000	6980.360000	75751.950000	6482.170000	30.820000	
max	17.170000	17386.600000	399469.310000	455829.720000	52.010000	

	tc_dispo/hab	prix	pv	rq	dist	\
count	43.000000	43.000000	43.000000	43.000000	43.000000	
mean	5.188140	1112.546512	-0.313256	-0.217674	5885.646512	
std	6.480909	285.697399	0.592628	0.540555	3201.149809	
min	-6.360000	502.500000	-2.380000	-1.430000	1352.500000	
25%	0.710000	960.350000	-0.510000	-0.555000	2735.500000	
50%	4.690000	1061.000000	-0.310000	-0.090000	6041.000000	
75%	7.730000	1324.000000	0.025000	0.100000	8913.050000	
max	24.910000	1866.000000	0.690000	0.920000	10751.600000	

	cluCH	cluKM
count	43.000000	43.0
mean	4.767442	1.0
std	0.750784	0.0
min	1.000000	1.0
25%	5.000000	1.0
50%	5.000000	1.0
75%	5.000000	1.0
max	5.000000	1.0

	cluKM	ISO3	Zone
3	3	ARE	Émirats arabes unis
4	3	ARG	Argentine
7	3	AUS	Australie
8	3	AUT	Autriche
11	3	BEL	Belgique
16	3	BHR	Bahreïn
22	3	BRA	Brésil
28	3	CAN	Canada
29	3	CHE	Suisse
30	3	CHL	Chili
31	3	CHN	Chine (continentale)
41	3	CYP	Chypre
42	3	CZE	Tchéquie
43	3	DEU	Allemagne
45	3	DNK	Danemark
51	3	ESP	Espagne

```
52       3  EST                                                Estonie
54       3  FIN                                               Finlande
56       3  FRA                                                France
58       3  GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…
65       3  GRC                                                Grèce
69       3  HKG                            Chine - RAS de Hong-Kong
71       3  HRV                                              Croatie
73       3  HUN                                              Hongrie
76       3  IRL                                              Irlande
80       3  ISR                                               Israël
81       3  ITA                                               Italie
84       3  JPN                                               Japon
91       3  KOR                                 République de Corée
92       3  KWT                                               Koweït
100      3  LTU                                             Lituanie
101      3  LUX                                          Luxembourg
102      3  LVA                                             Lettonie
107      3  MEX                                              Mexique
118      3  MYS                                             Malaisie
123      3  NLD                               Pays-Bas (Royaume des)
124      3  NOR                                              Norvège
127      3  NZL                                   Nouvelle-Zélande
128      3  OMN                                                Oman
134      3  POL                                              Pologne
136      3  PRT                                             Portugal
138      3  QAT                                                Qatar
139      3  ROU                                             Roumanie
140      3  RUS                                 Fédération de Russie
142      3  SAU                                      Arabie saoudite
145      3  SGP                                            Singapour
154      3  SVK                                            Slovaquie
155      3  SVN                                             Slovénie
156      3  SWE                                                Suède
171      3  TWN                                      Chine - Taïwan
175      3  URY                                             Uruguay
176      3  USA                                 États-Unis d'Amérique

       Unnamed: 0           pop     tc_pop        %urb           pib  \
count   52.000000  5.200000e+01  52.000000   52.000000  5.200000e+01
mean    84.692308  6.118159e+07   1.350769   79.606346  1.432800e+12
std     51.058420  1.998666e+08   1.751628   13.463658  3.461989e+12
min      3.000000  6.060651e+05  -1.890000   53.870000  2.580269e+10
25%     42.750000  4.875216e+06   0.290000   69.207500  1.765612e+11
50%     80.500000  1.025139e+07   1.035000   81.810000  3.935848e+11
75%    129.500000  3.981270e+07   2.297500   89.360000  1.331037e+12
max    176.000000  1.411122e+09   6.260000  103.100000  2.110548e+13

          tc_pib        pib/hab            imp           exp  dispo/hab  \
```

```
count  52.000000      52.000000      52.000000  5.200000e+01  52.000000
mean    5.690577   37060.624423  160104.497308  2.437207e+05  29.081154
std     4.161396   23735.168529  219035.649377  7.277304e+05  12.631250
min    -4.290000    9602.090000       0.000000  3.826000e+01  10.140000
25%     3.165000   19225.840000   26215.262500  7.672762e+03  18.702500
50%     4.960000   31141.955000   62287.670000  2.920534e+04  27.160000
75%     8.512500   48381.305000  162317.150000  1.150771e+05  35.132500
max    16.940000  118168.180000  856599.910000  3.996504e+06  62.100000

       tc_dispo/hab         prix          pv          rq          dist  \
count     52.000000    52.000000   52.000000   52.000000     52.000000
mean       2.714423  2194.025000    0.523654    1.033846   4541.267308
std        3.562595   736.314996    0.585860    0.678150   4605.194804
min       -9.930000   933.300000   -1.030000   -0.600000    262.400000
25%        1.267500  1699.250000    0.277500    0.667500   1075.025000
50%        2.845000  2039.000000    0.665000    1.155000   2004.900000
75%        4.462500  2468.500000    0.932500    1.652500   8414.275000
max       10.640000  4717.000000    1.450000    2.180000  19263.900000

           cluCH  cluKM
count  52.000000   52.0
mean    1.692308    3.0
std     1.528019    0.0
min     1.000000    3.0
25%     1.000000    3.0
50%     1.000000    3.0
75%     1.000000    3.0
max     5.000000    3.0
```

	cluKM	ISO3	Zone
6	4	ATG	Antigua-et-Barbuda
17	4	BHS	Bahamas
20	4	BLZ	Belize
23	4	BRB	Barbade
24	4	BRN	Brunéi Darussalam
25	4	BTN	Bhoutan
26	4	BWA	Botswana
38	4	CPV	Cabo Verde
44	4	DMA	Dominique
55	4	FJI	Fidji
66	4	GRD	Grenade
68	4	GUY	Guyana
79	4	ISL	Islande
89	4	KIR	Kiribati
90	4	KNA	Saint-Kitts-et-Nevis
97	4	LCA	Sainte-Lucie
103	4	MAC	Chine - RAS de Macao
110	4	MLT	Malte

```
112      4  MNE                        Monténégro
113      4  MNG                          Mongolie
116      4  MUS                           Maurice
126      4  NRU                             Nauru
152      4  STP               Sao Tomé-et-Principe
158      4  SYC                          Seychelles
166      4  TON                               Tonga
170      4  TUV                              Tuvalu
178      4  VCT  Saint-Vincent-et-les Grenadines
181      4  VUT                             Vanuatu
182      4  WSM                               Samoa
```

	Unnamed: 0	pop	tc_pop	%urb	pib	\
count	29.000000	2.900000e+01	29.000000	29.000000	2.900000e+01	
mean	90.827586	5.172364e+05	1.597586	55.388276	6.681713e+09	
std	56.818677	6.942532e+05	1.973045	24.408137	9.568906e+09	
min	6.000000	1.058909e+04	-2.330000	17.450000	4.957169e+07	
25%	38.000000	1.150606e+05	0.060000	34.060000	8.735290e+08	
50%	90.000000	2.860765e+05	1.570000	54.960000	2.081184e+09	
75%	126.000000	6.204121e+05	3.500000	71.950000	1.195069e+10	
max	182.000000	3.157713e+06	4.870000	98.960000	4.403003e+10	

	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	29.000000	29.000000	29.000000	29.000000	29.000000	
mean	6.891034	14770.039310	5171.495862	17.974828	40.892414	
std	6.786449	16766.811511	5278.504983	54.021267	24.311362	
min	-9.300000	1875.400000	8.310000	0.000000	2.070000	
25%	3.600000	4443.880000	989.960000	0.320000	19.340000	
50%	6.720000	8475.810000	3377.490000	2.900000	41.430000	
75%	9.750000	17446.450000	7427.310000	11.510000	59.050000	
max	30.940000	67694.730000	20546.900000	291.630000	92.210000	

	tc_dispo/hab	prix	pv	rq	dist	\
count	29.000000	29.000000	29.000000	29.000000	29.000000	
mean	6.228966	1820.431034	0.852069	0.142759	8905.086207	
std	6.810441	612.827285	0.345764	0.643283	4485.689144	
min	-3.520000	941.500000	-0.140000	-0.900000	1491.300000	
25%	0.770000	1408.000000	0.770000	-0.410000	6826.400000	
50%	4.370000	1816.000000	0.920000	0.080000	7320.100000	
75%	9.210000	2040.000000	1.040000	0.470000	11219.300000	
max	22.610000	4112.000000	1.340000	1.700000	16941.400000	

	cluCH	cluKM
count	29.000000	29.0
mean	2.862069	4.0
std	0.915117	0.0
min	1.000000	4.0
25%	3.000000	4.0

```
50%     3.000000    4.0
75%     3.000000    4.0
max     5.000000    4.0
```

	cluKM	ISO3	Zone
10	2	BDI	Burundi
12	2	BEN	Bénin
13	2	BFA	Burkina Faso
14	2	BGD	Bangladesh
27	2	CAF	République centrafricaine
32	2	CIV	Côte d'Ivoire
33	2	CMR	Cameroun
34	2	COD	République démocratique du Congo
37	2	COM	Comores
47	2	DZA	Algérie
53	2	ETH	Éthiopie
61	2	GIN	Guinée
62	2	GMB	Gambie
63	2	GNB	Guinée-Bissau
72	2	HTI	Haïti
74	2	IDN	Indonésie
75	2	IND	Inde
86	2	KEN	Kenya
87	2	KGZ	Kirghizistan
88	2	KHM	Cambodge
93	2	LAO	République démocratique populaire lao
95	2	LBR	Libéria
98	2	LKA	Sri Lanka
106	2	MDG	Madagascar
109	2	MLI	Mali
114	2	MOZ	Mozambique
115	2	MRT	Mauritanie
117	2	MWI	Malawi
120	2	NER	Niger
125	2	NPL	Népal
129	2	PAK	Pakistan
133	2	PNG	Papouasie-Nouvelle-Guinée
141	2	RWA	Rwanda
144	2	SEN	Sénégal
146	2	SLB	Îles Salomon
147	2	SLE	Sierra Leone
149	2	SOM	Somalie
157	2	SWZ	Eswatini
159	2	SYR	République arabe syrienne
160	2	TCD	Tchad
161	2	TGO	Togo
163	2	TJK	Tadjikistan
164	2	TKM	Turkménistan

```
165      2  TLS                              Timor-Leste
172      2  TZA            République-Unie de Tanzanie
173      2  UGA                                 Ouganda
185      2  ZMB                                  Zambie
186      2  ZWE                                Zimbabwe
```

	Unnamed: 0	pop	tc_pop	%urb	pib	\
count	48.000000	4.800000e+01	48.000000	48.000000	4.800000e+01	
mean	102.208333	6.092550e+07	4.468750	35.903958	1.150568e+11	
std	52.787393	2.006910e+08	1.565554	13.374275	4.117948e+11	
min	10.000000	7.022479e+05	1.450000	11.900000	1.178988e+09	
25%	61.750000	7.458268e+06	3.185000	26.340000	8.015305e+09	
50%	107.500000	1.608663e+07	4.720000	35.885000	1.609653e+10	
75%	147.500000	2.779411e+07	5.532500	44.065000	4.056836e+10	
max	186.000000	1.371249e+09	7.190000	71.720000	2.683277e+12	

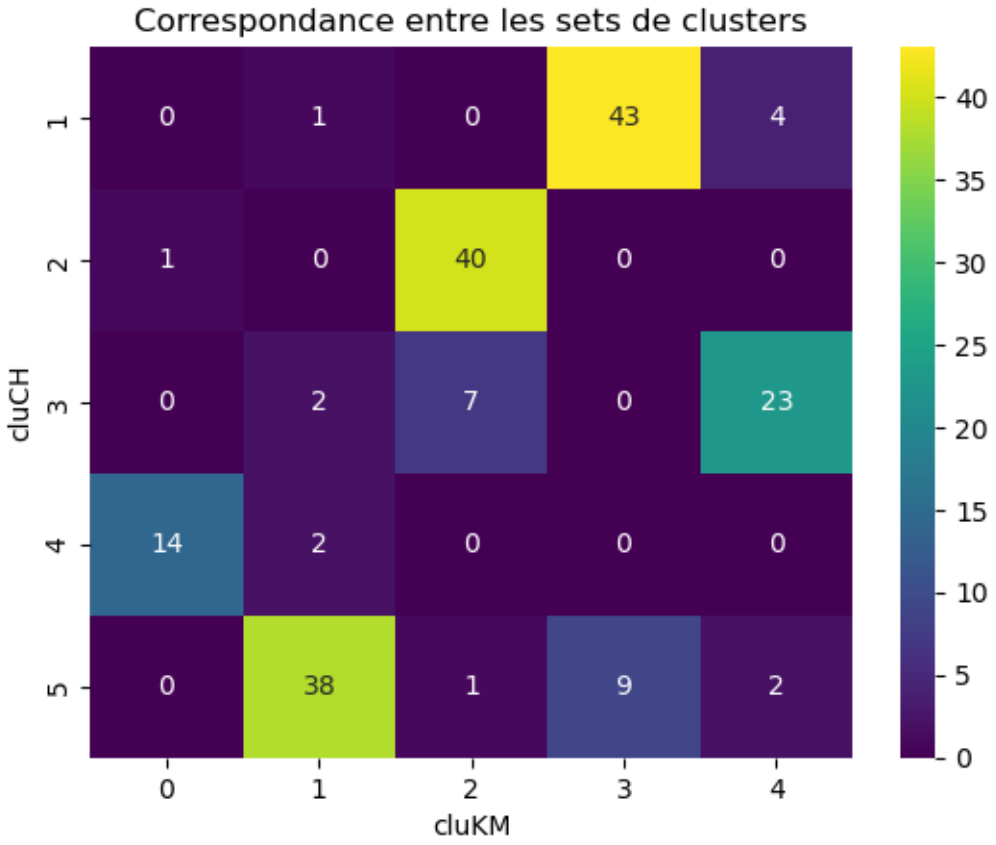
	tc_pib	pib/hab	imp	exp	dispo/hab	\
count	48.000000	48.000000	48.000000	48.000000	48.000000	
mean	9.589375	1524.594375	12388.673958	432.785417	4.540208	
std	4.824173	1226.048669	20897.084939	1126.377010	3.108907	
min	0.060000	274.730000	4.600000	0.000000	0.470000	
25%	6.537500	769.265000	197.585000	0.132500	1.915000	
50%	9.970000	1175.585000	3122.195000	8.145000	3.980000	
75%	12.750000	1817.897500	18220.452500	144.190000	6.472500	
max	21.350000	6677.140000	94154.900000	5133.530000	13.210000	

	tc_dispo/hab	prix	pv	rq	dist	\
count	48.000000	48.000000	48.000000	48.000000	48.000000	
mean	5.961042	1281.695833	-0.777292	-0.825625	6613.241667	
std	10.952356	581.235739	0.749581	0.467915	2807.766119	
min	-19.910000	473.700000	-2.750000	-2.060000	1340.400000	
25%	-0.440000	829.875000	-1.150000	-1.040000	4716.200000	
50%	5.805000	1103.000000	-0.635000	-0.775000	6089.300000	
75%	14.077500	1623.500000	-0.190000	-0.530000	7861.225000	
max	36.580000	2687.000000	0.540000	0.070000	15178.200000	

	cluCH	cluKM
count	48.000000	48.0
mean	2.208333	2.0
std	0.544150	0.0
min	2.000000	2.0
25%	2.000000	2.0
50%	2.000000	2.0
75%	2.000000	2.0
max	5.000000	2.0

4.3 - Caractérisation des clusters

[82]:
```
contingency_table = pd.crosstab(df_work['cluCH'], df_work['cluKM'])
fig=plt.figure()
sb.heatmap(contingency_table, annot=True, cmap='viridis')
plt.title("Correspondance entre les sets de clusters")
plt.show()
```



À y regarder de plus près, cela fait seulement 28 pays (1 pays sur 6) qui ne sont pas classés de la même façon par les 2 méthodes de clustering.

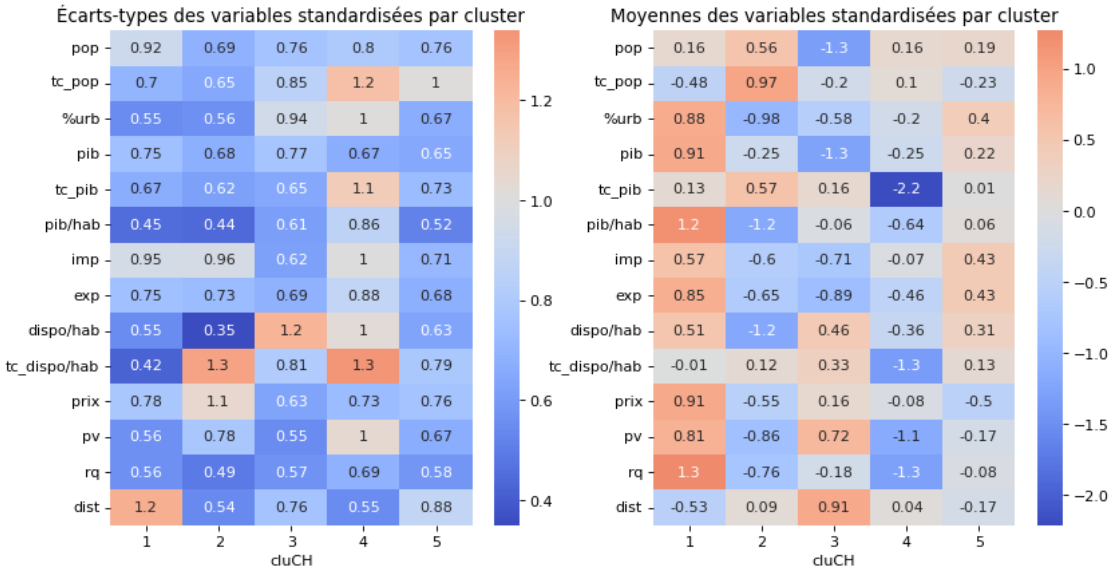
Afin de caractériser les clusters, on va s'intéresser (pour chaque cluster) à 2 grandeurs issues des distributions des variables : - l'écart-type pour chaque variable centrée-réduite : pour évaluer la dispersion des individus du cluster vis-à-vis de cette variable - la moyenne pour chaque variable initiale (non-transformée) : pour évaluer le comportement moyen des individus du cluster.

4.3.1 - Clusters obtenu par CAH

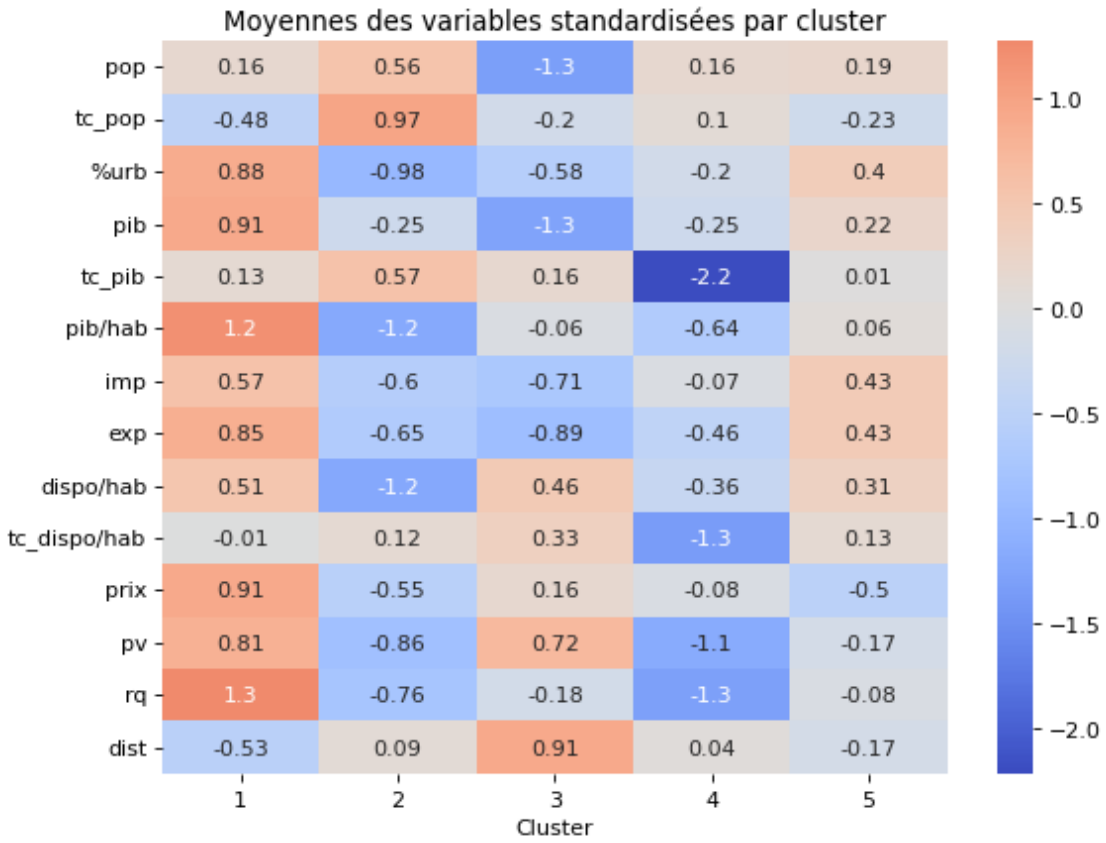
[83]:
```
df_scaled.columns = Lvar
```

[84]:
```
df_scaled["cluCH"] = df_work["cluCH"]
fig = plt.figure(figsize=(12,6), dpi=80)
plt.subplot(1,2,1)
```

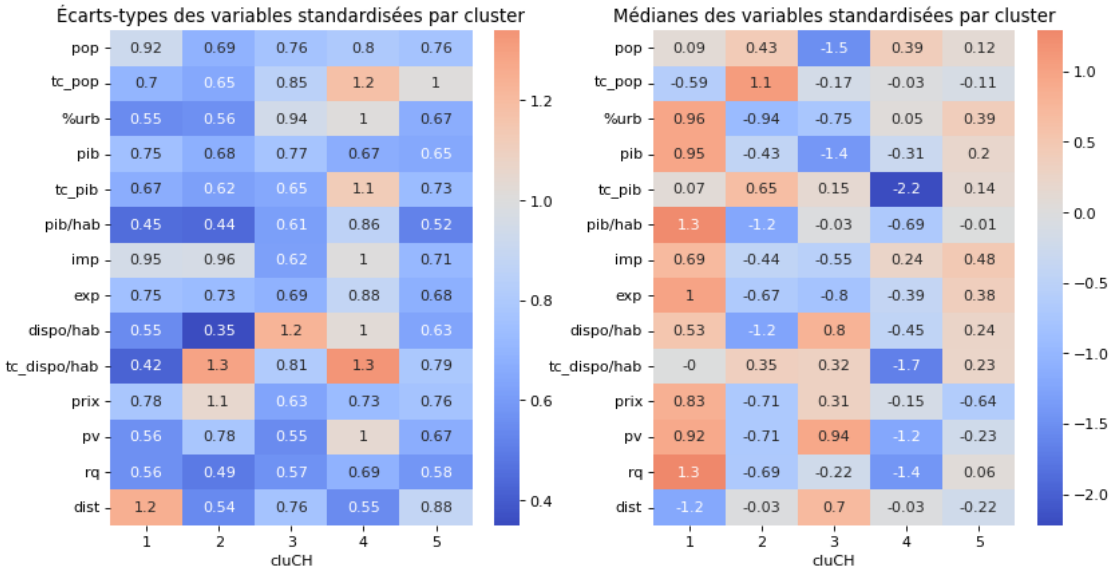
```
sb.heatmap((df_scaled.groupby("cluCH")[Lvar].std()).T.round(2), annot=True,␣
 ↪center=1, cmap='coolwarm')
plt.title("Écarts-types des variables standardisées par cluster")

plt.subplot(1,2,2)
sb.heatmap((df_scaled.groupby("cluCH")[Lvar].mean()).T.round(2), annot=True,␣
 ↪center=0, cmap='coolwarm')
plt.title("Moyennes des variables standardisées par cluster")
plt.savefig("Carac_clustersCH.png")
plt.show()
```



[85]:
```
fig = plt.figure(figsize=(8,6), dpi=80)

sb.heatmap((df_scaled.groupby("cluCH")[Lvar].mean()).T.round(2), annot=True,␣
 ↪center=0, cmap='coolwarm')
plt.xlabel("Cluster")
plt.title("Moyennes des variables standardisées par cluster")
plt.savefig("Centroids_clustersCH.png")
plt.show()
```



[86]:
```
fig = plt.figure(figsize=(12,6), dpi=80)
plt.subplot(1,2,1)
sb.heatmap((df_scaled.groupby("cluCH")[Lvar].std()).T.round(2), annot=True,␣
 ↪center=1, cmap='coolwarm')
plt.title("Écarts-types des variables standardisées par cluster")

plt.subplot(1,2,2)
sb.heatmap((df_scaled.groupby("cluCH")[Lvar].median()).T.round(2), annot=True,␣
 ↪center=0, cmap='coolwarm')
plt.title("Médianes des variables standardisées par cluster")
plt.show()
```



[87]:
```
df_zoid = (df_work.groupby("cluCH")[Lvar].mean().round(2)).reset_index()
df_nb = df_work.groupby("cluCH")["Zone"].count().reset_index().
 ↪rename(columns={"Zone": "Nb_Zones"})
display(pd.merge(df_nb, df_zoid, on="cluCH", how='inner'))
```

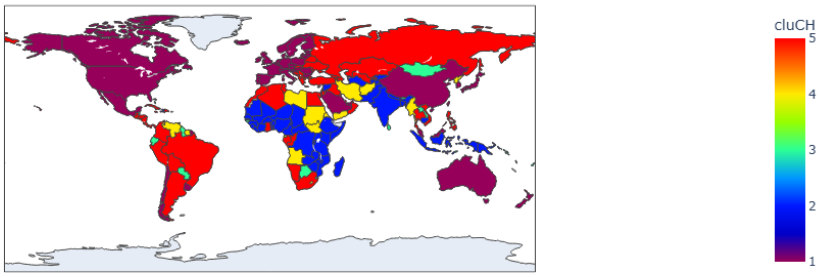
	cluCH	Nb_Zones	pop	tc_pop	%urb	pib	tc_pib	\
0	1	48	57621561.63	1.28	78.90	1.449925e+12	6.41	
1	2	41	74426653.33	4.66	36.41	1.383901e+11	9.98	
2	3	32	2204012.90	1.93	45.52	1.079715e+10	6.85	
3	4	16	23274893.46	2.64	54.22	6.720813e+10	-10.41	
4	5	50	28725036.28	1.87	68.02	2.049796e+11	5.70	

	pib/hab	imp	exp	dispo/hab	tc_dispo/hab	prix	pv	\
0	39804.14	159853.65	172461.90	28.39	3.30	2284.65	0.66	
1	1345.63	13942.78	452.70	4.06	5.64	1273.83	-0.93	
2	7703.00	4107.77	226.33	33.96	6.69	1621.31	0.58	
3	4436.55	42303.17	3086.32	16.01	-6.93	1485.19	-1.21	
4	9122.09	72265.93	114967.00	24.76	4.91	1241.16	-0.27	

	rq	dist
0	1.20	4514.30
1	-0.82	6185.59
2	-0.25	9819.98
3	-1.36	5993.99
4	-0.14	5413.54

[88]:
```
px.choropleth(df_work, locations="ISO3", locationmode="ISO-3",\
```

```
            color_continuous_scale=px.colors.sequential.Rainbow,␣
↪color='cluCH',\
            width=600, height=400)
```

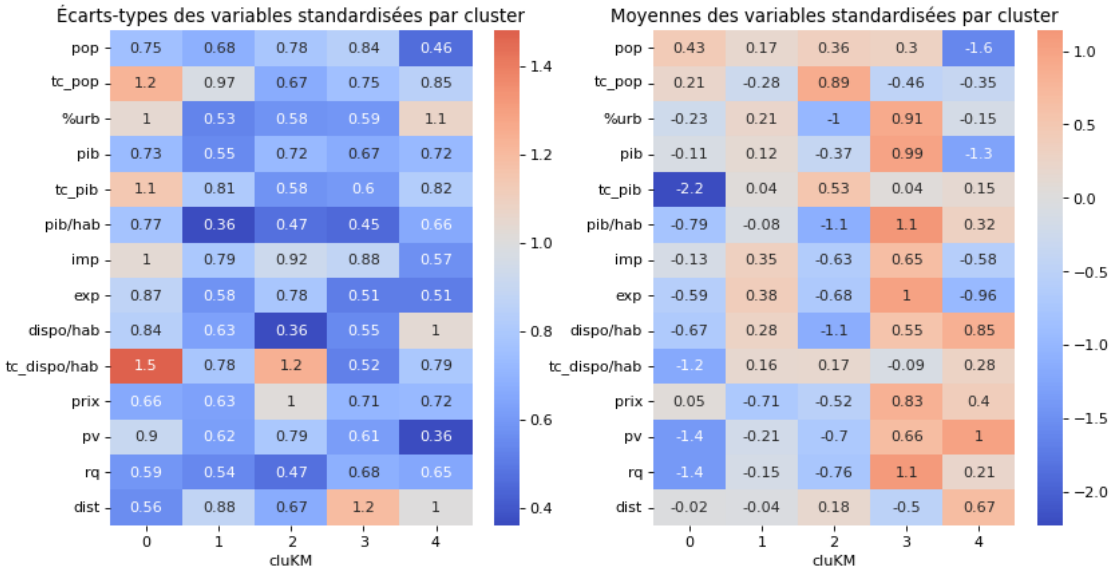


Lecture des clusters CH: - cluster 1 : se distingue des autres par un pib/hab particulièrement élevé, celui-ci étant homogène parmi les pays du cluster. Le cluster 1 CH rassemble les pays ayant le plus haut niveau de vie. La croissance de la consommation individuelle est aussi homogène. Ces pays sont également très urbanisés (>75% en moyenne). Bons indices de gouvernance pv et rq sur l'ensemble du cluster. - cluster 2 : consommation de poulet individuelle très faible (en moyenne 4 kg par personne et par an), PIB/hab lui aussi très faible. Qualité de la réglementation médiocre. Pays peu urbanisés et très peuplés. Ce cluster regroupe les pays les plus en retard dans leur développement. - cluster 3 : pays peu peuplés, plutôt en retrait du marché mondial de la viande de poulet (*exp* et *imp* faibles), mais qui connaissent une bonne croissance économique. Pays plutôt stables politiquement avec une réglementation pas totalement propice au business. - cluster 4 : pays qui ont connu des crises intérieures particulièrement graves. Cela se traduit par des récessions économiques de diverses amplitudes, mais en moyenne de -10% par an, des indices de gouvernances rédhibitoires. - cluster 5 : pays avec des niveaux de vie homogène mais bien en retrait par rapport au cluster 1. Les prix sont bas, la stabilité politique est incertaine et la réglementation médiocre. Pays bien urbanisés mais avec une croissance économique moins dynamique. Ces pays ont tendance à faire appel à des importations de viande de poulet de façon assez homogène.

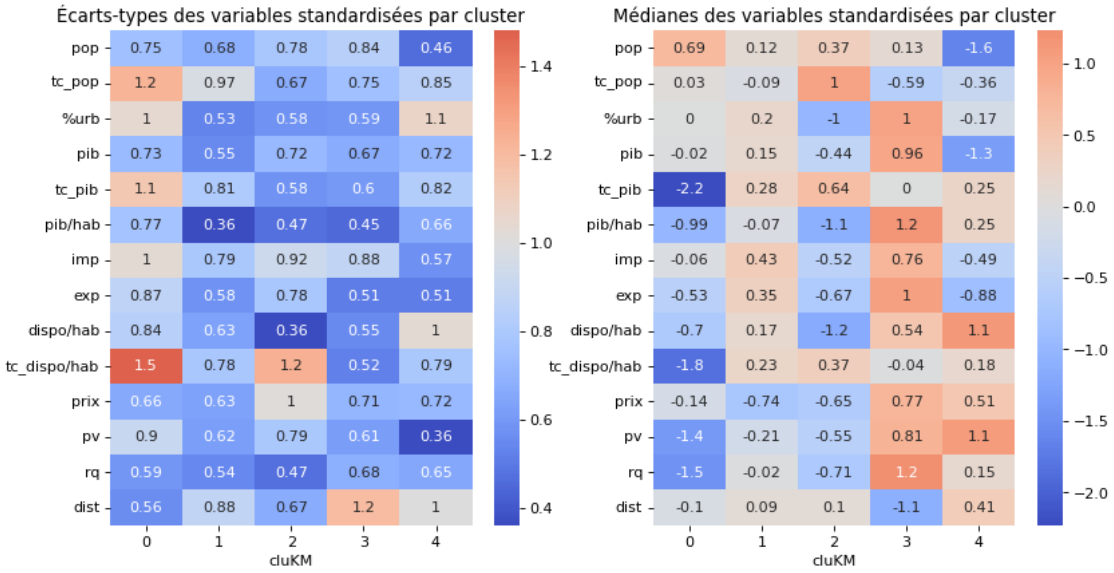
4.3.2 - Clusters obtenu par K-Means

[89]:
```
df_scaled["cluKM"] = df_work["cluKM"]
fig = plt.figure(figsize=(12,6), dpi=80)
plt.subplot(1,2,1)
sb.heatmap((df_scaled.groupby("cluKM")[Lvar].std()).T.round(2), annot=True,␣
 ↪center=1, cmap='coolwarm')
plt.title("Écarts-types des variables standardisées par cluster")

plt.subplot(1,2,2)
sb.heatmap((df_scaled.groupby("cluKM")[Lvar].mean()).T.round(2), annot=True,␣
 ↪center=0, cmap='coolwarm')
```

```
plt.title("Moyennes des variables standardisées par cluster")
plt.show()
```



[90]:
```
fig = plt.figure(figsize=(12,6), dpi=80)
plt.subplot(1,2,1)
sb.heatmap((df_scaled.groupby("cluKM")[Lvar].std()).T.round(2), annot=True,␣
 ↪center=1, cmap='coolwarm')
plt.title("Écarts-types des variables standardisées par cluster")

plt.subplot(1,2,2)
sb.heatmap((df_scaled.groupby("cluKM")[Lvar].median()).T.round(2), annot=True,␣
 ↪center=0, cmap='coolwarm')
plt.title("Médianes des variables standardisées par cluster")
plt.show()
```



[91]:
```
df_zoid = (df_work.groupby("cluKM")[Lvar].mean().round(2)).reset_index()
df_nb = df_work.groupby("cluKM")["Zone"].count().reset_index().
 ↪rename(columns={"Zone": "Nb_Zones"})
display(pd.merge(df_nb, df_zoid, on="cluKM", how='inner'))
```

	cluKM	Nb_Zones	pop	tc_pop	%urb	pib	tc_pib	\
0	0	15	38333443.45	2.90	53.54	9.997066e+10	-10.71	
1	1	43	23270658.29	1.76	63.55	1.213613e+11	6.02	
2	2	48	60925499.65	4.47	35.90	1.150568e+11	9.59	
3	3	52	61181585.99	1.35	79.61	1.432800e+12	5.69	
4	4	29	517236.36	1.60	55.39	6.681713e+09	6.89	

	pib/hab	imp	exp	dispo/hab	tc_dispo/hab	prix	pv	\
0	3253.84	42864.74	3139.75	10.77	-6.16	1556.73	-1.45	
1	6090.32	68678.86	31624.19	24.01	5.19	1112.55	-0.31	
2	1524.59	12388.67	432.79	4.54	5.96	1281.70	-0.78	
3	37060.62	160104.50	243720.66	29.08	2.71	2194.02	0.52	
4	14770.04	5171.50	17.97	40.89	6.23	1820.43	0.85	

	rq	dist
0	-1.46	5746.91
1	-0.22	5885.65
2	-0.83	6613.24
3	1.03	4541.27
4	0.14	8905.09

[92]:
```
px.choropleth(df_work, locations="ISO3", locationmode="ISO-3",\
```

```
            color_continuous_scale=px.colors.sequential.Rainbow,␣
↪color='cluKM',\
            width=600, height=400)
```

Lecture des clusters KM: - cluster 0 : pays qui ont connu des crises intérieures particulièrement graves. Cela se traduit par des récessions économiques de diverses amplitudes, mais en moyenne de -10% par an, des indices de gouvernances rédhibitoires. - cluster 1 : pays avec une croissance économique un peu molle, indices de gouvernance un peu médiocres - cluster 2 : consommation de poulet individuelle très faible (en moyenne 4 kg par personne et par an), PIB/hab lui aussi très faible. Qualité de la réglementation médiocre. Pays peu urbanisés et très peuplés. Ce cluster regroupe les pays les plus en retard dans leur développement. - cluster 3 : pays les plus développés, avec le niveau de vie le plus élevé, bons indicateurs de gouvernance - cluster 4 : pays qui ont un faible poids économique, représentent des débouchés commerciaux faibles en volume, mais dont la consommation individuelle de viande de poulet est particulièrement dynamique. pays également stables politiquement.

Comme on a 14 variables et 5 clusters, l'algorithme K-Means se révèle peu robuste car très dépendant des centroïdes initiaux, déterminés aléatoirement, alors que le clustering hiérarchique est déterministe car uniquement basé sur les distances entre individus (ou groupes d'individus).

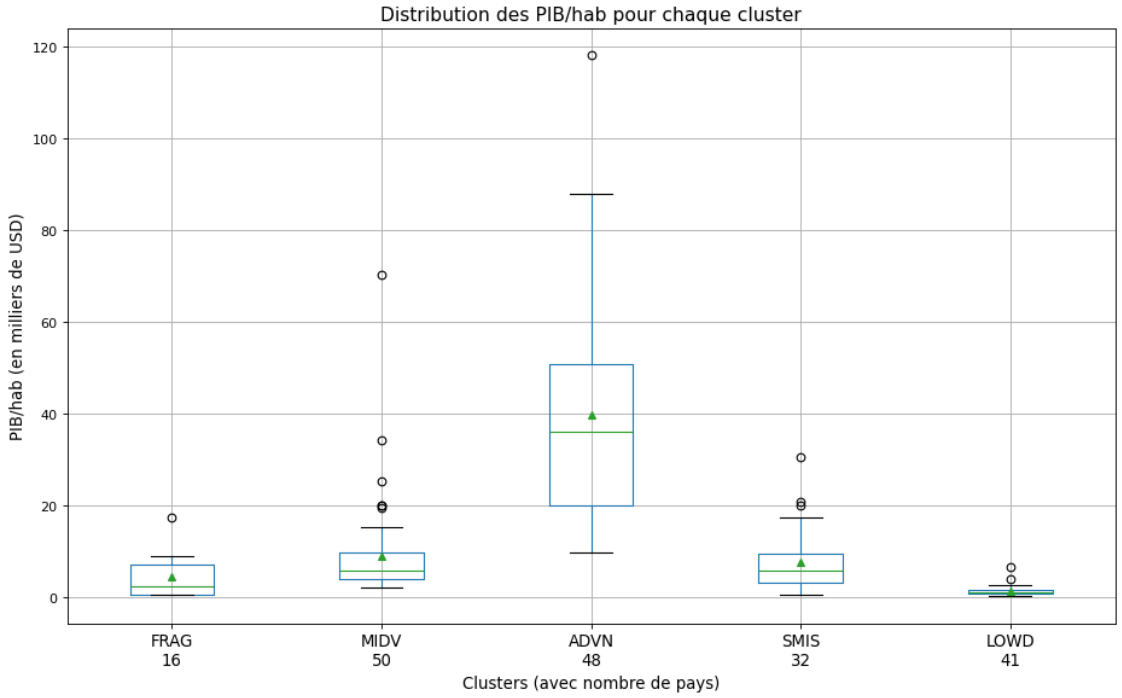
4.3.3 - Boxplots des clusters obtenu par CAH

On va donner des noms plus explicites aux clusters pour mieux les idnetifier.

[93]:
```
df_work = df_work.astype({"cluCH": str})

df_work.loc[df_work["cluCH"]=='1', "cluCH"] = "ADVN" # Pays les plus␣
↪développés, haut niveau de vie
df_work.loc[df_work["cluCH"]=='2', "cluCH"] = "LOWD" # Pays les moins␣
↪développés, bas niveau de vie
df_work.loc[df_work["cluCH"]=='3', "cluCH"] = "SMIS" # Pays de petite taille␣
↪(économique et démographique)
df_work.loc[df_work["cluCH"]=='4', "cluCH"] = "FRAG" # Pays fragiles, ont connu␣
↪de graves crises intérieures
df_work.loc[df_work["cluCH"]=='5', "cluCH"] = "MIDV" # Pays intermédiaires,␣
↪développés, mais avec un niveau de vie moyen
```

[94]:
```
# Boîtes à moustaches pour l'analyse de la variable pib/hab par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki["pib/hab"] /= 1e3
    df_worki.boxplot(column=["pib/hab"],positions=[i], widths=[0.4],␣
↪showmeans=True)
```

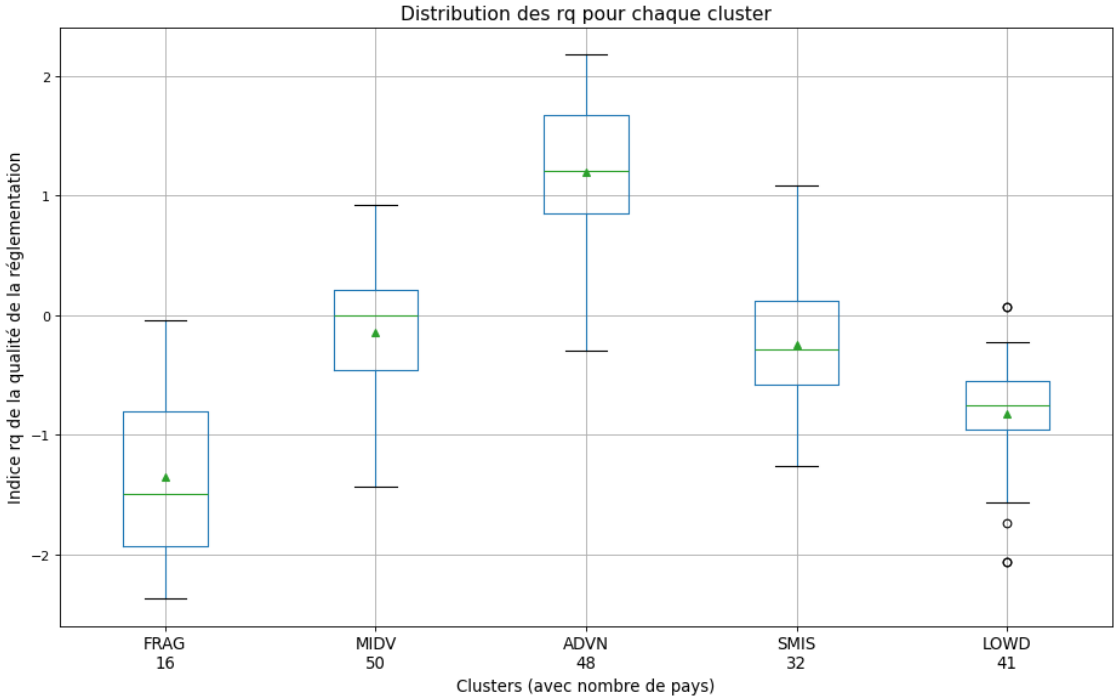
```
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("PIB/hab (en milliers de USD)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution des PIB/hab pour chaque cluster", fontsize=14)
plt.savefig("boxplot_PIBhab.png")
plt.show()
```



Les pays du cluster ADVN sont nettement devant les autres en terme de PIB/habitant. A contrario, les PIB/hab des pays du cluster LOWD sont nettement en retrait. Certaisn outliers du cluster MIDV (comme le Qatar) ont un PIB/hab similaire à ceux des pays du cluster ADVN.

[95]:
```
# Boîtes à moustaches pour l'analyse de la variable rq par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["rq"],positions=[i], widths=[0.4], showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))
```

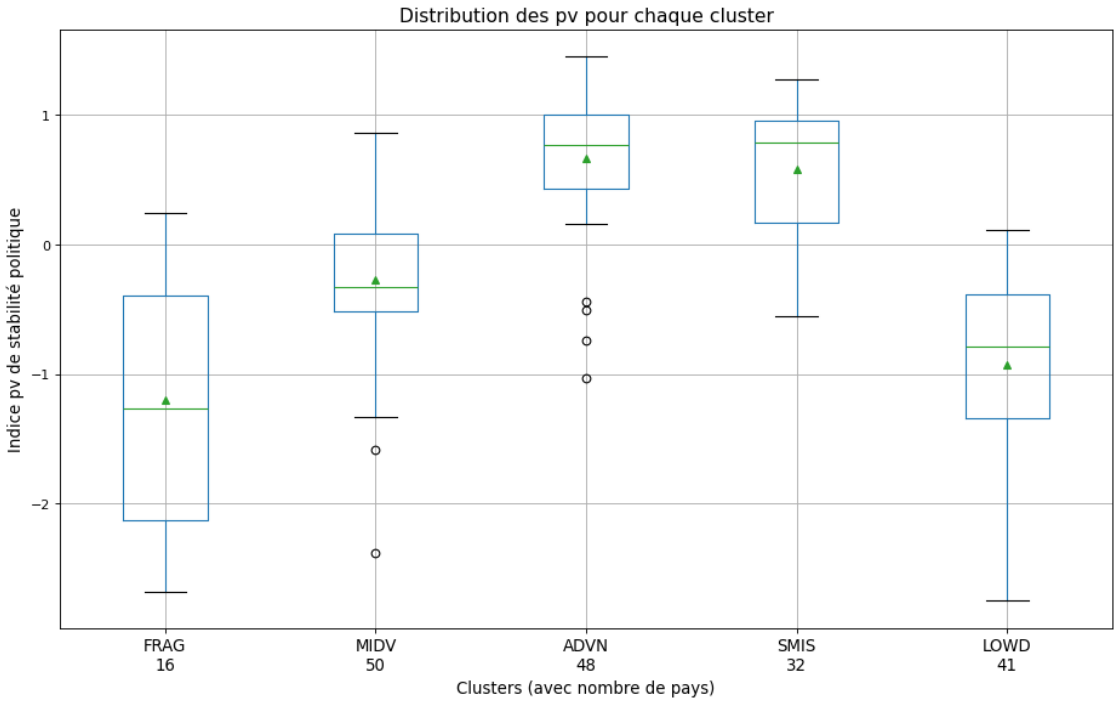
```
plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Indice rq de la qualité de la réglementation", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution des rq pour chaque cluster", fontsize=14)
plt.savefig("boxplot_rq.png")
plt.show()
```



On retrouve avec les rq une hiérarchie similaire à celle des PIB/hab. En effet, les variables PIB/hab et rq étaient toutes deux assez corrélées à l'axe F1.

[96]:
```
# Boîtes à moustaches pour l'analyse de la variable pv par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["pv"],positions=[i], widths=[0.4], showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
```

```
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Indice pv de stabilité politique", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution des pv pour chaque cluster", fontsize=14)
plt.savefig("boxplot_pv.png")
plt.show()
```

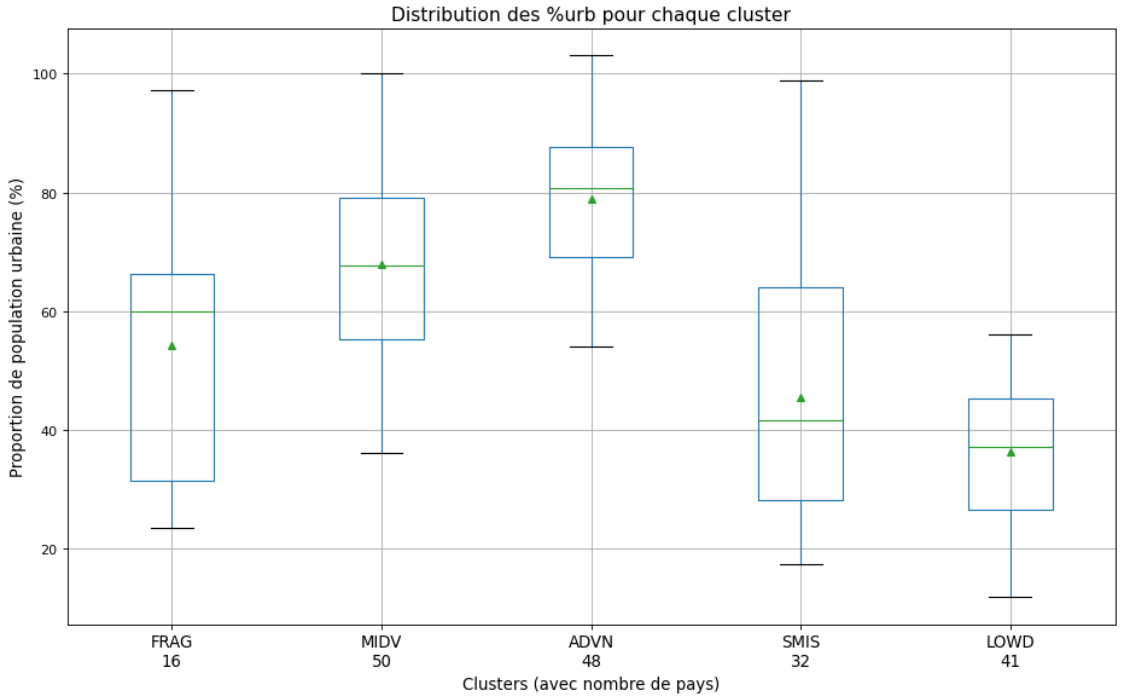


[97]:
```
display(df_work.loc[df_work["cluCH"]=='ADVN'].sort_values("pv").head())
```

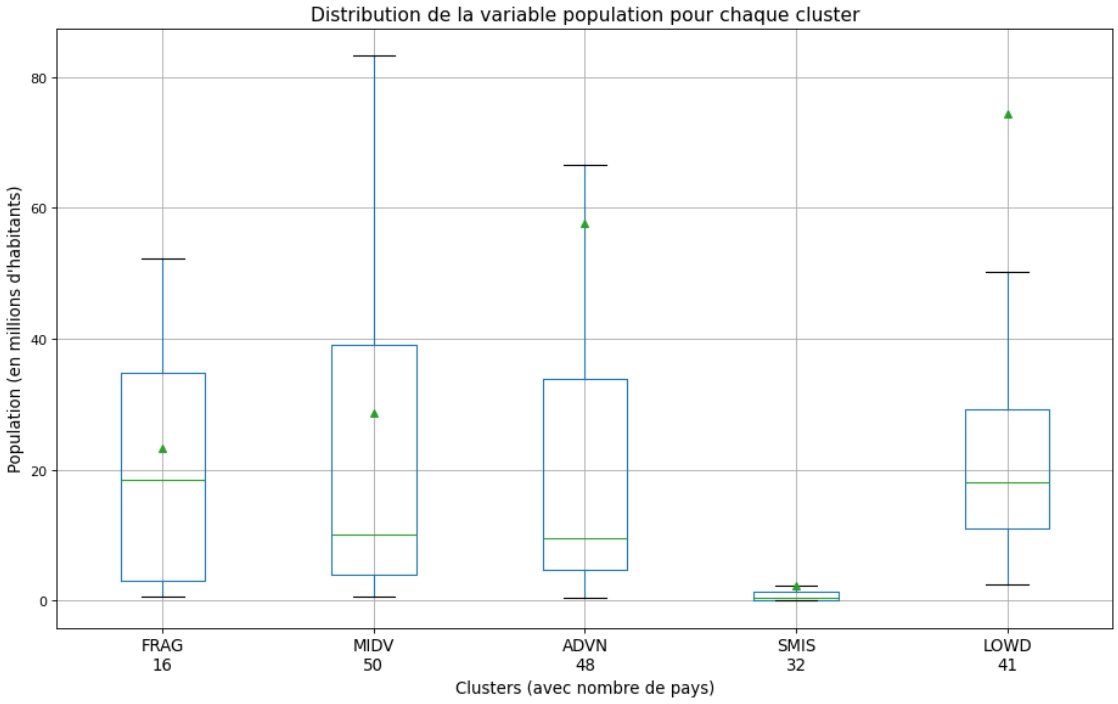
```
     Unnamed: 0 ISO3                 Zone           pop  tc_pop   %urb  \
80           80  ISR               Israël  8.503779e+06    3.53  91.93
107         107  MEX              Mexique  1.243626e+08    1.81  84.17
142         142  SAU       Arabie saoudite  3.064521e+07    2.56  91.32
31           31  CHN  Chine (continentale)  1.411122e+09    0.66  59.06
118         118  MYS              Malaisie  3.278126e+07    3.06  74.14

                pib  tc_pib   pib/hab        imp        exp  dispo/hab  \
80    3.928823e+11   12.58  45835.71       0.00     273.20      62.10
107   1.315136e+12    4.40  10564.96  856599.91    2605.72      33.59
142   8.218800e+11    8.43  26738.21  672511.84   34149.41      46.07
31    1.372828e+13   13.23   9709.00  840851.57  186777.57      10.14
118   3.477266e+11    5.20  10591.60   91960.16   10020.85      49.59

     tc_dispo/hab    prix    pv    rq     dist cluCH  cluKM
80           0.56  1566.0 -1.03  1.22   3281.9  ADVN      3
```

```
107          5.36  2077.0 -0.74  0.09   9206.8  ADVN      3
142          0.89  2033.0 -0.51  0.13   4694.5  ADVN      3
31           4.44  2532.0 -0.44 -0.30   8225.2  ADVN      3
118          0.93  2190.0  0.16  0.66  10436.4  ADVN      3
```

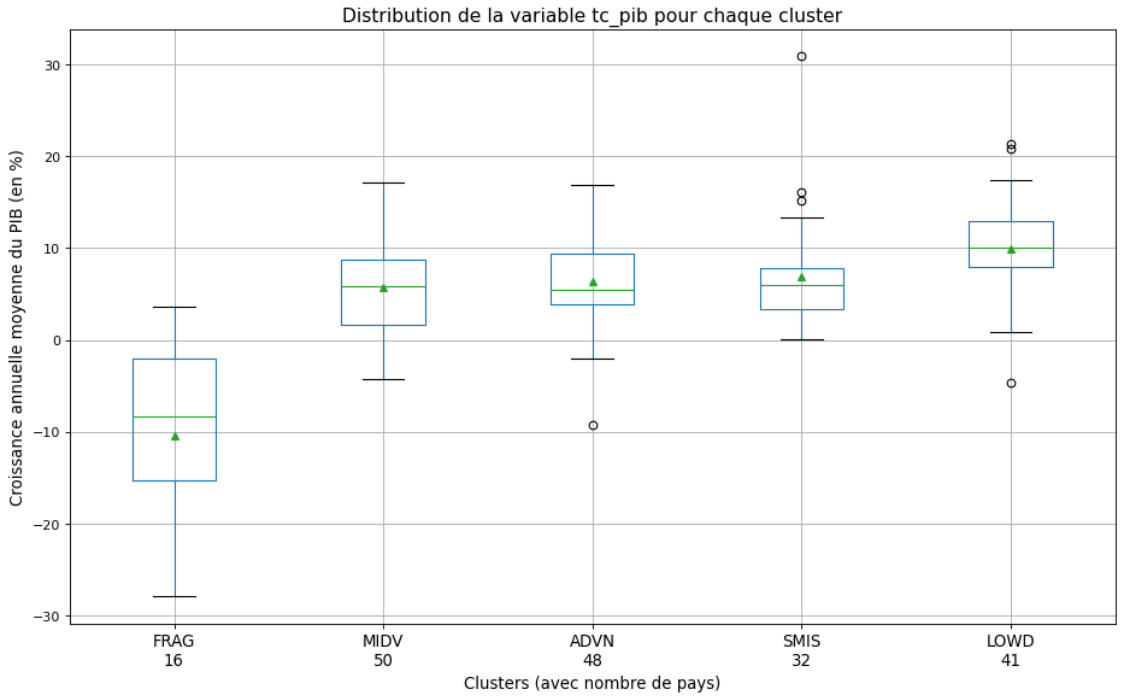
[98]:
```
# Boîtes à moustaches pour l'analyse de la variable %urb par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["%urb"],positions=[i], widths=[0.4],␣
 ↪showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Proportion de population urbaine (%)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution des %urb pour chaque cluster", fontsize=14)
plt.savefig("boxplot_%urb.png")
plt.show()
```



[99]:
```
# Boîtes à moustaches pour l'analyse de la variable population par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki["pop"] /= 1e6
    df_worki.boxplot(column=["pop"],positions=[i], widths=[0.4],␣
 ↪showmeans=True, showfliers=False)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Population (en millions d'habitants)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable population pour chaque cluster",␣
 ↪fontsize=14)
plt.savefig("boxplot_pop.png")
plt.show()
```



[100]:
```
# Boîtes à moustaches pour l'analyse de la variable tc_pib par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["tc_pib"],positions=[i], widths=[0.4],␣
 ↪showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Croissance annuelle moyenne du PIB (en %)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable tc_pib pour chaque cluster", fontsize=14)
plt.savefig("boxplot_tc-pib.png")
plt.show()
```

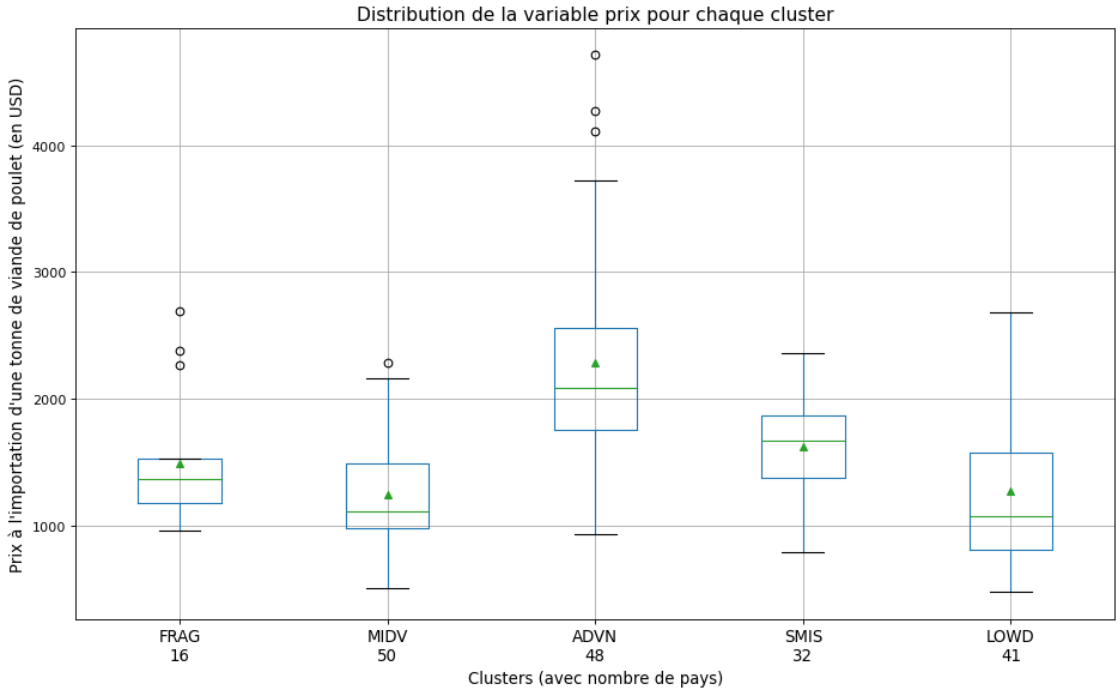


[101]:
```
display(df_work.loc[df_work["cluCH"]=='ADVN'].sort_values("tc_pib").head())
```

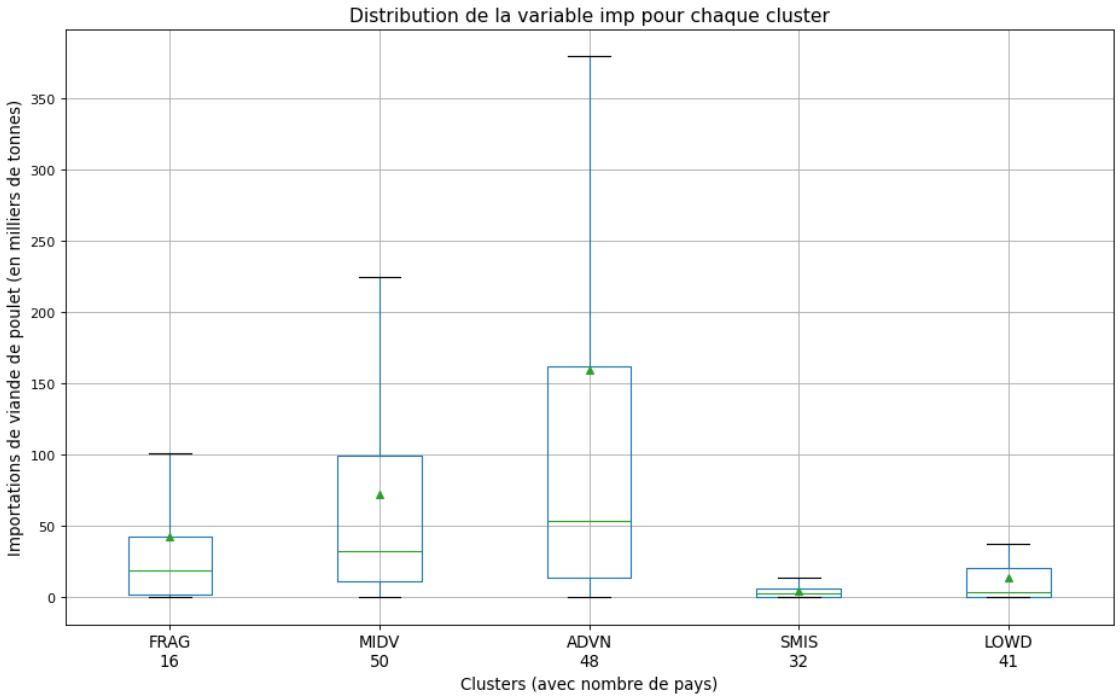
```
     Unnamed: 0 ISO3                  Zone          pop  tc_pop   %urb  \
103         103  MAC  Chine - RAS de Macao  6.579614e+05    3.57  95.94
```

```
84           84  JPN                 Japon  1.265160e+08   -0.48  91.99
24           24  BRN     Brunéi Darussalam  4.367667e+05    2.21  77.02
81           81  ITA                Italie  6.017546e+07   -0.42  69.40
156         156  SWE                 Suède  1.012190e+07    1.95  86.15

                pib  tc_pib   pib/hab        imp       exp  dispo/hab  \
103  4.403003e+10   -9.30  67694.73   20546.90      0.00      36.05
84   4.836251e+12   -2.08  38214.37  541214.28   8161.36      21.99
24   1.422809e+10   -1.37  32654.02     745.74      4.18      62.16
81   2.058128e+12    1.37  34210.04   55050.60  91653.81      16.11
156  5.585938e+11    1.55  55204.74   51720.17  22682.56      17.99

     tc_dispo/hab    prix    pv    rq     dist cluCH  cluKM
103          9.21  1986.0  1.16  1.70   9601.9  ADVN      4
84           3.82  2212.0  1.02  1.32   9725.6  ADVN      3
24           1.82  2229.0  1.20  0.86  11219.3  ADVN      4
81           3.66  2058.0  0.42  0.67   1109.9  ADVN      3
156          2.06  3071.0  0.97  1.78   1545.8  ADVN      3
```

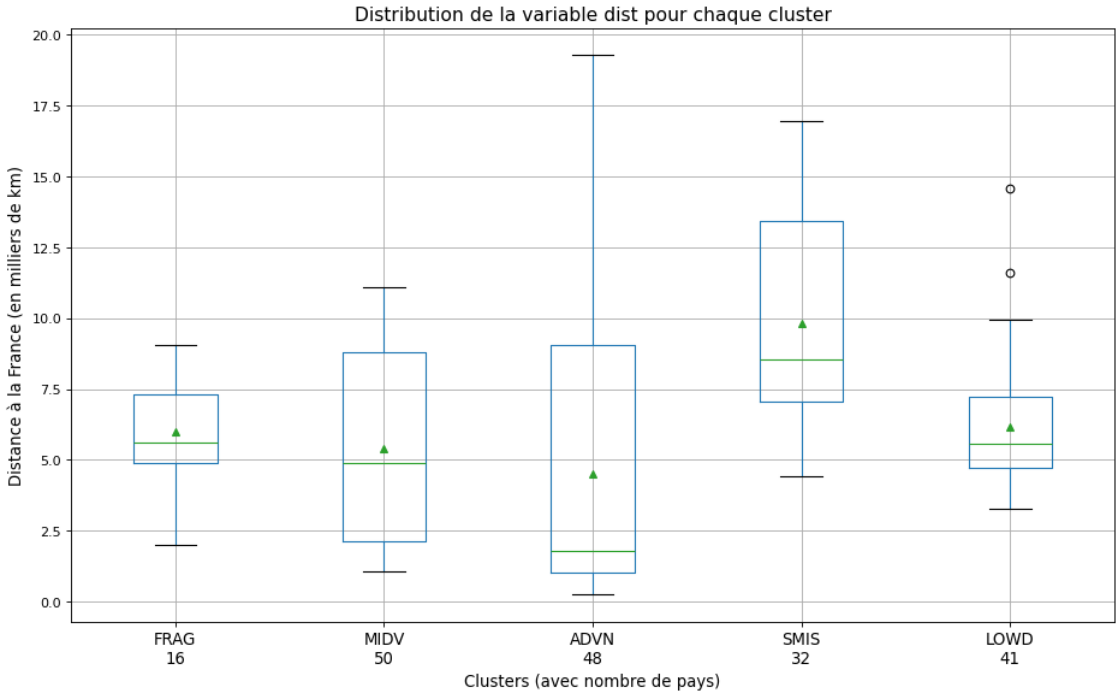
[102]:
```
# Boîtes à moustaches pour l'analyse de la variable prix par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["prix"],positions=[i], widths=[0.4],␣
 ↪showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Prix à l'importation d'une tonne de viande de poulet (en USD)",␣
 ↪fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable prix pour chaque cluster", fontsize=14)
plt.savefig("boxplot_prix.png")
plt.show()
```



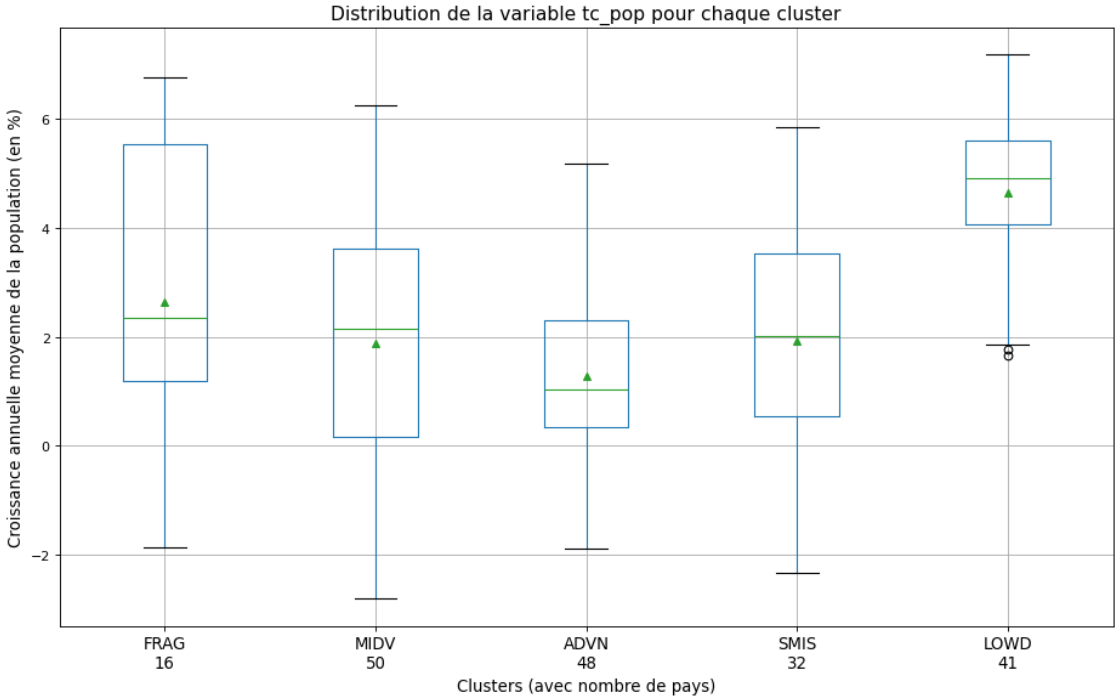
[103]:
```
# Boîtes à moustaches pour l'analyse de la variable importation par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki["imp"] /= 1e3
    df_worki.boxplot(column=["imp"],positions=[i], widths=[0.4],␣
 ↪showmeans=True, showfliers=False)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Importations de viande de poulet (en milliers de tonnes)",␣
 ↪fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable imp pour chaque cluster", fontsize=14)
plt.savefig("boxplot_imp.png")
plt.show()
```



[104]:
```
# Boîtes à moustaches pour l'analyse de la variable dist par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki["dist"] /= 1e3
    df_worki.boxplot(column=["dist"],positions=[i], widths=[0.4],␣
 ↪showmeans=True, showfliers=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Distance à la France (en milliers de km)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable dist pour chaque cluster", fontsize=14)
plt.savefig("boxplot_dist.png")
plt.show()
```



[105]:
```
# Boîtes à moustaches pour l'analyse de la variable tc_pop par cluster

fig = plt.figure(figsize=(14,8), dpi=80)

L_clu = list(df_work["cluCH"].unique())
L_lab = L_clu.copy()

for i in range(len(L_clu)):
    df_worki = df_work.loc[df_work["cluCH"]==L_clu[i]].copy()
    df_worki.boxplot(column=["tc_pop"],positions=[i], widths=[0.4],␣
 ↪showmeans=True)
    L_lab[i] = L_clu[i] + "\n" + str(len(df_worki))

plt.xticks(ticks=range(0,len(L_clu)), labels=L_lab, fontsize=12)
plt.xlabel("Clusters (avec nombre de pays)", fontsize=12)
plt.ylabel("Croissance annuelle moyenne de la population (en %)", fontsize=12)
plt.xlim([-0.5,len(L_clu)-0.5])
plt.title("Distribution de la variable tc_pop pour chaque cluster", fontsize=14)
plt.savefig("boxplot_tc-pop.png")
plt.show()
```



4.4 - Pays les plus intéressants au sein de chaque cluster

[106]:
```
display(df_work)
```

```
     Unnamed: 0 ISO3                 Zone          pop  tc_pop   %urb  \
0             0  AFG          Afghanistan  36758062.91    5.43  25.26
1             1  AGO               Angola  31358489.00    6.76  64.48
2             2  ALB              Albanie   2876923.27   -0.63  61.43
3             3  ARE  Émirats arabes unis   9259546.36    5.18  89.57
4             4  ARG            Argentine  44395603.91    1.36  92.41
..          …  …                    …           …     …    …
182         182  WSM                Samoa    207637.27    1.84  17.45
183         183  YEM                Yémen  34180357.18    5.87  31.04
184         184  ZAF       Afrique du Sud  58886231.00    2.86  64.58
185         185  ZMB               Zambie  18006950.64    5.92  42.75
186         186  ZWE             Zimbabwe  15097522.91    3.08  36.22

                pib  tc_pib   pib/hab        imp        exp  dispo/hab  \
0    1.812068e+10   -4.57    499.65   30146.73      52.98       1.59
1    1.031760e+11  -11.12   3393.95  265843.72      72.78      10.06
2    1.525616e+10   12.20   5315.70   24500.61       4.91      13.42
3    4.155386e+11    4.58  44935.94  536015.67   47777.40      58.23
4    5.591068e+11   -1.91  12614.01    4318.53  209064.85      43.46
..            …     …       …        …         …        …
182  8.705741e+08    2.86   4191.05   15739.14      34.87      77.46
```

```
183  1.888517e+10  -27.96    580.50  100922.62      15.84       8.35
184  3.789851e+11    1.16   6444.51  384106.89   51249.95      35.95
185  2.428533e+10    0.81   1359.56   17087.18    3800.64       3.40
186  2.272259e+10    7.95   1499.38    5285.46       0.00       6.48

     tc_dispo/hab    prix    pv    rq     dist cluCH  cluKM
0          -15.73  1270.0 -2.60 -1.23   5590.4  FRAG      0
1           -9.35  2693.0 -0.45 -0.85   6510.3  FRAG      0
2           12.14   880.9  0.24  0.22   1603.5  MIDV      1
3            1.29  1582.0  0.68  0.99   5249.5  ADVN      3
4            4.69  1611.0  0.00 -0.58  11072.2  MIDV      3
..            …     …   …   …       …   …    …
182          2.25   941.5  1.12 -0.20  16011.9  SMIS      4
183         -1.56  1371.0 -2.68 -1.53   5317.3  FRAG      0
184         -0.33   752.4 -0.34  0.05   9353.6  MIDV      1
185          3.83  1416.0  0.11 -0.55   7604.8  LOWD      2
186          6.02   657.0 -0.80 -1.57   7949.7  LOWD      2

[187 rows x 19 columns]
```

[107]:
```
df_eval = df_ra[["ISO3","ra_imp","per","score"]]
df_eval = df_work[["ISO3","Zone","cluCH","cluKM"]].merge(df_eval, on="ISO3",␣
 ↪how='inner')
display(df_eval)
```

```
    ISO3                 Zone cluCH  cluKM    ra_imp    per    score
0    AFG          Afghanistan  FRAG      0  -6288.90  0.996      0.0
1    AGO               Angola  FRAG      0 -15100.91  1.000      0.0
2    ALB              Albanie  MIDV      1  11211.02  1.000  11211.0
3    ARE  Émirats arabes unis  ADVN      3  26984.90  0.892  24070.5
4    ARG            Argentine  MIDV      3    589.14  0.000      0.0
..   …                    …   …    …       …     …       …
182  WSM                Samoa  SMIS      4    367.66  0.997    366.6
183  YEM                Yémen  FRAG      0  15764.76  1.000  15764.8
184  ZAF       Afrique du Sud  MIDV      1 -72087.50  0.882      0.0
185  ZMB               Zambie  LOWD      2   8353.39  0.746   6231.6
186  ZWE             Zimbabwe  LOWD      2    765.03  1.000    765.0

[187 rows x 7 columns]
```

[108]:
```
for n in list(df_eval["cluCH"].unique()):
    print("\nMeilleures opportunités commerciales cluster",str(n),":")
    display(df_eval.loc[df_eval["cluCH"]==n].sort_values("ra_imp",␣
 ↪ascending=False).head(10))
    print("\nOpportunités les plus pertinentes cluster",str(n),":")
    display(df_eval.loc[df_eval["cluCH"]==n].sort_values("score",␣
 ↪ascending=False).head(10))
```

Meilleures opportunités commerciales cluster FRAG :

	ISO3	Zone	cluCH	cluKM	ra_imp	\
77	IRN	Iran (République islamique d')	FRAG	0	36866.64	
96	LBY	Libye	FRAG	0	25213.86	
183	YEM	Yémen	FRAG	0	15764.76	
64	GNQ	Guinée équatoriale	FRAG	0	4121.74	
167	TTO	Trinité-et-Tobago	FRAG	1	979.00	
151	SSD	Soudan du Sud	FRAG	0	888.03	
135	PRK	République populaire démocratique de Corée	FRAG	0	354.72	
143	SDN	Soudan	FRAG	0	299.02	
50	ERI	Érythrée	FRAG	0	2.28	
111	MMR	Myanmar	FRAG	0	-50.75	

	per	score
77	0.337	12424.1
96	1.000	25213.9
183	1.000	15764.8
64	0.744	3066.6
167	0.996	975.1
151	1.000	888.0
135	1.000	354.7
143	0.984	294.2
50	1.000	2.3
111	0.947	0.0

Opportunités les plus pertinentes cluster FRAG :

	ISO3	Zone	cluCH	cluKM	ra_imp	\
96	LBY	Libye	FRAG	0	25213.86	
183	YEM	Yémen	FRAG	0	15764.76	
77	IRN	Iran (République islamique d')	FRAG	0	36866.64	
64	GNQ	Guinée équatoriale	FRAG	0	4121.74	
167	TTO	Trinité-et-Tobago	FRAG	1	979.00	
151	SSD	Soudan du Sud	FRAG	0	888.03	
135	PRK	République populaire démocratique de Corée	FRAG	0	354.72	
143	SDN	Soudan	FRAG	0	299.02	
50	ERI	Érythrée	FRAG	0	2.28	
0	AFG	Afghanistan	FRAG	0	-6288.90	

	per	score
96	1.000	25213.9
183	1.000	15764.8
77	0.337	12424.1
64	0.744	3066.6
167	0.996	975.1
151	1.000	888.0

```
135  1.000    354.7
143  0.984    294.2
50   1.000      2.3
0    0.996      0.0
```

Meilleures opportunités commerciales cluster MIDV :

```
    ISO3                    Zone cluCH  cluKM    ra_imp    per    score
180  VNM               Viet Nam  MIDV      1  38041.45  0.939  35720.9
60   GHA                  Ghana  MIDV      1  37502.98  1.000  37503.0
35   COG                  Congo  MIDV      1  35088.71  0.998  35018.5
177  UZB             Ouzbékistan  MIDV      1  23757.10  0.994  23614.6
169  TUR                 Türkiye  MIDV      1  21903.08  0.000      0.0
132  PHL             Philippines  MIDV      1  18494.87  0.999  18476.4
131  PER                   Pérou  MIDV      1  16496.45  1.000  16496.4
2    ALB                 Albanie  MIDV      1  11211.02  1.000  11211.0
46   DOM  République dominicaine  MIDV      1  10314.25  0.918   9468.5
154  SVK               Slovaquie  MIDV      3   9680.09  0.601   5817.7
```

Opportunités les plus pertinentes cluster MIDV :

```
    ISO3                    Zone cluCH  cluKM    ra_imp    per    score
60   GHA                  Ghana  MIDV      1  37502.98  1.000  37503.0
180  VNM               Viet Nam  MIDV      1  38041.45  0.939  35720.9
35   COG                  Congo  MIDV      1  35088.71  0.998  35018.5
177  UZB             Ouzbékistan  MIDV      1  23757.10  0.994  23614.6
132  PHL             Philippines  MIDV      1  18494.87  0.999  18476.4
131  PER                   Pérou  MIDV      1  16496.45  1.000  16496.4
2    ALB                 Albanie  MIDV      1  11211.02  1.000  11211.0
46   DOM  République dominicaine  MIDV      1  10314.25  0.918   9468.5
67   GTM               Guatemala  MIDV      1   7075.44  0.987   6983.5
154  SVK               Slovaquie  MIDV      3   9680.09  0.601   5817.7
```

Meilleures opportunités commerciales cluster ADVN :

```
    ISO3                                                Zone cluCH  cluKM  \
31   CHN                             Chine (continentale)  ADVN      3
123  NLD                          Pays-Bas (Royaume des)  ADVN      3
107  MEX                                         Mexique  ADVN      3
118  MYS                                        Malaisie  ADVN      3
56   FRA                                          France  ADVN      3
91   KOR                           République de Corée  ADVN      3
3    ARE                           Émirats arabes unis  ADVN      3
58   GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…  ADVN      3
145  SGP                                       Singapour  ADVN      3
171  TWN                                  Chine - Taïwan  ADVN      3

        ra_imp    per     score
```

```
31   341451.86  0.813  277600.4
123  107274.59  0.000       0.0
107   77126.89  0.998   76972.6
118   53348.08  0.903   48173.3
56    43370.71  0.392   17001.3
91    37421.45  0.790   29562.9
3     26984.90  0.892   24070.5
58    25425.61  0.272    6915.8
145   20733.96  0.810   16794.5
171   15451.10  0.987   15250.2
```

Opportunités les plus pertinentes cluster ADVN :

```
    ISO3                                                Zone cluCH  cluKM  \
31   CHN                                Chine (continentale)  ADVN      3
107  MEX                                             Mexique  ADVN      3
118  MYS                                            Malaisie  ADVN      3
91   KOR                                 République de Corée  ADVN      3
3    ARE                                 Émirats arabes unis  ADVN      3
56   FRA                                              France  ADVN      3
145  SGP                                           Singapour  ADVN      3
171  TWN                                Chine - Taïwan  ADVN      3
84   JPN                                               Japon  ADVN      3
58   GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…  ADVN      3

        ra_imp    per     score
31   341451.86  0.813  277600.4
107   77126.89  0.998   76972.6
118   53348.08  0.903   48173.3
91    37421.45  0.790   29562.9
3     26984.90  0.892   24070.5
56    43370.71  0.392   17001.3
145   20733.96  0.810   16794.5
171   15451.10  0.987   15250.2
84     7663.33  0.987    7563.7
58    25425.61  0.272    6915.8
```

Meilleures opportunités commerciales cluster SMIS :

```
    ISO3                                  Zone cluCH  cluKM   ra_imp    per  \
37   COM                               Comores  SMIS      2  7252.75  1.000
113  MNG                              Mongolie  SMIS      4  5395.43  1.000
97   LCA                         Sainte-Lucie  SMIS      4  1470.86  1.000
93   LAO  République démocratique populaire lao  SMIS      2  1231.64  0.995
17   BHS                               Bahamas  SMIS      4  1141.22  1.000
26   BWA                              Botswana  SMIS      4   952.21  0.998
146  SLB                         Îles Salomon  SMIS      2   832.61  1.000
181  VUT                               Vanuatu  SMIS      4   630.94  1.000
```

```
157  SWZ                              Eswatini  SMIS      2   536.19  0.814
25   BTN                               Bhoutan  SMIS      4   533.97  0.980

      score
37   7252.8
113  5395.4
97   1470.9
93   1225.5
17   1141.2
26    950.3
146   832.6
181   630.9
157   436.5
25    523.3
```

Opportunités les plus pertinentes cluster SMIS :

```
    ISO3                                  Zone cluCH  cluKM   ra_imp    per  \
37   COM                               Comores  SMIS      2  7252.75  1.000
113  MNG                              Mongolie  SMIS      4  5395.43  1.000
97   LCA                         Sainte-Lucie  SMIS      4  1470.86  1.000
93   LAO  République démocratique populaire lao  SMIS      2  1231.64  0.995
17   BHS                               Bahamas  SMIS      4  1141.22  1.000
26   BWA                              Botswana  SMIS      4   952.21  0.998
146  SLB                          Îles Salomon  SMIS      2   832.61  1.000
181  VUT                               Vanuatu  SMIS      4   630.94  1.000
25   BTN                               Bhoutan  SMIS      4   533.97  0.980
157  SWZ                              Eswatini  SMIS      2   536.19  0.814

      score
37   7252.8
113  5395.4
97   1470.9
93   1225.5
17   1141.2
26    950.3
146   832.6
181   630.9
25    523.3
157   436.5
```

Meilleures opportunités commerciales cluster LOWD :

```
    ISO3                             Zone cluCH  cluKM    ra_imp    per  \
61   GIN                           Guinée  LOWD      2  12546.04  1.000
164  TKM                     Turkménistan  LOWD      2   9893.53  1.000
115  MRT                       Mauritanie  LOWD      2   9838.74  1.000
147  SLE                     Sierra Leone  LOWD      2   8734.13  1.000
```

```
114  MOZ                          Mozambique  LOWD      2   8502.94  0.999
185  ZMB                              Zambie  LOWD      2   8353.39  0.746
62   GMB                              Gambie  LOWD      2   7796.71  1.000
72   HTI                               Haïti  LOWD      2   7464.53  1.000
34   COD  République démocratique du Congo  LOWD      2   7433.72  1.000
87   KGZ                        Kirghizistan  LOWD      2   7327.13  0.993

       score
61   12546.0
164   9893.5
115   9838.7
147   8734.1
114   8494.4
185   6231.6
62    7796.7
72    7464.5
34    7433.7
87    7275.8


Opportunités les plus pertinentes cluster LOWD :
    ISO3                            Zone cluCH  cluKM    ra_imp    per  \
61   GIN                          Guinée  LOWD      2  12546.04  1.000
164  TKM                    Turkménistan  LOWD      2   9893.53  1.000
115  MRT                      Mauritanie  LOWD      2   9838.74  1.000
147  SLE                    Sierra Leone  LOWD      2   8734.13  1.000
114  MOZ                      Mozambique  LOWD      2   8502.94  0.999
62   GMB                          Gambie  LOWD      2   7796.71  1.000
72   HTI                           Haïti  LOWD      2   7464.53  1.000
34   COD  République démocratique du Congo  LOWD      2   7433.72  1.000
87   KGZ                    Kirghizistan  LOWD      2   7327.13  0.993
185  ZMB                          Zambie  LOWD      2   8353.39  0.746

       score
61   12546.0
164   9893.5
115   9838.7
147   8734.1
114   8494.4
62    7796.7
72    7464.5
34    7433.7
87    7275.8
185   6231.6
```

[109]:
```
for n in list(df_eval["cluKM"].unique()):
    print("\nMeilleures opportunités commerciales cluster",str(n),":")
```

```
    display(df_eval.loc[df_eval["cluKM"]==n].sort_values("ra_imp",␣
↪ascending=False).head(10))
    print("\nOpportunités les plus pertinentes cluster",str(n),":")
    display(df_eval.loc[df_eval["cluKM"]==n].sort_values("score",␣
↪ascending=False).head(10))
```

```
Meilleures opportunités commerciales cluster 0 :
```

	ISO3	Zone	cluCH	cluKM	ra_imp	\
77	IRN	Iran (République islamique d')	FRAG	0	36866.64	
96	LBY	Libye	FRAG	0	25213.86	
183	YEM	Yémen	FRAG	0	15764.76	
64	GNQ	Guinée équatoriale	FRAG	0	4121.74	
151	SSD	Soudan du Sud	FRAG	0	888.03	
135	PRK	République populaire démocratique de Corée	FRAG	0	354.72	
143	SDN	Soudan	FRAG	0	299.02	
50	ERI	Érythrée	FRAG	0	2.28	
111	MMR	Myanmar	FRAG	0	-50.75	
99	LSO	Lesotho	FRAG	0	-303.59	

	per	score
77	0.337	12424.1
96	1.000	25213.9
183	1.000	15764.8
64	0.744	3066.6
151	1.000	888.0
135	1.000	354.7
143	0.984	294.2
50	1.000	2.3
111	0.947	0.0
99	1.000	0.0

```
Opportunités les plus pertinentes cluster 0 :
```

	ISO3	Zone	cluCH	cluKM	ra_imp	\
96	LBY	Libye	FRAG	0	25213.86	
183	YEM	Yémen	FRAG	0	15764.76	
77	IRN	Iran (République islamique d')	FRAG	0	36866.64	
64	GNQ	Guinée équatoriale	FRAG	0	4121.74	
151	SSD	Soudan du Sud	FRAG	0	888.03	
135	PRK	République populaire démocratique de Corée	FRAG	0	354.72	
143	SDN	Soudan	FRAG	0	299.02	
50	ERI	Érythrée	FRAG	0	2.28	
0	AFG	Afghanistan	FRAG	0	-6288.90	
1	AGO	Angola	FRAG	0	-15100.91	

	per	score

```
96   1.000  25213.9
183  1.000  15764.8
77   0.337  12424.1
64   0.744   3066.6
151  1.000    888.0
135  1.000    354.7
143  0.984    294.2
50   1.000      2.3
0    0.996      0.0
1    1.000      0.0
```

Meilleures opportunités commerciales cluster 1 :

```
    ISO3                    Zone cluCH  cluKM    ra_imp    per    score
180  VNM               Viet Nam  MIDV      1  38041.45  0.939  35720.9
60   GHA                  Ghana  MIDV      1  37502.98  1.000  37503.0
35   COG                  Congo  MIDV      1  35088.71  0.998  35018.5
177  UZB             Ouzbékistan  MIDV      1  23757.10  0.994  23614.6
169  TUR                 Türkiye  MIDV      1  21903.08  0.000      0.0
132  PHL             Philippines  MIDV      1  18494.87  0.999  18476.4
131  PER                   Pérou  MIDV      1  16496.45  1.000  16496.4
2    ALB                 Albanie  MIDV      1  11211.02  1.000  11211.0
46   DOM  République dominicaine  MIDV      1  10314.25  0.918   9468.5
67   GTM               Guatemala  MIDV      1   7075.44  0.987   6983.5
```

Opportunités les plus pertinentes cluster 1 :

```
    ISO3                    Zone cluCH  cluKM    ra_imp    per    score
60   GHA                  Ghana  MIDV      1  37502.98  1.000  37503.0
180  VNM               Viet Nam  MIDV      1  38041.45  0.939  35720.9
35   COG                  Congo  MIDV      1  35088.71  0.998  35018.5
177  UZB             Ouzbékistan  MIDV      1  23757.10  0.994  23614.6
132  PHL             Philippines  MIDV      1  18494.87  0.999  18476.4
131  PER                   Pérou  MIDV      1  16496.45  1.000  16496.4
2    ALB                 Albanie  MIDV      1  11211.02  1.000  11211.0
46   DOM  République dominicaine  MIDV      1  10314.25  0.918   9468.5
67   GTM               Guatemala  MIDV      1   7075.44  0.987   6983.5
57   GAB                   Gabon  MIDV      1   4730.50  1.000   4730.5
```

Meilleures opportunités commerciales cluster 3 :

```
    ISO3                                    Zone cluCH  cluKM  \
31   CHN              Chine (continentale)  ADVN      3
123  NLD            Pays-Bas (Royaume des)  ADVN      3
107  MEX                           Mexique  ADVN      3
118  MYS                          Malaisie  ADVN      3
56   FRA                            France  ADVN      3
91   KOR             République de Corée  ADVN      3
```

```
3    ARE                                    Émirats arabes unis  ADVN      3
58   GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…  ADVN      3
145  SGP                                            Singapour  ADVN      3
171  TWN                                        Chine - Taïwan  ADVN      3

        ra_imp    per     score
31   341451.86  0.813  277600.4
123  107274.59  0.000      0.0
107   77126.89  0.998   76972.6
118   53348.08  0.903   48173.3
56    43370.71  0.392   17001.3
91    37421.45  0.790   29562.9
3     26984.90  0.892   24070.5
58    25425.61  0.272    6915.8
145   20733.96  0.810   16794.5
171   15451.10  0.987   15250.2
```

Opportunités les plus pertinentes cluster 3 :

```
    ISO3                                                Zone cluCH  cluKM  \
31   CHN                                Chine (continentale)  ADVN      3
107  MEX                                             Mexique  ADVN      3
118  MYS                                            Malaisie  ADVN      3
91   KOR                                 République de Corée  ADVN      3
3    ARE                                 Émirats arabes unis  ADVN      3
56   FRA                                              France  ADVN      3
145  SGP                                            Singapour  ADVN      3
171  TWN                                        Chine - Taïwan  ADVN      3
84   JPN                                               Japon  ADVN      3
58   GBR  Royaume-Uni de Grande-Bretagne et d'Irlande du…  ADVN      3

        ra_imp    per     score
31   341451.86  0.813  277600.4
107   77126.89  0.998   76972.6
118   53348.08  0.903   48173.3
91    37421.45  0.790   29562.9
3     26984.90  0.892   24070.5
56    43370.71  0.392   17001.3
145   20733.96  0.810   16794.5
171   15451.10  0.987   15250.2
84     7663.33  0.987    7563.7
58    25425.61  0.272    6915.8
```

Meilleures opportunités commerciales cluster 4 :

```
    ISO3                          Zone cluCH  cluKM   ra_imp    per   score
113  MNG                      Mongolie  SMIS      4  5395.43  1.000  5395.4
103  MAC          Chine - RAS de Macao  ADVN      4  2150.08  1.000  2150.1
```

```
97   LCA                      Sainte-Lucie  SMIS      4  1470.86  1.000  1470.9
112  MNE                        Monténégro  MIDV      4  1201.71  0.999  1200.5
17   BHS                           Bahamas  SMIS      4  1141.22  1.000  1141.2
26   BWA                          Botswana  SMIS      4   952.21  0.998   950.3
181  VUT                           Vanuatu  SMIS      4   630.94  1.000   630.9
25   BTN                           Bhoutan  SMIS      4   533.97  0.980   523.3
178  VCT  Saint-Vincent-et-les Grenadines  SMIS      4   417.35  1.000   417.4
152  STP              Sao Tomé-et-Principe  SMIS      4   373.29  1.000   373.3
```

Opportunités les plus pertinentes cluster 4 :

```
    ISO3                              Zone cluCH  cluKM   ra_imp    per   score
113  MNG                           Mongolie  SMIS      4  5395.43  1.000  5395.4
103  MAC              Chine - RAS de Macao  ADVN      4  2150.08  1.000  2150.1
97   LCA                      Sainte-Lucie  SMIS      4  1470.86  1.000  1470.9
112  MNE                        Monténégro  MIDV      4  1201.71  0.999  1200.5
17   BHS                           Bahamas  SMIS      4  1141.22  1.000  1141.2
26   BWA                          Botswana  SMIS      4   952.21  0.998   950.3
181  VUT                           Vanuatu  SMIS      4   630.94  1.000   630.9
25   BTN                           Bhoutan  SMIS      4   533.97  0.980   523.3
178  VCT  Saint-Vincent-et-les Grenadines  SMIS      4   417.35  1.000   417.4
152  STP              Sao Tomé-et-Principe  SMIS      4   373.29  1.000   373.3
```

Meilleures opportunités commerciales cluster 2 :

```
    ISO3                                Zone cluCH  cluKM    ra_imp    per  \
61   GIN                              Guinée  LOWD      2  12546.04  1.000
164  TKM                        Turkménistan  LOWD      2   9893.53  1.000
115  MRT                          Mauritanie  LOWD      2   9838.74  1.000
147  SLE                        Sierra Leone  LOWD      2   8734.13  1.000
114  MOZ                          Mozambique  LOWD      2   8502.94  0.999
185  ZMB                              Zambie  LOWD      2   8353.39  0.746
62   GMB                              Gambie  LOWD      2   7796.71  1.000
72   HTI                               Haïti  LOWD      2   7464.53  1.000
34   COD  République démocratique du Congo  LOWD      2   7433.72  1.000
87   KGZ                        Kirghizistan  LOWD      2   7327.13  0.993

      score
61   12546.0
164   9893.5
115   9838.7
147   8734.1
114   8494.4
185   6231.6
62    7796.7
72    7464.5
34    7433.7
87    7275.8
```

Opportunités les plus pertinentes cluster 2 :

	ISO3	Zone	cluCH	cluKM	ra_imp	per	\
61	GIN	Guinée	LOWD	2	12546.04	1.000	
164	TKM	Turkménistan	LOWD	2	9893.53	1.000	
115	MRT	Mauritanie	LOWD	2	9838.74	1.000	
147	SLE	Sierra Leone	LOWD	2	8734.13	1.000	
114	MOZ	Mozambique	LOWD	2	8502.94	0.999	
62	GMB	Gambie	LOWD	2	7796.71	1.000	
72	HTI	Haïti	LOWD	2	7464.53	1.000	
34	COD	République démocratique du Congo	LOWD	2	7433.72	1.000	
87	KGZ	Kirghizistan	LOWD	2	7327.13	0.993	
37	COM	Comores	SMIS	2	7252.75	1.000	

	score
61	12546.0
164	9893.5
115	9838.7
147	8734.1
114	8494.4
62	7796.7
72	7464.5
34	7433.7
87	7275.8
37	7252.8