2-clustering-visualisations-112023

May 5, 2024

Partie 2 : ACP + clustering

[ ]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

import seaborn as sns

from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
```

[4]:
```
import sklearn
```

[5]:
```
from sklearn.preprocessing import StandardScaler #pour standaridiser les données
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans, DBSCAN
from sklearn.decomposition import PCA
```

[28]:
```
from statistics import mean
```

[395]:
```
data_set = pd.read_csv('/Users/helmisaddem/Documents/new_data_set.csv')
```

[396]:
```
data_set.head()
```

[396]:

	zone	population_milles_hab	evolution_population_pct	consommation_volaille_pct	TDI
0	Afrique du Sud	57009.756	1.40	6.37	24.27
1	Albanie	2884.169	-0.45	1.29	74.51
2	Algérie	41389.189	1.72	0.80	0.72
3	Allemagne	82658.409	0.12	2.06	49.24
4	Angola	29816.766	3.57	1.85	86.83

```
   disponibilite_alimentaire_en_quantite_(kg/personne/an)  PIB_par_habitant  \
0                                                  35.69          6723.928582
1                                                  16.36          4521.752217
2                                                   6.38          4134.936055
3                                                  19.47         44670.222282
4                                                  10.56          4042.681403

   indice_stabilite_politique  pib_pct_croissance  RNB_par_habitant
0                       -0.28           17.703799       6537.504869
1                        0.38            9.765943       4532.791977
2                       -0.92            6.287892       4081.775086
3                        0.59            6.441276      45734.557501
4                       -0.38           20.766646       3791.443851
```

[397]:
```
data_set_only_quant = data_set.drop('zone', axis='columns')
```

[ ]:
```
#standaridisation des données : les centrer et les réduire
scaler = StandardScaler()
data_set_only_quant_scaled = scaler.fit_transform(data_set_only_quant)
print(data_set_only_quant_scaled)
```

[399]:
```
pays = data_set['zone'].to_numpy()
indicators = data_set_only_quant.columns.to_numpy()
```

[ ]:
```
# ANALYSE EN COMPOSANTE PRINCIPALE :
#En utilisant l'ACP, nous allons chercher à synthétiser les informations
# tout en réduisant le nombre de dimensions à analyser.
```

[400]:
```
PCA().fit(data_set_only_quant_scaled)
```

[400]:
```
PCA()
```

[392]:
```
np.arange(1, (len(PCA().fit(data_set_only_quant_scaled).explained_variance_)+1))
```

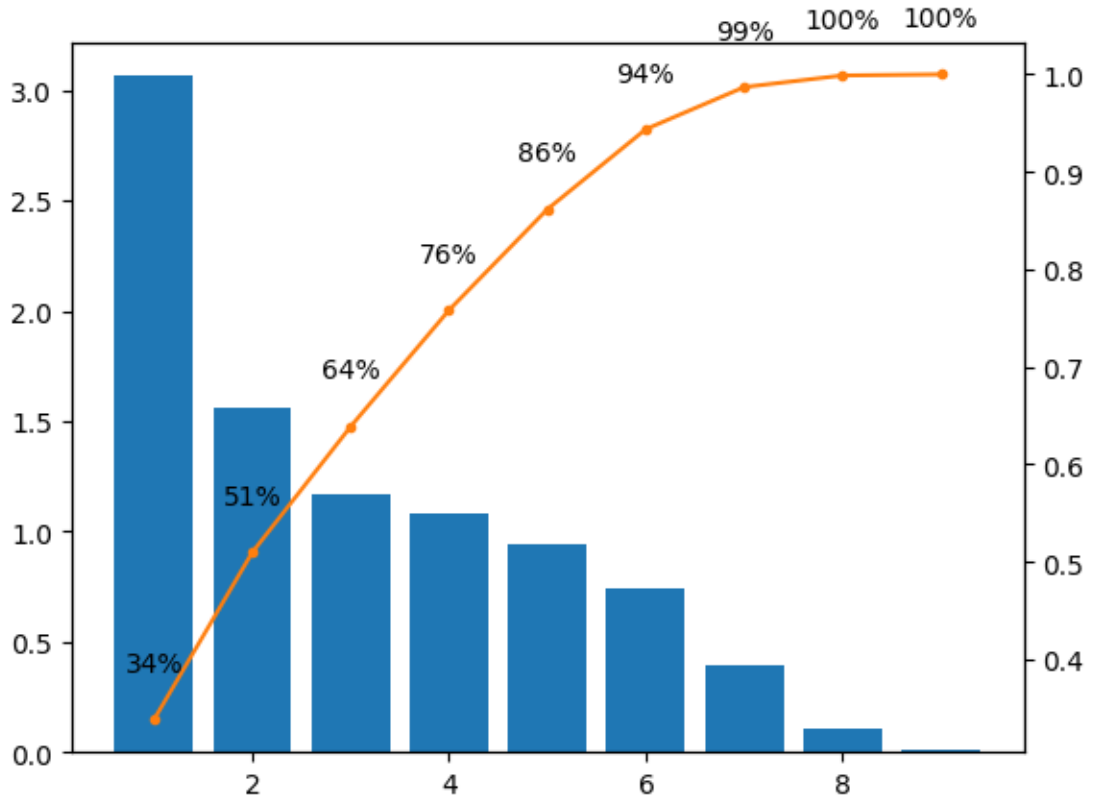
[392]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

[401]:
```
PCA().fit(data_set_only_quant_scaled).explained_variance_ratio_.cumsum()
```

[401]:
```
array([0.33786889, 0.5096151 , 0.63872055, 0.75821109, 0.86168613,
       0.94377094, 0.98696582, 0.99889778, 1.        ])
```

[402]:
```
plt.bar(np.arange(1, (len(PCA().fit(data_set_only_quant_scaled).
 ↪explained_variance_)+1)), PCA().fit(data_set_only_quant_scaled).
 ↪explained_variance_)
plt.twinx()
```

```
plt.plot(np.arange(1, (len(PCA().fit(data_set_only_quant_scaled).
  ↪explained_variance_)+1)), PCA().fit(data_set_only_quant_scaled).
  ↪explained_variance_ratio_.cumsum(), color='C1', marker='.')
for x, y in enumerate(PCA().fit(data_set_only_quant_scaled).
  ↪explained_variance_ratio_.cumsum()):
    plt.text(x+1, (y+0.05) , f'{y:.0%}', ha='center')
```



[403]:
```
#4 dimensions ou composantes explique 76% de la quantité de variance
pd.DataFrame(np.transpose(PCA(n_components=4).fit(data_set_only_quant_scaled).
  ↪components_),
             index=indicators,
    columns=['CP1', 'CP2', 'CP3', 'CP4'])
```

[403]:

	CP1	CP2	\
population_milles_hab	-0.127009	0.052949	
evolution_population_pct	-0.152917	-0.152026	
consommation_volaille_pct	0.271385	-0.681164	
TDI	0.234470	0.141201	
disponibilite_alimentaire_en_quantite_(kg/perso…	0.383099	-0.554544	
PIB_par_habitant	0.489474	0.264099	
indice_stabilite_politique	0.444386	0.201729	

```
pib_pct_croissance                                    0.059065  0.103357
RNB_par_habitant                                      0.494511  0.248029

                                                           CP3       CP4
population_milles_hab                                 0.300391  0.751046
evolution_population_pct                             -0.621634  0.246853
consommation_volaille_pct                             0.038969  0.055403
TDI                                                  -0.334228 -0.371010
disponibilite_alimentaire_en_quantite_(kg/perso…  0.106675  0.016416
PIB_par_habitant                                     -0.136532  0.292778
indice_stabilite_politique                            0.146291 -0.183411
pib_pct_croissance                                    0.585316 -0.163400
RNB_par_habitant                                     -0.126862  0.296560
```

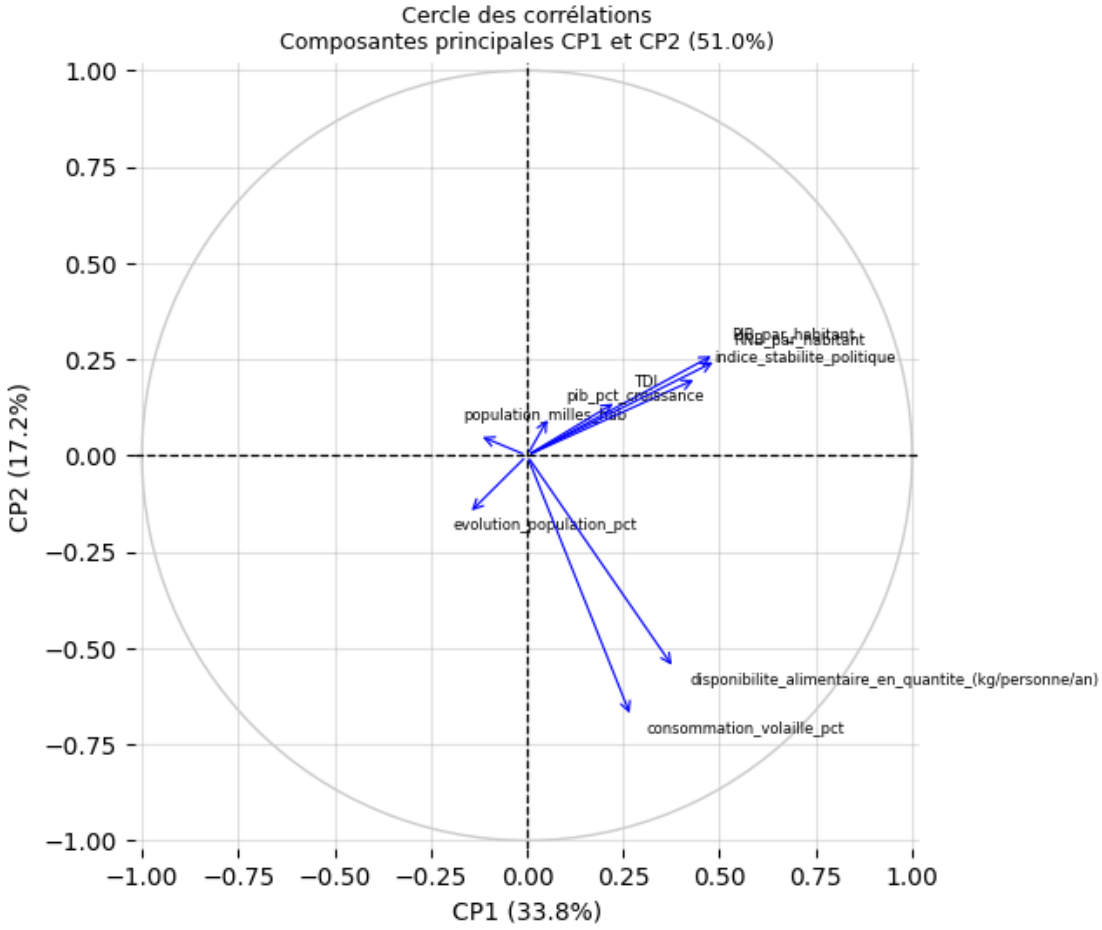
[ ]:

[404]:
```
def cercle_correlation_graph(dimension_sur_x, dimension_sur_y,␣
↪pca=PCA(n_components=4).fit(data_set_only_quant_scaled), text_offset=0.04):
    '''
    Trace le cercle des corrérations d'une ACP.
    Paramètres :
    - x_d : la dimension de l'ACP a représenter sur l'axe x.
    - y_d : la dimension de l'ACP a représenter sur l'axe y.
    - pca : la décomposition effectuée.
    - text_offset : le décalage pour le positionnement des noms des
    indicateurs.
    '''


    vecteurs_propres = pca.components_.T
    cum_variance_ratio = (pca.explained_variance_ratio_[dimension_sur_x]
                          + pca.explained_variance_ratio_[dimension_sur_y])


    fig, ax = plt.subplots(figsize=(6.4, 6.4))


    ax.set_aspect('equal')
    ax.grid(alpha=0.4)
    ax.set_axisbelow(True)
    ax.set_xlim(-1.02, 1.02)
    ax.set_ylim(-1.02, 1.02)

    for spine in ax.spines.values():
        spine.set_visible(False)

    circle = plt.Circle((0, 0), 1, fill=False, linewidth=1, color='0.8')
```

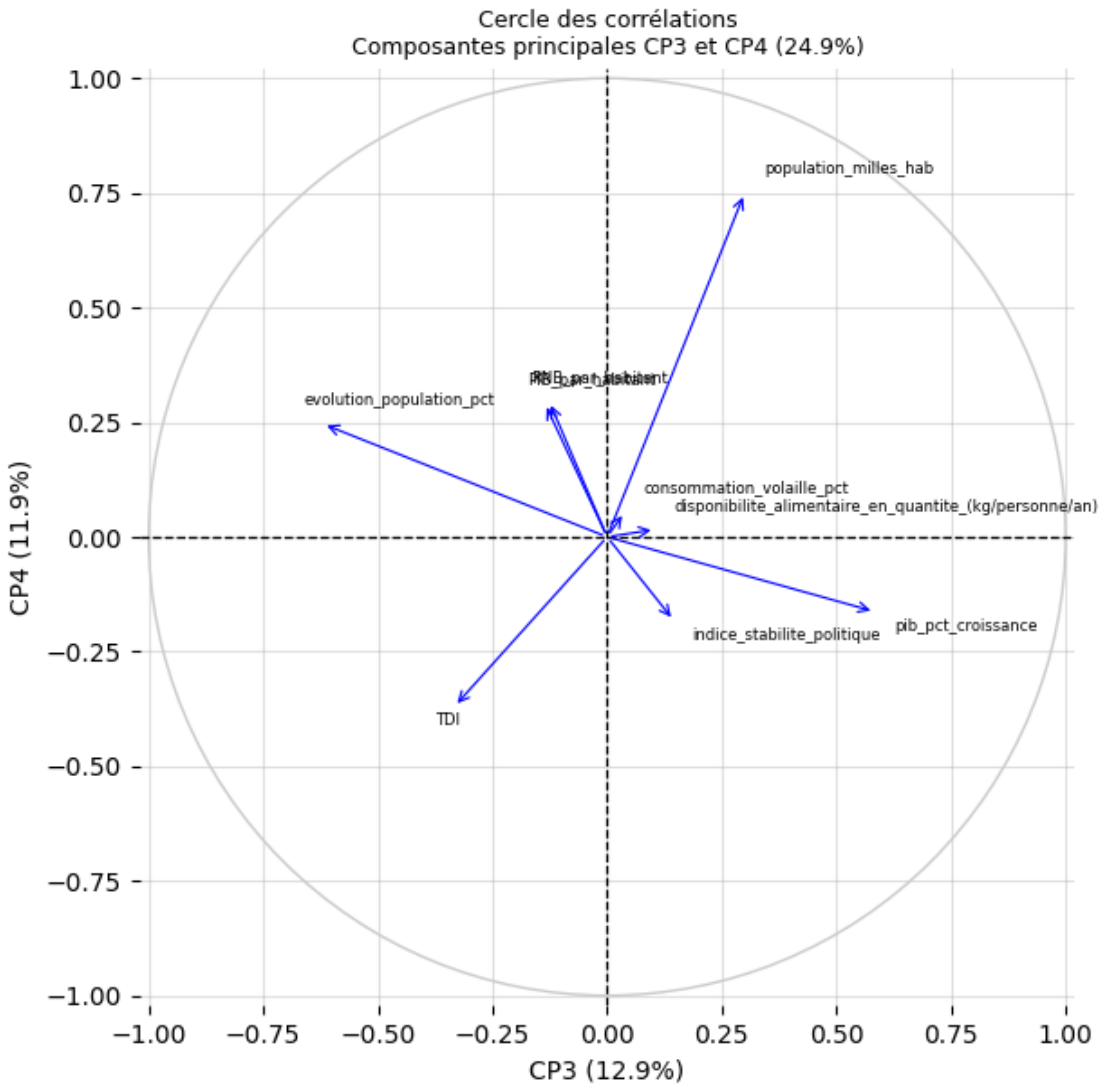
```
    ax.add_patch(circle)
    ax.axhline(y=0, linestyle = '--', linewidth=0.9, color='k')
    ax.axvline(x=0, linestyle = '--', linewidth=0.9, color='k')


    for i in range(len(vecteurs_propres)):

        ax.annotate(
            text='',
            xy=(vecteurs_propres[i, dimension_sur_x], vecteurs_propres[i,␣
↪dimension_sur_y]),
            xytext=(0, 0),
            arrowprops=dict(arrowstyle='->', linewidth=0.8, color='b')
        )


        if vecteurs_propres[i, dimension_sur_x] > 0:
            h_offset=text_offset
        elif vecteurs_propres[i, dimension_sur_x] < 0:
            h_offset=-text_offset

        if vecteurs_propres[i, dimension_sur_y] > 0:
            v_offset=text_offset
        elif vecteurs_propres[i, dimension_sur_y] < 0:
            v_offset=-text_offset


        ax.text(
            x=(vecteurs_propres[i, dimension_sur_x]+h_offset),
            y=(vecteurs_propres[i, dimension_sur_y]+v_offset),
            s=indicators[i], fontsize=6
        )


    ax.set_xlabel(f'CP{dimension_sur_x + 1} ({pca.
↪explained_variance_ratio_[dimension_sur_x]:.1%})')
    ax.set_ylabel(f'CP{dimension_sur_y + 1} ({pca.
↪explained_variance_ratio_[dimension_sur_y]:.1%})')

    ax.set_title(
        f'Cercle des corrélations\n'
        f'Composantes principales CP{dimension_sur_x + 1} et CP{dimension_sur_y␣
↪+ 1} ({cum_variance_ratio:.1%})',
        fontsize= 9)

    plt.tight_layout()
```

```
    plt.show()
```

[405]:
```
cercle_correlation_graph(0, 1)
```



[406]:
```
cercle_correlation_graph(2, 3)
```



[218]:
```
pd.DataFrame(np.transpose(PCA(n_components=4).fit(data_set_only_quant_scaled).
 ↪components_),
             index=indicators,
    columns=['CP1', 'CP2', 'CP3', 'CP4']).style.
 ↪background_gradient(axis='rows', cmap='coolwarm').format('{:.2f}')
```

[218]: <pandas.io.formats.style.Styler at 0x131090850>

[ ]:
```
#Projection des pays :

pca=PCA(n_components=4).fit(data_set_only_quant_scaled)
cum_variance_ratio = (pca.explained_variance_ratio_[0]
                          + pca.explained_variance_ratio_[1])
```

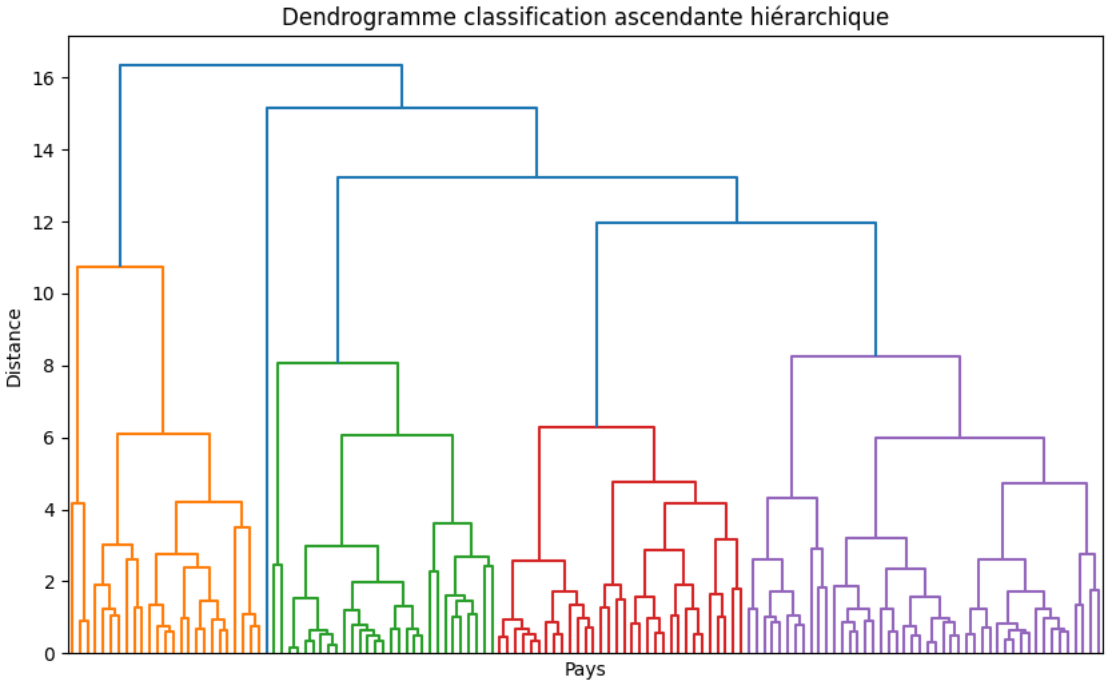
```
fig, ax = plt.subplots(figsize=(8, 8))


ax.set_aspect('equal')
ax.grid(alpha=0.4)
ax.set_axisbelow(True)

for spine in ax.spines.values():
        spine.set_visible(False)

# Tracer les lignes en pointillés pour x = 0  et y = 0
ax.axhline(y=0, linestyle = '--', linewidth=0.9, color='k')
ax.axvline(x=0, linestyle = '--', linewidth=0.9, color='k')

# Afficher la position des individus
ax.scatter(pca.transform(data_set_only_quant_scaled)[:, 0], pca.
 ↪transform(data_set_only_quant_scaled)[:, 1], marker='.')

# Ajouter les noms des individus
for i in range(len(pca.transform(data_set_only_quant_scaled))):
        ax.text(
            x=(pca.transform(data_set_only_quant_scaled)[i, 0]+0.04),
            y=(pca.transform(data_set_only_quant_scaled)[i, 1]+0.04),
            s=pays[i], fontsize='x-small'
        )

# Ajouter les titre (axes et figure)
ax.set_xlabel(f'CP{0 + 1} ({pca.explained_variance_ratio_[0]:.1%})')
ax.set_ylabel(f'CP{1 + 1} ({pca.explained_variance_ratio_[1]:.1%})')

ax.set_title(f'Observations sur CP{0 + 1} et CP{1 + 1} '
                f'({cum_variance_ratio:.1%})')

plt.tight_layout()

plt.show()
```

[408]:
```
data_set_only_quant = data_set_only_quant[["consommation_volaille_pct", "TDI",␣
 ↪"indice_stabilite_politique", "RNB_par_habitant", "population_milles_hab",␣
 ↪"pib_pct_croissance"]]
```

[409]:
```
scaler = StandardScaler()
data_set_only_quant_scaled = scaler.fit_transform(data_set_only_quant)
```

[410]:
```
indicators = data_set_only_quant.columns.to_numpy()
```

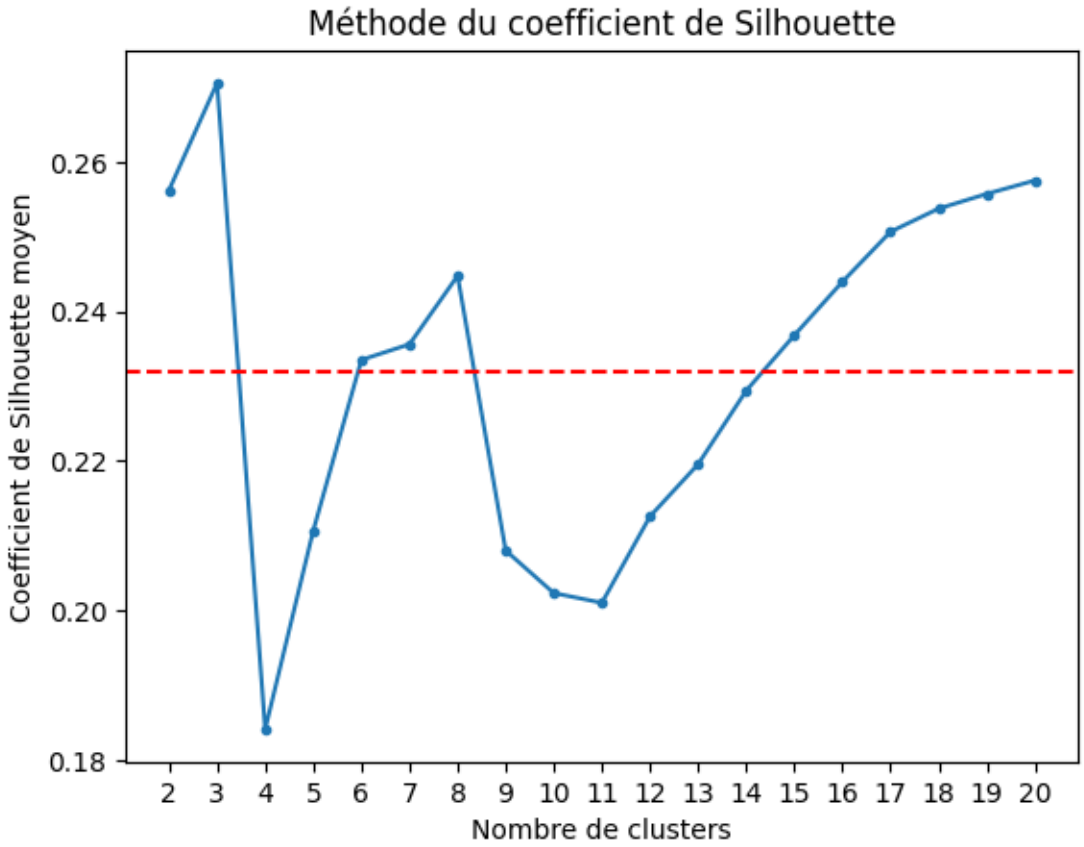
[411]:
```
#matrice de liaison
matrice_liaison = linkage(data_set_only_quant_scaled, method='ward',␣
 ↪metric='euclidean')

# dendrogramme
fig, ax = plt.subplots(figsize=(10, 6))

dendrogramme = dendrogram(matrice_liaison, no_labels=True, orientation='top',␣
 ↪ax=ax)

ax.set_title('Dendrogramme classification ascendante hiérarchique')
ax.set_xlabel('Pays')
ax.set_ylabel('Distance')

plt.show()

del  fig, ax, dendrogramme
```



[412]:
```
nbre_clusters = range(2, 21)
silhouette_coefs = []

# Calcul du score  Silhouette moyen pour chaque valeur du nombre de clusters
for i in nbre_clusters:
    clustering = fcluster(matrice_liaison, t = i, criterion = 'maxclust')
```

```
    silhouette_coef_moy = silhouette_score(data_set_only_quant_scaled,␣
 ↪clustering)
    silhouette_coefs.append(silhouette_coef_moy)
```

[288]:
```
# Traçage du graph
fig, ax = plt.subplots()

plt.plot(nbre_clusters, silhouette_coefs, marker='.')
plt.axhline(y=mean(silhouette_coefs), linestyle = '--', color='r')

ax.xaxis.set_major_locator(ticker.IndexLocator(1, 0))

ax.set_xlabel('Nombre de clusters')
ax.set_ylabel('Coefficient de Silhouette moyen')
ax.set_title('Méthode du coefficient de Silhouette')

plt.show()
```



[ ]:

[413]:
```
#nombre de clusters choisi = 15

# nombre de cluster = 15
clustering = fcluster(matrice_liaison, t=15, criterion='maxclust')


data_set['cluster'] = clustering
#Cela nous permet notamment d'affiche les pays qui composent chaque groupe.


clusters = np.sort(data_set['cluster'].unique())

# Liste des pays de chaque cluster
for cluster in clusters:
    n_pays = len(
        data_set.loc[data_set['cluster'] == cluster, 'zone'])
    liste_pays = list(
        data_set.loc[data_set['cluster'] == cluster, 'zone'])

    print(f'Cluster {cluster} - {n_pays} pays')
    print(f'{liste_pays}')

del clusters, cluster, n_pays, liste_pays
```

```
Cluster 1 - 3 pays
['Belgique', 'Chine - RAS de Hong-Kong', 'Pays-Bas']
Cluster 2 - 7 pays
['Chine - RAS de Macao', 'Danemark', 'Irlande', 'Islande', 'Luxembourg',
'Norvège', 'Suisse']
Cluster 3 - 15 pays
['Allemagne', 'Australie', 'Autriche', 'Canada', 'Espagne', "États-Unis
d'Amérique", 'Finlande', 'France', 'Grèce', 'Italie', 'Japon', 'Nouvelle-
Zélande', 'République de Corée', "Royaume-Uni de Grande-Bretagne et d'Irlande du
Nord", 'Suède']
Cluster 4 - 2 pays
['Égypte', 'Venezuela (République bolivarienne du)']
Cluster 5 - 18 pays
['Algérie', 'Azerbaïdjan', 'Bosnie-Herzégovine', 'Cameroun', "Côte d'Ivoire",
'Kenya', 'Niger', 'Ouganda', 'Paraguay', 'Pérou', 'République populaire
démocratique de Corée', 'Rwanda', 'Sénégal', 'Serbie', 'Sri Lanka', 'Thaïlande',
'Togo', 'Viet Nam']
Cluster 6 - 9 pays
['Indonésie', 'Liban', 'Nigéria', 'Pakistan', 'Philippines', 'Tchad', 'Tunisie',
'Turquie', 'Yémen']
Cluster 7 - 13 pays
['Belize', 'Colombie', 'El Salvador', 'Équateur', 'Guatemala', 'Guyana',
'Honduras', "Iran (République islamique d')", 'Jordanie', 'Maroc', 'Mexique',
```

```
'Myanmar', 'Nicaragua']
Cluster 8 - 4 pays
['Antigua-et-Barbuda', 'Émirats arabes unis', 'Koweït', 'Polynésie française']
Cluster 9 - 15 pays
['Afrique du Sud', 'Arabie saoudite', 'Argentine', 'Barbade', 'Bolivie (État
plurinational de)', 'Brésil', 'Chili', 'Costa Rica', 'Israël', 'Jamaïque',
'Malaisie', 'Maurice', 'Panama', 'République dominicaine', 'Trinité-et-Tobago']
Cluster 10 - 8 pays
['Angola', 'Bélarus', 'Eswatini', 'Kazakhstan', 'Namibie', 'République de
Moldova', 'Roumanie', 'Zambie']
Cluster 11 - 3 pays
['Fédération de Russie', 'Iraq', 'Ukraine']
Cluster 12 - 17 pays
['Botswana', 'Chypre', 'Croatie', 'Estonie', 'Fidji', 'Hongrie', 'Lettonie',
'Lituanie', 'Malte', 'Oman', 'Pologne', 'Portugal', 'Saint-Kitts-et-Nevis',
'Slovaquie', 'Slovénie', 'Tchéquie', 'Uruguay']
Cluster 13 - 14 pays
['Albanie', 'Arménie', 'Bénin', 'Bulgarie', 'Congo', 'Gabon', 'Géorgie',
'Ghana', 'Îles Salomon', 'Kirghizistan', 'Lesotho', 'Macédoine du Nord',
'Monténégro', 'Suriname']
Cluster 14 - 4 pays
['Dominique', 'Libéria', 'Samoa', 'Sierra Leone']
Cluster 15 - 1 pays
['Inde']
```

[414]:
```
exploration_clusters = data_set.groupby('cluster')[indicators].mean().
↪reset_index()
```

[ ]:
```
#exploration_kmeans_clusters = pd.DataFrame(
    #scaler.inverse_transform(clustering.cluster_centers_)).T


#exploration_kmeans_clusters = exploration_kmeans_clusters.set_index(indicators)
#exploration_clusters.columns = exploration_clusters.columns + 1
```

[385]:
```
exploration_clusters.head(15)
```

[385]:

	cluster	consommation_volaille_pct	TDI	\
0	1	3.373333	260.396667	
1	2	2.358571	59.858571	
2	3	3.006000	22.313333	
3	4	2.735000	6.480000	
4	5	1.013333	7.452778	
5	6	1.530000	8.952222	
6	7	4.136923	8.433846	
7	8	6.417500	94.532500	
8	9	6.244667	13.230667	

```
9        10                    1.843750   43.381250
10       11                    3.346667   33.143333
11       12                    2.307059   46.102353
12       13                    2.010000   81.818571
13       14                    2.252500   80.337500
14       15                    0.420000    0.000000

    indice_stabilite_politique  RNB_par_habitant  population_milles_hab  \
0                     0.726667      46731.755325           1.191581e+04
1                     1.201429      70869.080466           3.683796e+03
2                     0.685333      41967.226890           6.173718e+04
3                    -1.345000       4792.219893           6.292254e+04
4                    -0.515556       2995.464200           2.828821e+04
5                    -1.641111       3820.471020           1.012027e+05
6                    -0.460769       4674.970965           3.155107e+04
7                     0.400000      28012.604473           3.478707e+03
8                     0.122667      13501.765827           2.908794e+04
9                    -0.018750       5186.052115           1.268021e+04
10                   -1.606667       5823.204603           7.585686e+04
11                    0.805294      16690.372366           5.840611e+03
12                   -0.095714       3945.669188           5.472119e+03
13                    0.502500       3010.396534           3.114365e+03
14                   -0.760000       1916.967716           1.338677e+06

    pib_pct_croissance
0             6.155631
1             9.205294
2             5.233780
3           -20.932630
4             7.455744
5             1.187660
6             5.480069
7             6.671519
8             8.510214
9            18.286085
10           18.578571
11            9.451359
12            8.551799
13           -2.824631
14           14.569955
```

[ ]:
```
data_set.groupby('cluster')[indicators].mean().reset_index().
↪drop(columns=["cluster"]).T.style.background_gradient(axis='columns',␣
↪cmap='coolwarm').format('{:.2f}')
```

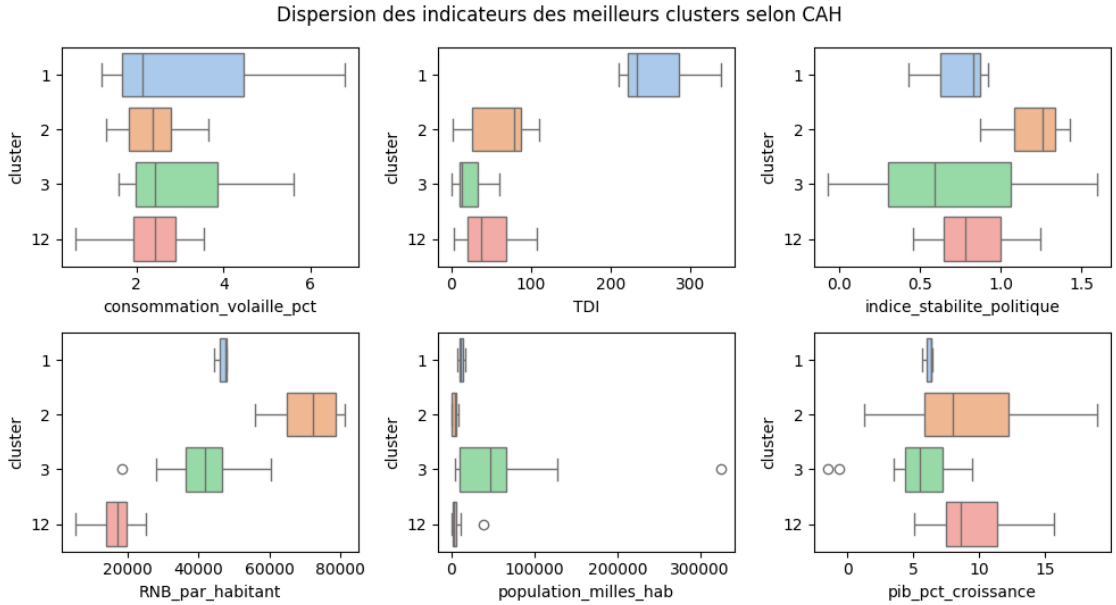
[374]:
```
exploration_clusters.loc[exploration_clusters['cluster'].isin([1, 2, 3, 12])].
↪round(2).reset_index().drop(columns=["index"])
```

[374]:

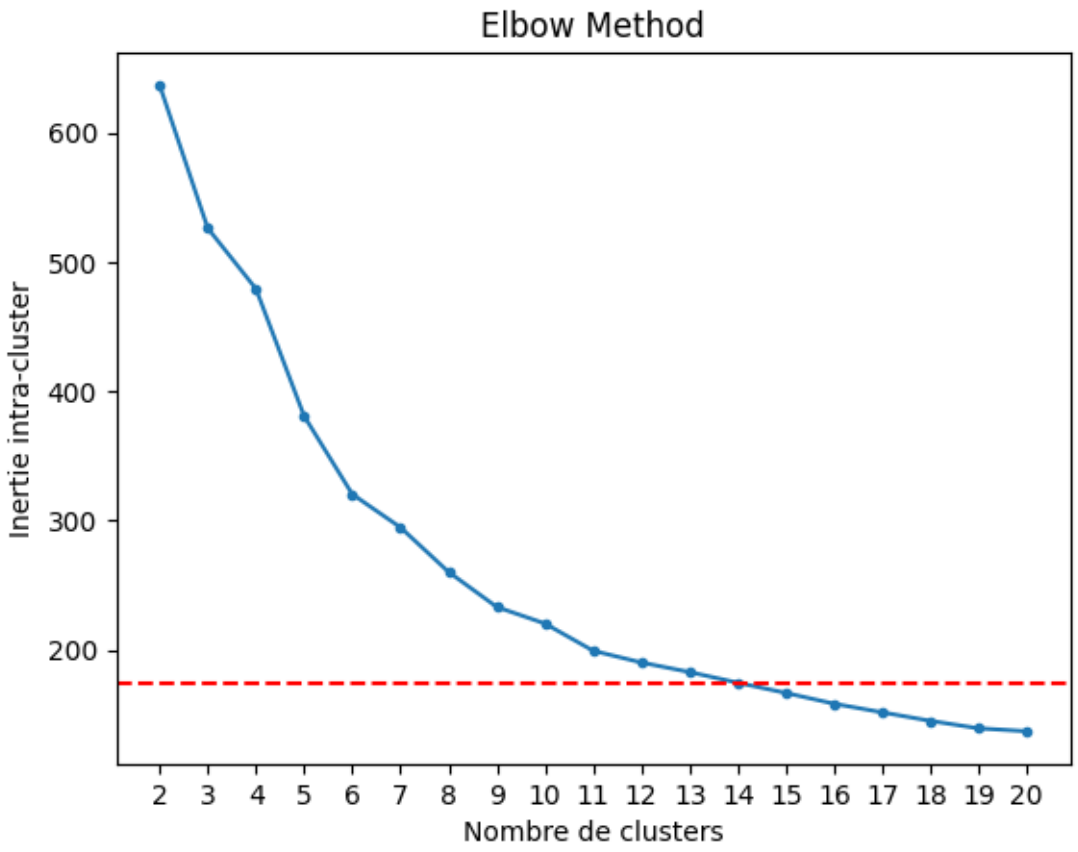
	cluster	consommation_volaille_pct	TDI	indice_stabilite_politique	\
0	1	3.37	260.40	0.73	
1	2	2.36	59.86	1.20	
2	3	3.01	22.31	0.69	
3	12	2.31	46.10	0.81	

	RNB_par_habitant	population_milles_hab	pib_pct_croissance
0	46731.76	11915.81	6.16
1	70869.08	3683.80	9.21
2	41967.23	61737.18	5.23
3	16690.37	5840.61	9.45

[377]:
```
fig, axs = plt.subplots(2, 3, figsize=(10, 5.5))

for i, indicateur in enumerate(indicators):
        r = i // 3
        c = i % 3
        sns.boxplot(data=data_set.loc[data_set["cluster"].isin([1, 2, 3, 12])].
 ↪round(2), x=indicateur,
                    y="cluster", orient='h',
                    ax=axs[r, c], palette="pastel", hue="cluster", legend=False)

    # Supprime les graphiques vides
[fig.delaxes(ax) for ax in axs.flatten() if not ax.has_data()]

fig.suptitle('Dispersion des indicateurs des meilleurs clusters selon CAH')

fig.tight_layout()

plt.show()
```



[416]:
```
# Méthode K-means :

#Calcul de l'inertie intra cluster
inertie_intra_cluster = []
for i in range(2,21):
    clustering = KMeans(n_clusters=i, n_init='auto', init='random',
                        random_state=0)
    clustering = clustering.fit(data_set_only_quant_scaled)
    inertie_intra_cluster.append(clustering.inertia_)

# Traçage du graphique
fig, ax = plt.subplots()

plt.plot(range(2,21), inertie_intra_cluster, marker='.')
plt.axhline(y=inertie_intra_cluster[12], linestyle = '--', color='r')


ax.xaxis.set_major_locator(ticker.IndexLocator(1, 0))

ax.set_xlabel('Nombre de clusters')
ax.set_ylabel('Inertie intra-cluster')
ax.set_title('Elbow Method')

plt.show()

del inertie_intra_cluster, i, clustering, fig, ax
```

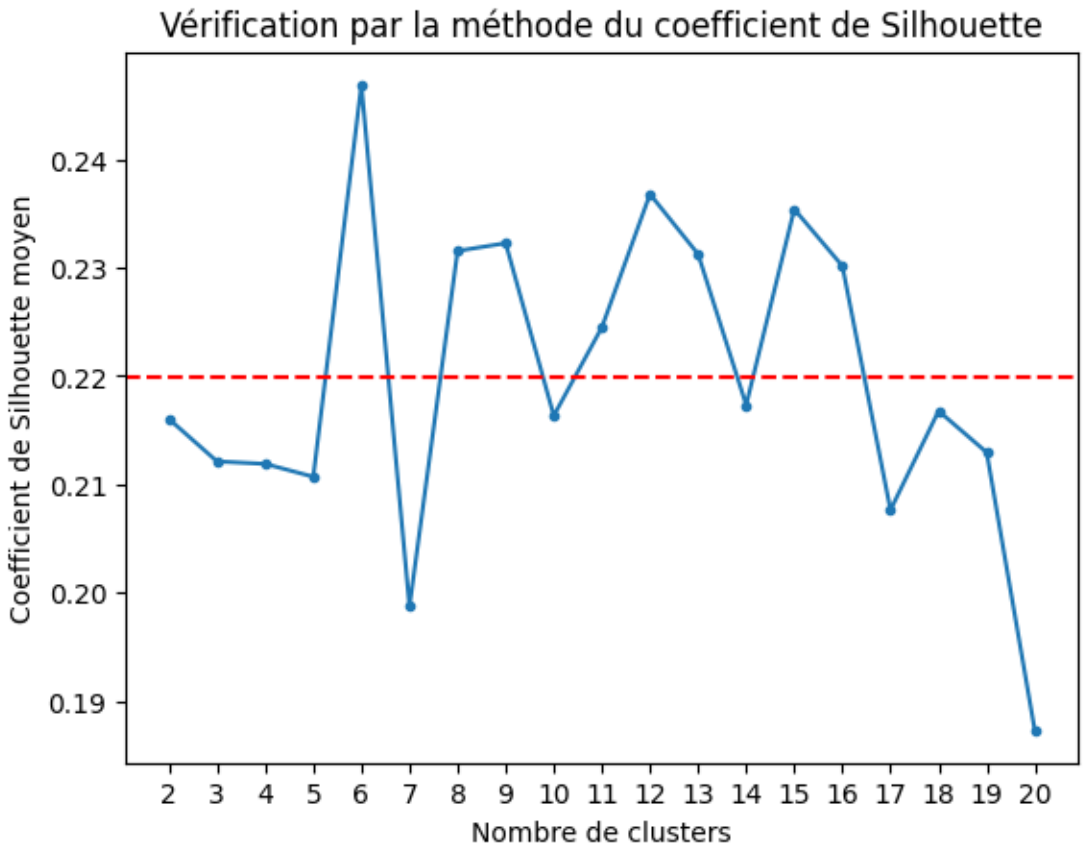


[ ]:
```
#droite rouge ou seuil : valeur à partir de laquelle la courbe commence à␣
↪s'aplatir
```

[378]:
```
# Verification par le coefficient de silouhette :

silhouette_coefs = []


for i in range(2, 21):
    clustering = KMeans(n_clusters=i, n_init='auto', init='random',
                        random_state=0)
    clustering = clustering.fit(data_set_only_quant_scaled)
    silhouette_coef_moy = silhouette_score(data_set_only_quant_scaled,␣
↪clustering.labels_)
    silhouette_coefs.append(silhouette_coef_moy)
```

[379]:
```
# Trace le graphique
fig, ax = plt.subplots()

plt.plot(range(2, 21), silhouette_coefs, marker='.')
```

```
plt.axhline(y=np.mean(silhouette_coefs), linestyle = '--', color='r')

# Définit la position des graduations
ax.xaxis.set_major_locator(ticker.IndexLocator(1, 0))

ax.set_xlabel('Nombre de clusters')
ax.set_ylabel('Coefficient de Silhouette moyen')
ax.set_title('Vérification par la méthode du coefficient de Silhouette')

plt.show()
```



[418]:
```
#Effectue le clustering avec fixation du nombre de clusters à 11
clustering = KMeans(n_clusters=12, n_init='auto', init='random',
                    random_state=0)
clustering.fit(data_set_only_quant_scaled)
```

[418]:
```
KMeans(init='random', n_clusters=12, random_state=0)
```

[419]:
```
data_set['cluster_kmeans'] = clustering.labels_ + 1

clusters = np.sort(data_set['cluster_kmeans'].unique())


for i in clusters:
    nbre_pays = len(
        data_set.loc[data_set['cluster_kmeans'] == i, 'zone'])
    liste_pays = list(
        data_set.loc[data_set['cluster_kmeans'] == i, 'zone'])

    print(f'Cluster {i} - {nbre_pays} pays')
    print(f'{liste_pays}')

del clusters, i, nbre_pays, liste_pays
```

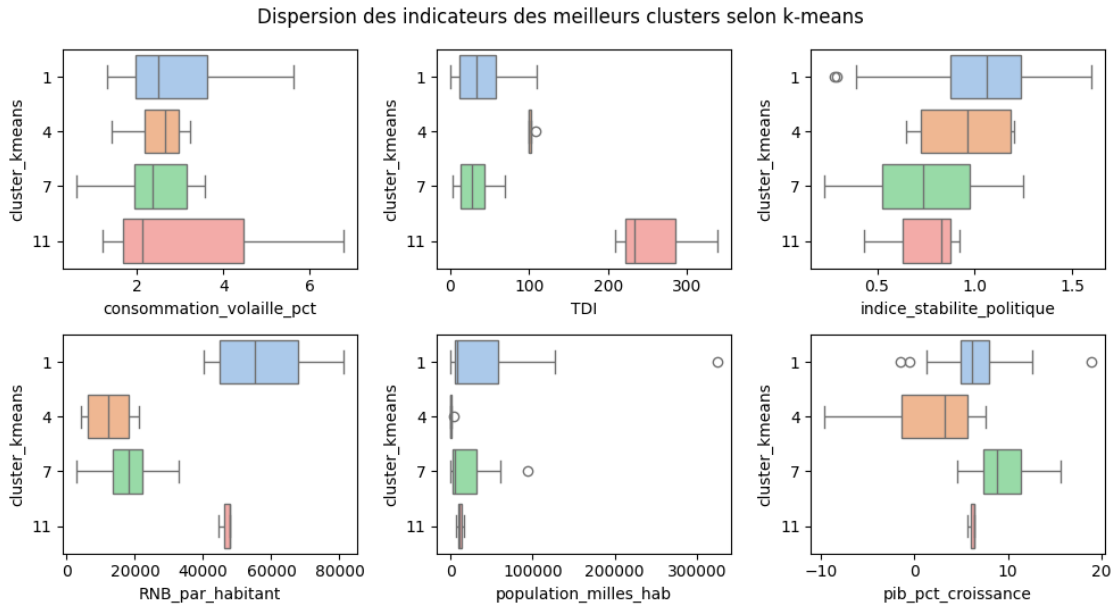
```
Cluster 1 - 18 pays
['Allemagne', 'Australie', 'Autriche', 'Canada', 'Chine - RAS de Macao',
'Danemark', "États-Unis d'Amérique", 'Finlande', 'France', 'Irlande', 'Islande',
'Japon', 'Luxembourg', 'Norvège', 'Nouvelle-Zélande', "Royaume-Uni de Grande-
Bretagne et d'Irlande du Nord", 'Suède', 'Suisse']
Cluster 2 - 19 pays
['Argentine', 'Belize', 'Bolivie (État plurinational de)', 'Brésil', 'Colombie',
'Costa Rica', 'El Salvador', 'Équateur', 'Guatemala', 'Guyana', 'Honduras',
"Iran (République islamique d')", 'Jordanie', 'Maroc', 'Mexique', 'Myanmar',
'Nicaragua', 'Panama', 'République dominicaine']
Cluster 3 - 1 pays
['Inde']
Cluster 4 - 4 pays
['Dominique', 'Oman', 'Saint-Kitts-et-Nevis', 'Samoa']
Cluster 5 - 16 pays
['Albanie', 'Arménie', 'Bénin', 'Bulgarie', 'Congo', 'Gabon', 'Géorgie',
'Ghana', 'Îles Salomon', 'Kirghizistan', 'Lesotho', 'Lettonie', 'Libéria',
'Macédoine du Nord', 'Monténégro', 'Suriname']
Cluster 6 - 21 pays
['Algérie', 'Azerbaïdjan', 'Bosnie-Herzégovine', 'Cameroun', "Côte d'Ivoire",
'Grèce', 'Indonésie', 'Kenya', 'Niger', 'Ouganda', 'Paraguay', 'Pérou',
'République populaire démocratique de Corée', 'Rwanda', 'Sénégal', 'Serbie',
'Sierra Leone', 'Sri Lanka', 'Tchad', 'Thaïlande', 'Togo']
Cluster 7 - 18 pays
['Botswana', 'Chypre', 'Croatie', 'Espagne', 'Estonie', 'Fidji', 'Hongrie',
'Italie', 'Lituanie', 'Malte', 'Pologne', 'Portugal', 'République de Corée',
'Slovaquie', 'Slovénie', 'Tchéquie', 'Uruguay', 'Viet Nam']
Cluster 8 - 2 pays
['Égypte', 'Venezuela (République bolivarienne du)']
Cluster 9 - 10 pays
['Angola', 'Bélarus', 'Eswatini', 'Fédération de Russie', 'Kazakhstan',
```

```
'Namibie', 'République de Moldova', 'Roumanie', 'Ukraine', 'Zambie']
Cluster 10 - 13 pays
['Afrique du Sud', 'Antigua-et-Barbuda', 'Arabie saoudite', 'Barbade', 'Chili',
'Émirats arabes unis', 'Israël', 'Jamaïque', 'Koweït', 'Malaisie', 'Maurice',
'Polynésie française', 'Trinité-et-Tobago']
Cluster 11 - 3 pays
['Belgique', 'Chine - RAS de Hong-Kong', 'Pays-Bas']
Cluster 12 - 8 pays
['Iraq', 'Liban', 'Nigéria', 'Pakistan', 'Philippines', 'Tunisie', 'Turquie',
'Yémen']
```

[382]:
```
#Création d'un DataFrame à partir des centroïdes avec les données non␣
 ↪normalisées
exploration_kmeans_clusters = pd.DataFrame(
    scaler.inverse_transform(clustering.cluster_centers_)).T


exploration_kmeans_clusters = exploration_kmeans_clusters.set_index(indicators)


exploration_kmeans_clusters.columns = exploration_kmeans_clusters.columns + 1


(exploration_kmeans_clusters
 .style.background_gradient(axis='columns', cmap="coolwarm").format('{:.2f}'))
```

[382]:
```
<pandas.io.formats.style.Styler at 0x1380ee3d0>
```

[386]:
```
fig, axs = plt.subplots(2, 3, figsize=(10, 5.5))

for i, indicateur in enumerate(indicators):
        r = i // 3
        c = i % 3
        sns.boxplot(data=data_set.loc[data_set["cluster_kmeans"].isin([1, 4, 7,␣
 ↪11])].round(2), x=indicateur,
                    y="cluster_kmeans", orient='h',
                    ax=axs[r, c], palette="pastel", hue="cluster_kmeans",␣
 ↪legend=False)


[fig.delaxes(ax) for ax in axs.flatten() if not ax.has_data()]

fig.suptitle('Dispersion des indicateurs des meilleurs clusters selon k-means')

fig.tight_layout()

plt.show()
```



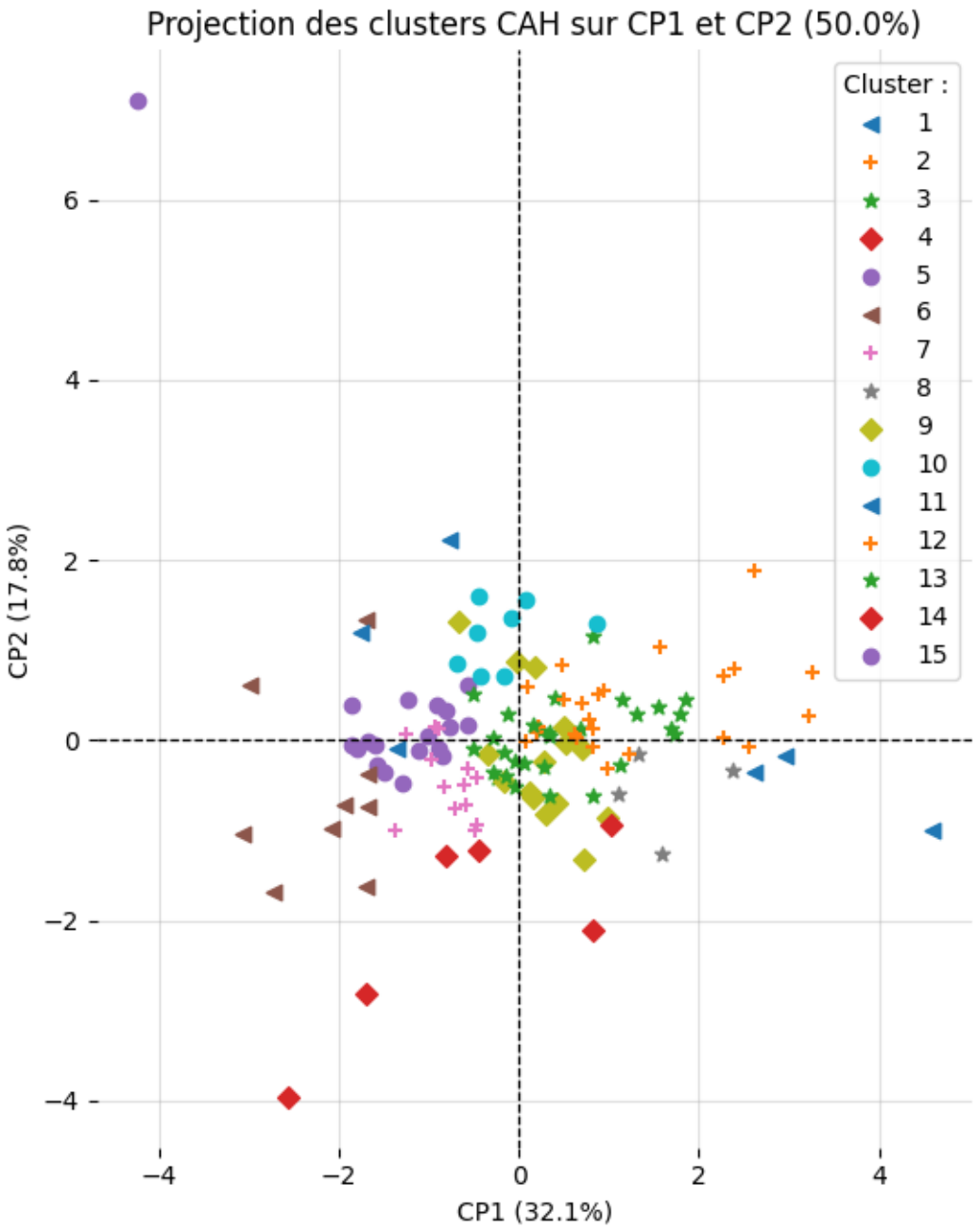
[ ]:
```
#Projection des pays :

pca=PCA(n_components=4).fit(data_set_only_quant_scaled)
cum_variance_ratio = (pca.explained_variance_ratio_[0]
                          + pca.explained_variance_ratio_[1])

fig, ax = plt.subplots(figsize=(8, 8))


ax.set_aspect('equal')
ax.grid(alpha=0.4)
ax.set_axisbelow(True)

for spine in ax.spines.values():
        spine.set_visible(False)


ax.axhline(y=0, linestyle = '--', linewidth=0.9, color='k')
ax.axvline(x=0, linestyle = '--', linewidth=0.9, color='k')

ax.scatter(pca.transform(data_set_only_quant_scaled)[:, 0], pca.
 ↪transform(data_set_only_quant_scaled)[:, 1], marker='.')


for i in range(len(pca.transform(data_set_only_quant_scaled))):
        ax.text(
            x=(pca.transform(data_set_only_quant_scaled)[i, 0]+0.04),
```

```
            y=(pca.transform(data_set_only_quant_scaled)[i, 1]+0.04),
            s=pays[i], fontsize='x-small'
        )


ax.set_xlabel(f'CP{0 + 1} ({pca.explained_variance_ratio_[0]:.1%})')
ax.set_ylabel(f'CP{1 + 1} ({pca.explained_variance_ratio_[1]:.1%})')

ax.set_title(f'Observations sur CP{0 + 1} et CP{1 + 1} '
                f'({cum_variance_ratio:.1%})')

plt.tight_layout()

plt.show()
```

[ ]:
```
PCA(n_components=4).fit(data_set_only_quant_scaled).
↪transform(data_set_only_quant_scaled)
```

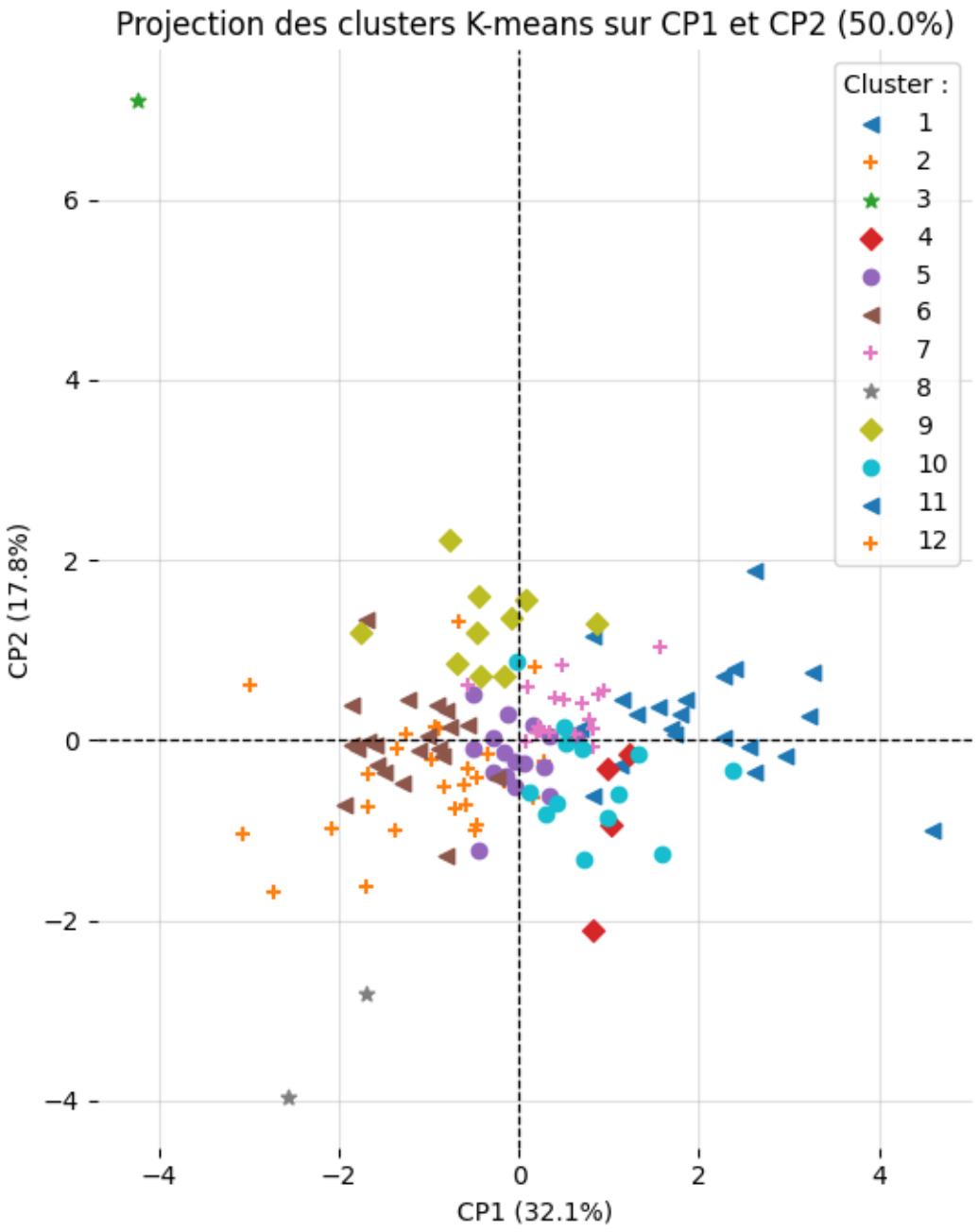
[421]:
```
#projection des clusters obtenus par classification ascendante hiérarichique

pca=PCA(n_components=4).fit(data_set_only_quant_scaled)
cum_variance_ratio = (pca.explained_variance_ratio_[0]
                          + pca.explained_variance_ratio_[1])


df = pd.DataFrame(pca.transform(data_set_only_quant_scaled))
df['cluster'] = data_set["cluster"]
n_clusters = len(df['cluster'].unique())
markers = ['<', '+', '*', 'D', 'o']

fig, ax = plt.subplots(figsize=(7, 7))


ax.set_aspect('equal')
ax.grid(alpha=0.4)
ax.set_axisbelow(True)

for spine in ax.spines.values():
        spine.set_visible(False)


ax.axhline(y=0, linestyle = '--', linewidth=0.9, color='k')
ax.axvline(x=0, linestyle = '--', linewidth=0.9, color='k')


for i in range(1, (n_clusters+1)):
        j = i % len(markers)
```

```
        ax.scatter(
            df.loc[df['cluster'] == i, 0],
            df.loc[df['cluster'] == i, 1],
            marker=markers[j-1], label=i
        )

ax.set_xlabel(f'CP{0 + 1} ({pca.explained_variance_ratio_[0]:.1%})')
ax.set_ylabel(f'CP{1 + 1} ({pca.explained_variance_ratio_[1]:.1%})')

ax.set_title(f'Projection des clusters CAH sur CP{0 + 1} et CP{1 + 1}'
                 f' ({cum_variance_ratio:.1%})')
fig.tight_layout()

plt.legend(title='Cluster :')

plt.show()
```



[420]:
```
#projection des clusters obtenus par K-Means:

pca=PCA(n_components=4).fit(data_set_only_quant_scaled)
cum_variance_ratio = (pca.explained_variance_ratio_[0]
                      + pca.explained_variance_ratio_[1])
```

```
df = pd.DataFrame(pca.transform(data_set_only_quant_scaled))
df['cluster'] = data_set["cluster_kmeans"]
n_clusters = len(df['cluster'].unique())
markers = ['<', '+', '*', 'D', 'o']

fig, ax = plt.subplots(figsize=(7, 7))


ax.set_aspect('equal')
ax.grid(alpha=0.4)
ax.set_axisbelow(True)

for spine in ax.spines.values():
        spine.set_visible(False)

ax.axhline(y=0, linestyle = '--', linewidth=0.9, color='k')
ax.axvline(x=0, linestyle = '--', linewidth=0.9, color='k')


for i in range(1, (n_clusters+1)):
        j = i % len(markers)
        ax.scatter(
            df.loc[df['cluster'] == i, 0],
            df.loc[df['cluster'] == i, 1],
            marker=markers[j-1], label=i
        )


ax.set_xlabel(f'CP{0 + 1} ({pca.explained_variance_ratio_[0]:.1%})')
ax.set_ylabel(f'CP{1 + 1} ({pca.explained_variance_ratio_[1]:.1%})')

ax.set_title(f'Projection des clusters K-means sur CP{0 + 1} et CP{1 + 1}'
                f' ({cum_variance_ratio:.1%})')
fig.tight_layout()

plt.legend(title='Cluster :')

plt.show()
```


Projection des clusters K-means sur CP1 et CP2 (50.0%)
Cluster :
1
2
3
4
5
6
7
8
9
10
11
12
CP2 (17.8%)
CP1 (32.1%)
6
4
2
0
−2
−4
−4
−2
0
2
4

[ ]: