P11_Poulet_Data_V5

August 30, 2025

ÉTUDE DE MARCHÉ SUR LE POULET - PRÉPARATION DES DONNÉES

1 OBJECTIFS DE CE NOTEBOOK

- Rassembler les données brutes nécessaires à l'étude ;
- Nettoyer et formater ces données ;
- Définir de nouvelles variables pertinentes (*feature engineering*) pour l'étude ;
- Générer un fichier prêt à l'emploi pour l'analyse (qui sera réalisée dans un autre Notebook).

Partie 1 - Importation des bibliothèques Python et chargement des fichiers

[1]:
```
#Importation de la librairie Pandas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from scipy import stats as st
from scipy.cluster.hierarchy import linkage, fcluster, dendrogram
from scipy.spatial.distance import squareform as sqf
```

[2]:
```
#Importation du fichier FAO_AO_Population_2010_2023.csv
df_population = pd.read_csv("FAO_AO_Population_2010_2023.csv", sep=';')

#importation du fichier FAO_MK_Macroéco_2010_2024.csv
df_eco = pd.read_csv("FAO_MK_Macroéco_2010_2024.csv", sep=';')

#Importation du fichier FAO_TCL_Import-Export_2010_2023.csv
df_commerce = pd.read_csv("FAO_TCL_Import-Export_2010_2023.csv", sep=';')

#importation du fichier FAO_FBS_Dispo_2010_2022.csv
df_dispo = pd.read_csv("FAO_FBS_Dispo_2010_2022.csv", sep=';')

#Importation du fichier FAO_QCL_Production_2010_2023.csv
df_prod = pd.read_csv("FAO_QCL_Production_2010_2023.csv", sep=';')

#Importation du fichier Prix_2019.xlsx
df_prix = pd.read_excel("Prix_2019.xlsx")
```

```
#Importation du fichier wgidataset.xlsx
df_gouv = pd.read_excel("wgidataset.xlsx")

#Importation du fichier dist_cepii.xlsx
df_dist = pd.read_excel("dist_cepii.xlsx")
```

Partie 2 - Nettoyage des fichiers

2.1 - Données de population

[3]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".
 ↪format(df_population.shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_population.shape[1]))
```

```
Le tableau comporte 16354 observation(s) ou article(s)
Le tableau comporte 15 colonne(s)
```

[4]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_population.shape[1])
#La nature des données dans chacune des colonnes
display(df_population.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_population):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_population[c]).
 ↪isna()).sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_population.
 ↪shape[0] - ((df_population[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_population.shape[0] - ((df_population[c]).isna()).sum() == 0:
        df_population.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 15
```

```
Code Domaine               object
Domaine                    object
Code zone (ISO3)           object
Zone                       object
Code Élément                int64
Élément                    object
Code Produit                int64
Produit                    object
Code année                  int64
Année                       int64
Unité                      object
Valeur                    float64
Symbole                    object
Description du Symbole     object
Note                      float64
dtype: object
```

```
Colonne Code Domaine - Nombre de valeurs NaN : 0
Colonne Code Domaine - Nombre de valeurs non-vides : 16354

Colonne Domaine - Nombre de valeurs NaN : 0
Colonne Domaine - Nombre de valeurs non-vides : 16354

Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 16354

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 16354

Colonne Code Élément - Nombre de valeurs NaN : 0
Colonne Code Élément - Nombre de valeurs non-vides : 16354

Colonne Élément - Nombre de valeurs NaN : 0
Colonne Élément - Nombre de valeurs non-vides : 16354

Colonne Code Produit - Nombre de valeurs NaN : 0
Colonne Code Produit - Nombre de valeurs non-vides : 16354

Colonne Produit - Nombre de valeurs NaN : 0
Colonne Produit - Nombre de valeurs non-vides : 16354

Colonne Code année - Nombre de valeurs NaN : 0
Colonne Code année - Nombre de valeurs non-vides : 16354

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 16354

Colonne Unité - Nombre de valeurs NaN : 0
Colonne Unité - Nombre de valeurs non-vides : 16354

Colonne Valeur - Nombre de valeurs NaN : 0
Colonne Valeur - Nombre de valeurs non-vides : 16354

Colonne Symbole - Nombre de valeurs NaN : 0
Colonne Symbole - Nombre de valeurs non-vides : 16354

Colonne Description du Symbole - Nombre de valeurs NaN : 0
Colonne Description du Symbole - Nombre de valeurs non-vides : 16354

Colonne Note - Nombre de valeurs NaN : 16354
Colonne Note - Nombre de valeurs non-vides : 0
```

[5]:
```
display(df_population)
```

```
      Code Domaine                      Domaine Code zone (ISO3)  \
0               OA  Séries temporelles annuelles              AFG
1               OA  Séries temporelles annuelles              AFG
2               OA  Séries temporelles annuelles              AFG
3               OA  Séries temporelles annuelles              AFG
4               OA  Séries temporelles annuelles              AFG
…              …                            …               …
16349           OA  Séries temporelles annuelles              ZWE
16350           OA  Séries temporelles annuelles              ZWE
16351           OA  Séries temporelles annuelles              ZWE
16352           OA  Séries temporelles annuelles              ZWE
16353           OA  Séries temporelles annuelles              ZWE

               Zone  Code Élément             Élément  Code Produit  \
0       Afghanistan           511   Population totale          3010
1       Afghanistan           512              Hommes          3010
2       Afghanistan           513              Femmes          3010
3       Afghanistan           551   Population rurale          3010
4       Afghanistan           561  Population urbaine          3010
…              …            …                 …            …
16349     Zimbabwe            511   Population totale          3010
16350     Zimbabwe            512              Hommes          3010
16351     Zimbabwe            513              Femmes          3010
16352     Zimbabwe            551   Population rurale          3010
16353     Zimbabwe            561  Population urbaine          3010

                      Produit  Code année  Année    Unité     Valeur Symbole  \
0       Population-Estimations        2010   2010  1000 No  28284.089       X
1       Population-Estimations        2010   2010  1000 No  14272.309       X
2       Population-Estimations        2010   2010  1000 No  14011.779       X
3       Population-Estimations        2010   2010  1000 No  21966.187       X
4       Population-Estimations        2010   2010  1000 No   6836.980       X
…                          …          …    …      …        …     …
16349   Population-Estimations        2023   2023  1000 No  16340.822       X
16350   Population-Estimations        2023   2023  1000 No   7780.934       X
16351   Population-Estimations        2023   2023  1000 No   8559.888       X
16352   Population-Estimations        2023   2023  1000 No  12695.580       X
16353   Population-Estimations        2023   2023  1000 No   6117.511       X

                   Description du Symbole
0       Chiffre de sources internationales
1       Chiffre de sources internationales
2       Chiffre de sources internationales
3       Chiffre de sources internationales
4       Chiffre de sources internationales
…                                     …
16349   Chiffre de sources internationales
16350   Chiffre de sources internationales
```

```
16351  Chiffre de sources internationales
16352  Chiffre de sources internationales
16353  Chiffre de sources internationales

[16354 rows x 14 columns]
```

[6]:
```
# Si la colonne présente tout le temps la même valeur (donc pas d'information),␣
 ↪on la supprime
for c in list(df_population):
    if (df_population.groupby(by=c).count()).iloc[0,0] == df_population.
 ↪shape[0]:
        df_population.drop(c,axis=1,inplace=True)

display(df_population)
```

```
      Code zone (ISO3)         Zone  Code Élément             Élément  \
0                  AFG  Afghanistan           511   Population totale
1                  AFG  Afghanistan           512              Hommes
2                  AFG  Afghanistan           513              Femmes
3                  AFG  Afghanistan           551   Population rurale
4                  AFG  Afghanistan           561  Population urbaine
…                  …          …            …                 …
16349              ZWE     Zimbabwe           511   Population totale
16350              ZWE     Zimbabwe           512              Hommes
16351              ZWE     Zimbabwe           513              Femmes
16352              ZWE     Zimbabwe           551   Population rurale
16353              ZWE     Zimbabwe           561  Population urbaine

       Code année  Année     Valeur Symbole  \
0            2010   2010  28284.089       X
1            2010   2010  14272.309       X
2            2010   2010  14011.779       X
3            2010   2010  21966.187       X
4            2010   2010   6836.980       X
…            …    …        …     …
16349        2023   2023  16340.822       X
16350        2023   2023   7780.934       X
16351        2023   2023   8559.888       X
16352        2023   2023  12695.580       X
16353        2023   2023   6117.511       X

                  Description du Symbole
0      Chiffre de sources internationales
1      Chiffre de sources internationales
2      Chiffre de sources internationales
3      Chiffre de sources internationales
4      Chiffre de sources internationales
…                                    …
```

```
16349  Chiffre de sources internationales
16350  Chiffre de sources internationales
16351  Chiffre de sources internationales
16352  Chiffre de sources internationales
16353  Chiffre de sources internationales

[16354 rows x 9 columns]
```

[7]:
```
#On multiplie les valeurs par 1000
df_population["Valeur"] *= 1000
# On supprime les colonnes Code Élément et Code année, qui sont redondantes␣
 ↪avec d'autres colonnes du DataFrame
df_population.drop(["Code Élément","Code année"],axis=1,inplace=True)
# On conserve la colonne Code zone pour de futures jointures avec d'autres␣
 ↪tables

display(df_population)
```

```
      Code zone (ISO3)         Zone              Élément  Année      Valeur  \
0                  AFG  Afghanistan   Population totale   2010  28284089.0
1                  AFG  Afghanistan              Hommes   2010  14272309.0
2                  AFG  Afghanistan              Femmes   2010  14011779.0
3                  AFG  Afghanistan   Population rurale   2010  21966187.0
4                  AFG  Afghanistan  Population urbaine   2010   6836980.0
…                  …           …                  …    …         …
16349              ZWE     Zimbabwe   Population totale   2023  16340822.0
16350              ZWE     Zimbabwe              Hommes   2023   7780934.0
16351              ZWE     Zimbabwe              Femmes   2023   8559888.0
16352              ZWE     Zimbabwe   Population rurale   2023  12695580.0
16353              ZWE     Zimbabwe  Population urbaine   2023   6117511.0

      Symbole              Description du Symbole
0           X  Chiffre de sources internationales
1           X  Chiffre de sources internationales
2           X  Chiffre de sources internationales
3           X  Chiffre de sources internationales
4           X  Chiffre de sources internationales
…       …                                  …
16349       X  Chiffre de sources internationales
16350       X  Chiffre de sources internationales
16351       X  Chiffre de sources internationales
16352       X  Chiffre de sources internationales
16353       X  Chiffre de sources internationales

[16354 rows x 7 columns]
```

[8]:
```
display(df_population.groupby(by=["Symbole","Description du Symbole"]).count())
```

```
                                          Code zone (ISO3)   Zone  Élément  \
Symbole Description du Symbole
E       Valeur estimée                                9      9        9
X       Chiffre de sources internationales            16345  16345    16345

                                          Année  Valeur
Symbole Description du Symbole
E       Valeur estimée                        9       9
X       Chiffre de sources internationales  16345   16345
```

[9]:
```
display(df_population.loc[df_population["Symbole"] == 'E'])
```

```
      Code zone (ISO3)                        Zone            Élément  Année  \
630                ANT  Antilles néerlandaises (ex)  Population totale   2010
631                ANT  Antilles néerlandaises (ex)             Hommes   2010
632                ANT  Antilles néerlandaises (ex)             Femmes   2010
14534             F206                 Soudan (ex)  Population totale   2010
14535             F206                 Soudan (ex)             Hommes   2010
14536             F206                 Soudan (ex)             Femmes   2010
14539             F206                 Soudan (ex)  Population totale   2011
14540             F206                 Soudan (ex)             Hommes   2011
14541             F206                 Soudan (ex)             Femmes   2011

           Valeur Symbole Description du Symbole
630      257053.0       E         Valeur estimée
631      120468.0       E         Valeur estimée
632      136585.0       E         Valeur estimée
14534  45160353.0       E         Valeur estimée
14535  22426702.0       E         Valeur estimée
14536  22733650.0       E         Valeur estimée
14539  46428915.0       E         Valeur estimée
14540  23058020.0       E         Valeur estimée
14541  23370895.0       E         Valeur estimée
```

[10]:
```
df_population.drop(["Symbole","Description du Symbole"],axis=1,inplace=True)
display(df_population)
```

```
      Code zone (ISO3)         Zone             Élément  Année      Valeur
0                  AFG  Afghanistan   Population totale   2010  28284089.0
1                  AFG  Afghanistan             Hommes   2010  14272309.0
2                  AFG  Afghanistan             Femmes   2010  14011779.0
3                  AFG  Afghanistan   Population rurale   2010  21966187.0
4                  AFG  Afghanistan  Population urbaine   2010   6836980.0
…                  …            …                  …    …         …
16349              ZWE     Zimbabwe   Population totale   2023  16340822.0
16350              ZWE     Zimbabwe             Hommes   2023   7780934.0
16351              ZWE     Zimbabwe             Femmes   2023   8559888.0
16352              ZWE     Zimbabwe   Population rurale   2023  12695580.0
16353              ZWE     Zimbabwe  Population urbaine   2023   6117511.0
```

[16354 rows x 5 columns]

[11]:
```
# On vérifie qu'on n'a pas de doublon : la clé primaire de cette table est le␣
 ↪triplet (Zone, Élément, Année)
display(df_population.groupby(by=["Zone","Élément","Année"]).count().
 ↪sort_values(by='Valeur',ascending=False))
```

```
                                                        Code zone (ISO3)  \
Zone                                Élément            Année
Afghanistan                         Femmes             2010                1
Roumanie                            Population urbaine 2016                1
                                    Population totale  2017                1
                                                       2018                1
                                                       2019                1
…                                                                        …
Guyana                              Population urbaine 2020                1
                                                       2021                1
                                                       2022                1
                                                       2023                1
Îles britanniques Atlantique Sud    Population urbaine 2023                1

                                                        Valeur
Zone                                Élément            Année
Afghanistan                         Femmes             2010        1
Roumanie                            Population urbaine 2016        1
                                    Population totale  2017        1
                                                       2018        1
                                                       2019        1
…                                                                  …
Guyana                              Population urbaine 2020        1
                                                       2021        1
                                                       2022        1
                                                       2023        1
Îles britanniques Atlantique Sud    Population urbaine 2023        1

[16354 rows x 2 columns]
```

[12]:
```
# On pivote la table de façon à ce que Élément soit retiré de la clé primaire
# remplaçant la colonne Élément en 5 colonnes avec des valeurs
df_population = df_population.pivot(index=["Code zone (ISO3)","Zone","Année"],␣
 ↪columns="Élément", values="Valeur").reset_index()
display(df_population)
```

```
Élément Code zone (ISO3)      Zone  Année     Femmes     Hommes  \
0                    ABW     Aruba   2010    52339.0    47775.0
1                    ABW     Aruba   2011    52801.0    48179.0
2                    ABW     Aruba   2012    53288.0    48475.0
```

```
3                    ABW      Aruba   2013    53779.0    48778.0
4                    ABW      Aruba   2014    54277.0    49096.0
…                    …          …     …          …          …
3293                 ZWE   Zimbabwe   2019  8027926.0  7243442.0
3294                 ZWE   Zimbabwe   2020  8156301.0  7370588.0
3295                 ZWE   Zimbabwe   2021  8291728.0  7505482.0
3296                 ZWE   Zimbabwe   2022  8426339.0  7642717.0
3297                 ZWE   Zimbabwe   2023  8559888.0  7780934.0

Élément  Population rurale  Population totale  Population urbaine
0                  57891.0           100114.0             43778.0
1                  58231.0           100981.0             43822.0
2                  58513.0           101763.0             44064.0
3                  58826.0           102558.0             44361.0
4                  59121.0           103373.0             44674.0
…                      …                 …                   …
3293            11725970.0         15271368.0           5571525.0
3294            11980005.0         15526888.0           5700460.0
3295            12226340.0         15797210.0           5834113.0
3296            12464597.0         16069056.0           5972826.0
3297            12695580.0         16340822.0           6117511.0

[3298 rows x 8 columns]
```

[13]:
```
#On renomme et permute les colonnes pour les placer dans un ordre plus logique
df_population = df_population.rename(columns = {"Code zone (ISO3)":␣
 ↪"ISO3","Population totale": "Totale",\
                                                 "Population rurale":␣
 ↪"Rurale","Population urbaine": "Urbaine"})
df_population = df_population.reindex(columns =␣
 ↪["ISO3","Zone","Année","Totale","Rurale","Urbaine","Femmes","Hommes"])
display(df_population)
```

```
Élément ISO3      Zone  Année      Totale      Rurale    Urbaine     Femmes  \
0         ABW     Aruba   2010    100114.0     57891.0    43778.0    52339.0
1         ABW     Aruba   2011    100981.0     58231.0    43822.0    52801.0
2         ABW     Aruba   2012    101763.0     58513.0    44064.0    53288.0
3         ABW     Aruba   2013    102558.0     58826.0    44361.0    53779.0
4         ABW     Aruba   2014    103373.0     59121.0    44674.0    54277.0
…       …         …       …          …           …          …          …
3293      ZWE  Zimbabwe   2019  15271368.0  11725970.0  5571525.0  8027926.0
3294      ZWE  Zimbabwe   2020  15526888.0  11980005.0  5700460.0  8156301.0
3295      ZWE  Zimbabwe   2021  15797210.0  12226340.0  5834113.0  8291728.0
3296      ZWE  Zimbabwe   2022  16069056.0  12464597.0  5972826.0  8426339.0
3297      ZWE  Zimbabwe   2023  16340822.0  12695580.0  6117511.0  8559888.0

Élément     Hommes
0          47775.0
```

```
1          48179.0
2          48475.0
3          48778.0
4          49096.0
…              …
3293     7243442.0
3294     7370588.0
3295     7505482.0
3296     7642717.0
3297     7780934.0

[3298 rows x 8 columns]
```

[14]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_population.shape[1])
#La nature des données dans chacune des colonnes
display(df_population.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_population):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_population[c]).
 ↪isna()).sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_population.
 ↪shape[0] - ((df_population[c]).isna()).sum())
```

```
Nombre de colonnes : 8

Élément
ISO3        object
Zone        object
Année        int64
Totale     float64
Rurale     float64
Urbaine    float64
Femmes     float64
Hommes     float64
dtype: object


Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 3298

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 3298

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 3298

Colonne Totale - Nombre de valeurs NaN : 28
Colonne Totale - Nombre de valeurs non-vides : 3270
```

```
Colonne Rurale - Nombre de valeurs NaN : 26
Colonne Rurale - Nombre de valeurs non-vides : 3272

Colonne Urbaine - Nombre de valeurs NaN : 26
Colonne Urbaine - Nombre de valeurs non-vides : 3272

Colonne Femmes - Nombre de valeurs NaN : 28
Colonne Femmes - Nombre de valeurs non-vides : 3270

Colonne Hommes - Nombre de valeurs NaN : 28
Colonne Hommes - Nombre de valeurs non-vides : 3270
```

[15]:
```
display(df_population.loc[df_population["Totale"].isna()])
missingval_zones_sr = pd.Series(df_population.loc[df_population["Totale"].
  ↪isna()]["ISO3"].unique(), name = "ISO3")
```

Élément	ISO3	Zone	Année	Totale	Rurale	Urbaine	Femmes	\
503	CHA	Îles Anglo-Normandes	2010	NaN	110025.0	49556.0	NaN	
504	CHA	Îles Anglo-Normandes	2011	NaN	110596.0	49901.0	NaN	
505	CHA	Îles Anglo-Normandes	2012	NaN	111242.0	50116.0	NaN	
506	CHA	Îles Anglo-Normandes	2013	NaN	111861.0	50319.0	NaN	
507	CHA	Îles Anglo-Normandes	2014	NaN	112457.0	50512.0	NaN	
508	CHA	Îles Anglo-Normandes	2015	NaN	113055.0	50703.0	NaN	
509	CHA	Îles Anglo-Normandes	2016	NaN	113648.0	50893.0	NaN	
510	CHA	Îles Anglo-Normandes	2017	NaN	114209.0	51105.0	NaN	
511	CHA	Îles Anglo-Normandes	2018	NaN	114740.0	51343.0	NaN	
512	CHA	Îles Anglo-Normandes	2019	NaN	115227.0	51601.0	NaN	
513	CHA	Îles Anglo-Normandes	2020	NaN	115690.0	51888.0	NaN	
514	CHA	Îles Anglo-Normandes	2021	NaN	116109.0	52195.0	NaN	
515	CHA	Îles Anglo-Normandes	2022	NaN	116490.0	52525.0	NaN	
516	CHA	Îles Anglo-Normandes	2023	NaN	116840.0	52884.0	NaN	
3116	VAT	Saint-Siège	2010	NaN	0.0	794.0	NaN	
3117	VAT	Saint-Siège	2011	NaN	0.0	796.0	NaN	
3118	VAT	Saint-Siège	2012	NaN	0.0	804.0	NaN	
3119	VAT	Saint-Siège	2013	NaN	0.0	801.0	NaN	
3120	VAT	Saint-Siège	2014	NaN	0.0	800.0	NaN	
3121	VAT	Saint-Siège	2015	NaN	0.0	803.0	NaN	
3122	VAT	Saint-Siège	2016	NaN	0.0	801.0	NaN	
3123	VAT	Saint-Siège	2017	NaN	0.0	792.0	NaN	
3124	VAT	Saint-Siège	2018	NaN	0.0	801.0	NaN	
3125	VAT	Saint-Siège	2019	NaN	0.0	799.0	NaN	
3126	VAT	Saint-Siège	2020	NaN	0.0	801.0	NaN	
3127	VAT	Saint-Siège	2021	NaN	0.0	800.0	NaN	
3128	VAT	Saint-Siège	2022	NaN	0.0	799.0	NaN	
3129	VAT	Saint-Siège	2023	NaN	0.0	799.0	NaN	

```
Élément  Hommes
```

```
503          NaN
504          NaN
505          NaN
506          NaN
507          NaN
508          NaN
509          NaN
510          NaN
511          NaN
512          NaN
513          NaN
514          NaN
515          NaN
516          NaN
3116         NaN
3117         NaN
3118         NaN
3119         NaN
3120         NaN
3121         NaN
3122         NaN
3123         NaN
3124         NaN
3125         NaN
3126         NaN
3127         NaN
3128         NaN
3129         NaN
```

[16]:
```
# À partir des données connues, on peut raisonnablement estimer la population␣
 ↪totale pour ces 2 zones
#df_tofill = df_population.loc[df_population["Totale"].isna()]
df_population["Totale"] = (df_population["Totale"]).fillna(value =␣
 ↪df_population.loc[df_population["Totale"].isna()]["Rurale"] +\
df_population.loc[df_population["Totale"].isna()]["Urbaine"])
display(pd.merge(left=df_population, right=missingval_zones_sr, on="ISO3"))
```

	ISO3	Zone	Année	Totale	Rurale	Urbaine	Femmes	\
0	CHA	Îles Anglo-Normandes	2010	159581.0	110025.0	49556.0	NaN	
1	CHA	Îles Anglo-Normandes	2011	160497.0	110596.0	49901.0	NaN	
2	CHA	Îles Anglo-Normandes	2012	161358.0	111242.0	50116.0	NaN	
3	CHA	Îles Anglo-Normandes	2013	162180.0	111861.0	50319.0	NaN	
4	CHA	Îles Anglo-Normandes	2014	162969.0	112457.0	50512.0	NaN	
5	CHA	Îles Anglo-Normandes	2015	163758.0	113055.0	50703.0	NaN	
6	CHA	Îles Anglo-Normandes	2016	164541.0	113648.0	50893.0	NaN	
7	CHA	Îles Anglo-Normandes	2017	165314.0	114209.0	51105.0	NaN	
8	CHA	Îles Anglo-Normandes	2018	166083.0	114740.0	51343.0	NaN	
9	CHA	Îles Anglo-Normandes	2019	166828.0	115227.0	51601.0	NaN	

10	CHA	Îles Anglo-Normandes	2020	167578.0	115690.0	51888.0	NaN
11	CHA	Îles Anglo-Normandes	2021	168304.0	116109.0	52195.0	NaN
12	CHA	Îles Anglo-Normandes	2022	169015.0	116490.0	52525.0	NaN
13	CHA	Îles Anglo-Normandes	2023	169724.0	116840.0	52884.0	NaN
14	VAT	Saint-Siège	2010	794.0	0.0	794.0	NaN
15	VAT	Saint-Siège	2011	796.0	0.0	796.0	NaN
16	VAT	Saint-Siège	2012	804.0	0.0	804.0	NaN
17	VAT	Saint-Siège	2013	801.0	0.0	801.0	NaN
18	VAT	Saint-Siège	2014	800.0	0.0	800.0	NaN
19	VAT	Saint-Siège	2015	803.0	0.0	803.0	NaN
20	VAT	Saint-Siège	2016	801.0	0.0	801.0	NaN
21	VAT	Saint-Siège	2017	792.0	0.0	792.0	NaN
22	VAT	Saint-Siège	2018	801.0	0.0	801.0	NaN
23	VAT	Saint-Siège	2019	799.0	0.0	799.0	NaN
24	VAT	Saint-Siège	2020	801.0	0.0	801.0	NaN
25	VAT	Saint-Siège	2021	800.0	0.0	800.0	NaN
26	VAT	Saint-Siège	2022	799.0	0.0	799.0	NaN
27	VAT	Saint-Siège	2023	799.0	0.0	799.0	NaN

	Hommes
0	NaN
1	NaN
2	NaN
3	NaN
4	NaN
5	NaN
6	NaN
7	NaN
8	NaN
9	NaN
10	NaN
11	NaN
12	NaN
13	NaN
14	NaN
15	NaN
16	NaN
17	NaN
18	NaN
19	NaN
20	NaN
21	NaN
22	NaN
23	NaN
24	NaN
25	NaN
26	NaN
27	NaN

[17]:
```
display(df_population.loc[df_population["Urbaine"].isna()])
```

Élément	ISO3	Zone	Année	Totale	Rurale	Urbaine	Femmes	Hommes
336	BLM	Saint-Barthélemy	2011	9088.0	NaN	NaN	4262.0	4826.0
337	BLM	Saint-Barthélemy	2012	9197.0	NaN	NaN	4318.0	4879.0
338	BLM	Saint-Barthélemy	2013	9318.0	NaN	NaN	4380.0	4938.0
339	BLM	Saint-Barthélemy	2014	9447.0	NaN	NaN	4449.0	4997.0
340	BLM	Saint-Barthélemy	2015	9590.0	NaN	NaN	4530.0	5060.0
341	BLM	Saint-Barthélemy	2016	9750.0	NaN	NaN	4619.0	5131.0
342	BLM	Saint-Barthélemy	2017	9928.0	NaN	NaN	4711.0	5218.0
343	BLM	Saint-Barthélemy	2018	10107.0	NaN	NaN	4859.0	5248.0
344	BLM	Saint-Barthélemy	2019	10299.0	NaN	NaN	5136.0	5162.0
345	BLM	Saint-Barthélemy	2020	10543.0	NaN	NaN	5497.0	5046.0
346	BLM	Saint-Barthélemy	2021	10764.0	NaN	NaN	5723.0	5040.0
347	BLM	Saint-Barthélemy	2022	10920.0	NaN	NaN	5806.0	5114.0
348	BLM	Saint-Barthélemy	2023	11085.0	NaN	NaN	5894.0	5191.0
1792	MAF	Saint-Martin (partie française)	2011	37069.0	NaN	NaN	19715.0	17354.0
1793	MAF	Saint-Martin (partie française)	2012	37276.0	NaN	NaN	19831.0	17444.0
1794	MAF	Saint-Martin (partie française)	2013	37419.0	NaN	NaN	19914.0	17505.0
1795	MAF	Saint-Martin (partie française)	2014	37450.0	NaN	NaN	19920.0	17531.0
1796	MAF	Saint-Martin (partie française)	2015	37369.0	NaN			
1797	MAF	Saint-Martin (partie française)	2016	37175.0	NaN			
1798	MAF	Saint-Martin (partie française)	2017	36837.0	NaN			
1799	MAF	Saint-Martin (partie française)	2018	36012.0	NaN			
1800	MAF	Saint-Martin (partie française)	2019	34267.0	NaN			
1801	MAF	Saint-Martin (partie française)	2020	31786.0	NaN			
1802	MAF	Saint-Martin (partie française)	2021	29961.0	NaN			
1803	MAF	Saint-Martin (partie française)	2022	28870.0	NaN			
1804	MAF	Saint-Martin (partie française)	2023	27515.0	NaN			

```
1796         NaN  19864.0  17505.0
1797         NaN  19762.0  17413.0
1798         NaN  19593.0  17245.0
1799         NaN  19078.0  16934.0
1800         NaN  17898.0  16368.0
1801         NaN  16651.0  15136.0
1802         NaN  15912.0  14049.0
1803         NaN  15336.0  13534.0
1804         NaN  14658.0  12857.0
```

[18]:
```
#Liste des pays les plus peuplés (par ordre décroissant) en 2023
df_pays_pop2023 = (df_population.loc[df_population["Année"] ==␣
↪2023,["ISO3","Zone","Totale"]]).sort_values(by="Totale", ascending=False).
↪reset_index()
display(df_pays_pop2023)
```

Élément	index	ISO3	Zone	Totale
0	951	F351	Chine	1.454059e+09
1	1385	IND	Inde	1.438070e+09
2	558	CHN	Chine (continentale)	1.422585e+09
3	3101	USA	États-Unis d'Amérique	3.434773e+08
4	1357	IDN	Indonésie	2.811901e+08
..	…	…	…	…
231	2042	MSR	Montserrat	4.420000e+03
232	993	FLK	Îles Falkland (Malvinas)	3.477000e+03
233	2919	TKL	Tokélaou	2.397000e+03
234	2196	NIU	Nioué	1.817000e+03
235	3129	VAT	Saint-Siège	7.990000e+02

[236 rows x 4 columns]

[19]:
```
df_population["Taille code"] = df_population["ISO3"].apply(lambda s: len(s))
display(df_population.loc[df_population["Taille code"] != 3])
```

Élément	ISO3	Zone	Année	Totale	Rurale	Urbaine	\
936	F206	Soudan (ex)	2010	4.516035e+07	31277063.0	13176092.0	
937	F206	Soudan (ex)	2011	4.642892e+07	32054105.0	13562066.0	
938	F351	Chine	2010	1.382450e+09	696249563.0	694170135.0	
939	F351	Chine	2011	1.391242e+09	682518202.0	715762326.0	
940	F351	Chine	2012	1.400655e+09	669007424.0	737124022.0	
941	F351	Chine	2013	1.410288e+09	655323811.0	758533671.0	
942	F351	Chine	2014	1.419355e+09	641352868.0	779954516.0	
943	F351	Chine	2015	1.427652e+09	627097042.0	801263909.0	
944	F351	Chine	2016	1.435681e+09	612549421.0	822422660.0	
945	F351	Chine	2017	1.444079e+09	597816520.0	843314783.0	
946	F351	Chine	2018	1.450798e+09	583199631.0	863601691.0	
947	F351	Chine	2019	1.455351e+09	568748198.0	883204937.0	
948	F351	Chine	2020	1.457943e+09	554487938.0	902077760.0	
949	F351	Chine	2021	1.458175e+09	540439367.0	920173874.0	

```
950      F351        Chine   2022  1.456770e+09  526626069.0  937461423.0
951      F351        Chine   2023  1.454059e+09  513079075.0  953931777.0

Élément       Femmes       Hommes  Taille code
936       22733650.0   22426702.0            4
937       23370895.0   23058020.0            4
938      675640944.0  706809041.0            4
939      679926627.0  711315716.0            4
940      684538666.0  716116667.0            4
941      689276305.0  721011688.0            4
942      693772000.0  725582593.0            4
943      697933695.0  729718321.0            4
944      702008575.0  733672520.0            4
945      706348457.0  737730664.0            4
946      709918983.0  740879243.0            4
947      712408482.0  742942800.0            4
948      714041712.0  743901142.0            4
949      714591397.0  743583206.0            4
950      714369838.0  742400113.0            4
951      713597370.0  740461354.0            4
```

Il faut retirer les lignes de données correspondant à Chine (continentale) + Macao + Hong Kong + Taïwan (code F351).

[20]:
```
df_population = (df_population.loc[df_population["ISO3"] != "F351"]).copy()
df_pays_pop2023 = (df_population.loc[df_population["Année"] ==␣
 ↪2023,["ISO3","Zone","Totale"]]).sort_values(by="Totale", ascending=False).
 ↪reset_index()
display(df_pays_pop2023)
df_population.drop("Taille code",axis=1,inplace=True)
display(df_population)
```

```
Élément  index ISO3                      Zone        Totale
0         1385  IND                      Inde  1.438070e+09
1          558  CHN     Chine (continentale)  1.422585e+09
2         3101  USA    États-Unis d'Amérique  3.434773e+08
3         1357  IDN                Indonésie  2.811901e+08
4         2294  PAK                 Pakistan  2.475045e+08
..         …  …                        …           …
230       2042  MSR               Montserrat  4.420000e+03
231        993  FLK  Îles Falkland (Malvinas)  3.477000e+03
232       2919  TKL                 Tokélaou  2.397000e+03
233       2196  NIU                    Nioué  1.817000e+03
234       3129  VAT             Saint-Siège  7.990000e+02

[235 rows x 4 columns]

Élément ISO3      Zone  Année      Totale      Rurale    Urbaine     Femmes  \
0         ABW     Aruba   2010    100114.0     57891.0    43778.0    52339.0
```

```
1         ABW     Aruba   2011    100981.0     58231.0    43822.0    52801.0
2         ABW     Aruba   2012    101763.0     58513.0    44064.0    53288.0
3         ABW     Aruba   2013    102558.0     58826.0    44361.0    53779.0
4         ABW     Aruba   2014    103373.0     59121.0    44674.0    54277.0
…       …       …       …          …          …          …          …
3293      ZWE  Zimbabwe   2019  15271368.0  11725970.0  5571525.0  8027926.0
3294      ZWE  Zimbabwe   2020  15526888.0  11980005.0  5700460.0  8156301.0
3295      ZWE  Zimbabwe   2021  15797210.0  12226340.0  5834113.0  8291728.0
3296      ZWE  Zimbabwe   2022  16069056.0  12464597.0  5972826.0  8426339.0
3297      ZWE  Zimbabwe   2023  16340822.0  12695580.0  6117511.0  8559888.0

Élément     Hommes
0          47775.0
1          48179.0
2          48475.0
3          48778.0
4          49096.0
…              …
3293     7243442.0
3294     7370588.0
3295     7505482.0
3296     7642717.0
3297     7780934.0

[3284 rows x 8 columns]
```

[21]:
```
Pop_tot2023 = (df_population.groupby(by="Année")["Totale"].sum()).loc[2023]
set_pays_ref = set(df_pays_pop2023["ISO3"].unique())
print("Population mondiale en 2023 :",Pop_tot2023)
```

```
Population mondiale en 2023 : 8090037234.0
```

[22]:
```
# Pour cette étude, on va particulièremeny s'intéresser à la population urbaine
  ↪des pays, plus proche des grands circuits de distribution
df_population.drop(["Rurale","Femmes","Hommes"], axis=1, inplace=True)
df_data_viz = df_population.copy()
df_population.dropna(inplace=True)
display(df_population)
```

```
Élément ISO3      Zone  Année      Totale    Urbaine
0         ABW     Aruba   2010    100114.0    43778.0
1         ABW     Aruba   2011    100981.0    43822.0
2         ABW     Aruba   2012    101763.0    44064.0
3         ABW     Aruba   2013    102558.0    44361.0
4         ABW     Aruba   2014    103373.0    44674.0
…       …       …       …          …          …
3293      ZWE  Zimbabwe   2019  15271368.0  5571525.0
3294      ZWE  Zimbabwe   2020  15526888.0  5700460.0
3295      ZWE  Zimbabwe   2021  15797210.0  5834113.0
```

```
3296     ZWE  Zimbabwe   2022  16069056.0  5972826.0
3297     ZWE  Zimbabwe   2023  16340822.0  6117511.0

[3258 rows x 5 columns]
```

[23]:
```
set_pays_data = set(df_population["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_data), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones,on="ISO3",how='inner'))
Pop_dat2023 = (df_population.groupby(by="Année")["Totale"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
 ↪2), "% de la population mondiale en 2023")
```

```
   index ISO3                            Zone   Totale
0   1804  MAF  Saint-Martin (partie française)  27515.0
1    348  BLM                 Saint-Barthélemy  11085.0

Les données en l'état couvrent 100.0 % de la population mondiale en 2023
```

[24]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_population):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_population[c]).
 ↪isna()).sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_population.
 ↪shape[0] - ((df_population[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_population["Zone"]).nunique())

df_groupby = df_population.groupby(by="Zone").count().
 ↪sort_values(by='Année',ascending=False)
display(df_groupby)
display(df_groupby.loc[df_groupby["Année"]<14])
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 3258

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 3258

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 3258

Colonne Totale - Nombre de valeurs NaN : 0
Colonne Totale - Nombre de valeurs non-vides : 3258

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 3258

Nombre de zones dans les données : 235
```

Élément	ISO3	Année	Totale	Urbaine
Zone				
Afghanistan	14	14	14	14
Papouasie-Nouvelle-Guinée	14	14	14	14
Pays-Bas (Royaume des)	14	14	14	14
Philippines	14	14	14	14
Pologne	14	14	14	14
…	…	…	…	…
Sint Maarten (partie néerlandaise)	13	13	13	13
Soudan	12	12	12	12
Soudan du Sud	12	12	12	12
Soudan (ex)	2	2	2	2
Antilles néerlandaises (ex)	1	1	1	1

[235 rows x 4 columns]

Élément	ISO3	Année	Totale	Urbaine
Zone				
Pays-Bas caribéens	13	13	13	13
Curaçao	13	13	13	13
Sint Maarten (partie néerlandaise)	13	13	13	13
Soudan	12	12	12	12
Soudan du Sud	12	12	12	12
Soudan (ex)	2	2	2	2
Antilles néerlandaises (ex)	1	1	1	1

Pour les Antilles néerlandaises qui ont été redécoupées en 2011, la part de population est négligeable par rapport à la population mondiale (quelques dizaine sde milliers de personnes). Pour le Soudan qui a été scindé en 2 états en 2011, une solution peut être de restreindre notre étude en ne la faisant commencer qu'à partir de 2012. Nous verrons cela plus tard au moment de traiter les autres fichiers.

2.2 - Données macro-économiques

[25]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_eco.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_eco.shape[1]))
```

```
Le tableau comporte 6122 observation(s) ou article(s)
Le tableau comporte 15 colonne(s)
```

[26]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_eco.shape[1])
#La nature des données dans chacune des colonnes
display(df_eco.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_eco):
```

```
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_eco[c]).isna()).
↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_eco.shape[0] -␣
↪((df_eco[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_eco.shape[0] - ((df_eco[c]).isna()).sum() == 0:
        df_eco.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 15
```

```
Code Domaine               object
Domaine                    object
Code zone (ISO3)           object
Zone                       object
Code Élément                int64
Élément                    object
Code Produit                int64
Produit                    object
Code année                  int64
Année                       int64
Unité                      object
Valeur                    float64
Symbole                    object
Description du Symbole     object
Note                      float64
dtype: object
```

```
Colonne Code Domaine - Nombre de valeurs NaN : 0
Colonne Code Domaine - Nombre de valeurs non-vides : 6122

Colonne Domaine - Nombre de valeurs NaN : 0
Colonne Domaine - Nombre de valeurs non-vides : 6122

Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 6122

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 6122

Colonne Code Élément - Nombre de valeurs NaN : 0
Colonne Code Élément - Nombre de valeurs non-vides : 6122

Colonne Élément - Nombre de valeurs NaN : 0
Colonne Élément - Nombre de valeurs non-vides : 6122

Colonne Code Produit - Nombre de valeurs NaN : 0
Colonne Code Produit - Nombre de valeurs non-vides : 6122
```

```
Colonne Produit - Nombre de valeurs NaN : 0
Colonne Produit - Nombre de valeurs non-vides : 6122

Colonne Code année - Nombre de valeurs NaN : 0
Colonne Code année - Nombre de valeurs non-vides : 6122

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 6122

Colonne Unité - Nombre de valeurs NaN : 0
Colonne Unité - Nombre de valeurs non-vides : 6122

Colonne Valeur - Nombre de valeurs NaN : 0
Colonne Valeur - Nombre de valeurs non-vides : 6122

Colonne Symbole - Nombre de valeurs NaN : 0
Colonne Symbole - Nombre de valeurs non-vides : 6122

Colonne Description du Symbole - Nombre de valeurs NaN : 0
Colonne Description du Symbole - Nombre de valeurs non-vides : 6122

Colonne Note - Nombre de valeurs NaN : 6122
Colonne Note - Nombre de valeurs non-vides : 0
```

[27]:
```
display(df_eco)
```

```
     Code Domaine            Domaine Code zone (ISO3)         Zone  \
0              MK  Indicateurs macro              AFG  Afghanistan
1              MK  Indicateurs macro              AFG  Afghanistan
2              MK  Indicateurs macro              AFG  Afghanistan
3              MK  Indicateurs macro              AFG  Afghanistan
4              MK  Indicateurs macro              AFG  Afghanistan
…             …                 …               …           …
6117           MK  Indicateurs macro              ZWE     Zimbabwe
6118           MK  Indicateurs macro              ZWE     Zimbabwe
6119           MK  Indicateurs macro              ZWE     Zimbabwe
6120           MK  Indicateurs macro              ZWE     Zimbabwe
6121           MK  Indicateurs macro              ZWE     Zimbabwe

      Code Élément     Élément  Code Produit                 Produit  \
0             6110  Valeur US$         22008  Produit Intérieur Brut
1             6110  Valeur US$         22008  Produit Intérieur Brut
2             6110  Valeur US$         22008  Produit Intérieur Brut
3             6110  Valeur US$         22008  Produit Intérieur Brut
4             6110  Valeur US$         22008  Produit Intérieur Brut
…            …         …           …                      …
6117          6110  Valeur US$         22011    Revenu national brut
```

```
6118          6110  Valeur US$          22011    Revenu national brut
6119          6110  Valeur US$          22011    Revenu national brut
6120          6110  Valeur US$          22011    Revenu national brut
6121          6110  Valeur US$          22011    Revenu national brut

      Code année  Année           Unité       Valeur Symbole  \
0           2010   2010  Millions d'USD  15144.56966       X
1           2011   2011  Millions d'USD  17923.10025       X
2           2012   2012  Millions d'USD  19794.40586       X
3           2013   2013  Millions d'USD  19904.40934       X
4           2014   2014  Millions d'USD  19500.46092       X
…            …    …             …          …     …
6117        2019   2019  Millions d'USD  22238.09701       X
6118        2020   2020  Millions d'USD  21183.19622       X
6119        2021   2021  Millions d'USD  23523.94698       X
6120        2022   2022  Millions d'USD  25804.13719       X
6121        2023   2023  Millions d'USD  29992.98921       X

                Description du Symbole
0     Chiffre de sources internationales
1     Chiffre de sources internationales
2     Chiffre de sources internationales
3     Chiffre de sources internationales
4     Chiffre de sources internationales
…                                   …
6117  Chiffre de sources internationales
6118  Chiffre de sources internationales
6119  Chiffre de sources internationales
6120  Chiffre de sources internationales
6121  Chiffre de sources internationales

[6122 rows x 14 columns]
```

[28]:
```python
# Si la colonne présente tout le temps la même valeur (donc pas d'information), on la supprime
for c in list(df_eco):
    if (df_eco.groupby(by=c).count()).iloc[0,0] == df_eco.shape[0]:
        df_eco.drop(c,axis=1,inplace=True)

display(df_eco)
```

```
     Code zone (ISO3)         Zone  Code Produit                 Produit  \
0                 AFG  Afghanistan         22008  Produit Intérieur Brut
1                 AFG  Afghanistan         22008  Produit Intérieur Brut
2                 AFG  Afghanistan         22008  Produit Intérieur Brut
3                 AFG  Afghanistan         22008  Produit Intérieur Brut
4                 AFG  Afghanistan         22008  Produit Intérieur Brut
…                  …           …           …                     …
```

```
6117              ZWE     Zimbabwe         22011    Revenu national brut
6118              ZWE     Zimbabwe         22011    Revenu national brut
6119              ZWE     Zimbabwe         22011    Revenu national brut
6120              ZWE     Zimbabwe         22011    Revenu national brut
6121              ZWE     Zimbabwe         22011    Revenu national brut

      Code année  Année       Valeur Symbole  \
0           2010   2010  15144.56966       X
1           2011   2011  17923.10025       X
2           2012   2012  19794.40586       X
3           2013   2013  19904.40934       X
4           2014   2014  19500.46092       X
…            …    …           …     …
6117        2019   2019  22238.09701       X
6118        2020   2020  21183.19622       X
6119        2021   2021  23523.94698       X
6120        2022   2022  25804.13719       X
6121        2023   2023  29992.98921       X

                 Description du Symbole
0     Chiffre de sources internationales
1     Chiffre de sources internationales
2     Chiffre de sources internationales
3     Chiffre de sources internationales
4     Chiffre de sources internationales
…                                   …
6117  Chiffre de sources internationales
6118  Chiffre de sources internationales
6119  Chiffre de sources internationales
6120  Chiffre de sources internationales
6121  Chiffre de sources internationales

[6122 rows x 9 columns]
```

[29]:
```
display(df_eco.groupby(by=["Symbole","Description du Symbole"]).count())
```

```
                                          Code zone (ISO3)  Zone  \
Symbole Description du Symbole
E       Valeur estimée                                 210   210
X       Chiffre de sources internationales            5912  5912

                                          Code Produit  Produit  Code année  \
Symbole Description du Symbole
E       Valeur estimée                             210      210         210
X       Chiffre de sources internationales         5912     5912        5912

                                          Année  Valeur
Symbole Description du Symbole
```

```
E       Valeur estimée                          210     210
X       Chiffre de sources internationales     5912    5912
```

[30]:
```
display((df_eco.loc[df_eco["Symbole"] == 'E',"Année"]).unique())
```

```
array([2024], dtype=int64)
```

Les données macro-économiques pour l'année 2024 sont des estimations. Comme on ne dispose pas de données pour cette année dans les autres datasets, on peut se passer des données de 2024.

[31]:
```
df_eco = df_eco.loc[df_eco["Symbole"] == 'X']
for c in list(df_eco):
    if (df_eco.groupby(by=c).count()).iloc[0,0] == df_eco.shape[0]:
        df_eco.drop(c,axis=1,inplace=True)

display(df_eco)
```

```
     Code zone (ISO3)         Zone  Code Produit                 Produit  \
0                 AFG  Afghanistan         22008  Produit Intérieur Brut
1                 AFG  Afghanistan         22008  Produit Intérieur Brut
2                 AFG  Afghanistan         22008  Produit Intérieur Brut
3                 AFG  Afghanistan         22008  Produit Intérieur Brut
4                 AFG  Afghanistan         22008  Produit Intérieur Brut
…                 …           …             …                      …
6117              ZWE     Zimbabwe         22011    Revenu national brut
6118              ZWE     Zimbabwe         22011    Revenu national brut
6119              ZWE     Zimbabwe         22011    Revenu national brut
6120              ZWE     Zimbabwe         22011    Revenu national brut
6121              ZWE     Zimbabwe         22011    Revenu national brut

      Code année  Année       Valeur
0           2010   2010  15144.56966
1           2011   2011  17923.10025
2           2012   2012  19794.40586
3           2013   2013  19904.40934
4           2014   2014  19500.46092
…           …    …           …
6117        2019   2019  22238.09701
6118        2020   2020  21183.19622
6119        2021   2021  23523.94698
6120        2022   2022  25804.13719
6121        2023   2023  29992.98921

[5912 rows x 7 columns]
```

[32]:
```
# On supprime les colonnes Code ÉlémentCode Produit  et Code année, qui sont␣
 ↪redondantes avec d'autres colonnes du DataFrame
df_eco.drop(["Code Produit","Code année"],axis=1,inplace=True)
```

```
# On conserve la colonne Code zone pour de futures jointures avec d'autres tables

#On multiplie la colonne Valeur par 1e6 pour avoir une valeur en dollar US
df_eco["Valeur"] *= 1e6

display(df_eco)
```

```
     Code zone (ISO3)         Zone                 Produit  Année  \
0                 AFG  Afghanistan  Produit Intérieur Brut   2010
1                 AFG  Afghanistan  Produit Intérieur Brut   2011
2                 AFG  Afghanistan  Produit Intérieur Brut   2012
3                 AFG  Afghanistan  Produit Intérieur Brut   2013
4                 AFG  Afghanistan  Produit Intérieur Brut   2014
…                 …          …                    …     …
6117              ZWE     Zimbabwe   Revenu national brut   2019
6118              ZWE     Zimbabwe   Revenu national brut   2020
6119              ZWE     Zimbabwe   Revenu national brut   2021
6120              ZWE     Zimbabwe   Revenu national brut   2022
6121              ZWE     Zimbabwe   Revenu national brut   2023

            Valeur
0     1.514457e+10
1     1.792310e+10
2     1.979441e+10
3     1.990441e+10
4     1.950046e+10
…              …
6117  2.223810e+10
6118  2.118320e+10
6119  2.352395e+10
6120  2.580414e+10
6121  2.999299e+10

[5912 rows x 5 columns]
```

[33]:
```
# On pivote la table de façon à ce que Produit soit retiré de la clé primaire
# remplaçant la colonne Produit en 2 colonnes avec les valeurs de PIB et RNB
df_eco = df_eco.pivot(index=["Code zone (ISO3)","Zone","Année"], columns="Produit", values="Valeur").reset_index()
display(df_eco)
```

```
Produit Code zone (ISO3)      Zone  Année  Produit Intérieur Brut  \
0                    ABW     Aruba   2010            2.453598e+09
1                    ABW     Aruba   2011            2.637860e+09
2                    ABW     Aruba   2012            2.615207e+09
3                    ABW     Aruba   2013            2.727849e+09
4                    ABW     Aruba   2014            2.790849e+09
```

```
…                    …        …     …                      …
2951                   ZWE  Zimbabwe   2019            2.259452e+10
2952                   ZWE  Zimbabwe   2020            2.166475e+10
2953                   ZWE  Zimbabwe   2021            2.411815e+10
2954                   ZWE  Zimbabwe   2022            2.641859e+10
2955                   ZWE  Zimbabwe   2023            3.036820e+10

Produit  Revenu national brut
0                 2.313386e+09
1                 2.391842e+09
2                 2.499116e+09
3                 2.563517e+09
4                 2.688102e+09
…                          …
2951              2.223810e+10
2952              2.118320e+10
2953              2.352395e+10
2954              2.580414e+10
2955              2.999299e+10

[2956 rows x 5 columns]
```

[34]:
```
df_eco = df_eco.rename(columns = {"Code zone (ISO3)": "ISO3", "Produit␣
 ↪Intérieur Brut": "PIB", "Revenu national brut": "RNB"})
df_eco.drop("RNB", axis=1, inplace=True)
display(df_eco)
```

```
Produit ISO3      Zone  Année           PIB
0         ABW     Aruba   2010  2.453598e+09
1         ABW     Aruba   2011  2.637860e+09
2         ABW     Aruba   2012  2.615207e+09
3         ABW     Aruba   2013  2.727849e+09
4         ABW     Aruba   2014  2.790849e+09
…       …         …     …            …
2951      ZWE  Zimbabwe   2019  2.259452e+10
2952      ZWE  Zimbabwe   2020  2.166475e+10
2953      ZWE  Zimbabwe   2021  2.411815e+10
2954      ZWE  Zimbabwe   2022  2.641859e+10
2955      ZWE  Zimbabwe   2023  3.036820e+10

[2956 rows x 4 columns]
```

[35]:
```
df_eco.drop("Zone",axis=1,inplace=True)
df_data = pd.merge(left=df_population, right=df_eco, how='inner',␣
 ↪on=["ISO3","Année"])
display(df_data)
```

```
     ISO3      Zone  Année      Totale    Urbaine           PIB
```

```
0      ABW      Aruba    2010    100114.0    43778.0  2.453598e+09
1      ABW      Aruba    2011    100981.0    43822.0  2.637860e+09
2      ABW      Aruba    2012    101763.0    44064.0  2.615207e+09
3      ABW      Aruba    2013    102558.0    44361.0  2.727849e+09
4      ABW      Aruba    2014    103373.0    44674.0  2.790849e+09
…    …        …      …          …         …          …
2943   ZWE   Zimbabwe    2019  15271368.0  5571525.0  2.259452e+10
2944   ZWE   Zimbabwe    2020  15526888.0  5700460.0  2.166475e+10
2945   ZWE   Zimbabwe    2021  15797210.0  5834113.0  2.411815e+10
2946   ZWE   Zimbabwe    2022  16069056.0  5972826.0  2.641859e+10
2947   ZWE   Zimbabwe    2023  16340822.0  6117511.0  3.036820e+10

[2948 rows x 6 columns]
```

[36]:
```
# Données avec jointure externe pour visualisation
df_data_viz = pd.merge(left=df_population, right=df_eco, how='left',␣
 ↪on=["ISO3","Année"])
```

[37]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_data):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_data[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_data.shape[0] -␣
 ↪((df_data[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_data["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 2948

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2948

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2948

Colonne Totale - Nombre de valeurs NaN : 0
Colonne Totale - Nombre de valeurs non-vides : 2948

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 2948

Colonne PIB - Nombre de valeurs NaN : 0
Colonne PIB - Nombre de valeurs non-vides : 2948

Nombre de zones dans les données : 212
```

Il apparaît pertinent de créer des colonnes où les données sont rapportées à la population totale du pays. On fera cela quand on aura rassemblé toutes nos données.

[38]:
```
set_pays_data = set(df_data["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_data), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones,on="ISO3",how='inner'))
Pop_dat2023 = (df_data.groupby(by="Année")["Totale"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
↪2), "% de la population mondiale en 2023")
```

	index	ISO3	Zone	Totale
0	2490	REU	Réunion	874883.0
1	893	ESH	Sahara occidental	579729.0
2	1133	GLP	Guadeloupe	376517.0
3	2056	MTQ	Martinique	346002.0
4	2112	MYT	Mayotte	316015.0
5	1245	GUF	Guyane française	303402.0
6	516	CHA	Îles Anglo-Normandes	169724.0
7	1259	GUM	Guam	166506.0
8	3185	VIR	Îles Vierges américaines	85701.0
9	1371	IMN	Île de Man	84165.0
10	1021	FRO	Îles Féroé	54714.0
11	140	ASM	Samoa américaines	47521.0
12	2000	MNP	Îles Mariannes du Nord	45143.0
13	1105	GIB	Gibraltar	38471.0
14	251	BES	Pays-Bas caribéens	29898.0
15	1804	MAF	Saint-Martin (partie française)	27515.0
16	3227	WLF	Îles Wallis-et-Futuna	11370.0
17	348	BLM	Saint-Barthélemy	11085.0
18	2684	SPM	Saint-Pierre-et-Miquelon	5681.0
19	2600	SHN	Îles britanniques Atlantique Sud	5289.0
20	993	FLK	Îles Falkland (Malvinas)	3477.0
21	2919	TKL	Tokélaou	2397.0
22	2196	NIU	Nioué	1817.0
23	3129	VAT	Saint-Siège	799.0

```
Les données en l'état couvrent 99.96 % de la population mondiale en 2023
```

2.3 - Données de commerce

[39]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_commerce.
↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_commerce.shape[1]))
```

```
Le tableau comporte 8743 observation(s) ou article(s)
Le tableau comporte 15 colonne(s)
```

[40]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_commerce.shape[1])
#La nature des données dans chacune des colonnes
display(df_commerce.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_commerce):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_commerce[c]).
 ↪isna()).sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_commerce.shape[0]␣
 ↪- ((df_commerce[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_commerce.shape[0] - ((df_commerce[c]).isna()).sum() == 0:
        df_commerce.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 15
```

```
Code Domaine               object
Domaine                    object
Code zone (ISO3)           object
Zone                       object
Code Élément                int64
Élément                    object
Code Produit (CPC)          int64
Produit                    object
Code année                  int64
Année                       int64
Unité                      object
Valeur                    float64
Symbole                    object
Description du Symbole     object
Note                       object
dtype: object
```

```
Colonne Code Domaine - Nombre de valeurs NaN : 0
Colonne Code Domaine - Nombre de valeurs non-vides : 8743

Colonne Domaine - Nombre de valeurs NaN : 0
Colonne Domaine - Nombre de valeurs non-vides : 8743

Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 8743

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 8743

Colonne Code Élément - Nombre de valeurs NaN : 0
Colonne Code Élément - Nombre de valeurs non-vides : 8743
```

```
Colonne Élément - Nombre de valeurs NaN : 0
Colonne Élément - Nombre de valeurs non-vides : 8743

Colonne Code Produit (CPC) - Nombre de valeurs NaN : 0
Colonne Code Produit (CPC) - Nombre de valeurs non-vides : 8743

Colonne Produit - Nombre de valeurs NaN : 0
Colonne Produit - Nombre de valeurs non-vides : 8743

Colonne Code année - Nombre de valeurs NaN : 0
Colonne Code année - Nombre de valeurs non-vides : 8743

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 8743

Colonne Unité - Nombre de valeurs NaN : 0
Colonne Unité - Nombre de valeurs non-vides : 8743

Colonne Valeur - Nombre de valeurs NaN : 0
Colonne Valeur - Nombre de valeurs non-vides : 8743

Colonne Symbole - Nombre de valeurs NaN : 0
Colonne Symbole - Nombre de valeurs non-vides : 8743

Colonne Description du Symbole - Nombre de valeurs NaN : 0
Colonne Description du Symbole - Nombre de valeurs non-vides : 8743

Colonne Note - Nombre de valeurs NaN : 7053
Colonne Note - Nombre de valeurs non-vides : 1690
```

[41]:
```
display(df_commerce)
```

```
     Code Domaine                      Domaine Code zone (ISO3)         Zone  \
0             TCL  Cultures et produits animaux              AFG  Afghanistan
1             TCL  Cultures et produits animaux              AFG  Afghanistan
2             TCL  Cultures et produits animaux              AFG  Afghanistan
3             TCL  Cultures et produits animaux              AFG  Afghanistan
4             TCL  Cultures et produits animaux              AFG  Afghanistan
…             …                             …               …           …
8738          TCL  Cultures et produits animaux              ZWE     Zimbabwe
8739          TCL  Cultures et produits animaux              ZWE     Zimbabwe
8740          TCL  Cultures et produits animaux              ZWE     Zimbabwe
8741          TCL  Cultures et produits animaux              ZWE     Zimbabwe
8742          TCL  Cultures et produits animaux              ZWE     Zimbabwe

      Code Élément                 Élément  Code Produit (CPC)  \
0             5609  Importations - quantité                2151
1             5909  Exportations - quantité                2151
```

```
2             5609  Importations - quantité               2151
3             5909  Exportations - quantité               2151
4             5609  Importations - quantité               2151
…              …                      …                  …
8738          5610  Importations - quantité              21121
8739          5610  Importations - quantité              21121
8740          5610  Importations - quantité              21121
8741          5610  Importations - quantité              21121
8742          5610  Importations - quantité              21121

                Produit  Code année  Année       Unité   Valeur Symbole  \
0               Poulets        2010   2010  1000 têtes  3892.00       A
1               Poulets        2010   2010  1000 têtes     0.00       E
2               Poulets        2011   2011  1000 têtes  4720.00       A
3               Poulets        2011   2011  1000 têtes     0.00       E
4               Poulets        2012   2012  1000 têtes  1094.00       A
…                 …          …    …          …      …     …
8738  Viande de poulet        2019   2019      tonnes  1353.12       A
8739  Viande de poulet        2020   2020      tonnes  1288.10       A
8740  Viande de poulet        2021   2021      tonnes  4071.36       A
8741  Viande de poulet        2022   2022      tonnes  5124.70       A
8742  Viande de poulet        2023   2023      tonnes  7629.48       A

     Description du Symbole Note
0           Chiffre officiel  NaN
1             Valeur estimée  NaN
2           Chiffre officiel  NaN
3             Valeur estimée  NaN
4           Chiffre officiel  NaN
…                        …  …
8738        Chiffre officiel  NaN
8739        Chiffre officiel  NaN
8740        Chiffre officiel  NaN
8741        Chiffre officiel  NaN
8742        Chiffre officiel  NaN

[8743 rows x 15 columns]
```

[42]:
```
# Si la colonne présente tout le temps la même valeur (donc pas d'information),␣
 ↪on la supprime
for c in list(df_commerce):
    if (df_commerce.groupby(by=c).count()).iloc[0,0] == df_commerce.shape[0]:
        df_commerce.drop(c,axis=1,inplace=True)

display(df_commerce)
```

```
     Code zone (ISO3)         Zone  Code Élément                 Élément  \
0                 AFG  Afghanistan          5609  Importations - quantité
```

```
1                AFG  Afghanistan          5909  Exportations - quantité
2                AFG  Afghanistan          5609  Importations - quantité
3                AFG  Afghanistan          5909  Exportations - quantité
4                AFG  Afghanistan          5609  Importations - quantité
…                …          …            …                    …
8738             ZWE     Zimbabwe          5610  Importations - quantité
8739             ZWE     Zimbabwe          5610  Importations - quantité
8740             ZWE     Zimbabwe          5610  Importations - quantité
8741             ZWE     Zimbabwe          5610  Importations - quantité
8742             ZWE     Zimbabwe          5610  Importations - quantité

      Code Produit (CPC)           Produit  Code année  Année       Unité  \
0                   2151           Poulets        2010   2010  1000 têtes
1                   2151           Poulets        2010   2010  1000 têtes
2                   2151           Poulets        2011   2011  1000 têtes
3                   2151           Poulets        2011   2011  1000 têtes
4                   2151           Poulets        2012   2012  1000 têtes
…                    …                 …         …    …         …
8738               21121  Viande de poulet        2019   2019      tonnes
8739               21121  Viande de poulet        2020   2020      tonnes
8740               21121  Viande de poulet        2021   2021      tonnes
8741               21121  Viande de poulet        2022   2022      tonnes
8742               21121  Viande de poulet        2023   2023      tonnes

       Valeur Symbole Description du Symbole Note
0     3892.00       A       Chiffre officiel  NaN
1        0.00       E         Valeur estimée  NaN
2     4720.00       A       Chiffre officiel  NaN
3        0.00       E         Valeur estimée  NaN
4     1094.00       A       Chiffre officiel  NaN
…       …     …                   …  …
8738  1353.12       A       Chiffre officiel  NaN
8739  1288.10       A       Chiffre officiel  NaN
8740  4071.36       A       Chiffre officiel  NaN
8741  5124.70       A       Chiffre officiel  NaN
8742  7629.48       A       Chiffre officiel  NaN

[8743 rows x 13 columns]
```

[43]:
```
# On supprime les colonnes Code ÉlémentCode Produit  et Code année, qui sont␣
 ↪redondantes avec d'autres colonnes du DataFrame
df_commerce.drop(["Code Élément","Code Produit (CPC)","Code␣
 ↪année"],axis=1,inplace=True)
# On conserve la colonne Code zone pour de futures jointures avec d'autres␣
 ↪tables

display(df_commerce)
```

```
     Code zone (ISO3)         Zone                  Élément           Produit  \
0                 AFG  Afghanistan  Importations - quantité           Poulets
1                 AFG  Afghanistan  Exportations - quantité           Poulets
2                 AFG  Afghanistan  Importations - quantité           Poulets
3                 AFG  Afghanistan  Exportations - quantité           Poulets
4                 AFG  Afghanistan  Importations - quantité           Poulets
…                 …          …                       …               …
8738              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8739              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8740              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8741              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8742              ZWE     Zimbabwe  Importations - quantité  Viande de poulet

      Année       Unité   Valeur Symbole Description du Symbole Note
0      2010  1000 têtes  3892.00       A       Chiffre officiel  NaN
1      2010  1000 têtes     0.00       E         Valeur estimée  NaN
2      2011  1000 têtes  4720.00       A       Chiffre officiel  NaN
3      2011  1000 têtes     0.00       E         Valeur estimée  NaN
4      2012  1000 têtes  1094.00       A       Chiffre officiel  NaN
…      …          …       …     …                       …  …
8738   2019      tonnes  1353.12       A       Chiffre officiel  NaN
8739   2020      tonnes  1288.10       A       Chiffre officiel  NaN
8740   2021      tonnes  4071.36       A       Chiffre officiel  NaN
8741   2022      tonnes  5124.70       A       Chiffre officiel  NaN
8742   2023      tonnes  7629.48       A       Chiffre officiel  NaN

[8743 rows x 10 columns]
```

[44]:
```
display(df_commerce.groupby(by="Note").count())
```

```
                                                    Code zone (ISO3)  Zone  \
Note
Données estimées en utilisant les données des p…              1690  1690

                                                    Élément  Produit  Année  \
Note
Données estimées en utilisant les données des p…     1690     1690   1690

                                                    Unité  Valeur  Symbole  \
Note
Données estimées en utilisant les données des p…   1690    1690     1690

                                                    Description du Symbole
Note
Données estimées en utilisant les données des p…                    1690
```

[45]:
```
display(df_commerce.groupby(by=["Symbole","Description du Symbole"]).count())
```

```
                                          Code zone (ISO3)  Zone  Élément  \
```

```
Symbole Description du Symbole
A       Chiffre officiel                          6170  6170     6170
E       Valeur estimée                             478   478      478
I       Valeur imputée                             399   399      399
X       Chiffre de sources internationales         1696  1696     1696

                                                Produit  Année  Unité  Valeur  \
Symbole Description du Symbole
A       Chiffre officiel                           6170   6170   6170    6170
E       Valeur estimée                              478    478    478     478
I       Valeur imputée                              399    399    399     399
X       Chiffre de sources internationales          1696   1696   1696    1696

                                                Note
Symbole Description du Symbole
A       Chiffre officiel                           0
E       Valeur estimée                             0
I       Valeur imputée                             0
X       Chiffre de sources internationales  1690
```

[46]:
```
df_commerce.drop(["Symbole","Description du Symbole","Note"],axis=1,inplace=True)
display(df_commerce)
```

```
     Code zone (ISO3)         Zone                  Élément           Produit  \
0                 AFG  Afghanistan  Importations - quantité           Poulets
1                 AFG  Afghanistan  Exportations - quantité           Poulets
2                 AFG  Afghanistan  Importations - quantité           Poulets
3                 AFG  Afghanistan  Exportations - quantité           Poulets
4                 AFG  Afghanistan  Importations - quantité           Poulets
…                 …          …                     …                 …
8738              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8739              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8740              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8741              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8742              ZWE     Zimbabwe  Importations - quantité  Viande de poulet

      Année       Unité   Valeur
0      2010  1000 têtes  3892.00
1      2010  1000 têtes     0.00
2      2011  1000 têtes  4720.00
3      2011  1000 têtes     0.00
4      2012  1000 têtes  1094.00
…      …          …       …
8738   2019      tonnes  1353.12
8739   2020      tonnes  1288.10
8740   2021      tonnes  4071.36
8741   2022      tonnes  5124.70
```

```
8742   2023      tonnes  7629.48

[8743 rows x 7 columns]
```

[47]:
```
#Les données de commerce de poulets (vivants) ne sont peut-être pas pertinentes␣
 ↪pour notre étude
df_commerce = df_commerce.loc[df_commerce["Produit"]=="Viande de poulet"]
display(df_commerce)
```

```
     Code zone (ISO3)         Zone                  Élément           Produit  \
21                AFG  Afghanistan  Importations - quantité  Viande de poulet
22                AFG  Afghanistan  Importations - quantité  Viande de poulet
23                AFG  Afghanistan  Importations - quantité  Viande de poulet
24                AFG  Afghanistan  Importations - quantité  Viande de poulet
25                AFG  Afghanistan  Importations - quantité  Viande de poulet
…                 …           …                       …                 …
8738              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8739              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8740              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8741              ZWE     Zimbabwe  Importations - quantité  Viande de poulet
8742              ZWE     Zimbabwe  Importations - quantité  Viande de poulet

      Année   Unité    Valeur
21     2010  tonnes  38591.00
22     2011  tonnes  51004.00
23     2012  tonnes  21750.00
24     2013  tonnes  48389.00
25     2014  tonnes  42300.00
…      …     …       …
8738   2019  tonnes   1353.12
8739   2020  tonnes   1288.10
8740   2021  tonnes   4071.36
8741   2022  tonnes   5124.70
8742   2023  tonnes   7629.48

[4622 rows x 7 columns]
```

[48]:
```
# Si la colonne présente tout le temps la même valeur (donc pas d'information),␣
 ↪on la supprime
for c in list(df_commerce):
    if (df_commerce.groupby(by=c).count()).iloc[0,0] == df_commerce.shape[0]:
        df_commerce.drop(c,axis=1,inplace=True)

display(df_commerce)
```

```
     Code zone (ISO3)         Zone                  Élément  Année    Valeur
21                AFG  Afghanistan  Importations - quantité   2010  38591.00
22                AFG  Afghanistan  Importations - quantité   2011  51004.00
```

```
23                AFG  Afghanistan  Importations - quantité   2012  21750.00
24                AFG  Afghanistan  Importations - quantité   2013  48389.00
25                AFG  Afghanistan  Importations - quantité   2014  42300.00
…                 …          …                        …      …       …
8738              ZWE     Zimbabwe  Importations - quantité   2019   1353.12
8739              ZWE     Zimbabwe  Importations - quantité   2020   1288.10
8740              ZWE     Zimbabwe  Importations - quantité   2021   4071.36
8741              ZWE     Zimbabwe  Importations - quantité   2022   5124.70
8742              ZWE     Zimbabwe  Importations - quantité   2023   7629.48

[4622 rows x 5 columns]
```

[49]:
```
# On pivote la table de façon à ce que Élément soit retiré de la clé primaire
# remplaçant la colonne Élément en 2 colonnes avec des valeurs
df_commerce = df_commerce.pivot(index=["Code zone (ISO3)","Zone","Année"],␣
 ↪columns="Élément", values="Valeur").reset_index()
display(df_commerce)
```

```
Élément Code zone (ISO3)         Zone  Année  Exportations - quantité  \
0                    AFG  Afghanistan   2010                      NaN
1                    AFG  Afghanistan   2011                      NaN
2                    AFG  Afghanistan   2012                      NaN
3                    AFG  Afghanistan   2013                      NaN
4                    AFG  Afghanistan   2014                      NaN
…                    …          …     …                        …
2695                 ZWE     Zimbabwe   2019                      NaN
2696                 ZWE     Zimbabwe   2020                      NaN
2697                 ZWE     Zimbabwe   2021                      NaN
2698                 ZWE     Zimbabwe   2022                      NaN
2699                 ZWE     Zimbabwe   2023                      NaN

Élément  Importations - quantité
0                        38591.00
1                        51004.00
2                        21750.00
3                        48389.00
4                        42300.00
…                             …
2695                      1353.12
2696                      1288.10
2697                      4071.36
2698                      5124.70
2699                      7629.48

[2700 rows x 5 columns]
```

[50]:
```
# On remplace les NaN par des 0
df_commerce = df_commerce.fillna(0.0)
```

```
display(df_commerce)
```

```
Élément Code zone (ISO3)         Zone  Année  Exportations - quantité  \
0                     AFG  Afghanistan   2010                      0.0
1                     AFG  Afghanistan   2011                      0.0
2                     AFG  Afghanistan   2012                      0.0
3                     AFG  Afghanistan   2013                      0.0
4                     AFG  Afghanistan   2014                      0.0
…                     …           …    …                        …
2695                  ZWE     Zimbabwe   2019                      0.0
2696                  ZWE     Zimbabwe   2020                      0.0
2697                  ZWE     Zimbabwe   2021                      0.0
2698                  ZWE     Zimbabwe   2022                      0.0
2699                  ZWE     Zimbabwe   2023                      0.0

Élément  Importations - quantité
0                        38591.00
1                        51004.00
2                        21750.00
3                        48389.00
4                        42300.00
…                             …
2695                      1353.12
2696                      1288.10
2697                      4071.36
2698                      5124.70
2699                      7629.48

[2700 rows x 5 columns]
```

[51]:
```
#On renomme les colonnes
df_commerce = df_commerce.rename(columns = {"Code zone (ISO3)":"ISO3", \
                                            "Exportations - quantité": "exp",␣
 ↪"Importations - quantité": "imp"})
display(df_commerce)
```

```
Élément ISO3         Zone  Année  exp       imp
0         AFG  Afghanistan   2010  0.0  38591.00
1         AFG  Afghanistan   2011  0.0  51004.00
2         AFG  Afghanistan   2012  0.0  21750.00
3         AFG  Afghanistan   2013  0.0  48389.00
4         AFG  Afghanistan   2014  0.0  42300.00
…       …           …     …   …        …
2695      ZWE     Zimbabwe   2019  0.0   1353.12
2696      ZWE     Zimbabwe   2020  0.0   1288.10
2697      ZWE     Zimbabwe   2021  0.0   4071.36
2698      ZWE     Zimbabwe   2022  0.0   5124.70
2699      ZWE     Zimbabwe   2023  0.0   7629.48
```

```
[2700 rows x 5 columns]
```

[52]:
```
df_groupby = df_commerce.groupby(by="Zone").count().
  ↪sort_values(by='Année',ascending=False)
display(df_groupby)
```

```
Élément                                    ISO3  Année  exp  imp
Zone
Afghanistan                                  14     14   14   14
Pologne                                      14     14   14   14
Ouganda                                      14     14   14   14
Ouzbékistan                                  14     14   14   14
Pakistan                                     14     14   14   14
…                                           …     …  …  …
Équateur                                      9      9    9    9
Érythrée                                      9      9    9    9
Myanmar                                       8      8    8    8
République populaire démocratique de Corée    6      6    6    6
Soudan (ex)                                   2      2    2    2

[197 rows x 4 columns]
```

On peut remarquer qu'on n'a pas des données pour toutes les années (les données manquantes doivent probablement être des années où exportations et importations sont égales à 0 pour un pays donné). Ce problème sera résolu en faisant une jointure à gauche à partir des données de population.

[53]:
```
sr_annees = pd.Series(df_commerce["Année"].unique(), name="Année")
sr_codzone = pd.Series(df_commerce["ISO3"].unique(), name="ISO3")
empty_pk = pd.merge(sr_codzone, sr_annees, how='cross')
df_commerce = pd.merge(left=empty_pk, right=df_commerce, how='left',␣
  ↪on=["ISO3","Année"])
df_commerce = df_commerce.fillna(0.0)
df_commerce.drop("Zone",axis=1,inplace=True)
display(df_commerce)
```

```
      ISO3  Année  exp       imp
0      AFG   2010  0.0  38591.00
1      AFG   2011  0.0  51004.00
2      AFG   2012  0.0  21750.00
3      AFG   2013  0.0  48389.00
4      AFG   2014  0.0  42300.00
…    …    …  …       …
2753   ZWE   2019  0.0   1353.12
2754   ZWE   2020  0.0   1288.10
2755   ZWE   2021  0.0   4071.36
2756   ZWE   2022  0.0   5124.70
2757   ZWE   2023  0.0   7629.48
```

```
[2758 rows x 4 columns]
```

[54]:
```
df_data = pd.merge(left=df_data, right=df_commerce, how='inner',␣
 ↪on=["ISO3","Année"])
display(df_data)
```

```
     ISO3         Zone  Année      Totale    Urbaine           PIB  exp  \
0     AFG  Afghanistan   2010  28284089.0  6836980.0  1.514457e+10  0.0
1     AFG  Afghanistan   2011  29347708.0  7114473.0  1.792310e+10  0.0
2     AFG  Afghanistan   2012  30560034.0  7416295.0  1.979441e+10  0.0
3     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10  0.0
4     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10  0.0
…     …           …     …          …          …           …    …
2692  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10  0.0
2693  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10  0.0
2694  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10  0.0
2695  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10  0.0
2696  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10  0.0

           imp
0     38591.00
1     51004.00
2     21750.00
3     48389.00
4     42300.00
…          …
2692   1353.12
2693   1288.10
2694   4071.36
2695   5124.70
2696   7629.48

[2697 rows x 8 columns]
```

[55]:
```
# Données avec jointure externe pour visualisation
df_data_viz = pd.merge(left=df_data_viz, right=df_commerce, how='left',␣
 ↪on=["ISO3","Année"])
```

[56]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_data):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_data[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_data.shape[0] -␣
 ↪((df_data[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_data["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 2697

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2697

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2697

Colonne Totale - Nombre de valeurs NaN : 0
Colonne Totale - Nombre de valeurs non-vides : 2697

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 2697

Colonne PIB - Nombre de valeurs NaN : 0
Colonne PIB - Nombre de valeurs non-vides : 2697

Colonne exp - Nombre de valeurs NaN : 0
Colonne exp - Nombre de valeurs non-vides : 2697

Colonne imp - Nombre de valeurs NaN : 0
Colonne imp - Nombre de valeurs non-vides : 2697

Nombre de zones dans les données : 193
```

[57]:
```
set_pays_ref = set(df_pays_pop2023["ISO3"].unique())
set_pays_data = set(df_data["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_data), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones,on="ISO3",how='inner'))
```

	index	ISO3	Zone	Totale
0	2392	PRI	Porto Rico	3242023.0
1	2490	REU	Réunion	874883.0
2	893	ESH	Sahara occidental	579729.0
3	1133	GLP	Guadeloupe	376517.0
4	2056	MTQ	Martinique	346002.0
5	2112	MYT	Mayotte	316015.0
6	1245	GUF	Guyane française	303402.0
7	711	CUW	Curaçao	185427.0
8	516	CHA	Îles Anglo-Normandes	169724.0
9	1259	GUM	Guam	166506.0
10	1035	FSM	Micronésie (États fédérés de)	112630.0
11	13	ABW	Aruba	107939.0
12	3185	VIR	Îles Vierges américaines	85701.0
13	1371	IMN	Île de Man	84165.0
14	83	AND	Andorre	80856.0

```
15    725  CYM                        Îles Caïmanes    73038.0
16    390  BMU                             Bermudes    64698.0
17   1217  GRL                            Groenland    55922.0
18   1021  FRO                           Îles Féroé    54714.0
19    140  ASM                 Samoa américaines      47521.0
20   2849  TCA             Îles Turques-et-Caïques    46198.0
21   2000  MNP              Îles Mariannes du Nord    45143.0
22   2807  SXM  Sint Maarten (partie néerlandaise)    42749.0
23   1707  LIE                         Liechtenstein    39598.0
24   3171  VGB           Îles Vierges britanniques    38985.0
25   1832  MCO                               Monaco    38956.0
26   1902  MHL                         Îles Marshall    38827.0
27   1105  GIB                             Gibraltar    38471.0
28   2656  SMR                           Saint-Marin    33733.0
29    251  BES                  Pays-Bas caribéens      29898.0
30   1804  MAF     Saint-Martin (partie française)     27515.0
31   2350  PLW                               Palaos     17727.0
32     55  AIA                              Anguilla    14410.0
33   3227  WLF               Îles Wallis-et-Futuna     11370.0
34    348  BLM                     Saint-Barthélemy    11085.0
35   2684  SPM             Saint-Pierre-et-Miquelon      5681.0
36   2600  SHN    Îles britanniques Atlantique Sud      5289.0
37   2042  MSR                            Montserrat     4420.0
38    993  FLK             Îles Falkland (Malvinas)      3477.0
39   2919  TKL                               Tokélaou     2397.0
40   2196  NIU                                 Nioué      1817.0
41   3129  VAT                           Saint-Siège       799.0
```

[58]:
```
Pop_dat2023 = (df_data.groupby(by="Année")["Totale"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
 ↪2), "% de la population mondiale en 2023")
```

```
Les données en l'état couvrent 99.9 % de la population mondiale en 2023
```

[59]:
```
Imp_tot2023 = (df_data_viz.groupby("Année")["imp"].sum()).loc[2023]
Imp_dat2023 = (df_data.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données en l'état couvrent",round(100*Imp_dat2023/Imp_tot2023, 2),"%␣
 ↪des importations totales")
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données en l'état couvrent 100.0 % des importations totales
```

[60]:
```
print("Liste des zones exclues de l'analyse faute de données complètes :")
set_imp_ref = set(df_data_viz["ISO3"].unique())
set_imp_data = set(df_data["ISO3"].unique())
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
```

```
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_data), name="ISO3")
display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner'))
```

Liste des zones exclues de l'analyse faute de données complètes :

	ISO3	Zone	imp
0	FRO	Îles Féroé	459.71
1	NIU	Nioué	75.50
2	ABW	Aruba	NaN
3	AIA	Anguilla	NaN
4	AND	Andorre	NaN
5	ASM	Samoa américaines	NaN
6	BES	Pays-Bas caribéens	NaN
7	BMU	Bermudes	NaN
8	CHA	Îles Anglo-Normandes	NaN
9	CUW	Curaçao	NaN
10	CYM	Îles Caïmanes	NaN
11	ESH	Sahara occidental	NaN
12	FLK	Îles Falkland (Malvinas)	NaN
13	FSM	Micronésie (États fédérés de)	NaN
14	GIB	Gibraltar	NaN
15	GLP	Guadeloupe	NaN
16	GRL	Groenland	NaN
17	GUF	Guyane française	NaN
18	GUM	Guam	NaN
19	IMN	Île de Man	NaN
20	LIE	Liechtenstein	NaN
21	MCO	Monaco	NaN
22	MHL	Îles Marshall	NaN
23	MNP	Îles Mariannes du Nord	NaN
24	MSR	Montserrat	NaN
25	MTQ	Martinique	NaN
26	MYT	Mayotte	NaN
27	PLW	Palaos	NaN
28	PRI	Porto Rico	NaN
29	REU	Réunion	NaN
30	SHN	Îles britanniques Atlantique Sud	NaN
31	SMR	Saint-Marin	NaN
32	SPM	Saint-Pierre-et-Miquelon	NaN
33	SXM	Sint Maarten (partie néerlandaise)	NaN
34	TCA	Îles Turques-et-Caïques	NaN
35	TKL	Tokélaou	NaN
36	VAT	Saint-Siège	NaN
37	VGB	Îles Vierges britanniques	NaN
38	VIR	Îles Vierges américaines	NaN
39	WLF	Îles Wallis-et-Futuna	NaN

2.4 - Données de production

[61]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_prod.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_prod.shape[1]))
```

```
Le tableau comporte 2737 observation(s) ou article(s)
Le tableau comporte 15 colonne(s)
```

[62]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_prod.shape[1])
#La nature des données dans chacune des colonnes
display(df_prod.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_prod):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_prod[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_prod.shape[0] -␣
 ↪((df_prod[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_prod.shape[0] - ((df_prod[c]).isna()).sum() == 0:
        df_prod.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 15
```

```
Code Domaine               object
Domaine                    object
Code zone (ISO3)           object
Zone                       object
Code Élément                int64
Élément                    object
Code Produit (CPC)          int64
Produit                    object
Code année                  int64
Année                       int64
Unité                      object
Valeur                    float64
Symbole                    object
Description du Symbole     object
Note                       object
dtype: object


Colonne Code Domaine - Nombre de valeurs NaN : 0
Colonne Code Domaine - Nombre de valeurs non-vides : 2737

Colonne Domaine - Nombre de valeurs NaN : 0
Colonne Domaine - Nombre de valeurs non-vides : 2737

Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
```

```
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 2737

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2737

Colonne Code Élément - Nombre de valeurs NaN : 0
Colonne Code Élément - Nombre de valeurs non-vides : 2737

Colonne Élément - Nombre de valeurs NaN : 0
Colonne Élément - Nombre de valeurs non-vides : 2737

Colonne Code Produit (CPC) - Nombre de valeurs NaN : 0
Colonne Code Produit (CPC) - Nombre de valeurs non-vides : 2737

Colonne Produit - Nombre de valeurs NaN : 0
Colonne Produit - Nombre de valeurs non-vides : 2737

Colonne Code année - Nombre de valeurs NaN : 0
Colonne Code année - Nombre de valeurs non-vides : 2737

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2737

Colonne Unité - Nombre de valeurs NaN : 0
Colonne Unité - Nombre de valeurs non-vides : 2737

Colonne Valeur - Nombre de valeurs NaN : 0
Colonne Valeur - Nombre de valeurs non-vides : 2737

Colonne Symbole - Nombre de valeurs NaN : 0
Colonne Symbole - Nombre de valeurs non-vides : 2737

Colonne Description du Symbole - Nombre de valeurs NaN : 0
Colonne Description du Symbole - Nombre de valeurs non-vides : 2737

Colonne Note - Nombre de valeurs NaN : 2650
Colonne Note - Nombre de valeurs non-vides : 87
```

[63]:
```
display(df_prod)
```

```
     Code Domaine                      Domaine Code zone (ISO3)         Zone  \
0             QCL  Cultures et produits animaux              AFG  Afghanistan
1             QCL  Cultures et produits animaux              AFG  Afghanistan
2             QCL  Cultures et produits animaux              AFG  Afghanistan
3             QCL  Cultures et produits animaux              AFG  Afghanistan
4             QCL  Cultures et produits animaux              AFG  Afghanistan
…             …                          …                …          …
2732          QCL  Cultures et produits animaux              ZWE     Zimbabwe
```

```
2733          QCL  Cultures et produits animaux              ZWE     Zimbabwe
2734          QCL  Cultures et produits animaux              ZWE     Zimbabwe
2735          QCL  Cultures et produits animaux              ZWE     Zimbabwe
2736          QCL  Cultures et produits animaux              ZWE     Zimbabwe

      Code Élément     Élément  Code Produit (CPC)  \
0             5510  Production               21121
1             5510  Production               21121
2             5510  Production               21121
3             5510  Production               21121
4             5510  Production               21121
…              …         …                   …
2732          5510  Production               21121
2733          5510  Production               21121
2734          5510  Production               21121
2735          5510  Production               21121
2736          5510  Production               21121

                                    Produit  Code année  Année   Unité  \
0     Viande, poulet, fraîche ou réfrigérée        2010   2010  tonnes
1     Viande, poulet, fraîche ou réfrigérée        2011   2011  tonnes
2     Viande, poulet, fraîche ou réfrigérée        2012   2012  tonnes
3     Viande, poulet, fraîche ou réfrigérée        2013   2013  tonnes
4     Viande, poulet, fraîche ou réfrigérée        2014   2014  tonnes
…                                       …         …    …     …
2732  Viande, poulet, fraîche ou réfrigérée        2019   2019  tonnes
2733  Viande, poulet, fraîche ou réfrigérée        2020   2020  tonnes
2734  Viande, poulet, fraîche ou réfrigérée        2021   2021  tonnes
2735  Viande, poulet, fraîche ou réfrigérée        2022   2022  tonnes
2736  Viande, poulet, fraîche ou réfrigérée        2023   2023  tonnes

         Valeur Symbole Description du Symbole Note
0      28000.00       E         Valeur estimée  NaN
1      25600.00       E         Valeur estimée  NaN
2      24800.00       E         Valeur estimée  NaN
3      26400.00       E         Valeur estimée  NaN
4      24809.32       I         Valeur imputée  NaN
…        …     …                   …  …
2732  114344.00       A       Chiffre officiel  NaN
2733  111546.00       A       Chiffre officiel  NaN
2734  113000.00       E         Valeur estimée  NaN
2735  118225.03       I         Valeur imputée  NaN
2736   97773.99       E         Valeur estimée  NaN

[2737 rows x 15 columns]
```

[64]:
```
# Si la colonne présente tout le temps la même valeur (donc pas d'information), on la supprime
for c in list(df_prod):
    if (df_prod.groupby(by=c).count()).iloc[0,0] == df_prod.shape[0]:
        df_prod.drop(c,axis=1,inplace=True)

display(df_prod)
```

```
     Code zone (ISO3)         Zone  Code année  Année     Valeur Symbole  \
0                 AFG  Afghanistan        2010   2010   28000.00       E
1                 AFG  Afghanistan        2011   2011   25600.00       E
2                 AFG  Afghanistan        2012   2012   24800.00       E
3                 AFG  Afghanistan        2013   2013   26400.00       E
4                 AFG  Afghanistan        2014   2014   24809.32       I
…                 …            …          …      …        …      …
2732              ZWE     Zimbabwe        2019   2019  114344.00       A
2733              ZWE     Zimbabwe        2020   2020  111546.00       A
2734              ZWE     Zimbabwe        2021   2021  113000.00       E
2735              ZWE     Zimbabwe        2022   2022  118225.03       I
2736              ZWE     Zimbabwe        2023   2023   97773.99       E

     Description du Symbole Note
0             Valeur estimée  NaN
1             Valeur estimée  NaN
2             Valeur estimée  NaN
3             Valeur estimée  NaN
4             Valeur imputée  NaN
…                        …  …
2732       Chiffre officiel  NaN
2733       Chiffre officiel  NaN
2734         Valeur estimée  NaN
2735         Valeur imputée  NaN
2736         Valeur estimée  NaN

[2737 rows x 8 columns]
```

[65]:
```
display(df_prod.groupby(by=["Symbole","Description du Symbole"]).count())
```

```
                                          Code zone (ISO3)  Zone  \
Symbole Description du Symbole
A       Chiffre officiel                              1573  1573
E       Valeur estimée                                 285   285
I       Valeur imputée                                 792   792
X       Chiffre de sources internationales              87    87

                                          Code année  Année  Valeur  Note
Symbole Description du Symbole
A       Chiffre officiel                        1573   1573    1573     0
```

```
E       Valeur estimée                          285    285     285     0
I       Valeur imputée                          792    792     792     0
X       Chiffre de sources internationales       87     87      87    87
```

[66]:
```
# On supprime les colonnes inutiles ou redondantes
df_prod.drop(["Symbole","Description du Symbole","Code␣
 ↪année","Note"],axis=1,inplace=True)
# On conserve la colonne Code zone pour de futures jointures avec d'autres␣
 ↪tables

df_prod = df_prod.rename(columns = {"Code zone (ISO3)": "ISO3","Valeur":␣
 ↪"prod"})

display(df_prod)
```

```
     ISO3         Zone  Année       prod
0     AFG  Afghanistan   2010   28000.00
1     AFG  Afghanistan   2011   25600.00
2     AFG  Afghanistan   2012   24800.00
3     AFG  Afghanistan   2013   26400.00
4     AFG  Afghanistan   2014   24809.32
…     …          …     …         …
2732  ZWE     Zimbabwe   2019  114344.00
2733  ZWE     Zimbabwe   2020  111546.00
2734  ZWE     Zimbabwe   2021  113000.00
2735  ZWE     Zimbabwe   2022  118225.03
2736  ZWE     Zimbabwe   2023   97773.99

[2737 rows x 4 columns]
```

[67]:
```
display(df_prod.groupby(by="Zone").count().sort_values("Année",ascending=False))
```

```
              ISO3  Année  prod
Zone
Afghanistan     14     14    14
Madagascar      14     14    14
Ouzbékistan     14     14    14
Pakistan        14     14    14
Palestine       14     14    14
…              …     …     …
Roumanie        13     13    13
Soudan          12     12    12
Soudan du Sud   12     12    12
Slovaquie       10     10    10
Soudan (ex)      2      2     2

[197 rows x 3 columns]
```

[68]:
```
sr_annees = pd.Series(df_prod["Année"].unique(), name="Année")
sr_codzone = pd.Series(df_prod["ISO3"].unique(), name="ISO3")
empty_pk = pd.merge(sr_codzone, sr_annees, how='cross')
df_prod = pd.merge(left=empty_pk, right=df_prod, how='left',␣
 ↪on=["ISO3","Année"])
df_prod = df_prod.fillna(0.0)
df_prod.drop("Zone",axis=1,inplace=True)
display(df_prod)
```

```
     ISO3  Année       prod
0     AFG   2010   28000.00
1     AFG   2011   25600.00
2     AFG   2012   24800.00
3     AFG   2013   26400.00
4     AFG   2014   24809.32
…   …     …         …
2753  ZWE   2019  114344.00
2754  ZWE   2020  111546.00
2755  ZWE   2021  113000.00
2756  ZWE   2022  118225.03
2757  ZWE   2023   97773.99

[2758 rows x 3 columns]
```

[69]:
```
df_data = pd.merge(left=df_data, right=df_prod, how='inner',␣
 ↪on=["ISO3","Année"])
display(df_data)
```

```
     ISO3         Zone  Année      Totale    Urbaine           PIB  exp  \
0     AFG  Afghanistan   2010  28284089.0  6836980.0  1.514457e+10  0.0
1     AFG  Afghanistan   2011  29347708.0  7114473.0  1.792310e+10  0.0
2     AFG  Afghanistan   2012  30560034.0  7416295.0  1.979441e+10  0.0
3     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10  0.0
4     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10  0.0
…   …          …    …         …          …          …             …
2664  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10  0.0
2665  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10  0.0
2666  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10  0.0
2667  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10  0.0
2668  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10  0.0

           imp       prod
0     38591.00   28000.00
1     51004.00   25600.00
2     21750.00   24800.00
3     48389.00   26400.00
4     42300.00   24809.32
…         …         …
```

```
2664   1353.12  114344.00
2665   1288.10  111546.00
2666   4071.36  113000.00
2667   5124.70  118225.03
2668   7629.48   97773.99

[2669 rows x 9 columns]
```

[70]:
```
# Données avec jointure externe pour visualisation
df_data_viz = pd.merge(left=df_data_viz, right=df_prod, how='left',␣
 ↪on=["ISO3","Année"])
```

[71]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_data):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_data[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_data.shape[0] -␣
 ↪((df_data[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_data["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 2669

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2669

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2669

Colonne Totale - Nombre de valeurs NaN : 0
Colonne Totale - Nombre de valeurs non-vides : 2669

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 2669

Colonne PIB - Nombre de valeurs NaN : 0
Colonne PIB - Nombre de valeurs non-vides : 2669

Colonne exp - Nombre de valeurs NaN : 0
Colonne exp - Nombre de valeurs non-vides : 2669

Colonne imp - Nombre de valeurs NaN : 0
Colonne imp - Nombre de valeurs non-vides : 2669

Colonne prod - Nombre de valeurs NaN : 0
Colonne prod - Nombre de valeurs non-vides : 2669
```

```
Nombre de zones dans les données : 191
```

[72]:
```
set_pays_data = set(df_data["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_data), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones,on="ISO3",how='inner'))
Pop_dat2023 = (df_data.groupby(by="Année")["Totale"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
↪2), "% de la population mondiale en 2023")
```

	index	ISO3	Zone	Totale
0	2392	PRI	Porto Rico	3242023.0
1	781	DJI	Djibouti	1152944.0
2	2490	REU	Réunion	874883.0
3	893	ESH	Sahara occidental	579729.0
4	1874	MDV	Maldives	525994.0
5	1133	GLP	Guadeloupe	376517.0
6	2056	MTQ	Martinique	346002.0
7	2112	MYT	Mayotte	316015.0
8	1245	GUF	Guyane française	303402.0
9	711	CUW	Curaçao	185427.0
10	516	CHA	Îles Anglo-Normandes	169724.0
11	1259	GUM	Guam	166506.0
12	1035	FSM	Micronésie (États fédérés de)	112630.0
13	13	ABW	Aruba	107939.0
14	3185	VIR	Îles Vierges américaines	85701.0
15	1371	IMN	Île de Man	84165.0
16	83	AND	Andorre	80856.0
17	725	CYM	Îles Caïmanes	73038.0
18	390	BMU	Bermudes	64698.0
19	1217	GRL	Groenland	55922.0
20	1021	FRO	Îles Féroé	54714.0
21	140	ASM	Samoa américaines	47521.0
22	2849	TCA	Îles Turques-et-Caïques	46198.0
23	2000	MNP	Îles Mariannes du Nord	45143.0
24	2807	SXM	Sint Maarten (partie néerlandaise)	42749.0
25	1707	LIE	Liechtenstein	39598.0
26	3171	VGB	Îles Vierges britanniques	38985.0
27	1832	MCO	Monaco	38956.0
28	1902	MHL	Îles Marshall	38827.0
29	1105	GIB	Gibraltar	38471.0
30	2656	SMR	Saint-Marin	33733.0
31	251	BES	Pays-Bas caribéens	29898.0
32	1804	MAF	Saint-Martin (partie française)	27515.0
33	2350	PLW	Palaos	17727.0
34	55	AIA	Anguilla	14410.0
35	3227	WLF	Îles Wallis-et-Futuna	11370.0
36	348	BLM	Saint-Barthélemy	11085.0

```
37   2684  SPM            Saint-Pierre-et-Miquelon     5681.0
38   2600  SHN    Îles britanniques Atlantique Sud     5289.0
39   2042  MSR                           Montserrat     4420.0
40    993  FLK           Îles Falkland (Malvinas)     3477.0
41   2919  TKL                             Tokélaou     2397.0
42   2196  NIU                                Nioué     1817.0
43   3129  VAT                          Saint-Siège      799.0
```

Les données en l'état couvrent 99.88 % de la population mondiale en 2023

[73]:
```
Imp_dat2023 = (df_data.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données en l'état couvrent",round(100*Imp_dat2023/Imp_tot2023, 2),"%␣
 ↪des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")
set_imp_ref = set(df_data_viz["ISO3"].unique())
set_imp_data = set(df_data["ISO3"].unique())
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_data), name="ISO3")

display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données en l'état couvrent 99.87 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :
```

	ISO3	Zone	imp
0	MDV	Maldives	14119.76
1	DJI	Djibouti	3389.86
2	FRO	Îles Féroé	459.71
3	NIU	Nioué	75.50

2.5 - Données de disponibilités

[74]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_dispo.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_dispo.shape[1]))
```

```
Le tableau comporte 23764 observation(s) ou article(s)
Le tableau comporte 15 colonne(s)
```

[75]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_dispo.shape[1])
#La nature des données dans chacune des colonnes
display(df_dispo.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
```

```
for c in list(df_dispo):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_dispo[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_dispo.shape[0] -␣
 ↪((df_dispo[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_dispo.shape[0] - ((df_dispo[c]).isna()).sum() == 0:
        df_dispo.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 15
```

```
Code Domaine               object
Domaine                    object
Code zone (ISO3)           object
Zone                       object
Code Élément                int64
Élément                    object
Code Produit (FBS)         object
Produit                    object
Code année                  int64
Année                       int64
Unité                      object
Valeur                      int64
Symbole                    object
Description du Symbole     object
Note                      float64
dtype: object
```

```
Colonne Code Domaine - Nombre de valeurs NaN : 0
Colonne Code Domaine - Nombre de valeurs non-vides : 23764

Colonne Domaine - Nombre de valeurs NaN : 0
Colonne Domaine - Nombre de valeurs non-vides : 23764

Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 23764

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 23764

Colonne Code Élément - Nombre de valeurs NaN : 0
Colonne Code Élément - Nombre de valeurs non-vides : 23764

Colonne Élément - Nombre de valeurs NaN : 0
Colonne Élément - Nombre de valeurs non-vides : 23764

Colonne Code Produit (FBS) - Nombre de valeurs NaN : 0
```

```
Colonne Code Produit (FBS) - Nombre de valeurs non-vides : 23764

Colonne Produit - Nombre de valeurs NaN : 0
Colonne Produit - Nombre de valeurs non-vides : 23764

Colonne Code année - Nombre de valeurs NaN : 0
Colonne Code année - Nombre de valeurs non-vides : 23764

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 23764

Colonne Unité - Nombre de valeurs NaN : 0
Colonne Unité - Nombre de valeurs non-vides : 23764

Colonne Valeur - Nombre de valeurs NaN : 0
Colonne Valeur - Nombre de valeurs non-vides : 23764

Colonne Symbole - Nombre de valeurs NaN : 0
Colonne Symbole - Nombre de valeurs non-vides : 23764

Colonne Description du Symbole - Nombre de valeurs NaN : 0
Colonne Description du Symbole - Nombre de valeurs non-vides : 23764

Colonne Note - Nombre de valeurs NaN : 23764
Colonne Note - Nombre de valeurs non-vides : 0
```

[76]:
```
display(df_dispo)
```

```
      Code Domaine                      Domaine Code zone (ISO3)         Zone  \
0              FBS  Bilans Alimentaires (2010-)              AFG  Afghanistan
1              FBS  Bilans Alimentaires (2010-)              AFG  Afghanistan
2              FBS  Bilans Alimentaires (2010-)              AFG  Afghanistan
3              FBS  Bilans Alimentaires (2010-)              AFG  Afghanistan
4              FBS  Bilans Alimentaires (2010-)              AFG  Afghanistan
…              …                            …               …           …
23759          FBS  Bilans Alimentaires (2010-)              ZWE     Zimbabwe
23760          FBS  Bilans Alimentaires (2010-)              ZWE     Zimbabwe
23761          FBS  Bilans Alimentaires (2010-)              ZWE     Zimbabwe
23762          FBS  Bilans Alimentaires (2010-)              ZWE     Zimbabwe
23763          FBS  Bilans Alimentaires (2010-)              ZWE     Zimbabwe

       Code Élément                   Élément Code Produit (FBS)  \
0              5301  Disponibilité intérieure              S2731
1              5142                Nourriture              S2731
2              5301  Disponibilité intérieure              S2731
3              5142                Nourriture              S2731
4              5301  Disponibilité intérieure              S2731
…              …                           …               …
```

```
23759          5142                Nourriture              S2735
23760          5301  Disponibilité intérieure              S2735
23761          5142                Nourriture              S2735
23762          5301  Disponibilité intérieure              S2735
23763          5142                Nourriture              S2735

                Produit  Code année  Année   Unité  Valeur Symbole  \
0      Viande de Bovins        2010   2010  1000 t     134       I
1      Viande de Bovins        2010   2010  1000 t     134       I
2      Viande de Bovins        2011   2011  1000 t     140       I
3      Viande de Bovins        2011   2011  1000 t     140       I
4      Viande de Bovins        2012   2012  1000 t     134       I
…                   …         …    …     …      …     …
23759     Viande, Autre        2020   2020  1000 t      37       I
23760     Viande, Autre        2021   2021  1000 t      37       I
23761     Viande, Autre        2021   2021  1000 t      38       I
23762     Viande, Autre        2022   2022  1000 t      38       I
23763     Viande, Autre        2022   2022  1000 t      38       I

      Description du Symbole
0              Valeur imputée
1              Valeur imputée
2              Valeur imputée
3              Valeur imputée
4              Valeur imputée
…                        …
23759          Valeur imputée
23760          Valeur imputée
23761          Valeur imputée
23762          Valeur imputée
23763          Valeur imputée

[23764 rows x 14 columns]
```

[77]:
```python
# Si la colonne présente tout le temps la même valeur (donc pas d'information), on la supprime
for c in list(df_dispo):
    if (df_dispo.groupby(by=c).count()).iloc[0,0] == df_dispo.shape[0]:
        df_dispo.drop(c,axis=1,inplace=True)

display(df_dispo)
```

```
      Code zone (ISO3)         Zone  Code Élément                  Élément  \
0                  AFG  Afghanistan          5301  Disponibilité intérieure
1                  AFG  Afghanistan          5142                Nourriture
2                  AFG  Afghanistan          5301  Disponibilité intérieure
3                  AFG  Afghanistan          5142                Nourriture
4                  AFG  Afghanistan          5301  Disponibilité intérieure
```

```
…                 …          …           …                          …
23759               ZWE     Zimbabwe          5142                Nourriture
23760               ZWE     Zimbabwe          5301  Disponibilité intérieure
23761               ZWE     Zimbabwe          5142                Nourriture
23762               ZWE     Zimbabwe          5301  Disponibilité intérieure
23763               ZWE     Zimbabwe          5142                Nourriture

      Code Produit (FBS)           Produit  Code année  Année  Valeur Symbole  \
0                  S2731  Viande de Bovins        2010   2010     134       I
1                  S2731  Viande de Bovins        2010   2010     134       I
2                  S2731  Viande de Bovins        2011   2011     140       I
3                  S2731  Viande de Bovins        2011   2011     140       I
4                  S2731  Viande de Bovins        2012   2012     134       I
…                  …                 …         …     …      …     …
23759              S2735     Viande, Autre        2020   2020      37       I
23760              S2735     Viande, Autre        2021   2021      37       I
23761              S2735     Viande, Autre        2021   2021      38       I
23762              S2735     Viande, Autre        2022   2022      38       I
23763              S2735     Viande, Autre        2022   2022      38       I

      Description du Symbole
0              Valeur imputée
1              Valeur imputée
2              Valeur imputée
3              Valeur imputée
4              Valeur imputée
…                        …
23759          Valeur imputée
23760          Valeur imputée
23761          Valeur imputée
23762          Valeur imputée
23763          Valeur imputée

[23764 rows x 11 columns]
```

[78]:
```
df_dispo["Valeur"] *= 1000 # Les valeurs étaient données en milliers de tonnes
display(display(df_dispo.groupby(by=["Symbole","Description du Symbole"]).
  ↪count()))
```

```
                                Code zone (ISO3)   Zone  Code Élément  \
Symbole Description du Symbole
A       Chiffre officiel                       4      4             4
E       Valeur estimée                       130    130           130
I       Valeur imputée                     23630  23630         23630

                                Élément  Code Produit (FBS)  Produit  \
Symbole Description du Symbole
A       Chiffre officiel              4                   4        4
```

```
E       Valeur estimée              130                 130      130
I       Valeur imputée            23630               23630    23630

                                Code année  Année  Valeur
Symbole Description du Symbole
A       Chiffre officiel                 4      4       4
E       Valeur estimée                 130    130     130
I       Valeur imputée               23630  23630   23630

None
```

Le degré de confiance dans les données est ici significativement plus faible que précédemment.

[79]:
```
# On supprime les colonnes inutiles ou redondantes
df_dispo.drop(["Symbole","Description du Symbole","Code année","Code␣
 ↪Élément","Code Produit (FBS)"],axis=1,inplace=True)
# On conserve la colonne Code zone pour de futures jointures avec d'autres␣
 ↪tables

display(df_dispo)
```

```
      Code zone (ISO3)         Zone                    Élément  \
0                  AFG  Afghanistan  Disponibilité intérieure
1                  AFG  Afghanistan                 Nourriture
2                  AFG  Afghanistan  Disponibilité intérieure
3                  AFG  Afghanistan                 Nourriture
4                  AFG  Afghanistan  Disponibilité intérieure
…                  …          …                       …
23759              ZWE     Zimbabwe                 Nourriture
23760              ZWE     Zimbabwe  Disponibilité intérieure
23761              ZWE     Zimbabwe                 Nourriture
23762              ZWE     Zimbabwe  Disponibilité intérieure
23763              ZWE     Zimbabwe                 Nourriture

                Produit  Année  Valeur
0      Viande de Bovins   2010  134000
1      Viande de Bovins   2010  134000
2      Viande de Bovins   2011  140000
3      Viande de Bovins   2011  140000
4      Viande de Bovins   2012  134000
…                  …    …      …
23759     Viande, Autre   2020   37000
23760     Viande, Autre   2021   37000
23761     Viande, Autre   2021   38000
23762     Viande, Autre   2022   38000
23763     Viande, Autre   2022   38000

[23764 rows x 6 columns]
```

[80]:
```
# On remplace "Disponibilité intérieure" par "Disponibilité" et on concatène␣
 ↪les colonnes Élément et Produit pour préparer le pivot
df_dispo["Élément"] = df_dispo["Élément"].replace("Disponibilité␣
 ↪intérieure","Disponibilité")
df_dispo["Élément"] = df_dispo["Élément"] + " - " + df_dispo["Produit"]
df_dispo.drop(["Produit"],axis=1,inplace=True)
display(df_dispo)
```

```
      Code zone (ISO3)         Zone                              Élément  Année  \
0                  AFG  Afghanistan  Disponibilité - Viande de Bovins   2010
1                  AFG  Afghanistan     Nourriture - Viande de Bovins   2010
2                  AFG  Afghanistan  Disponibilité - Viande de Bovins   2011
3                  AFG  Afghanistan     Nourriture - Viande de Bovins   2011
4                  AFG  Afghanistan  Disponibilité - Viande de Bovins   2012
…                  …           …                                 …    …
23759              ZWE     Zimbabwe        Nourriture - Viande, Autre   2020
23760              ZWE     Zimbabwe     Disponibilité - Viande, Autre   2021
23761              ZWE     Zimbabwe        Nourriture - Viande, Autre   2021
23762              ZWE     Zimbabwe     Disponibilité - Viande, Autre   2022
23763              ZWE     Zimbabwe        Nourriture - Viande, Autre   2022

       Valeur
0      134000
1      134000
2      140000
3      140000
4      134000
…       …
23759   37000
23760   37000
23761   38000
23762   38000
23763   38000

[23764 rows x 5 columns]
```

[81]:
```
df_dispo = df_dispo.pivot(index=["Code zone (ISO3)","Zone","Année"],␣
 ↪columns="Élément", values="Valeur").reset_index()
display(df_dispo)
```

```
Élément Code zone (ISO3)         Zone  Année  \
0                   AFG  Afghanistan   2010
1                   AFG  Afghanistan   2011
2                   AFG  Afghanistan   2012
3                   AFG  Afghanistan   2013
4                   AFG  Afghanistan   2014
…                   …           …     …
2378                ZWE     Zimbabwe   2018
```

```
2379                 ZWE     Zimbabwe   2019
2380                 ZWE     Zimbabwe   2020
2381                 ZWE     Zimbabwe   2021
2382                 ZWE     Zimbabwe   2022

Élément  Disponibilité - Viande d'Ovins/Caprins  \
0                                      142000.0
1                                      136000.0
2                                      153000.0
3                                      164000.0
4                                      166000.0
…                                          …
2378                                    27000.0
2379                                    27000.0
2380                                    28000.0
2381                                    31000.0
2382                                    32000.0

Élément  Disponibilité - Viande de Bovins  \
0                                 134000.0
1                                 140000.0
2                                 134000.0
3                                 132000.0
4                                 163000.0
…                                    …
2378                              617000.0
2379                              629000.0
2380                              629000.0
2381                              700000.0
2382                              725000.0

Élément  Disponibilité - Viande de Volailles  \
0                                    65000.0
1                                    56000.0
2                                    62000.0
3                                    67000.0
4                                    69000.0
…                                      …
2378                                 66000.0
2379                                 69000.0
2380                                113000.0
2381                                117000.0
2382                                117000.0

Élément  Disponibilité - Viande de porcins  Disponibilité - Viande, Autre  \
0                                      0.0                         11000.0
1                                      NaN                         10000.0
2                                      NaN                         11000.0
```

```
3                                    NaN                          12000.0
4                                 1000.0                          14000.0
…                                    …                               …
2378                              8000.0                          37000.0
2379                             11000.0                          37000.0
2380                             10000.0                          37000.0
2381                             12000.0                          37000.0
2382                             11000.0                          38000.0

Élément  Nourriture - Viande d'Ovins/Caprins  Nourriture - Viande de Bovins  \
0                                   142000.0                        134000.0
1                                   136000.0                        140000.0
2                                   153000.0                        134000.0
3                                   164000.0                        132000.0
4                                   166000.0                        163000.0
…                                      …                               …
2378                                27000.0                        617000.0
2379                                27000.0                        629000.0
2380                                28000.0                        629000.0
2381                                31000.0                        700000.0
2382                                32000.0                        725000.0

Élément  Nourriture - Viande de Volailles  Nourriture - Viande de porcins  \
0                                  65000.0                            NaN
1                                  56000.0                            NaN
2                                  62000.0                            NaN
3                                  67000.0                            NaN
4                                  69000.0                         1000.0
…                                     …                               …
2378                               66000.0                         8000.0
2379                               69000.0                        11000.0
2380                              113000.0                        10000.0
2381                              117000.0                        12000.0
2382                              117000.0                        11000.0

Élément  Nourriture - Viande, Autre
0                           11000.0
1                           10000.0
2                           11000.0
3                           12000.0
4                           14000.0
…                              …
2378                        37000.0
2379                        38000.0
2380                        37000.0
2381                        38000.0
2382                        38000.0
```

[2383 rows x 13 columns]

[82]:
```
df_dispo = df_dispo.fillna(0.0)
display(df_dispo)
```

```
Élément Code zone (ISO3)         Zone  Année  \
0                    AFG  Afghanistan   2010
1                    AFG  Afghanistan   2011
2                    AFG  Afghanistan   2012
3                    AFG  Afghanistan   2013
4                    AFG  Afghanistan   2014
…                    …            …     …
2378                 ZWE     Zimbabwe   2018
2379                 ZWE     Zimbabwe   2019
2380                 ZWE     Zimbabwe   2020
2381                 ZWE     Zimbabwe   2021
2382                 ZWE     Zimbabwe   2022

Élément  Disponibilité - Viande d'Ovins/Caprins  \
0                                       142000.0
1                                       136000.0
2                                       153000.0
3                                       164000.0
4                                       166000.0
…                                            …
2378                                     27000.0
2379                                     27000.0
2380                                     28000.0
2381                                     31000.0
2382                                     32000.0

Élément  Disponibilité - Viande de Bovins  \
0                                 134000.0
1                                 140000.0
2                                 134000.0
3                                 132000.0
4                                 163000.0
…                                      …
2378                              617000.0
2379                              629000.0
2380                              629000.0
2381                              700000.0
2382                              725000.0

Élément  Disponibilité - Viande de Volailles  \
0                                    65000.0
1                                    56000.0
2                                    62000.0
```

```
3                                    67000.0
4                                    69000.0
…                                      …
2378                                 66000.0
2379                                 69000.0
2380                                113000.0
2381                                117000.0
2382                                117000.0

Élément  Disponibilité - Viande de porcins  Disponibilité - Viande, Autre  \
0                                      0.0                         11000.0
1                                      0.0                         10000.0
2                                      0.0                         11000.0
3                                      0.0                         12000.0
4                                   1000.0                         14000.0
…                                      …                             …
2378                                8000.0                         37000.0
2379                               11000.0                         37000.0
2380                               10000.0                         37000.0
2381                               12000.0                         37000.0
2382                               11000.0                         38000.0

Élément  Nourriture - Viande d'Ovins/Caprins  Nourriture - Viande de Bovins  \
0                                   142000.0                        134000.0
1                                   136000.0                        140000.0
2                                   153000.0                        134000.0
3                                   164000.0                        132000.0
4                                   166000.0                        163000.0
…                                      …                             …
2378                                27000.0                        617000.0
2379                                27000.0                        629000.0
2380                                28000.0                        629000.0
2381                                31000.0                        700000.0
2382                                32000.0                        725000.0

Élément  Nourriture - Viande de Volailles  Nourriture - Viande de porcins  \
0                                  65000.0                             0.0
1                                  56000.0                             0.0
2                                  62000.0                             0.0
3                                  67000.0                             0.0
4                                  69000.0                          1000.0
…                                    …                               …
2378                               66000.0                          8000.0
2379                               69000.0                         11000.0
2380                              113000.0                         10000.0
2381                              117000.0                         12000.0
2382                              117000.0                         11000.0
```

```
Élément  Nourriture - Viande, Autre
0                             11000.0
1                             10000.0
2                             11000.0
3                             12000.0
4                             14000.0
…                                …
2378                          37000.0
2379                          38000.0
2380                          37000.0
2381                          38000.0
2382                          38000.0

[2383 rows x 13 columns]
```

[83]:
```
(df_dispo.iloc[:,[8,9,10,11,12]]).sum(axis=1)
```

[83]:
```
0       352000.0
1       342000.0
2       360000.0
3       375000.0
4       413000.0
          …
2378    755000.0
2379    774000.0
2380    817000.0
2381    898000.0
2382    923000.0
Length: 2383, dtype: float64
```

[84]:
```
df_dispo["Disponibilité - Viandes"] = (df_dispo.iloc[:,[3,4,5,6,7]]).sum(axis=1)
df_dispo["Nourriture - Viandes"] = (df_dispo.iloc[:,[8,9,10,11,12]]).sum(axis=1)
df_dispo.drop(df_dispo.columns[[3,4,6,7,8,9,11,12]],axis=1,inplace=True)
display(df_dispo)
```

```
Élément Code zone (ISO3)         Zone  Année  \
0                     AFG  Afghanistan   2010
1                     AFG  Afghanistan   2011
2                     AFG  Afghanistan   2012
3                     AFG  Afghanistan   2013
4                     AFG  Afghanistan   2014
…                   …            …     …
2378                  ZWE     Zimbabwe   2018
2379                  ZWE     Zimbabwe   2019
2380                  ZWE     Zimbabwe   2020
2381                  ZWE     Zimbabwe   2021
2382                  ZWE     Zimbabwe   2022
```

```
Élément  Disponibilité - Viande de Volailles  \
0                                      65000.0
1                                      56000.0
2                                      62000.0
3                                      67000.0
4                                      69000.0
…                                          …
2378                                   66000.0
2379                                   69000.0
2380                                  113000.0
2381                                  117000.0
2382                                  117000.0

Élément  Nourriture - Viande de Volailles  Disponibilité - Viandes  \
0                                    65000.0                 352000.0
1                                    56000.0                 342000.0
2                                    62000.0                 360000.0
3                                    67000.0                 375000.0
4                                    69000.0                 413000.0
…                                        …                       …
2378                                 66000.0                 755000.0
2379                                 69000.0                 773000.0
2380                                113000.0                 817000.0
2381                                117000.0                 897000.0
2382                                117000.0                 923000.0

Élément  Nourriture - Viandes
0                     352000.0
1                     342000.0
2                     360000.0
3                     375000.0
4                     413000.0
…                         …
2378                  755000.0
2379                  774000.0
2380                  817000.0
2381                  898000.0
2382                  923000.0

[2383 rows x 7 columns]
```

[85]:
```
for c in list(df_dispo):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_dispo[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_dispo.shape[0] -␣
 ↪((df_dispo[c]).isna()).sum())
```

```
Colonne Code zone (ISO3) - Nombre de valeurs NaN : 0
Colonne Code zone (ISO3) - Nombre de valeurs non-vides : 2383

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2383

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2383

Colonne Disponibilité - Viande de Volailles - Nombre de valeurs NaN : 0
Colonne Disponibilité - Viande de Volailles - Nombre de valeurs non-vides : 2383

Colonne Nourriture - Viande de Volailles - Nombre de valeurs NaN : 0
Colonne Nourriture - Viande de Volailles - Nombre de valeurs non-vides : 2383

Colonne Disponibilité - Viandes - Nombre de valeurs NaN : 0
Colonne Disponibilité - Viandes - Nombre de valeurs non-vides : 2383

Colonne Nourriture - Viandes - Nombre de valeurs NaN : 0
Colonne Nourriture - Viandes - Nombre de valeurs non-vides : 2383
```

[86]:
```
sr_annees = pd.Series(df_dispo["Année"].unique(), name="Année")
sr_codzone = pd.Series(df_dispo["Code zone (ISO3)"].unique(), name="Code zone␣
↪(ISO3)")
empty_pk = pd.merge(sr_codzone, sr_annees, how='cross')
df_dispo = pd.merge(left=empty_pk, right=df_dispo, how='left', on=["Code zone␣
↪(ISO3)","Année"])
df_dispo = df_dispo.fillna(0.0)
df_dispo.drop("Zone",axis=1,inplace=True)
display(df_dispo)
```

```
     Code zone (ISO3)  Année  Disponibilité - Viande de Volailles  \
0                 AFG   2010                                 65000.0
1                 AFG   2011                                 56000.0
2                 AFG   2012                                 62000.0
3                 AFG   2013                                 67000.0
4                 AFG   2014                                 69000.0
…                 …     …                                    …
2465              ZWE   2018                                 66000.0
2466              ZWE   2019                                 69000.0
2467              ZWE   2020                                113000.0
2468              ZWE   2021                                117000.0
2469              ZWE   2022                                117000.0

      Nourriture - Viande de Volailles  Disponibilité - Viandes  \
0                              65000.0                352000.0
1                              56000.0                342000.0
2                              62000.0                360000.0
```

```
3                                67000.0                 375000.0
4                                69000.0                 413000.0
…                                    …                      …
2465                             66000.0                 755000.0
2466                             69000.0                 773000.0
2467                            113000.0                 817000.0
2468                            117000.0                 897000.0
2469                            117000.0                 923000.0

      Nourriture - Viandes
0                 352000.0
1                 342000.0
2                 360000.0
3                 375000.0
4                 413000.0
…                      …
2465              755000.0
2466              774000.0
2467              817000.0
2468              898000.0
2469              923000.0

[2470 rows x 6 columns]
```

Les données de disponibilité ne vont pas nous servir pour l'analyse des pays. En revanche, cela donne des informations intéressantes sur la consommation de poulet vis-à-vis des autres types de viandes.

[87]:
```
# Données avec jointure externe pour visualisation
df_data_viz = pd.merge(left=df_data_viz, right=df_dispo, how='left',\
                       left_on=["ISO3","Année"],\
                       right_on=["Code zone (ISO3)","Année"])
```

2.6 - Données de prix

[88]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_prix.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_prix.shape[1]))
```

```
Le tableau comporte 214 observation(s) ou article(s)
Le tableau comporte 2 colonne(s)
```

[89]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_prix.shape[1])
#La nature des données dans chacune des colonnes
display(df_prix.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_prix):
```

```
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_prix[c]).isna()).
↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_prix.shape[0] -␣
↪((df_prix[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_prix.shape[0] - ((df_prix[c]).isna()).sum() == 0:
        df_prix.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 2

ISO3     object
Prix    float64
dtype: object


Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 214

Colonne Prix - Nombre de valeurs NaN : 0
Colonne Prix - Nombre de valeurs non-vides : 214
```

[90]:
```
display(df_prix)
```

```
    ISO3    Prix
0    AFG  1270.0
1    ALB   880.9
2    DZA   945.9
3    ASM  1776.0
4    AND  3221.0
..   …     …
209  VEN  1476.0
210  VNM   990.9
211  YEM  1371.0
212  ZMB  1416.0
213  ZWE   657.0

[214 rows x 2 columns]
```

[91]:
```
set_imp_prix = set(df_prix["ISO3"].unique())
df_prixdata = df_data.merge(df_prix, how='inner', on="ISO3")
Imp_prixdat2023 = (df_prixdata.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données du DataFrame couvrent",round(100*Imp_prixdat2023/
↪Imp_tot2023, 2),"% des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_prix), name="ISO3")
```

```
display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données du DataFrame couvrent 99.87 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :

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

2.7 - Données de gouvernance

[92]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_gouv.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_gouv.shape[1]))
```

```
Le tableau comporte 32100 observation(s) ou article(s)
Le tableau comporte 11 colonne(s)
```

[93]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_gouv.shape[1])
#La nature des données dans chacune des colonnes
display(df_gouv.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_gouv):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_gouv[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_gouv.shape[0] -␣
 ↪((df_gouv[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_gouv.shape[0] - ((df_gouv[c]).isna()).sum() == 0:
        df_gouv.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 11

codeindyr       object
code            object
countryname     object
year             int64
indicator       object
estimate        object
stddev          object
nsource         object
pctrank         object
pctranklower    object
pctrankupper    object
dtype: object
```

```
Colonne codeindyr - Nombre de valeurs NaN : 0
Colonne codeindyr - Nombre de valeurs non-vides : 32100

Colonne code - Nombre de valeurs NaN : 0
Colonne code - Nombre de valeurs non-vides : 32100

Colonne countryname - Nombre de valeurs NaN : 0
Colonne countryname - Nombre de valeurs non-vides : 32100

Colonne year - Nombre de valeurs NaN : 0
Colonne year - Nombre de valeurs non-vides : 32100

Colonne indicator - Nombre de valeurs NaN : 0
Colonne indicator - Nombre de valeurs non-vides : 32100

Colonne estimate - Nombre de valeurs NaN : 0
Colonne estimate - Nombre de valeurs non-vides : 32100

Colonne stddev - Nombre de valeurs NaN : 0
Colonne stddev - Nombre de valeurs non-vides : 32100

Colonne nsource - Nombre de valeurs NaN : 0
Colonne nsource - Nombre de valeurs non-vides : 32100

Colonne pctrank - Nombre de valeurs NaN : 0
Colonne pctrank - Nombre de valeurs non-vides : 32100

Colonne pctranklower - Nombre de valeurs NaN : 0
Colonne pctranklower - Nombre de valeurs non-vides : 32100

Colonne pctrankupper - Nombre de valeurs NaN : 0
Colonne pctrankupper - Nombre de valeurs non-vides : 32100
```

Ça fait beaucoup de colonnes qui contiennent des valeurs non-chiffrées.

[94]:
```
display(df_gouv)
```

```
       codeindyr code            countryname  year indicator  estimate  \
0      AFGcc1996  AFG            Afghanistan  1996        cc -1.291705
1      ALBcc1996  ALB                Albania  1996        cc -0.893903
2      DZAcc1996  DZA                Algeria  1996        cc -0.566741
3      ASMcc1996  ASM         American Samoa  1996        cc        ..
4      ADOcc1996  ADO                Andorra  1996        cc  1.318143
…          …  …                      …   …       …       …
32095  VIRva2023  VIR  Virgin Islands (U.S.)  2023        va        ..
32096  WBGva2023  WBG     West Bank and Gaza  2023        va -1.118067
32097  YEMva2023  YEM            Yemen, Rep.  2023        va -1.550217
32098  ZMBva2023  ZMB                 Zambia  2023        va -0.047946
```

```
32099  ZWEva2023  ZWE              Zimbabwe  2023        va -1.092633

         stddev nsource    pctrank pctranklower pctrankupper
0      0.340507       2   4.301075            0    27.419355
1      0.315914       3  19.354839     2.688172    43.010754
2      0.262077       4  33.333332    16.666666    52.688171
3            ..      ..         ..           ..           ..
4      0.480889       1  87.096771    72.043015    96.774193
…         …     …         …           …           …
32095        ..      ..         ..           ..           ..
32096  0.149837       6  18.137255    11.764706    24.509804
32097  0.131432       8   6.372549      2.45098    11.764706
32098  0.118482      12  45.098038    39.215687    52.450981
32099  0.118235      13  19.117647    12.745098    24.509804

[32100 rows x 11 columns]
```

Il faut forcer le type float et enlever les lignes où apparaissent des NaN.

[95]:
```
display(df_gouv.groupby("nsource").count())
```

```
         codeindyr  code  countryname  year  indicator  estimate  stddev  \
nsource
1             2145  2145         2145  2145       2145      2145    2145
2             1688  1688         1688  1688       1688      1688    1688
3             1839  1839         1839  1839       1839      1839    1839
4             2165  2165         2165  2165       2165      2165    2165
5             2004  2004         2004  2004       2004      2004    2004
6             2880  2880         2880  2880       2880      2880    2880
7             3057  3057         3057  3057       3057      3057    3057
8             3148  3148         3148  3148       3148      3148    3148
9             2921  2921         2921  2921       2921      2921    2921
10            2254  2254         2254  2254       2254      2254    2254
11            1884  1884         1884  1884       1884      1884    1884
12            1532  1532         1532  1532       1532      1532    1532
13            1200  1200         1200  1200       1200      1200    1200
14             859   859          859   859        859       859     859
15             682   682          682   682        682       682     682
16             373   373          373   373        373       373     373
17             189   189          189   189        189       189     189
18             107   107          107   107        107       107     107
19              44    44           44    44         44        44      44
20               3     3            3     3          3         3       3
..            1126  1126         1126  1126       1126      1126    1126

         pctrank  pctranklower  pctrankupper
nsource
1           2145          2145          2145
2           1688          1688          1688
```

```
3           1839          1839          1839
4           2165          2165          2165
5           2004          2004          2004
6           2880          2880          2880
7           3057          3057          3057
8           3148          3148          3148
9           2921          2921          2921
10          2254          2254          2254
11          1884          1884          1884
12          1532          1532          1532
13          1200          1200          1200
14           859           859           859
15           682           682           682
16           373           373           373
17           189           189           189
18           107           107           107
19            44            44            44
20             3             3             3
..          1126          1126          1126
```

[96]:
```
# Les seules colonnes qui vont nous intéresser sont code, countryname, year,␣
 ↪indicator et estimate
# On supprime les autres
df_gouv.
 ↪drop(["codeindyr","stddev","nsource","pctrank","pctranklower","pctrankupper"],␣
 ↪axis=1, inplace=True)
display(df_gouv)
```

```
      code            countryname  year indicator  estimate
0      AFG            Afghanistan  1996        cc -1.291705
1      ALB                Albania  1996        cc -0.893903
2      DZA                Algeria  1996        cc -0.566741
3      ASM         American Samoa  1996        cc        ..
4      ADO                Andorra  1996        cc  1.318143
…    …                        …    …       …       …
32095  VIR  Virgin Islands (U.S.)  2023        va        ..
32096  WBG     West Bank and Gaza  2023        va -1.118067
32097  YEM            Yemen, Rep.  2023        va -1.550217
32098  ZMB                 Zambia  2023        va -0.047946
32099  ZWE               Zimbabwe  2023        va -1.092633

[32100 rows x 5 columns]
```

[97]:
```
# On force le changement de type
df_gouv['estimate'] = pd.to_numeric(df_gouv['estimate'], errors='coerce')
df_gouv = df_gouv.astype({'estimate': float})
df_gouv.dropna(inplace=True)
display(df_gouv.dtypes)
```

```
display(df_gouv)
```

```
code            object
countryname     object
year             int64
indicator       object
estimate       float64
dtype: object
```

	code	countryname	year	indicator	estimate
0	AFG	Afghanistan	1996	cc	-1.291705
1	ALB	Albania	1996	cc	-0.893903
2	DZA	Algeria	1996	cc	-0.566741
4	ADO	Andorra	1996	cc	1.318143
5	AGO	Angola	1996	cc	-1.167702
…	…	…	…	…	…
32094	VNM	Viet Nam	2023	va	-1.241854
32096	WBG	West Bank and Gaza	2023	va	-1.118067
32097	YEM	Yemen, Rep.	2023	va	-1.550217
32098	ZMB	Zambia	2023	va	-0.047946
32099	ZWE	Zimbabwe	2023	va	-1.092633

[30974 rows x 5 columns]

Les différents indicateurs agrégés de la Banque Mondiale sont les suivants : - cc : contrôle de la corruption - ge : efficacité gouvernementale - pv : stabilité politique et absence de violence/terrorisme - rl : respect de la loi, criminalité - rq : qualité de la réglementation, notamment dans le développement du secteur privé et en matière de commerce - va : capacité des citoyens à choisir librement leur gouvernement Les indicateurs qui nous intéressent ici sont cc, pv et rq

[98]:
```
# On ne conserve que les années et les indicateurs pertinents
df_gouv = df_gouv.loc[(df_gouv["year"]>2011)].copy()
df_gouv = df_gouv.loc[(df_gouv["indicator"]=="cc") |␣
 ↪(df_gouv["indicator"]=="pv") | (df_gouv["indicator"]=="rq")].copy()
display(df_gouv)
```

	code	countryname	year	indicator	estimate
16692	AFG	Afghanistan	2012	cc	-1.430373
16693	ALB	Albania	2012	cc	-0.778729
16694	DZA	Algeria	2012	cc	-0.521545
16695	ASM	American Samoa	2012	cc	0.299411
16696	ADO	Andorra	2012	cc	1.260531
…	…	…	…	…	…
31881	VIR	Virgin Islands (U.S.)	2023	rq	1.350546
31882	WBG	West Bank and Gaza	2023	rq	-0.279209
31883	YEM	Yemen, Rep.	2023	rq	-1.843051
31884	ZMB	Zambia	2023	rq	-0.498100
31885	ZWE	Zimbabwe	2023	rq	-1.338359

```
[7617 rows x 5 columns]
```

[99]:
```
df_gouv = df_gouv.pivot(index=["code","year"], columns="indicator",␣
  ↪values="estimate").reset_index()
display(df_gouv)
```

```
indicator code  year        cc        pv        rq
0           ABW  2012  1.095236  1.263882  1.402131
1           ABW  2013  1.119601  1.312226  1.420880
2           ABW  2014  1.014001  1.164568  1.245361
3           ABW  2015  1.248301  1.210240  1.358142
4           ABW  2016  1.232693  1.260759  1.335745
…         …   …       …         …         …
2545        ZWE  2019 -1.290334 -0.943303 -1.486515
2546        ZWE  2020 -1.308788 -1.052743 -1.434415
2547        ZWE  2021 -1.277147 -0.954443 -1.386109
2548        ZWE  2022 -1.259969 -0.894974 -1.425967
2549        ZWE  2023 -1.261216 -0.934447 -1.338359

[2550 rows x 5 columns]
```

[100]:
```
display(df_gouv.groupby("code")["year"].count().sort_values())
```

```
code
ANT     2
NIU     8
COK     8
ABW    12
MWI    12
       ..
GRD    12
GRL    12
GTM    12
GUM    12
ZWE    12
Name: year, Length: 214, dtype: int64
```

Les zones pour lesquelles on n'a pas les données pour toutes les années sont les Antilles Néerlandaises (qui ont été administrativement réorganisées), les Îles Cook et Niué.

[101]:
```
df_data = pd.merge(left=df_data, right=df_gouv, how='inner',␣
  ↪left_on=["ISO3","Année"], right_on=["code","year"])
df_data.drop(["code","year"], axis=1, inplace=True)
display(df_data)
```

```
     ISO3         Zone  Année      Totale    Urbaine           PIB  exp  \
0     AFG  Afghanistan   2012  30560034.0  7416295.0  1.979441e+10  0.0
1     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10  0.0
2     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10  0.0
3     AFG  Afghanistan   2015  33831764.0  8367571.0  1.869957e+10  0.0
```

```
4     AFG  Afghanistan   2016  34700612.0  8670939.0  1.822435e+10  0.0
…   …          …     …         …          …           …   …
2247  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10  0.0
2248  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10  0.0
2249  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10  0.0
2250  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10  0.0
2251  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10  0.0

           imp       prod        cc        pv        rq
0     21750.00   24800.00 -1.430373 -2.418561 -1.192580
1     48389.00   26400.00 -1.445961 -2.519349 -1.193090
2     42300.00   24809.32 -1.364934 -2.411068 -1.124134
3     41811.00   24558.74 -1.354713 -2.562625 -1.018826
4     35473.00   24297.68 -1.540228 -2.662156 -1.339695
…         …         …       …         …         …
2247   1353.12  114344.00 -1.290334 -0.943303 -1.486515
2248   1288.10  111546.00 -1.308788 -1.052743 -1.434415
2249   4071.36  113000.00 -1.277147 -0.954443 -1.386109
2250   5124.70  118225.03 -1.259969 -0.894974 -1.425967
2251   7629.48   97773.99 -1.261216 -0.934447 -1.338359

[2252 rows x 12 columns]
```

[102]:
```
# Données avec jointure externe pour visualisation
df_data_viz = pd.merge(left=df_data_viz, right=df_gouv, how='left',␣
 ↪left_on=["ISO3","Année"], right_on=["code","year"])
df_data_viz.drop(["code","year"], axis=1, inplace=True)
```

[103]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_data):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_data[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_data.shape[0] -␣
 ↪((df_data[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_data["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 2252

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2252

Colonne Année - Nombre de valeurs NaN : 0
Colonne Année - Nombre de valeurs non-vides : 2252

Colonne Totale - Nombre de valeurs NaN : 0
```

```
Colonne Totale - Nombre de valeurs non-vides : 2252

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 2252

Colonne PIB - Nombre de valeurs NaN : 0
Colonne PIB - Nombre de valeurs non-vides : 2252

Colonne exp - Nombre de valeurs NaN : 0
Colonne exp - Nombre de valeurs non-vides : 2252

Colonne imp - Nombre de valeurs NaN : 0
Colonne imp - Nombre de valeurs non-vides : 2252

Colonne prod - Nombre de valeurs NaN : 0
Colonne prod - Nombre de valeurs non-vides : 2252

Colonne cc - Nombre de valeurs NaN : 4
Colonne cc - Nombre de valeurs non-vides : 2248

Colonne pv - Nombre de valeurs NaN : 2
Colonne pv - Nombre de valeurs non-vides : 2250

Colonne rq - Nombre de valeurs NaN : 4
Colonne rq - Nombre de valeurs non-vides : 2248

Nombre de zones dans les données : 188
```

Les valeurs manquantes concernent les Îles Cook.

[104]:
```
df_data = (df_data.loc[df_data["ISO3"]!="COK"]).copy()
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_data):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_data[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_data.shape[0] -␣
 ↪((df_data[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_data["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
Colonne ISO3 - Nombre de valeurs non-vides : 2244

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 2244

Colonne Année - Nombre de valeurs NaN : 0
```

```
Colonne Année - Nombre de valeurs non-vides : 2244

Colonne Totale - Nombre de valeurs NaN : 0
Colonne Totale - Nombre de valeurs non-vides : 2244

Colonne Urbaine - Nombre de valeurs NaN : 0
Colonne Urbaine - Nombre de valeurs non-vides : 2244

Colonne PIB - Nombre de valeurs NaN : 0
Colonne PIB - Nombre de valeurs non-vides : 2244

Colonne exp - Nombre de valeurs NaN : 0
Colonne exp - Nombre de valeurs non-vides : 2244

Colonne imp - Nombre de valeurs NaN : 0
Colonne imp - Nombre de valeurs non-vides : 2244

Colonne prod - Nombre de valeurs NaN : 0
Colonne prod - Nombre de valeurs non-vides : 2244

Colonne cc - Nombre de valeurs NaN : 0
Colonne cc - Nombre de valeurs non-vides : 2244

Colonne pv - Nombre de valeurs NaN : 0
Colonne pv - Nombre de valeurs non-vides : 2244

Colonne rq - Nombre de valeurs NaN : 0
Colonne rq - Nombre de valeurs non-vides : 2244

Nombre de zones dans les données : 187
```

[105]:
```
set_pays_data = set(df_data["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_data), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones,on="ISO3",how='inner'))
Pop_dat2023 = (df_data.groupby(by="Année")["Totale"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
 ↪2), "% de la population mondiale en 2023")
```

	index	ISO3	Zone	Totale
0	2448	PSE	Palestine	5409202.0
1	2392	PRI	Porto Rico	3242023.0
2	781	DJI	Djibouti	1152944.0
3	2490	REU	Réunion	874883.0
4	893	ESH	Sahara occidental	579729.0
5	1874	MDV	Maldives	525994.0
6	1133	GLP	Guadeloupe	376517.0
7	2056	MTQ	Martinique	346002.0
8	2112	MYT	Mayotte	316015.0

```
9    1245  GUF                    Guyane française   303402.0
10   2140  NCL                 Nouvelle-Calédonie   289870.0
11   2462  PYF                Polynésie française   281118.0
12    711  CUW                            Curaçao   185427.0
13    516  CHA             Îles Anglo-Normandes     169724.0
14   1259  GUM                               Guam   166506.0
15   1035  FSM       Micronésie (États fédérés de)  112630.0
16     13  ABW                              Aruba   107939.0
17   3185  VIR          Îles Vierges américaines    85701.0
18   1371  IMN                        Île de Man    84165.0
19     83  AND                           Andorre    80856.0
20    725  CYM                     Îles Caïmanes    73038.0
21    390  BMU                          Bermudes    64698.0
22   1217  GRL                         Groenland    55922.0
23   1021  FRO                        Îles Féroé    54714.0
24    140  ASM                 Samoa américaines    47521.0
25   2849  TCA           Îles Turques-et-Caïques    46198.0
26   2000  MNP            Îles Mariannes du Nord    45143.0
27   2807  SXM  Sint Maarten (partie néerlandaise)  42749.0
28   1707  LIE                      Liechtenstein   39598.0
29   3171  VGB         Îles Vierges britanniques    38985.0
30   1832  MCO                            Monaco    38956.0
31   1902  MHL                     Îles Marshall    38827.0
32   1105  GIB                         Gibraltar    38471.0
33   2656  SMR                       Saint-Marin    33733.0
34    251  BES                Pays-Bas caribéens    29898.0
35   1804  MAF    Saint-Martin (partie française)   27515.0
36   2350  PLW                            Palaos    17727.0
37     55  AIA                          Anguilla    14410.0
38    628  COK                          Îles Cook    14222.0
39   3227  WLF             Îles Wallis-et-Futuna    11370.0
40    348  BLM                  Saint-Barthélemy    11085.0
41   2684  SPM           Saint-Pierre-et-Miquelon     5681.0
42   2600  SHN   Îles britanniques Atlantique Sud     5289.0
43   2042  MSR                        Montserrat     4420.0
44    993  FLK          Îles Falkland (Malvinas)     3477.0
45   2919  TKL                           Tokélaou     2397.0
46   2196  NIU                              Nioué     1817.0
47   3129  VAT                        Saint-Siège      799.0

Les données en l'état couvrent 99.81 % de la population mondiale en 2023
```

[106]:
```
Imp_dat2023 = (df_data.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données en l'état couvrent",round(100*Imp_dat2023/Imp_tot2023, 2),"%␣
 ↪des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")
```

```
set_imp_ref = set(df_data_viz["ISO3"].unique())
set_imp_data = set(df_data["ISO3"].unique())
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_data), name="ISO3")

display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données en l'état couvrent 99.63 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :
```

	ISO3	Zone	imp
0	PYF	Polynésie française	15818.64
1	MDV	Maldives	14119.76
2	PSE	Palestine	9200.00
3	NCL	Nouvelle-Calédonie	8540.08
4	DJI	Djibouti	3389.86
5	COK	Îles Cook	1225.47
6	FRO	Îles Féroé	459.71
7	NIU	Nioué	75.50

2.8 - Données de distance

[107]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(df_dist.
 ↪shape[0]))
print("Le tableau comporte {} colonne(s)".format(df_dist.shape[1]))

#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_dist.shape[1])
#La nature des données dans chacune des colonnes
display(df_dist.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_dist):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_dist[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_dist.shape[0] -␣
 ↪((df_dist[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_dist.shape[0] - ((df_dist[c]).isna()).sum() == 0:
        df_dist.drop(c,axis=1,inplace=True)
```

```
Le tableau comporte 50178 observation(s) ou article(s)
Le tableau comporte 14 colonne(s)
Nombre de colonnes : 14

iso_o             object
```

```
iso_d             object
contig             int64
comlang_off        int64
comlang_ethno      int64
colony             int64
comcol             int64
curcol             int64
col45              int64
smctry             int64
dist             float64
distcap          float64
distw             object
distwces          object
dtype: object


Colonne iso_o - Nombre de valeurs NaN : 0
Colonne iso_o - Nombre de valeurs non-vides : 50178

Colonne iso_d - Nombre de valeurs NaN : 0
Colonne iso_d - Nombre de valeurs non-vides : 50178

Colonne contig - Nombre de valeurs NaN : 0
Colonne contig - Nombre de valeurs non-vides : 50178

Colonne comlang_off - Nombre de valeurs NaN : 0
Colonne comlang_off - Nombre de valeurs non-vides : 50178

Colonne comlang_ethno - Nombre de valeurs NaN : 0
Colonne comlang_ethno - Nombre de valeurs non-vides : 50178

Colonne colony - Nombre de valeurs NaN : 0
Colonne colony - Nombre de valeurs non-vides : 50178

Colonne comcol - Nombre de valeurs NaN : 0
Colonne comcol - Nombre de valeurs non-vides : 50178

Colonne curcol - Nombre de valeurs NaN : 0
Colonne curcol - Nombre de valeurs non-vides : 50178

Colonne col45 - Nombre de valeurs NaN : 0
Colonne col45 - Nombre de valeurs non-vides : 50178

Colonne smctry - Nombre de valeurs NaN : 0
Colonne smctry - Nombre de valeurs non-vides : 50178

Colonne dist - Nombre de valeurs NaN : 0
Colonne dist - Nombre de valeurs non-vides : 50178
```

```
Colonne distcap - Nombre de valeurs NaN : 0
Colonne distcap - Nombre de valeurs non-vides : 50178

Colonne distw - Nombre de valeurs NaN : 2
Colonne distw - Nombre de valeurs non-vides : 50176

Colonne distwces - Nombre de valeurs NaN : 2
Colonne distwces - Nombre de valeurs non-vides : 50176
```

[108]:
```
display(df_dist)
```

```
      iso_o iso_d  contig  comlang_off  comlang_ethno  colony  comcol  curcol  \
0       ABW   ABW       0            0              0       0       0       0
1       ABW   AFG       0            0              0       0       0       0
2       ABW   AGO       0            0              0       0       0       0
3       ABW   AIA       0            0              1       0       0       0
4       ABW   ALB       0            0              0       0       0       0
…     …   …     …          …            …      …       …       …
50173   ZWE   COD       0            0              0       0       0       0
50174   ZWE   ZMB       1            1              1       0       1       0
50175   ZWE   ZWE       0            0              0       0       0       0
50176   FRA   MNE       0            0              0       0       0       0
50177   FRA   SSD       0            0              0       0       0       0

       col45  smctry          dist       distcap     distw  distwces
0          0       0      5.225315      5.225315  25.09354  23.04723
1          0       0  13257.810000  13257.810000  13168.22  13166.37
2          0       0   9516.913000   9516.913000  9587.316  9584.193
3          0       0    983.268200    983.268200  976.8974  976.8916
4          0       0   9091.742000   9091.742000  9091.576  9091.466
…      …      …            …             …       …       …
50173      0       0   2283.061000   2283.061000  1930.976  1719.147
50174      0       0    396.804100    396.804100  583.7954  525.0734
50175      0       0    235.119300    235.119300  199.8205  67.83789
50176      0       0   1491.320000   1491.320000       NaN       NaN
50177      0       0   5626.700000   5626.700000       NaN       NaN

[50178 rows x 14 columns]
```

[109]:
```
# On ne conserve que les lignes pour lesquelles le pays d'origine (iso_o) est␣
 ↪la France FRA
# et les colonnes iso_d et dist
df_dist = (df_dist.loc[df_dist["iso_o"]=="FRA"][["iso_d","dist"]]).copy()
df_dist["dist"] = df_dist["dist"].round(1)
display(df_dist)
```

```
      iso_d    dist
15008   ABW  7685.9
```

```
15009   AFG  5590.4
15010   AGO  6510.3
15011   AIA  6710.6
15012   ALB  1603.5
…     …      …
15229   COD  6049.7
15230   ZMB  7604.8
15231   ZWE  7949.7
50176   MNE  1491.3
50177   SSD  5626.7

[226 rows x 2 columns]
```

[110]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes :", df_dist.shape[1])
#La nature des données dans chacune des colonnes
display(df_dist.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_dist):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_dist[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_dist.shape[0] -␣
 ↪((df_dist[c]).isna()).sum())
    # Si la colonne est entièrement vide, on la supprime
    if df_dist.shape[0] - ((df_dist[c]).isna()).sum() == 0:
        df_dist.drop(c,axis=1,inplace=True)
```

```
Nombre de colonnes : 2
```

```
iso_d     object
dist     float64
dtype: object
```

```
Colonne iso_d - Nombre de valeurs NaN : 0
Colonne iso_d - Nombre de valeurs non-vides : 226

Colonne dist - Nombre de valeurs NaN : 0
Colonne dist - Nombre de valeurs non-vides : 226
```

[111]:
```
set_imp_dist = set(df_dist["iso_d"].unique())
df_distdata = df_data.merge(df_dist, how='inner', left_on="ISO3",␣
 ↪right_on="iso_d")
Imp_distdat2023 = (df_distdata.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données du DataFrame couvrent",round(100*Imp_distdat2023/
 ↪Imp_tot2023, 2),"% des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")
```

```
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_dist), name="ISO3")

display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données du DataFrame couvrent 99.63 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :

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

Partie 3 - Feature Engineering

[112]:
```
#Récapitulons les données rassemblées
display(df_data.columns)
```

```
Index(['ISO3', 'Zone', 'Année', 'Totale', 'Urbaine', 'PIB', 'exp', 'imp',
       'prod', 'cc', 'pv', 'rq'],
      dtype='object')
```

Il se pose également la question des variables qui devraient être rapportées à la population du pays : cela semble naturel de le faire pour calculer les taux de population urbaine/rurale ou femmes/hommes. Mais par exemple, s'il paraît là-aussi logique de rapporter le PIB au nombre d'habitants (revenu moyen par habitant), faut-il pour autant ne pas conserver la donnée de PIB (taille de l'économie du pays) ? Le risque étant que notre analyse par ACP et clustering soit biaisée par ces données faisant intervenir la "taille" du pays.

Cela fait déjà beaucoup de données, d'autant plus qu'on a des données sur plusieurs années. Il faut donc définir avec soin le traitement réalisé sur ces variables temporelles : moyenne, valeur la plus récente, taux de croissance moyen, sur quelle période de référence ?

[113]:
```
display(df_data)
```

	ISO3	Zone	Année	Totale	Urbaine	PIB	exp	\
0	AFG	Afghanistan	2012	30560034.0	7416295.0	1.979441e+10	0.0	
1	AFG	Afghanistan	2013	31622704.0	7733832.0	1.990441e+10	0.0	
2	AFG	Afghanistan	2014	32792523.0	8054222.0	1.950046e+10	0.0	
3	AFG	Afghanistan	2015	33831764.0	8367571.0	1.869957e+10	0.0	
4	AFG	Afghanistan	2016	34700612.0	8670939.0	1.822435e+10	0.0	
…	…	…	…	…	…	…	…	
2247	ZWE	Zimbabwe	2019	15271368.0	5571525.0	2.259452e+10	0.0	
2248	ZWE	Zimbabwe	2020	15526888.0	5700460.0	2.166475e+10	0.0	
2249	ZWE	Zimbabwe	2021	15797210.0	5834113.0	2.411815e+10	0.0	
2250	ZWE	Zimbabwe	2022	16069056.0	5972826.0	2.641859e+10	0.0	
2251	ZWE	Zimbabwe	2023	16340822.0	6117511.0	3.036820e+10	0.0	

	imp	prod	cc	pv	rq

```
0     21750.00   24800.00 -1.430373 -2.418561 -1.192580
1     48389.00   26400.00 -1.445961 -2.519349 -1.193090
2     42300.00   24809.32 -1.364934 -2.411068 -1.124134
3     41811.00   24558.74 -1.354713 -2.562625 -1.018826
4     35473.00   24297.68 -1.540228 -2.662156 -1.339695
…        …         …        …        …        …
2247   1353.12  114344.00 -1.290334 -0.943303 -1.486515
2248   1288.10  111546.00 -1.308788 -1.052743 -1.434415
2249   4071.36  113000.00 -1.277147 -0.954443 -1.386109
2250   5124.70  118225.03 -1.259969 -0.894974 -1.425967
2251   7629.48   97773.99 -1.261216 -0.934447 -1.338359

[2244 rows x 12 columns]
```

[114]:
```
df_data = df_data.rename(columns={"Totale": "pop", "Urbaine": "urb", "PIB":␣
  ↪"pib"})
df_data_ini = df_data.copy()
display(df_data)
```

```
     ISO3         Zone  Année         pop        urb           pib  exp  \
0     AFG  Afghanistan   2012  30560034.0  7416295.0  1.979441e+10  0.0
1     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10  0.0
2     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10  0.0
3     AFG  Afghanistan   2015  33831764.0  8367571.0  1.869957e+10  0.0
4     AFG  Afghanistan   2016  34700612.0  8670939.0  1.822435e+10  0.0
…   …          …    …          …          …             …   …
2247  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10  0.0
2248  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10  0.0
2249  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10  0.0
2250  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10  0.0
2251  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10  0.0

           imp       prod        cc        pv        rq
0     21750.00   24800.00 -1.430373 -2.418561 -1.192580
1     48389.00   26400.00 -1.445961 -2.519349 -1.193090
2     42300.00   24809.32 -1.364934 -2.411068 -1.124134
3     41811.00   24558.74 -1.354713 -2.562625 -1.018826
4     35473.00   24297.68 -1.540228 -2.662156 -1.339695
…        …         …        …        …        …
2247   1353.12  114344.00 -1.290334 -0.943303 -1.486515
2248   1288.10  111546.00 -1.308788 -1.052743 -1.434415
2249   4071.36  113000.00 -1.277147 -0.954443 -1.386109
2250   5124.70  118225.03 -1.259969 -0.894974 -1.425967
2251   7629.48   97773.99 -1.261216 -0.934447 -1.338359

[2244 rows x 12 columns]
```

[115]:
```
# Calcul de variables supplémentaires (feature engineering)

# Disponibilités en viande de poulet (en tonnes)
df_data["dispo"] = df_data["imp"] + df_data["prod"] - df_data["exp"]

# Propotion de la population vivant en milieu urbain, plus proche des grands␣
 ↪circuits de distribution
df_data["%urb"] = round(100*df_data["urb"] / df_data["pop"],2)

# PIB/habitant indicateur du niveau de vie d'un pays (en dollars US)
df_data["pib/hab"] = round(df_data["pib"]/df_data["pop"],2)

# Disponibilité de viande de poulet par habitant, en kg/an, info sur la␣
 ↪consommation
df_data["dispo/hab"] = round(1e3*df_data["dispo"]/df_data["pop"],2)
```

Données dont il faut calculer le taux de croissance : - population urbaine - impoortations - PIB - PIB/habitant - disponibilités par habitant

Récapitulons nos données :

[116]:
```
display(list(df_data.columns))
```

```
['ISO3',
 'Zone',
 'Année',
 'pop',
 'urb',
 'pib',
 'exp',
 'imp',
 'prod',
 'cc',
 'pv',
 'rq',
 'dispo',
 '%urb',
 'pib/hab',
 'dispo/hab']
```

Avec la variable *Population* on essaie d'obtenir des informations sur la démographie des pays. Avec la variable *%Urbaine* on essaie d'obtenir des informations sur le mode de vie des habitants d'un pays. Avec la variable *PIB*, on essaie de connaître la taille de l'économie d'un pays. Avec la variable *PIB/hab*, on essaie de connaître le niveau de vie moyen de la population d'un pays. Avec les variables *Exp*, *Imp*, *Prod* on essaie d'obtenir des informations commerciales d'un pays (est-il acheteur ou vendeur ?) vis-s-vis de la viande de poulet. Avec la variables *Dispo* on essaie de connaître la taille des marchés économiques (~ consommation totale) de la viande de poulet. Avec les variables *Dispo/hab*, on essaie de connaître les habitudes alimentaires (~consommation/habitant) de la population d'un pays.

Il se pose alors une question fondamentale pour notre étude : quelle est la donnée clé que l'on essaie de notre étude ? Du point de vue de l'entreprise, il s'agit d'établir un positionnement stratégique non pas pour la consommation de poulet des différents pays, mais pour les **importations** de poulet. C'est donc la variable *imp* qui va nous intéresser. Comme nous disposons de données temporelles, on peut aller plus loin dans l'analyse en identifiant les pays dont les importations croissent au cours du temps. L'analyse de la variable *imp* peut nous fournir un positionnement pertinent à court et moyen termes, l'analyse de la variable "croissance de *imp*" (qui serait une variable "dérivée" de *imp*) nous fournirait un positionnement pertinent à moyen et long termes. On s'intéressera aux valeurs de cette variable (variation de *imp*) lors de l'évaluation des pays des différents clusters, afin d'éviter de multiplier les données. En revanche, avoir une idée du taux de croissance de la population ou du PIB est tout à fait pertinent à ce stade.

Il faut définir la croissance d'une variable. En effet, se contenter de diviser la valeur la plus récente par la plus ancienne (puis retrancher 1 pour obtenir un taux de croissance en %) pourrait susciter des attentes démesurées et n'est pas très robuste car cela passerait sous silence toutes les données recueillies concernant les années intermédiaires.

Une idée pourrait être de s'appuyer sur un modèle de progression arithmétique de la grandeur $ V_n $ où n est un indice représentant l'année à laquelle est considérée la valeur de la variable V :

$$V_n = a * (n - \overline{n}) + b$$

Les paramètres de ce modèle sont b, une sorte de valeur moyenne de V, et la croissance annuelle a. $\overline{n}$ représente la valeur moyenne des années considérées : par exemple pour les années 2013 à 2023, $\overline{n} = 2018$ En cherchant les paramètres optimaux a et b qui minimisent l'erreur entre le modèle et les observations (au sens des moindres carrés), on obtient :

$$b = \overline{V}$$

(b est exactement la moyenne arithmétique des V_n) et

$$a = \frac{\sum ((n - \overline{n}) * V_n)}{\sum (n - \overline{n})^2}$$

On peut étudier les pays sur la période allant de 2013 à 2023.

[117]:
```
df_cibles = df_data.copy()
set_pays_cibles = set(df_cibles["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_cibles), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones, on="ISO3", how='inner'))
#print("Liste des 20 pays les plus peuplés en 2023 exclus de notre analyse :")
#display(df_pays_pop2023.merge(rmvd_zones, left_on="Code zone (ISO3)",␣
 ↪right_on="ISO3", how='inner').head(20))
Pop_dat2023 = (df_cibles.groupby(by="Année")["pop"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
 ↪2), "% de la population mondiale en 2023")
```

	index	ISO3	Zone	Totale
0	2448	PSE	Palestine	5409202.0
1	2392	PRI	Porto Rico	3242023.0

2	781	DJI	Djibouti	1152944.0
3	2490	REU	Réunion	874883.0
4	893	ESH	Sahara occidental	579729.0
5	1874	MDV	Maldives	525994.0
6	1133	GLP	Guadeloupe	376517.0
7	2056	MTQ	Martinique	346002.0
8	2112	MYT	Mayotte	316015.0
9	1245	GUF	Guyane française	303402.0
10	2140	NCL	Nouvelle-Calédonie	289870.0
11	2462	PYF	Polynésie française	281118.0
12	711	CUW	Curaçao	185427.0
13	516	CHA	Îles Anglo-Normandes	169724.0
14	1259	GUM	Guam	166506.0
15	1035	FSM	Micronésie (États fédérés de)	112630.0
16	13	ABW	Aruba	107939.0
17	3185	VIR	Îles Vierges américaines	85701.0
18	1371	IMN	Île de Man	84165.0
19	83	AND	Andorre	80856.0
20	725	CYM	Îles Caïmanes	73038.0
21	390	BMU	Bermudes	64698.0
22	1217	GRL	Groenland	55922.0
23	1021	FRO	Îles Féroé	54714.0
24	140	ASM	Samoa américaines	47521.0
25	2849	TCA	Îles Turques-et-Caïques	46198.0
26	2000	MNP	Îles Mariannes du Nord	45143.0
27	2807	SXM	Sint Maarten (partie néerlandaise)	42749.0
28	1707	LIE	Liechtenstein	39598.0
29	3171	VGB	Îles Vierges britanniques	38985.0
30	1832	MCO	Monaco	38956.0
31	1902	MHL	Îles Marshall	38827.0
32	1105	GIB	Gibraltar	38471.0
33	2656	SMR	Saint-Marin	33733.0
34	251	BES	Pays-Bas caribéens	29898.0
35	1804	MAF	Saint-Martin (partie française)	27515.0
36	2350	PLW	Palaos	17727.0
37	55	AIA	Anguilla	14410.0
38	628	COK	Îles Cook	14222.0
39	3227	WLF	Îles Wallis-et-Futuna	11370.0
40	348	BLM	Saint-Barthélemy	11085.0
41	2684	SPM	Saint-Pierre-et-Miquelon	5681.0
42	2600	SHN	Îles britanniques Atlantique Sud	5289.0
43	2042	MSR	Montserrat	4420.0
44	993	FLK	Îles Falkland (Malvinas)	3477.0
45	2919	TKL	Tokélaou	2397.0
46	2196	NIU	Nioué	1817.0
47	3129	VAT	Saint-Siège	799.0

Les données en l'état couvrent 99.81 % de la population mondiale en 2023

[118]:
```
Imp_dat2023 = (df_data.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données en l'état couvrent",round(100*Imp_dat2023/Imp_tot2023, 2),"%␣
 ↪des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")
set_imp_ref = set(df_data_viz["ISO3"].unique())
set_imp_data = set(df_data["ISO3"].unique())
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_data), name="ISO3")

display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données en l'état couvrent 99.63 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :
```

	ISO3	Zone	imp
0	PYF	Polynésie française	15818.64
1	MDV	Maldives	14119.76
2	PSE	Palestine	9200.00
3	NCL	Nouvelle-Calédonie	8540.08
4	DJI	Djibouti	3389.86
5	COK	Îles Cook	1225.47
6	FRO	Îles Féroé	459.71
7	NIU	Nioué	75.50

Création des variables croissance annuelle pour les pays cibles :

[119]:
```
def VarTC_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 la moyenne arithmétique des valeurs
        mean_dfb = (dfb[["ISO3", col]]).groupby("ISO3")[col].mean().
↪reset_index()

        # calcul de a du modèle en divisant par le dénominateur
        wsum_dfb["ra_"+col] = round(wsum_dfb["ra"] / swgt2, 2)
        merged_dfb = wsum_dfb.merge(mean_dfb, on="ISO3", how='inner')

        merged_dfb["tc_"+col] = round(100*wsum_dfb["ra_"+col] /␣
↪merged_dfb[col],2)

        resultats["tc_"+col] = merged_dfb[["ISO3", "tc_"+col]].
↪set_index("ISO3")["tc_"+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))
```

[120]:
```
display(df_cibles)
```

```
     ISO3         Zone  Année         pop        urb           pib  exp  \
0     AFG  Afghanistan   2012  30560034.0  7416295.0  1.979441e+10  0.0
1     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10  0.0
2     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10  0.0
3     AFG  Afghanistan   2015  33831764.0  8367571.0  1.869957e+10  0.0
4     AFG  Afghanistan   2016  34700612.0  8670939.0  1.822435e+10  0.0
…   …          …    …         …          …           …   …
2247  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10  0.0
2248  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10  0.0
2249  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10  0.0
2250  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10  0.0
2251  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10  0.0

           imp       prod        cc        pv        rq      dispo   %urb  \
0     21750.00   24800.00 -1.430373 -2.418561 -1.192580   46550.00  24.27
1     48389.00   26400.00 -1.445961 -2.519349 -1.193090   74789.00  24.46
2     42300.00   24809.32 -1.364934 -2.411068 -1.124134   67109.32  24.56
3     41811.00   24558.74 -1.354713 -2.562625 -1.018826   66369.74  24.73
4     35473.00   24297.68 -1.540228 -2.662156 -1.339695   59770.68  24.99
…        …         …       …         …         …         …    …
2247   1353.12  114344.00 -1.290334 -0.943303 -1.486515  115697.12  36.48
2248   1288.10  111546.00 -1.308788 -1.052743 -1.434415  112834.10  36.71
```

```
2249   4071.36  113000.00 -1.277147 -0.954443 -1.386109  117071.36  36.93
2250   5124.70  118225.03 -1.259969 -0.894974 -1.425967  123349.73  37.17
2251   7629.48   97773.99 -1.261216 -0.934447 -1.338359  105403.47  37.44

      pib/hab  dispo/hab
0      647.72       1.52
1      629.43       2.37
2      594.66       2.05
3      552.72       1.96
4      525.19       1.72
…        …          …
2247  1479.53       7.58
2248  1395.31       7.27
2249  1526.73       7.41
2250  1644.07       7.68
2251  1858.43       6.45

[2244 rows x 16 columns]
```

[121]:
```
df_cibles_tc = VarTC_multicol(df=df_cibles, debut=2013, fin=2023,\
                              colonnes=["pop","urb","pib","pib/
 ↪hab","dispo","dispo/hab"]).reset_index()
display(df_cibles_tc)
```

```
    ISO3  tc_pop  tc_urb  tc_pib  tc_pib/hab  tc_dispo  tc_dispo/hab
0    AFG    5.43    6.82   -4.57       -9.89    -10.25        -15.73
1    AGO    6.76    8.55  -11.12      -18.20     -2.35         -9.35
2    ALB   -0.63    3.28   12.20       12.91     11.34         12.14
3    ARE    5.18    3.21    4.58       -1.01      5.93          1.29
4    ARG    1.36    2.13   -1.91       -3.35      6.00          4.69
..   …      …       …      …           …          …             …
182  WSM    1.84   -0.78    2.86        1.02      4.03          2.25
183  YEM    5.87    8.00  -27.96      -33.07      4.59         -1.56
184  ZAF    2.86    3.89    1.16       -1.75      2.47         -0.33
185  ZMB    5.92    8.41    0.81       -5.50      9.67          3.83
186  ZWE    3.08    4.40    7.95        4.80      9.03          6.02

[187 rows x 7 columns]
```

Liste des données à moyenner (moyenne arithmétique) sur la même période (2013-2023) : - population - pib - imp - cc - pv - rq - dep - %urb - pib/hab - dispo/hab

[122]:
```
# On se restreint aux années 2013 à 2023
df_cibles = df_cibles.loc[df_cibles["Année"]>2012].copy()
display(df_cibles)
```

```
     ISO3         Zone  Année         pop        urb           pib    exp  \
1     AFG  Afghanistan   2013  31622704.0  7733832.0  1.990441e+10   0.00
2     AFG  Afghanistan   2014  32792523.0  8054222.0  1.950046e+10   0.00
```

```
3     AFG  Afghanistan   2015  33831764.0  8367571.0  1.869957e+10   0.00
4     AFG  Afghanistan   2016  34700612.0  8670939.0  1.822435e+10   0.00
5     AFG  Afghanistan   2017  35688935.0  8971472.0  1.903430e+10  36.99
…   …          …    …         …         …          …    …
2247  ZWE     Zimbabwe   2019  15271368.0  5571525.0  2.259452e+10   0.00
2248  ZWE     Zimbabwe   2020  15526888.0  5700460.0  2.166475e+10   0.00
2249  ZWE     Zimbabwe   2021  15797210.0  5834113.0  2.411815e+10   0.00
2250  ZWE     Zimbabwe   2022  16069056.0  5972826.0  2.641859e+10   0.00
2251  ZWE     Zimbabwe   2023  16340822.0  6117511.0  3.036820e+10   0.00

           imp       prod        cc        pv        rq      dispo   %urb  \
1     48389.00   26400.00 -1.445961 -2.519349 -1.193090   74789.00  24.46
2     42300.00   24809.32 -1.364934 -2.411068 -1.124134   67109.32  24.56
3     41811.00   24558.74 -1.354713 -2.562625 -1.018826   66369.74  24.73
4     35473.00   24297.68 -1.540228 -2.662156 -1.339695   59770.68  24.99
5     28937.00   27637.84 -1.530075 -2.794976 -1.363010   56537.85  25.14
…          …          …         …         …         …          …    …
2247   1353.12  114344.00 -1.290334 -0.943303 -1.486515  115697.12  36.48
2248   1288.10  111546.00 -1.308788 -1.052743 -1.434415  112834.10  36.71
2249   4071.36  113000.00 -1.277147 -0.954443 -1.386109  117071.36  36.93
2250   5124.70  118225.03 -1.259969 -0.894974 -1.425967  123349.73  37.17
2251   7629.48   97773.99 -1.261216 -0.934447 -1.338359  105403.47  37.44

      pib/hab  dispo/hab
1      629.43       2.37
2      594.66       2.05
3      552.72       1.96
4      525.19       1.72
5      533.34       1.58
…         …          …
2247  1479.53       7.58
2248  1395.31       7.27
2249  1526.73       7.41
2250  1644.07       7.68
2251  1858.43       6.45

[2057 rows x 16 columns]
```

[123]:
```
# On récupère les moyennes arithmétiques au cours du temps des données par pays␣
 ↪pour l'ACP
L_avg_var = list(df_cibles.columns)[3:]
df_cibles_avg = round(df_cibles.groupby(["ISO3","Zone"])[list(L_avg_var)].
 ↪mean().reset_index(),2)
display(df_cibles_avg)
```

```
    ISO3                Zone          pop          urb           pib  \
0    AFG         Afghanistan  36758062.91   9303056.55  1.812068e+10
1    AGO              Angola  31358489.00  20281821.45  1.031760e+11
```

```
2    ALB              Albanie   2876923.27   1766194.09  1.525616e+10
3    ARE  Émirats arabes unis   9259546.36   8268234.82  4.155386e+11
4    ARG            Argentine  44395603.91  41034761.09  5.591068e+11
..   …                    …          …           …            …
182  WSM                Samoa    207637.27     36180.09  8.705741e+08
183  YEM                Yémen  34180357.18  10642219.91  1.888517e+10
184  ZAF       Afrique du Sud  58886231.00  38055900.45  3.789851e+11
185  ZMB               Zambie  18006950.64   7725972.09  2.428533e+10
186  ZWE             Zimbabwe  15097522.91   5474848.45  2.272259e+10

            exp        imp        prod    cc    pv    rq       dispo   %urb  \
0         52.98   30146.73    27156.60 -1.38 -2.60 -1.23    57250.35  25.26
1         72.78  265843.72    44621.98 -1.11 -0.45 -0.85   310392.91  64.48
2          4.91   24500.61    14010.37 -0.53  0.24  0.22    38506.07  61.43
3      47777.40  536015.67    51563.00  1.13  0.68  0.99   539801.27  89.57
4     209064.85    4318.53  2137418.79 -0.34  0.00 -0.58  1932672.46  92.41
..          …          …          …   …   …   …   …          …     …
182       34.87   15739.14      395.99  0.52  1.12 -0.20    16100.26  17.45
183       15.84  100922.62   183741.98 -1.61 -2.68 -1.53   284648.76  31.04
184    51249.95  384106.89  1783399.45 -0.12 -0.34  0.05  2116256.39  64.58
185     3800.64   17087.18    48216.33 -0.53  0.11 -0.55    61502.86  42.75
186        0.00    5285.46    93027.04 -1.31 -0.80 -1.57    98312.50  36.22

      pib/hab  dispo/hab
0      499.65       1.59
1     3393.95      10.06
2     5315.70      13.42
3    44935.94      58.23
4    12614.01      43.46
..        …          …
182   4191.05      77.46
183    580.50       8.35
184   6444.51      35.95
185   1359.56       3.40
186   1499.38       6.48

[187 rows x 15 columns]
```

[124]:
```
# Reste à faire la jointure des données pour les pays cibles
df_work = pd.merge(left=df_cibles_avg, right=df_cibles_tc, on="ISO3",␣
 ↪how='inner')
df_work = df_work.merge(df_prix, on="ISO3", how='inner')
df_work = df_work.merge(df_dist, left_on="ISO3", right_on="iso_d", how='inner')
df_work = df_work.rename(columns={"Prix": "prix"})
df_work.drop(["urb","iso_d"], axis=1, inplace=True)
display(df_work)
```

```
    ISO3                Zone          pop           pib        exp  \
```

```
0    AFG          Afghanistan  36758062.91  1.812068e+10      52.98
1    AGO               Angola  31358489.00  1.031760e+11      72.78
2    ALB              Albanie   2876923.27  1.525616e+10       4.91
3    ARE  Émirats arabes unis   9259546.36  4.155386e+11   47777.40
4    ARG            Argentine  44395603.91  5.591068e+11  209064.85
..   …                    …           …           …        …
182  WSM                Samoa    207637.27  8.705741e+08      34.87
183  YEM                Yémen  34180357.18  1.888517e+10      15.84
184  ZAF       Afrique du Sud  58886231.00  3.789851e+11   51249.95
185  ZMB               Zambie  18006950.64  2.428533e+10    3800.64
186  ZWE             Zimbabwe  15097522.91  2.272259e+10       0.00

           imp        prod    cc    pv    rq  …   pib/hab  dispo/hab  \
0     30146.73    27156.60 -1.38 -2.60 -1.23  …    499.65       1.59
1    265843.72    44621.98 -1.11 -0.45 -0.85  …   3393.95      10.06
2     24500.61    14010.37 -0.53  0.24  0.22  …   5315.70      13.42
3    536015.67    51563.00  1.13  0.68  0.99  …  44935.94      58.23
4      4318.53  2137418.79 -0.34  0.00 -0.58  …  12614.01      43.46
..         …          …   …   …   …  …       …        …
182   15739.14      395.99  0.52  1.12 -0.20  …   4191.05      77.46
183  100922.62   183741.98 -1.61 -2.68 -1.53  …    580.50       8.35
184  384106.89  1783399.45 -0.12 -0.34  0.05  …   6444.51      35.95
185   17087.18    48216.33 -0.53  0.11 -0.55  …   1359.56       3.40
186    5285.46    93027.04 -1.31 -0.80 -1.57  …   1499.38       6.48

     tc_pop  tc_urb  tc_pib  tc_pib/hab  tc_dispo  tc_dispo/hab    prix  \
0      5.43    6.82   -4.57       -9.89    -10.25        -15.73  1270.0
1      6.76    8.55  -11.12      -18.20     -2.35         -9.35  2693.0
2     -0.63    3.28   12.20       12.91     11.34         12.14   880.9
3      5.18    3.21    4.58       -1.01      5.93          1.29  1582.0
4      1.36    2.13   -1.91       -3.35      6.00          4.69  1611.0
..      …       …      …          …         …             …      …
182    1.84   -0.78    2.86        1.02      4.03          2.25   941.5
183    5.87    8.00  -27.96      -33.07      4.59         -1.56  1371.0
184    2.86    3.89    1.16       -1.75      2.47         -0.33   752.4
185    5.92    8.41    0.81       -5.50      9.67          3.83  1416.0
186    3.08    4.40    7.95        4.80      9.03          6.02   657.0

        dist
0     5590.4
1     6510.3
2     1603.5
3     5249.5
4    11072.2
..       …
182  16011.9
183   5317.3
184   9353.6
```

```
185   7604.8
186   7949.7

[187 rows x 22 columns]
```

Pour les variables pop, urb et pib, conaître la variation annuelle nous intéresse peu. On préférerait avoir un taux de croissance.

[125]:
```
set_pays_cibles = set(df_work["ISO3"].unique())
rmvd_zones = pd.Series(list(set_pays_ref - set_pays_cibles), name="ISO3")
display(df_pays_pop2023.merge(rmvd_zones, on="ISO3", how='inner'))
#print("Liste des 20 pays les plus peuplés en 2023 exclus de notre analyse :")
#display(df_pays_pop2023.merge(rmvd_zones, left_on="Code zone (ISO3)",␣
 ↪right_on="ISO3", how='inner').head(20))
Pop_dat2023 = (df_cibles.groupby(by="Année")["pop"].sum()).loc[2023]
print("Les données en l'état couvrent", round(100*Pop_dat2023 / Pop_tot2023,␣
 ↪2), "% de la population mondiale en 2023")
```

	index	ISO3	Zone	Totale
0	2448	PSE	Palestine	5409202.0
1	2392	PRI	Porto Rico	3242023.0
2	781	DJI	Djibouti	1152944.0
3	2490	REU	Réunion	874883.0
4	893	ESH	Sahara occidental	579729.0
5	1874	MDV	Maldives	525994.0
6	1133	GLP	Guadeloupe	376517.0
7	2056	MTQ	Martinique	346002.0
8	2112	MYT	Mayotte	316015.0
9	1245	GUF	Guyane française	303402.0
10	2140	NCL	Nouvelle-Calédonie	289870.0
11	2462	PYF	Polynésie française	281118.0
12	711	CUW	Curaçao	185427.0
13	516	CHA	Îles Anglo-Normandes	169724.0
14	1259	GUM	Guam	166506.0
15	1035	FSM	Micronésie (États fédérés de)	112630.0
16	13	ABW	Aruba	107939.0
17	3185	VIR	Îles Vierges américaines	85701.0
18	1371	IMN	Île de Man	84165.0
19	83	AND	Andorre	80856.0
20	725	CYM	Îles Caïmanes	73038.0
21	390	BMU	Bermudes	64698.0
22	1217	GRL	Groenland	55922.0
23	1021	FRO	Îles Féroé	54714.0
24	140	ASM	Samoa américaines	47521.0
25	2849	TCA	Îles Turques-et-Caïques	46198.0
26	2000	MNP	Îles Mariannes du Nord	45143.0
27	2807	SXM	Sint Maarten (partie néerlandaise)	42749.0
28	1707	LIE	Liechtenstein	39598.0

```
29   3171  VGB           Îles Vierges britanniques    38985.0
30   1832  MCO                             Monaco    38956.0
31   1902  MHL                       Îles Marshall    38827.0
32   1105  GIB                          Gibraltar    38471.0
33   2656  SMR                        Saint-Marin    33733.0
34    251  BES                  Pays-Bas caribéens    29898.0
35   1804  MAF     Saint-Martin (partie française)    27515.0
36   2350  PLW                              Palaos    17727.0
37     55  AIA                            Anguilla    14410.0
38    628  COK                           Îles Cook    14222.0
39   3227  WLF               Îles Wallis-et-Futuna    11370.0
40    348  BLM                    Saint-Barthélemy    11085.0
41   2684  SPM            Saint-Pierre-et-Miquelon     5681.0
42   2600  SHN    Îles britanniques Atlantique Sud     5289.0
43   2042  MSR                           Montserrat     4420.0
44    993  FLK            Îles Falkland (Malvinas)     3477.0
45   2919  TKL                             Tokélaou     2397.0
46   2196  NIU                                Nioué     1817.0
47   3129  VAT                          Saint-Siège      799.0
```

Les données en l'état couvrent 99.81 % de la population mondiale en 2023

[126]:
```
set_imp_ref = set(df_data_viz["ISO3"].unique())
set_imp_data = set(df_work["ISO3"].unique())
df_imp2023 = df_data_viz.loc[df_data_viz["Année"]==2023][["ISO3","Zone","imp"]].
 ↪sort_values("imp", ascending=False)
rmvd_zones = pd.Series(list(set_imp_ref - set_imp_data), name="ISO3")
kept_zones = pd.Series(list(set_imp_data), name="ISO3")

df_check = df_data_viz.merge(kept_zones, on="ISO3", how='inner')

Imp_dat2023 = (df_check.groupby("Année")["imp"].sum()).loc[2023]
print("Importations totales de viande de poulet en 2023 (en t) :",Imp_tot2023)
print("Les données en l'état couvrent",round(100*Imp_dat2023/Imp_tot2023, 2),"%␣
 ↪des importations totales\n")

print("Liste des zones exclues de l'analyse faute de données complètes :")

display(df_imp2023.merge(rmvd_zones,on="ISO3",how='inner').dropna())
```

```
Importations totales de viande de poulet en 2023 (en t) : 14131333.32
Les données en l'état couvrent 99.63 % des importations totales

Liste des zones exclues de l'analyse faute de données complètes :
```

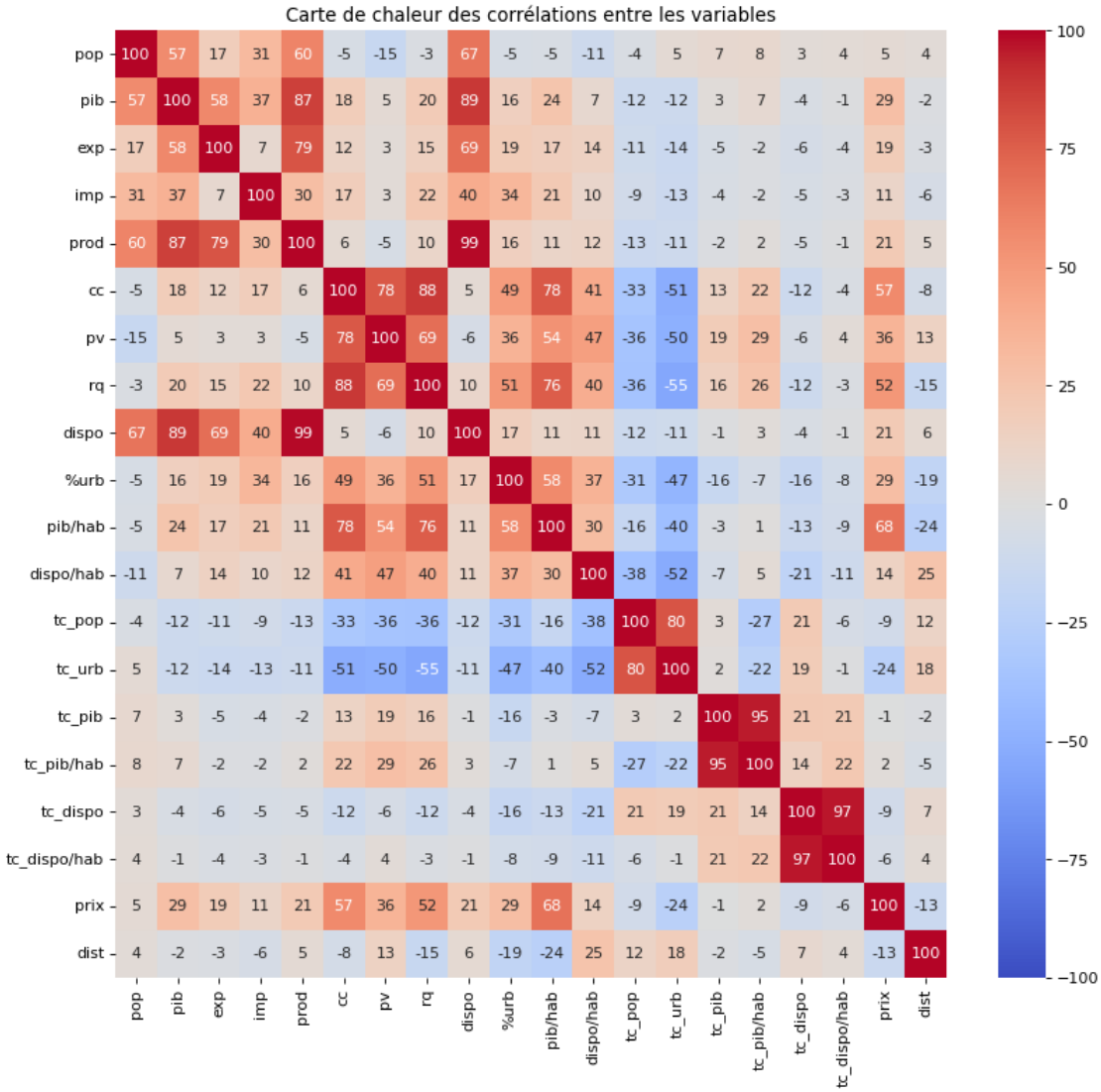
	ISO3	Zone	imp
0	PYF	Polynésie française	15818.64
1	MDV	Maldives	14119.76
2	PSE	Palestine	9200.00

```
3  NCL   Nouvelle-Calédonie   8540.08
4  DJI              Djibouti   3389.86
5  COK            Îles Cook   1225.47
6  FRO           Îles Féroé    459.71
7  NIU                Nioué     75.50
```

Partie 4 - Choix des variables

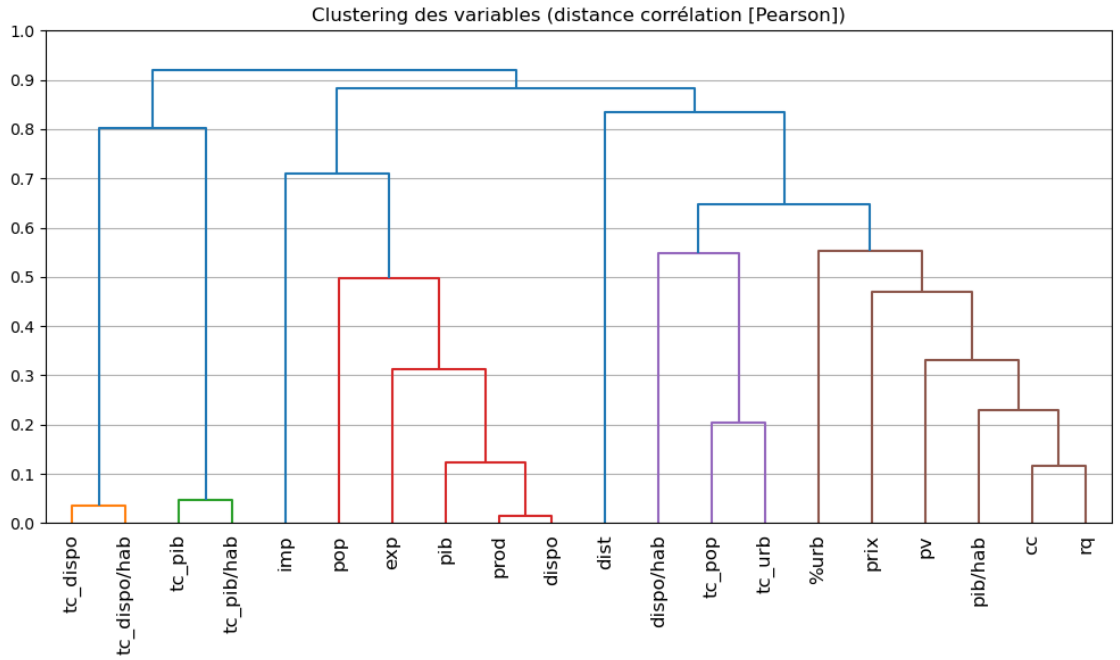
J'ai collecté un assez grand nombre de variables : il y en a 20 par pays. Il serait judicieux de vérifier qu'il n'y a pas de redondances entre ces variables, c'est-à-dire de vérifier que les corrélations entre variables ne sont pas trop grandes.

[127]:
```
fig = plt.figure(figsize=(12,11), dpi=80)

sb.heatmap(100*df_work[list(df_work.columns)[2:]].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")
#plt.savefig("Heatmap.png")
plt.show()
```

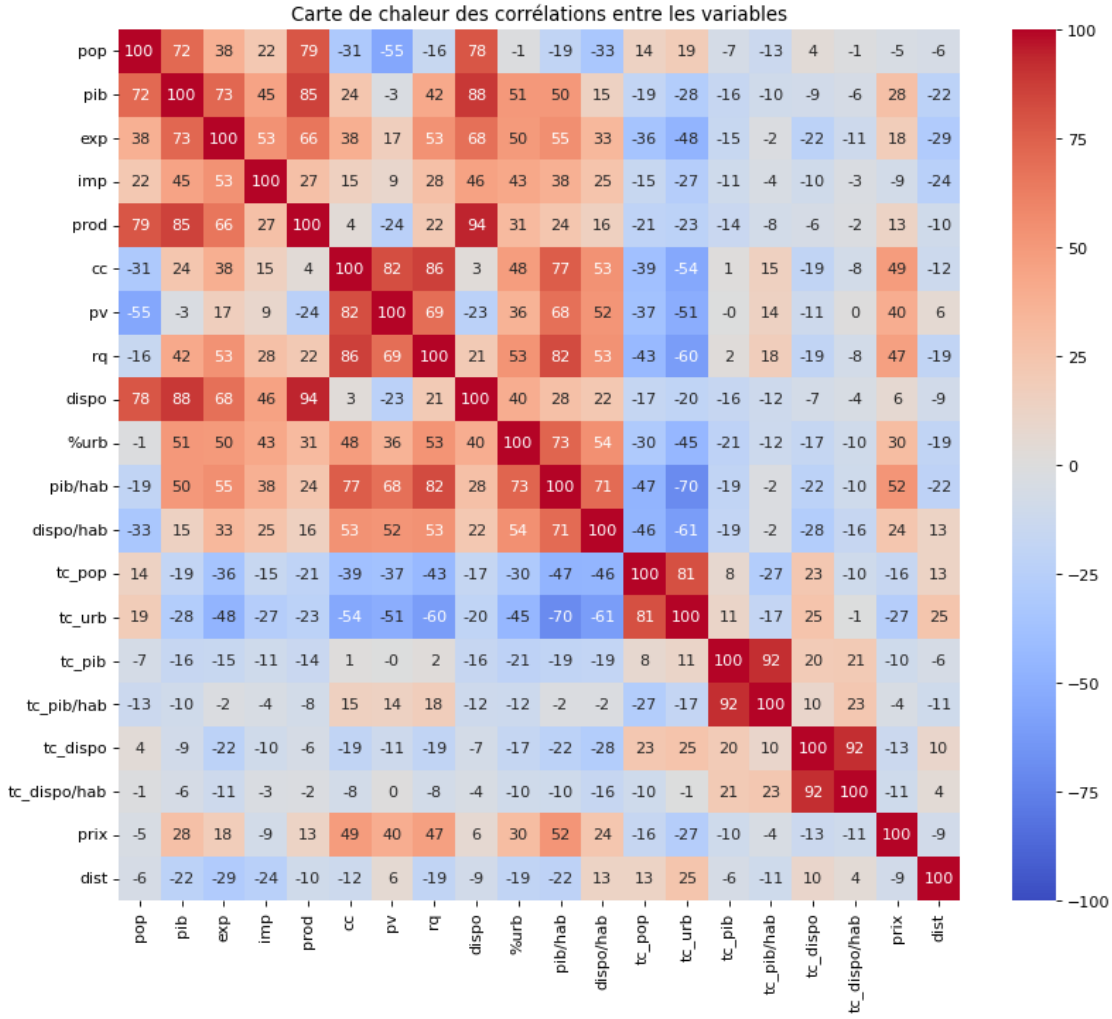


[128]:
```
vcorr = (df_work[list(df_work.columns)[2:]].corr(method='pearson')).abs()

# 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)[2:], 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.show()
```

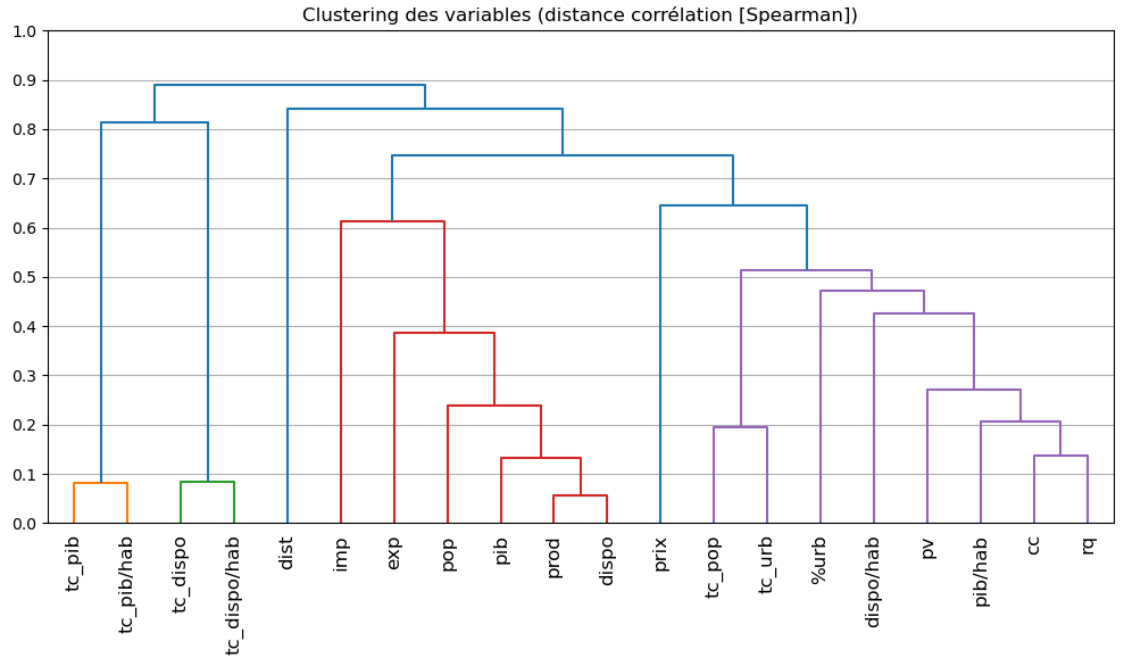


[129]:
```
fig = plt.figure(figsize=(12,10), dpi=80)

sb.heatmap(100*df_work[list(df_work.columns)[2:]].corr(method='spearman'),\
           vmin=-100,vmax=100,cmap='coolwarm', fmt='.0f', annot=True)
#plt.xticks(rotation=45)
plt.title("Carte de chaleur des corrélations entre les variables")
#plt.savefig("Heatmap.png")
plt.show()
```



[130]:
```
vcorr = (df_work[list(df_work.columns)[2:]].corr(method='spearman')).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)[2:], leaf_rotation=90)
plt.yticks(np.linspace(0,1,11))
plt.grid(axis='y')
plt.title("Clustering des variables (distance corrélation [Spearman])")
```

```
plt.tight_layout()
plt.savefig("Dendrogramme_20variables.png")
plt.show()
```



L'avantage de la méthode de Spearman dans le calul des taux de corrélation, c'est qu'elle ne tient compte que des rangs, ce qui permet de faire une première itération avec les outliers. On retient des variables de façon à ce qu'aucune des distances entre 2 d'entre elles soit inférieure à 0.2 : - *tc_pib* (taux de croissance économique, plutôt que *tc_pib/hab*, taux d'évolution du niveau de vie) ; - *tc_dispo/hab* (taux de croissance de la consommation individuelle de poulet plutôt que *tc_dispo* qui concerne la consommation totale) - *dist* (distance à la France) ; - *prix* à l'importation ; - *tc_pop* (taux d'évolution de la population générale) ; - *pv* (indicateur de stabilité politique) ; - *pib/hab* ; - *rq* (indicateur de la qualité réglemntaire) ; - *%urb* (proportion de population urbaine) ; - *dispo/hab* ; - *imp* (importations) ; - *exp* (exportations) ; - *pop* (population) ; - *pib*

Partie 5 - Analyse univariée des variables et traitement des outliers

[131]:
```
df_stats = df_work.describe()
display(df_stats)
df_tmp = df_work.copy()
outzones = set()
```

	pop	pib	exp	imp	prod	\
count	1.870000e+02	1.870000e+02	1.870000e+02	187.000000	1.870000e+02	
mean	4.115779e+07	4.649210e+11	7.541019e+04	67733.826524	6.046585e+05	
std	1.485712e+08	1.923886e+12	3.978649e+05	139838.323091	2.050315e+06	
min	1.058909e+04	4.957169e+07	0.000000e+00	0.000000	4.020000e+00	
25%	2.615376e+06	1.281903e+10	5.160000e+00	2571.430000	1.015940e+04	

```
50%    9.384180e+06  4.261156e+10  2.732000e+02   15739.140000  7.088046e+04
75%    3.003372e+07  2.442004e+11  9.950875e+03   61549.495000  3.130800e+05
max    1.411122e+09  2.110548e+13  3.996504e+06  856599.910000  1.899368e+07

                cc           pv           rq         dispo        %urb  \
count  187.000000  187.000000  187.000000  1.870000e+02  187.000000
mean    -0.087540   -0.109786   -0.069519  5.969822e+05   58.850214
std      0.988897    0.955977    0.995871  1.801934e+06   22.940540
min     -1.760000   -2.750000   -2.370000  4.747300e+02   11.900000
25%     -0.845000   -0.675000   -0.760000  2.782382e+04   40.355000
50%     -0.320000   -0.040000   -0.150000  9.983053e+04   59.820000
75%      0.595000    0.690000    0.605000  3.310951e+05   77.805000
max      2.260000    1.450000    2.180000  1.564719e+07  103.100000

             pib/hab   dispo/hab      tc_pop      tc_urb      tc_pib  \
count     187.000000  187.000000  187.000000  187.000000  187.000000
mean    14648.959519   21.978877    2.408235    3.845668    5.637647
std     20316.983508   18.380063    2.329081    2.907300    7.648627
min       274.730000    0.440000   -2.810000   -1.820000  -27.960000
25%      1929.380000    6.475000    0.615000    1.530000    2.515000
50%      6167.400000   18.810000    2.360000    3.330000    6.500000
75%     17987.650000   31.755000    4.180000    6.160000   10.115000
max    118168.180000   92.210000    7.190000   11.240000   30.940000

       tc_pib/hab    tc_dispo  tc_dispo/hab         prix          dist
count  187.000000  187.000000    187.000000   187.000000    187.000000
mean     3.175615    6.310963      3.949786  1602.104813   6155.697861
std      7.941339    8.768887      8.638932   717.418848   3932.848129
min    -33.070000  -24.600000    -25.760000   473.700000    262.400000
25%      0.315000    2.630000      0.460000  1047.500000   3247.950000
50%      3.900000    5.220000      3.660000  1476.000000   5626.700000
75%      7.695000    9.755000      7.625000  2027.000000   8773.450000
max     28.700000   38.010000     36.580000  4717.000000  19263.900000
```

Certaines variables présentent une grande variabilité. On va essayer de réduire cette variabilité en appliquant des fonctions logarithmes sur certaines d'entre elles.

[132]:
```
sout = set()
for col in list(df_stats.columns):
    iq = (df_stats[col]["75%"] - df_stats[col]["25%"])
    if df_stats[col]["min"] < df_stats[col]["25%"] - 1.5*iq :
        sout = sout | {col}
        print("Colonne", col, "outliers valeurs basses")
    if df_stats[col]["max"] > df_stats[col]["75%"] + 1.5*iq :
        sout = sout | {col}
        print("Colonne", col, "outliers valeurs hautes")
    if col in sout:
        print("")
```

```
lout = list(sout)
```

```
Colonne pop outliers valeurs hautes

Colonne pib outliers valeurs hautes

Colonne exp outliers valeurs hautes

Colonne imp outliers valeurs hautes

Colonne prod outliers valeurs hautes

Colonne pv outliers valeurs basses

Colonne dispo outliers valeurs hautes

Colonne pib/hab outliers valeurs hautes

Colonne dispo/hab outliers valeurs hautes

Colonne tc_pib outliers valeurs basses
Colonne tc_pib outliers valeurs hautes

Colonne tc_pib/hab outliers valeurs basses
Colonne tc_pib/hab outliers valeurs hautes

Colonne tc_dispo outliers valeurs basses
Colonne tc_dispo outliers valeurs hautes

Colonne tc_dispo/hab outliers valeurs basses
Colonne tc_dispo/hab outliers valeurs hautes

Colonne prix outliers valeurs hautes

Colonne dist outliers valeurs hautes
```

[133]:
```
print("Variables avec outliers :")
display(df_stats[lout])
```

```
Variables avec outliers :
```

	tc_pib/hab	exp	tc_pib	pib/hab	pib	\
count	187.000000	1.870000e+02	187.000000	187.000000	1.870000e+02	
mean	3.175615	7.541019e+04	5.637647	14648.959519	4.649210e+11	
std	7.941339	3.978649e+05	7.648627	20316.983508	1.923886e+12	
min	-33.070000	0.000000e+00	-27.960000	274.730000	4.957169e+07	
25%	0.315000	5.160000e+00	2.515000	1929.380000	1.281903e+10	
50%	3.900000	2.732000e+02	6.500000	6167.400000	4.261156e+10	

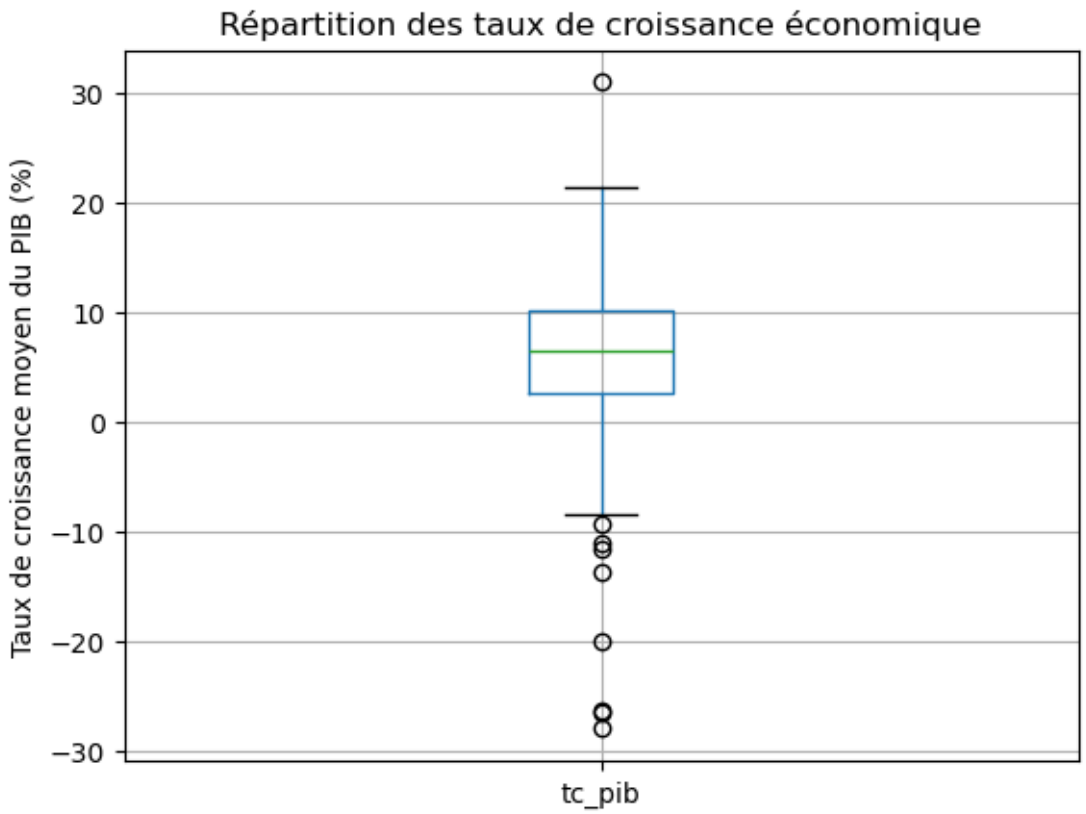
```
75%       7.695000  9.950875e+03   10.115000   17987.650000  2.442004e+11
max      28.700000  3.996504e+06   30.940000  118168.180000  2.110548e+13

                pv          prix    tc_dispo   dispo/hab         dispo  \
count   187.000000    187.000000  187.000000  187.000000  1.870000e+02
mean     -0.109786   1602.104813    6.310963   21.978877  5.969822e+05
std       0.955977    717.418848    8.768887   18.380063  1.801934e+06
min      -2.750000    473.700000  -24.600000    0.440000  4.747300e+02
25%      -0.675000   1047.500000    2.630000    6.475000  2.782382e+04
50%      -0.040000   1476.000000    5.220000   18.810000  9.983053e+04
75%       0.690000   2027.000000    9.755000   31.755000  3.310951e+05
max       1.450000   4717.000000   38.010000   92.210000  1.564719e+07

                 pop          dist             imp  tc_dispo/hab          prod
count   1.870000e+02    187.000000     187.000000    187.000000  1.870000e+02
mean    4.115779e+07   6155.697861   67733.826524      3.949786  6.046585e+05
std     1.485712e+08   3932.848129  139838.323091      8.638932  2.050315e+06
min     1.058909e+04    262.400000       0.000000    -25.760000  4.020000e+00
25%     2.615376e+06   3247.950000    2571.430000      0.460000  1.015940e+04
50%     9.384180e+06   5626.700000   15739.140000      3.660000  7.088046e+04
75%     3.003372e+07   8773.450000   61549.495000      7.625000  3.130800e+05
max     1.411122e+09  19263.900000  856599.910000     36.580000  1.899368e+07
```

On veut utiliser nos données pour faire une ACP. Or, cette technique s'appuie sur le calcul de coefficients de corrélation de Pearson, qui suppose que les données suivent des lois normales. Il n'y a *a priori* aucune raison que nos données suivent des lois normales : en effet, nos variables peuvent ne même pas présenter une symétrie dans leur répartititon.

5.1 - Analyse univariée des variables

Variable tc_pib (taux de croissance du pib)

[134]:
```
df_tmp.boxplot(column=["tc_pib"], showfliers=True)
plt.ylabel("Taux de croissance moyen du PIB (%)")
plt.title("Répartition des taux de croissance économique")
plt.show()
```



[135]:
```
display((df_tmp.loc[df_tmp["tc_pib"]>20][["ISO3","Zone","tc_pib"]]).
  ↪sort_values("tc_pib",ascending=False))
display((df_tmp.loc[df_tmp["tc_pib"]<-9][["ISO3","Zone","tc_pib"]]).
  ↪sort_values("tc_pib",ascending=True))
```

	ISO3	Zone	tc_pib
68	GUY	Guyana	30.94
61	GIN	Guinée	21.35
53	ETH	Éthiopie	20.80

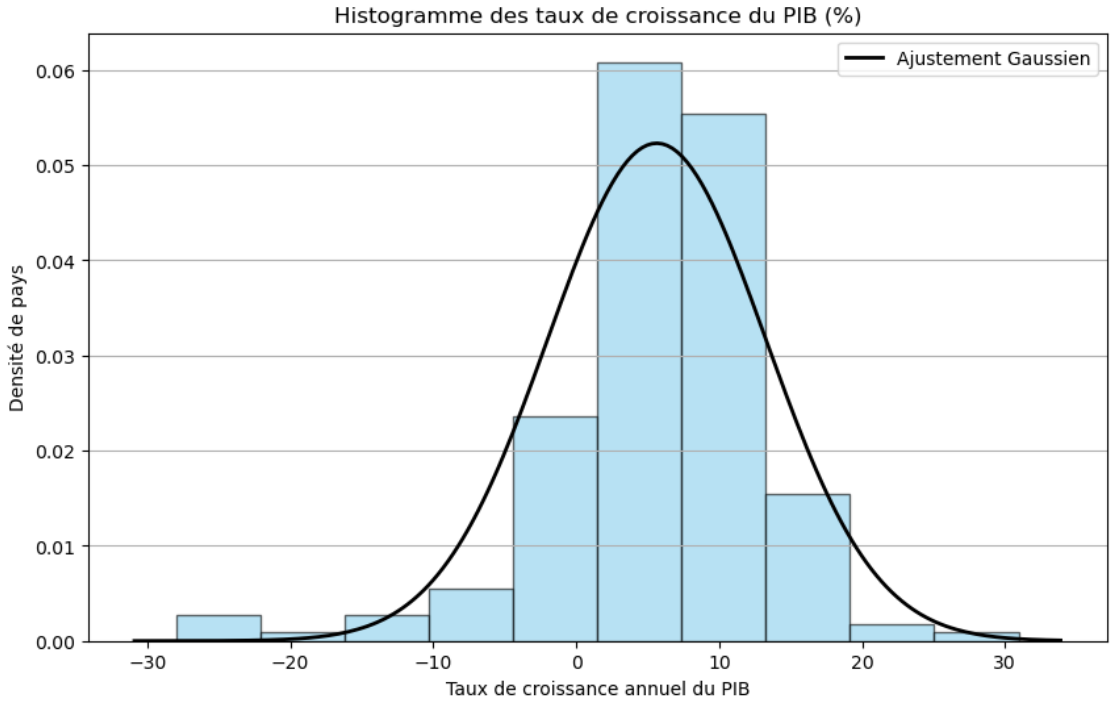
	ISO3	Zone	tc_pib
183	YEM	Yémen	-27.96
179	VEN	Venezuela (République bolivarienne du)	-26.53
151	SSD	Soudan du Sud	-26.38
143	SDN	Soudan	-20.07
94	LBN	Liban	-13.69
64	GNQ	Guinée équatoriale	-11.63
1	AGO	Angola	-11.12
103	MAC	Chine - RAS de Macao	-9.30

Les pays avec un taux de croissance du PIB atypiquement haut (Guyana) ont connu une très forte croissance économique. A contrario, ceux avec un taux de croissance du PIB atypiquement bas ont

connu de graves problèmes économiques (guerres, effondrement des services publics)

[136]:
```
data = df_work["tc_pib"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des taux de croissance du PIB (%)")
plt.xlabel('Taux de croissance annuel du PIB')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 5.64
Écart type : 7.65
Z_min : -4.39
Z_max : 3.31
```

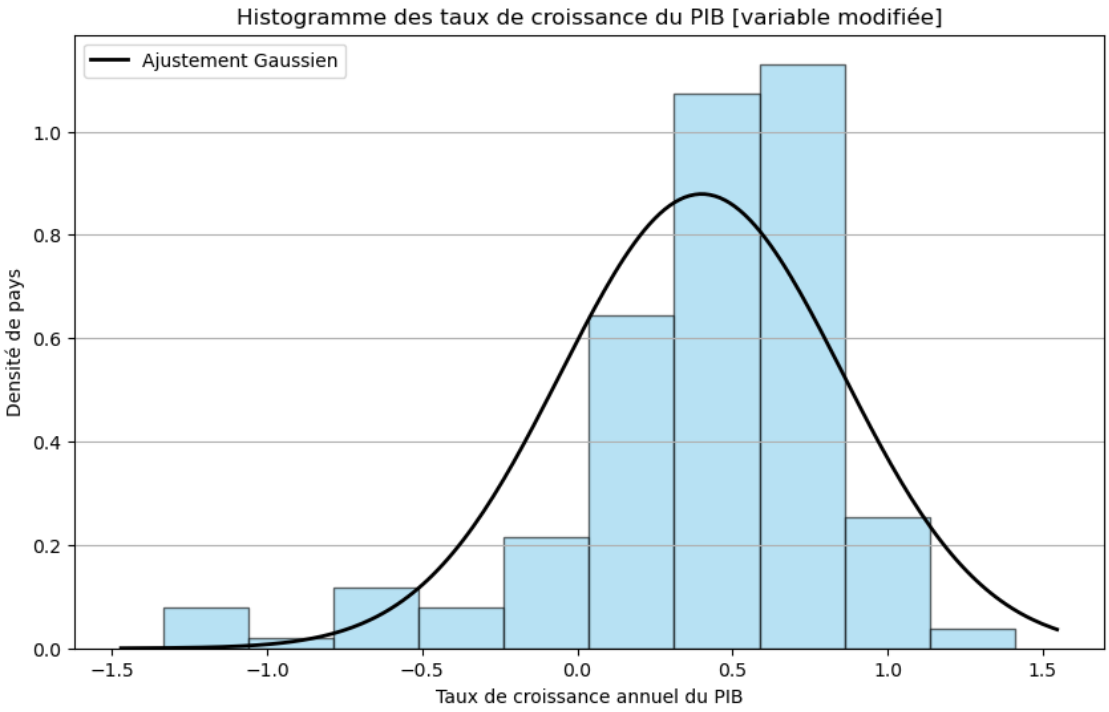



```
Statistique du test de Shapiro-Wilk : 0.892012810601244
Valeur p : 2.1965148598361224e-10
```

[137]:
```
df_tmp["tc_pib"] = df_work["tc_pib"].apply(lambda x: np.sign(x)*(np.
  ↪log(1+abs(x)/10)))
data = df_tmp["tc_pib"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))
```

```python
# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des taux de croissance du PIB [variable modifiée]")
plt.xlabel('Taux de croissance annuel du PIB')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

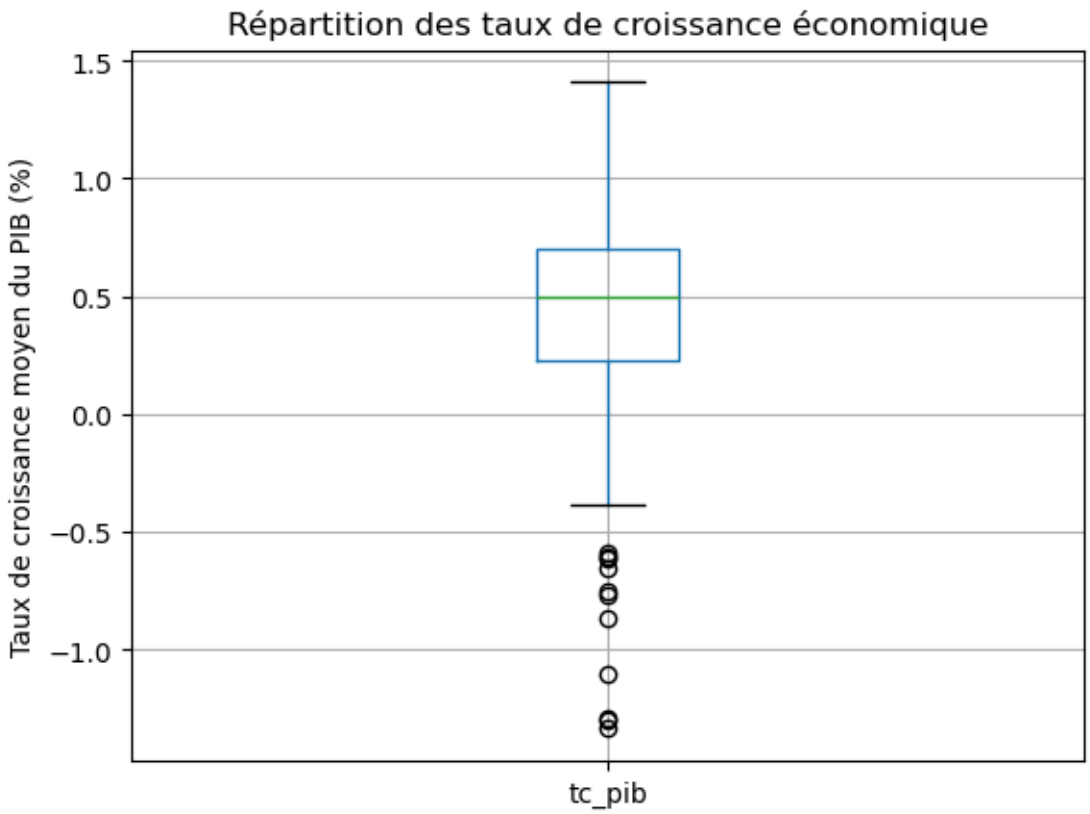
```
Moyenne : 0.4
Écart type : 0.46
Z_min : -3.81
Z_max : 2.22
```




```
Statistique du test de Shapiro-Wilk : 0.8900097702889682
Valeur p : 1.67633747152252e-10
```

On n'a pas amélioré la normalité des données, mais on a réduit le z-score des outliers.

[138]:
```
df_tmp.boxplot(column=["tc_pib"], showfliers=True)
plt.ylabel("Taux de croissance moyen du PIB (%)")
plt.title("Répartition des taux de croissance économique")
plt.show()
```



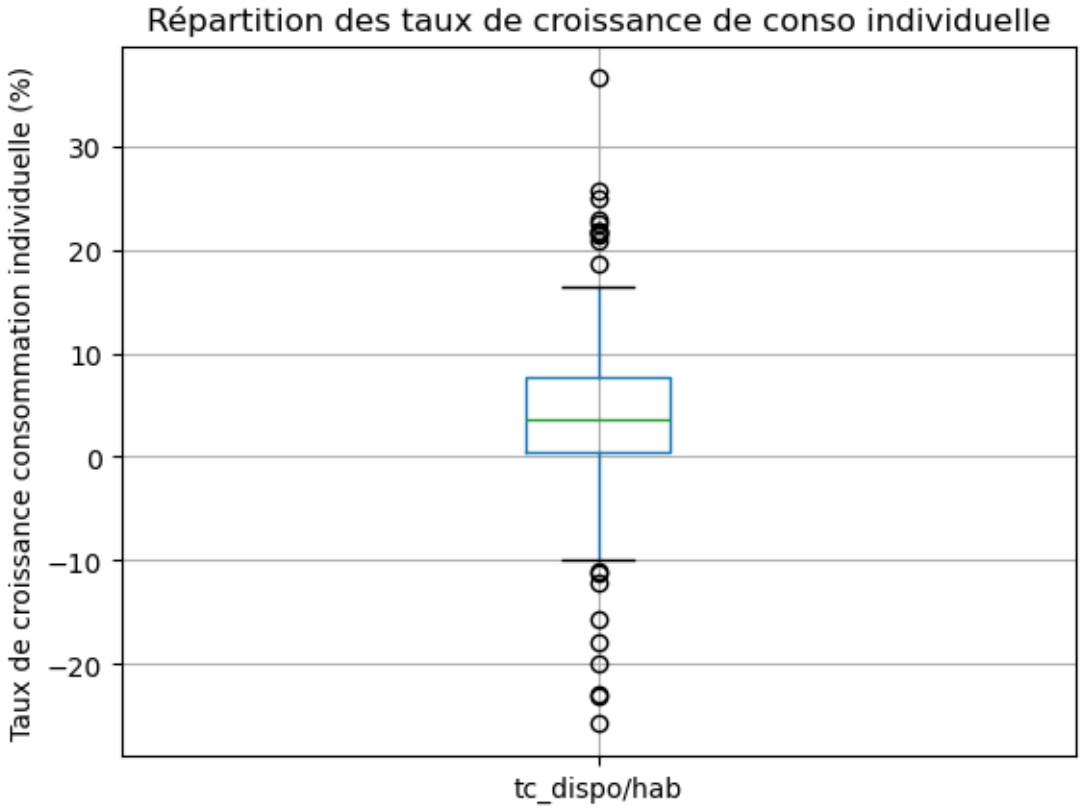
[139]:
```
#display((df_tmp.loc[df_tmp["tc_pib"]>2][["ISO3","Zone","tc_pib"]]).
 ↪sort_values("tc_pib",ascending=False))
display((df_tmp.loc[df_tmp["tc_pib"]<-0.4][["ISO3","Zone","tc_pib"]]).
 ↪sort_values("tc_pib",ascending=True))
```

	ISO3	Zone	tc_pib
183	YEM	Yémen	-1.333948
179	VEN	Venezuela (République bolivarienne du)	-1.295549
151	SSD	Soudan du Sud	-1.291434
143	SDN	Soudan	-1.100943
94	LBN	Liban	-0.862468
64	GNQ	Guinée équatoriale	-0.771496
1	AGO	Angola	-0.747635
103	MAC	Chine - RAS de Macao	-0.657520
96	LBY	Libye	-0.609222
153	SUR	Suriname	-0.603222
77	IRN	Iran (République islamique d')	-0.588897

La régularisation des données qu'on propose pour cette variable améliore la normalité des données mais repousse plus loin les outliers. Opérer une transformation n'est pas crucial.

Variable tc_dispo/hab

[140]:
```
df_tmp.boxplot(column=["tc_dispo/hab"], showfliers=True)
plt.ylabel("Taux de croissance consommation individuelle (%)")
plt.title("Répartition des taux de croissance de conso individuelle")
plt.show()
```



[141]:
```
display((df_tmp.loc[df_tmp["tc_dispo/hab"]>15][["ISO3","Zone","tc_dispo/hab"]]).
  ↪sort_values("tc_dispo/hab",ascending=False))
display((df_tmp.loc[df_tmp["tc_dispo/hab"]<-10][["ISO3","Zone","tc_dispo/
  ↪hab"]]).sort_values("tc_dispo/hab",ascending=True))
```

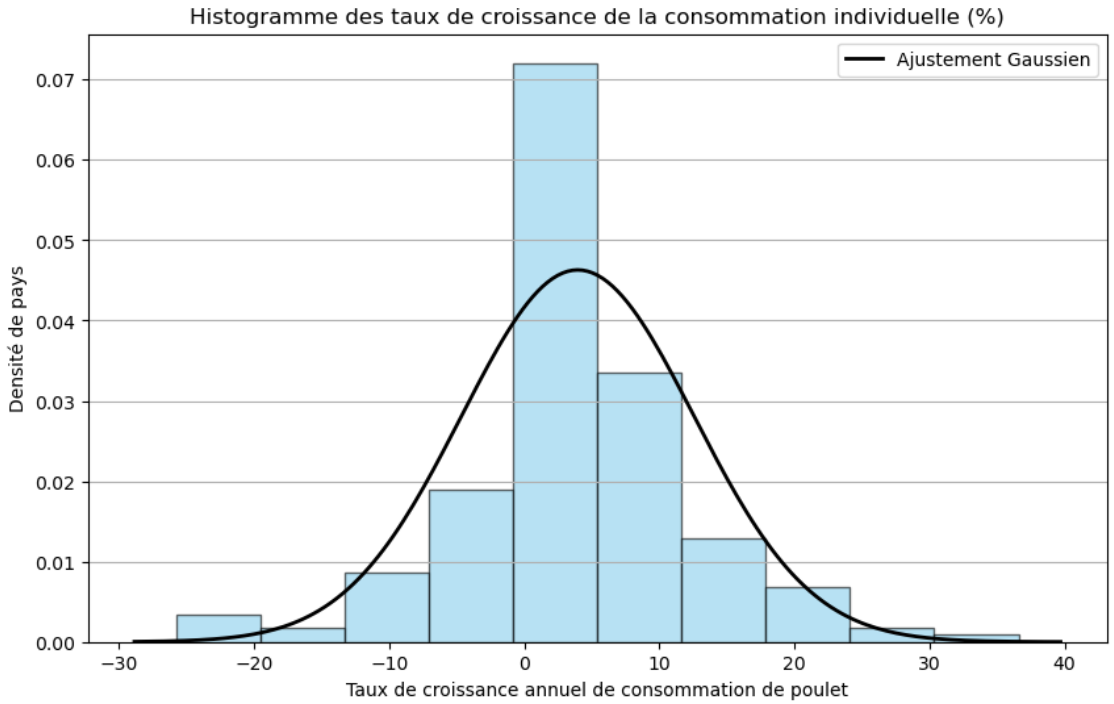
	ISO3	Zone	tc_dispo/hab
125	NPL	Népal	36.58
62	GMB	Gambie	25.67
177	UZB	Ouzbékistan	24.91
151	SSD	Soudan du Sud	22.90
113	MNG	Mongolie	22.61
25	BTN	Bhoutan	21.75
35	COG	Congo	21.75
27	CAF	République centrafricaine	21.59
61	GIN	Guinée	21.40
37	COM	Comores	20.81

```
180  VNM                    Viet Nam         18.59
55   FJI                       Fidji         16.47
86   KEN                       Kenya         16.37
149  SOM                     Somalie         16.21
74   IDN                   Indonésie         15.50

     ISO3                                        Zone  tc_dispo/hab
99   LSO                                     Lesotho        -25.76
111  MMR                                     Myanmar        -23.07
179  VEN      Venezuela (République bolivarienne du)        -22.93
95   LBR                                     Libéria        -19.91
12   BEN                                       Bénin        -17.83
0    AFG                                  Afghanistan        -15.73
64   GNQ                        Guinée équatoriale        -12.15
50   ERI                                    Érythrée        -11.25
135  PRK  République populaire démocratique de Corée        -11.07
```

[142]:
```python
data = df_work["tc_dispo/hab"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
```

```
plt.title("Histogramme des taux de croissance de la consommation individuelle␣
  ↪(%)")
plt.xlabel('Taux de croissance annuel de consommation de poulet')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 3.95
Écart type : 8.64
Z_min : -3.44
Z_max : 3.78
```

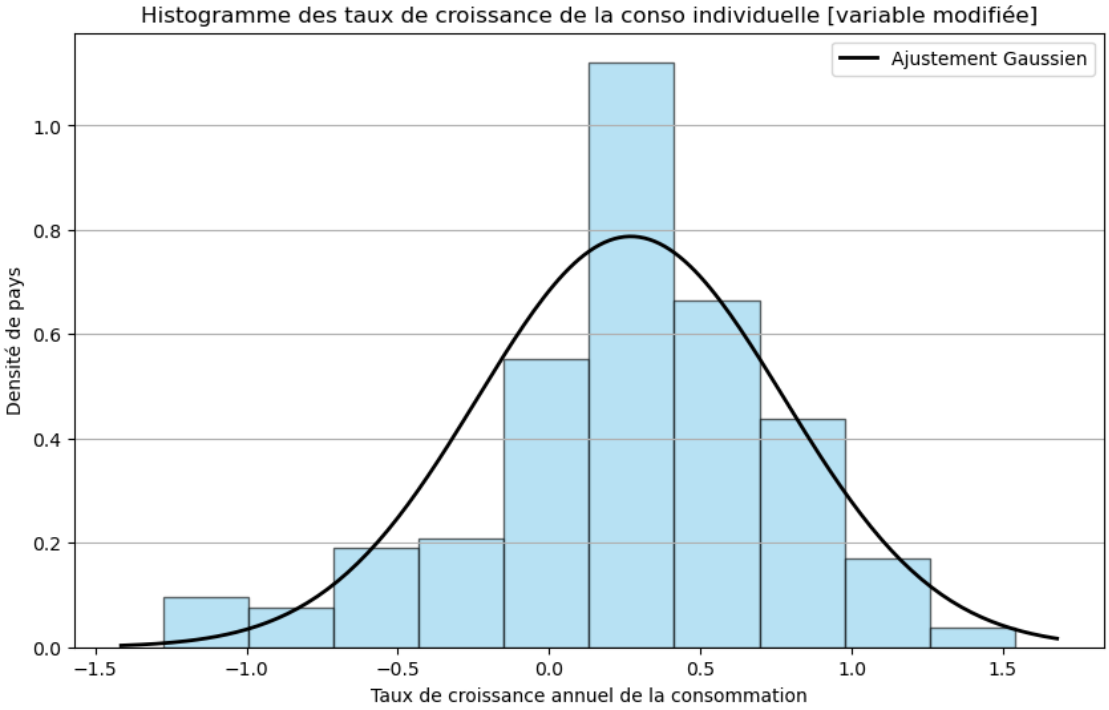



```
Statistique du test de Shapiro-Wilk : 0.9479463028030349
Valeur p : 2.4845849754292485e-06
```

[143]:
```
df_tmp["tc_dispo/hab"] = df_work["tc_dispo/hab"].apply(lambda x: np.sign(x)*(np.
 ↪log(1+abs(x/10))))
data = df_tmp["tc_dispo/hab"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des taux de croissance de la conso individuelle␣
 ↪[variable modifiée]")
plt.xlabel('Taux de croissance annuel de la consommation')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
```

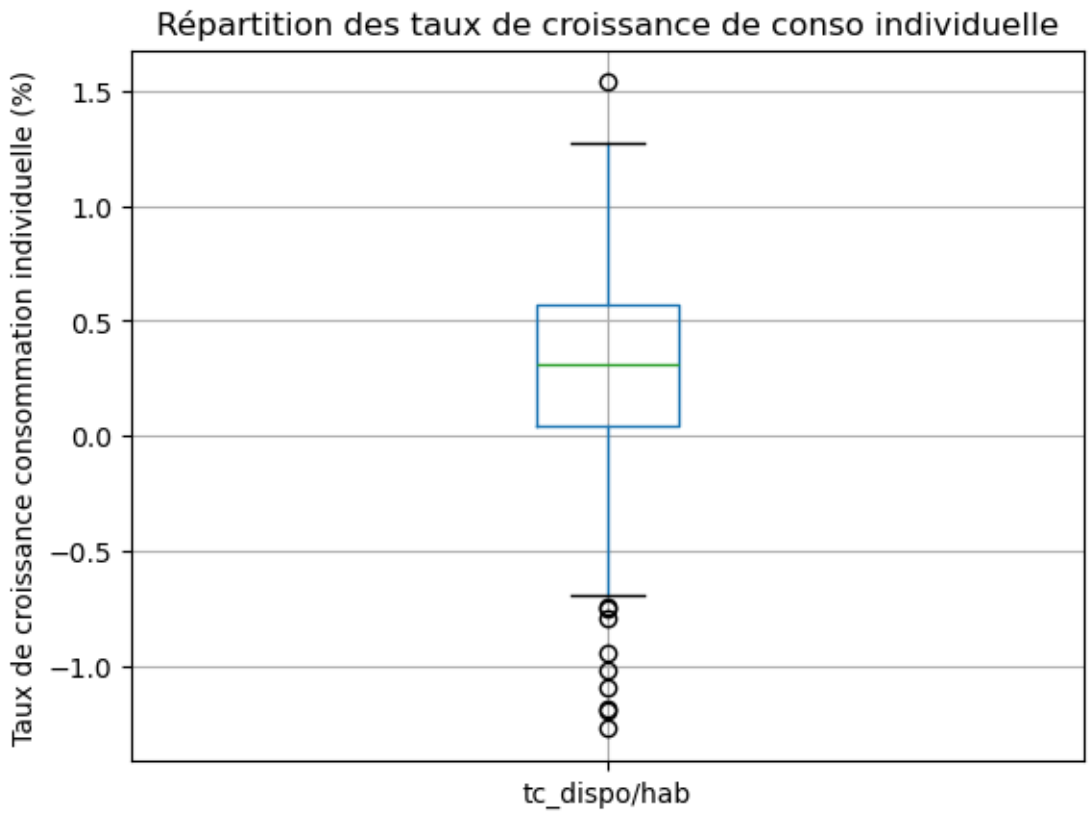
```
print(f"Valeur p : {p_value}")
```

```
Moyenne : 0.27
Écart type : 0.51
Z_min : -3.04
Z_max : 2.5
```




```
Statistique du test de Shapiro-Wilk : 0.9714375630708643
Valeur p : 0.0007117182193731307
```

[144]:
```
df_tmp.boxplot(column=["tc_dispo/hab"], showfliers=True)
plt.ylabel("Taux de croissance consommation individuelle (%)")
plt.title("Répartition des taux de croissance de conso individuelle")
plt.show()
```



[145]:
```
display((df_tmp.loc[df_tmp["tc_dispo/hab"]>1.3][["ISO3","Zone","tc_dispo/
↪hab"]]).sort_values("tc_dispo/hab",ascending=False))
display((df_tmp.loc[df_tmp["tc_dispo/hab"]<-0.7][["ISO3","Zone","tc_dispo/
↪hab"]]).sort_values("tc_dispo/hab",ascending=True))
```

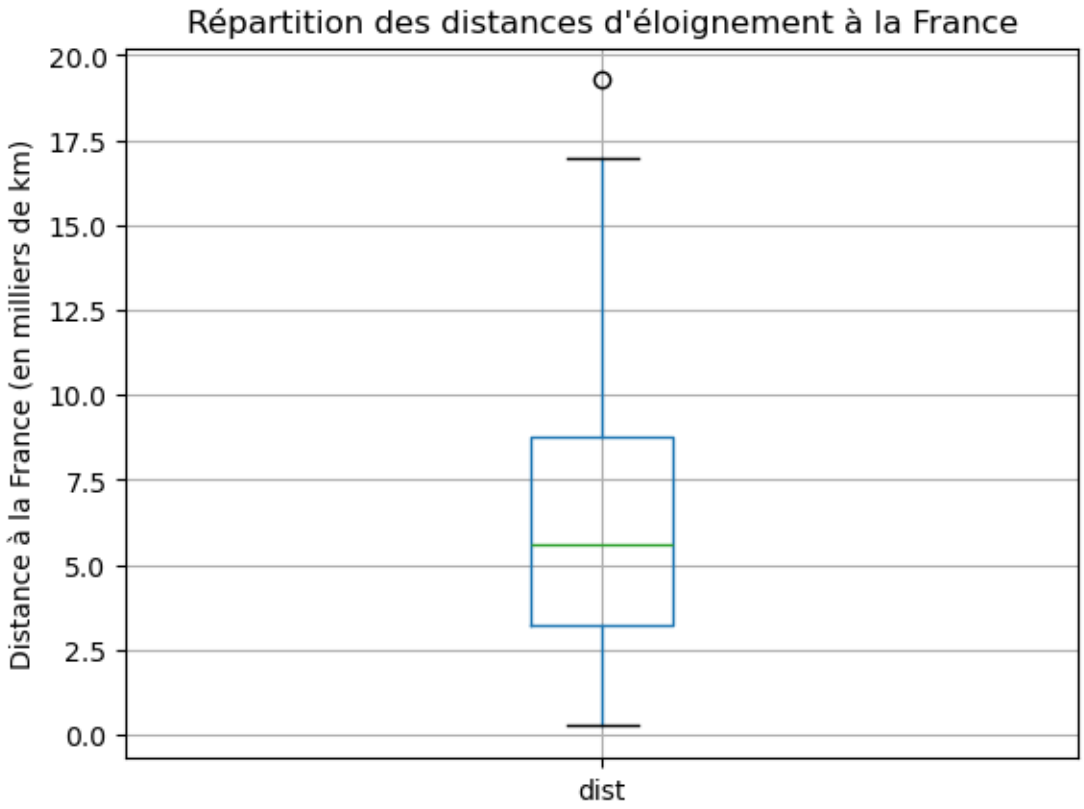
```
    ISO3   Zone  tc_dispo/hab
125  NPL  Népal      1.538586
```

```
    ISO3                                          Zone  tc_dispo/hab
99   LSO                                       Lesotho     -1.274245
111  MMR                                       Myanmar     -1.196041
179  VEN      Venezuela (République bolivarienne du)     -1.191799
95   LBR                                       Libéria     -1.095608
12   BEN                                         Bénin     -1.023529
0    AFG                                   Afghanistan     -0.945073
64   GNQ                       Guinée équatoriale     -0.795252
50   ERI                                     Érythrée     -0.753772
135  PRK  République populaire démocratique de Corée     -0.745265
```

On a significativement amélioré la normalité des données, grandement réduit les z-scores positifs et légèrement amélioré les z-scores négatifs

Variable dist

[146]:
```
df_tmp["dist"] = df_work["dist"]/1e3
df_tmp.boxplot(column=["dist"], showfliers=True)
plt.ylabel("Distance à la France (en milliers de km)")
plt.title("Répartition des distances d'éloignement à la France")
plt.show()
```



[147]:
```
display((df_tmp.loc[df_tmp["dist"]>17][["ISO3","Zone","dist"]]).
 ↪sort_values("dist",ascending=False))
```

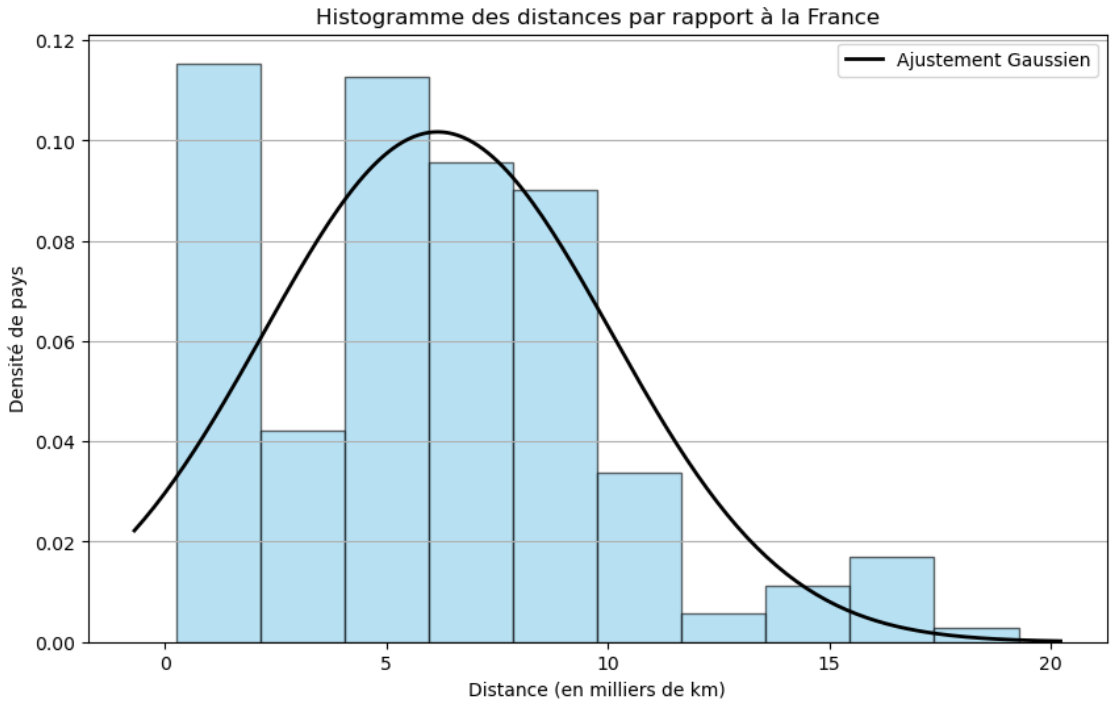
```
    ISO3              Zone     dist
127  NZL  Nouvelle-Zélande  19.2639
```

Le seul outlier correspond au pays quasiment aux antipodes de la France méttropolitaine : la Nouvelle-Zélande.

[148]:
```
data = df_tmp["dist"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
```

```
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des distances par rapport à la France")
plt.xlabel('Distance (en milliers de km)')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 6.16
Écart type : 3.93
Z_min : -1.5
Z_max : 3.33
```

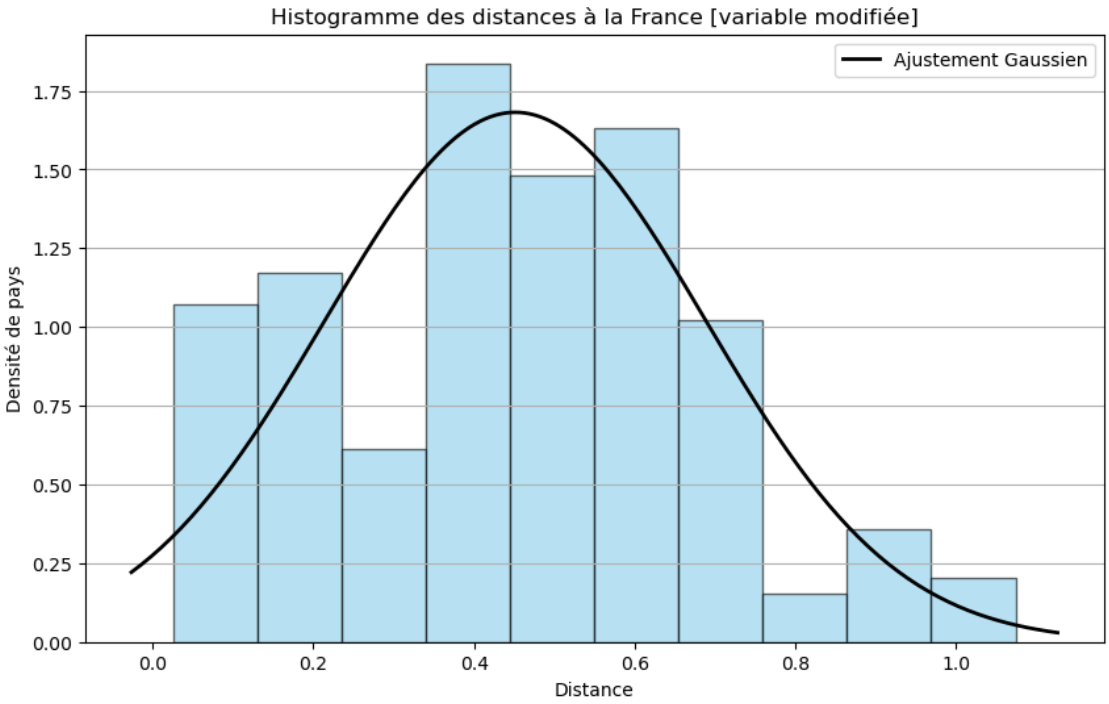



```
Statistique du test de Shapiro-Wilk : 0.9484446006539282
Valeur p : 2.759454076968214e-06
```

[149]:
```python
df_tmp["dist"] = df_work["dist"].apply(lambda x: (np.log(1+x/10000)))
data = df_tmp["dist"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)
```

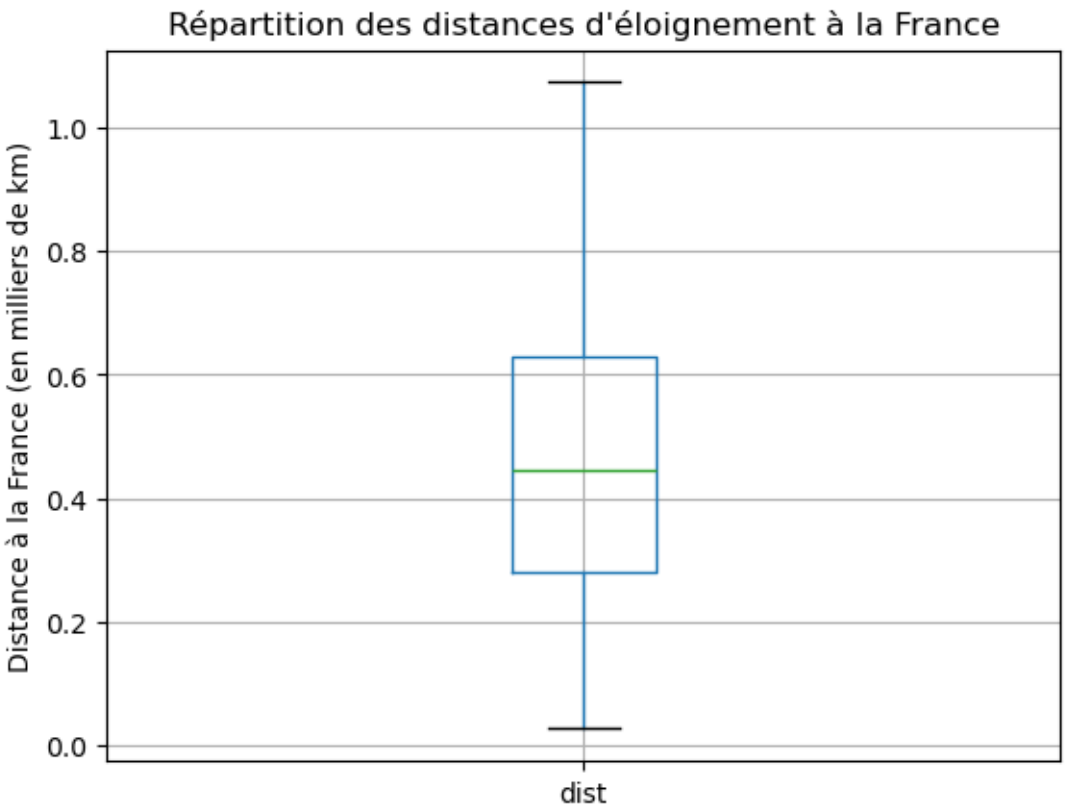
```python
# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des distances à la France [variable modifiée]")
plt.xlabel('Distance')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 0.45
Écart type : 0.24
Z_min : -1.79
Z_max : 2.62
```




```
Statistique du test de Shapiro-Wilk : 0.9739179268730834
Valeur p : 0.0014339708035155955
```

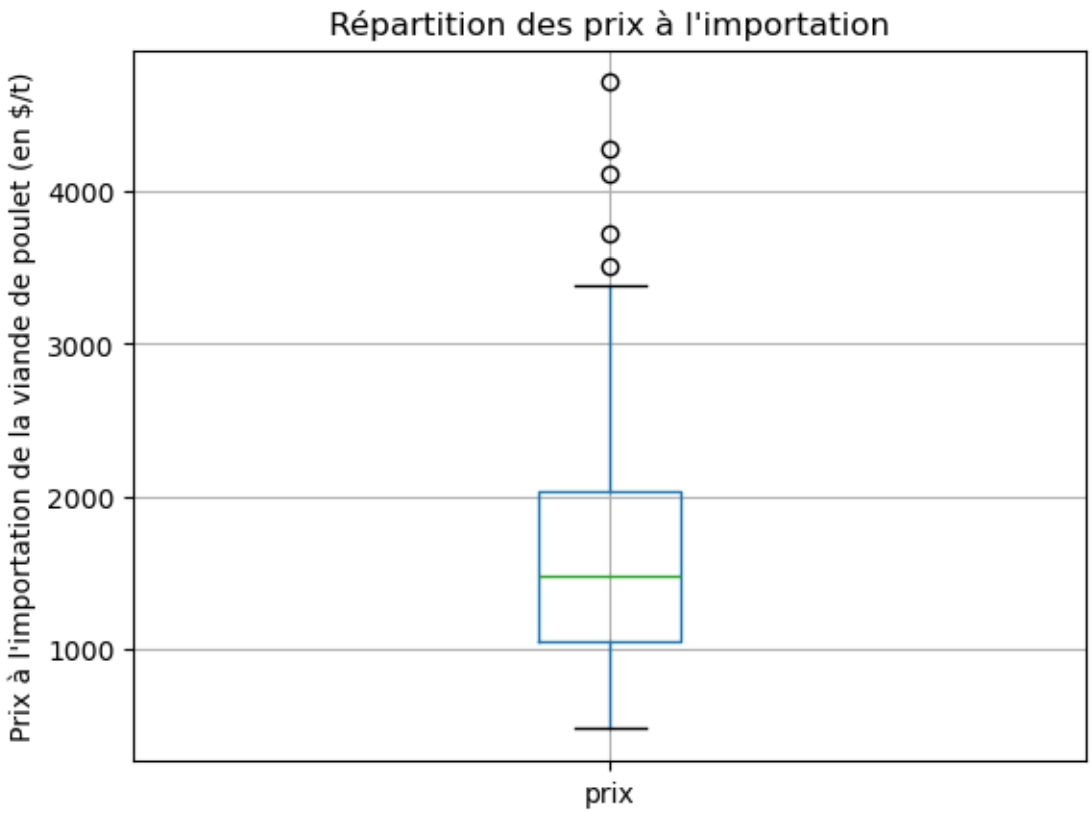
[150]:
```
df_tmp.boxplot(column=["dist"], showfliers=True)
plt.ylabel("Distance à la France (en milliers de km)")
plt.title("Répartition des distances d'éloignement à la France")
plt.show()
```



On améliore un peu la normalité et on répartit mieux les outliers.

Variable prix

[151]:
```
df_tmp.boxplot(column=["prix"], showfliers=True)
plt.ylabel("Prix à l'importation de la viande de poulet (en $/t)")
plt.title("Répartition des prix à l'importation")
plt.show()
```



[152]:
```
display((df_tmp.loc[df_tmp["prix"]>3500][["ISO3","Zone","prix"]]).
  ↪sort_values("prix",ascending=False))
```

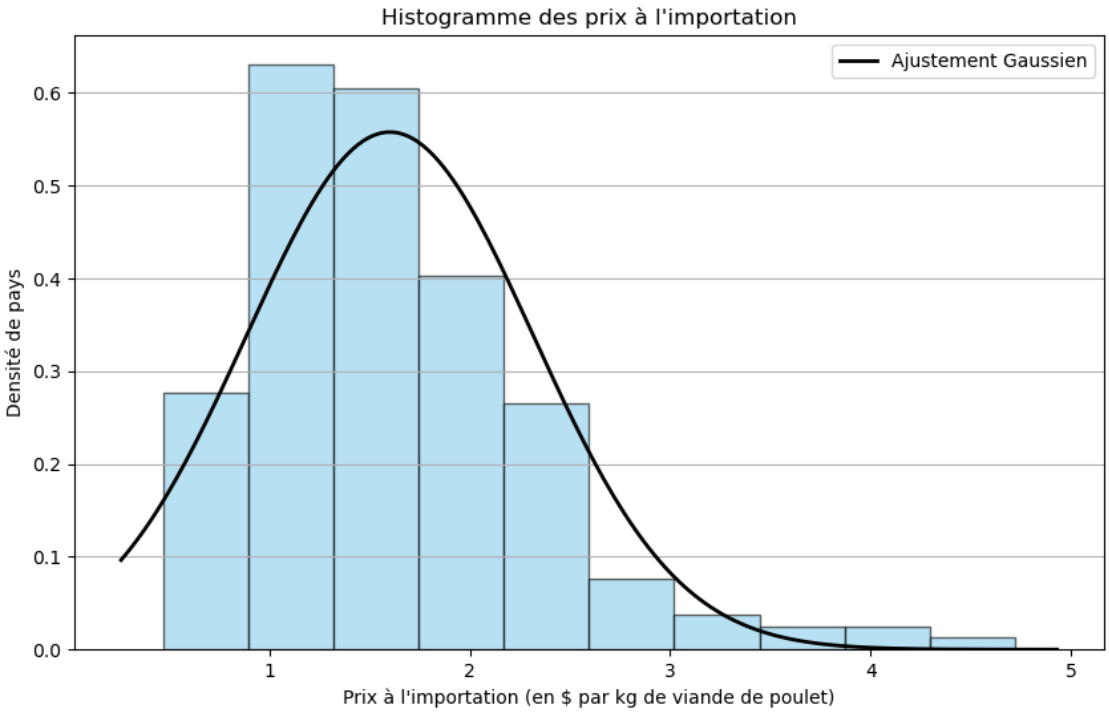
	ISO3	Zone	prix
101	LUX	Luxembourg	4717.0
29	CHE	Suisse	4276.0
79	ISL	Islande	4112.0
76	IRL	Irlande	3720.0
176	USA	États-Unis d'Amérique	3506.0

Des prix à l'importation anormalement élevés peuvent traduire des barrières douanières ou non tarifaires (barrières sanitaires).

[153]:
```
data = (df_tmp["prix"]/1000).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
```

```
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des prix à l'importation")
plt.xlabel("Prix à l'importation (en $ par kg de viande de poulet)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 1.6
Écart type : 0.72
Z_min : -1.57
Z_max : 4.34
```

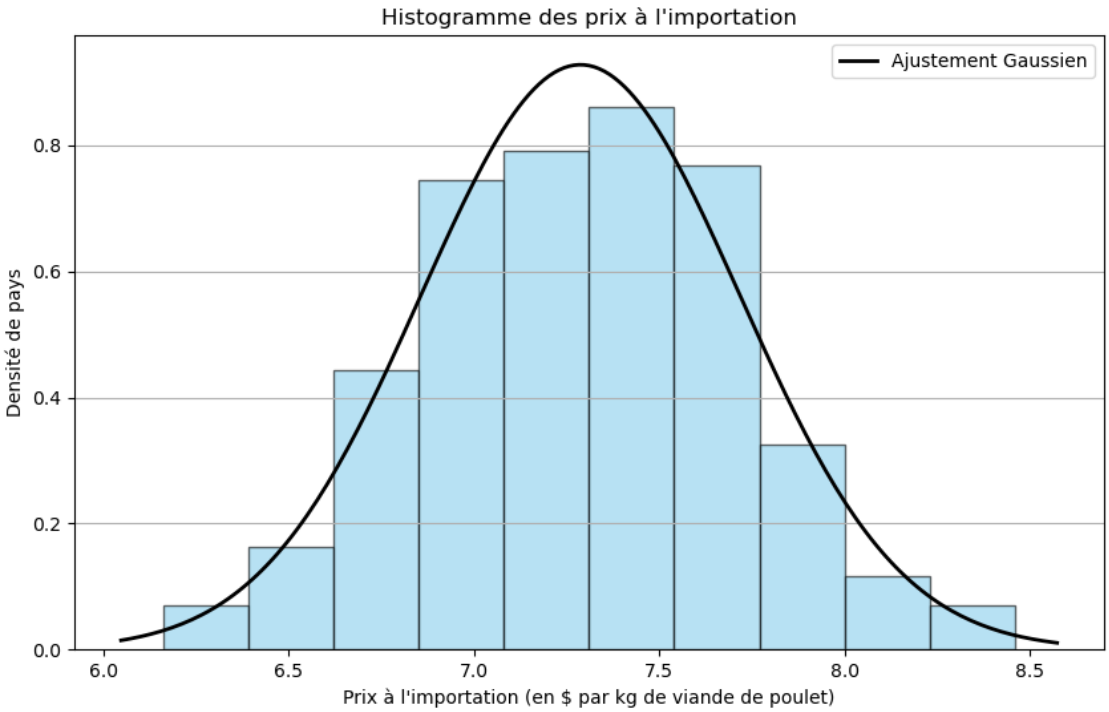



```
Statistique du test de Shapiro-Wilk : 0.9139698106708004
Valeur p : 5.384597998081091e-09
```

[154]:
```python
df_tmp["prix"] = df_work["prix"].apply(lambda x: (np.log(1+x)))
data = df_tmp["prix"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)
```

```
# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des prix à l'importation")
plt.xlabel("Prix à l'importation (en $ par kg de viande de poulet)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 7.29
Écart type : 0.43
Z_min : -2.61
Z_max : 2.72
```

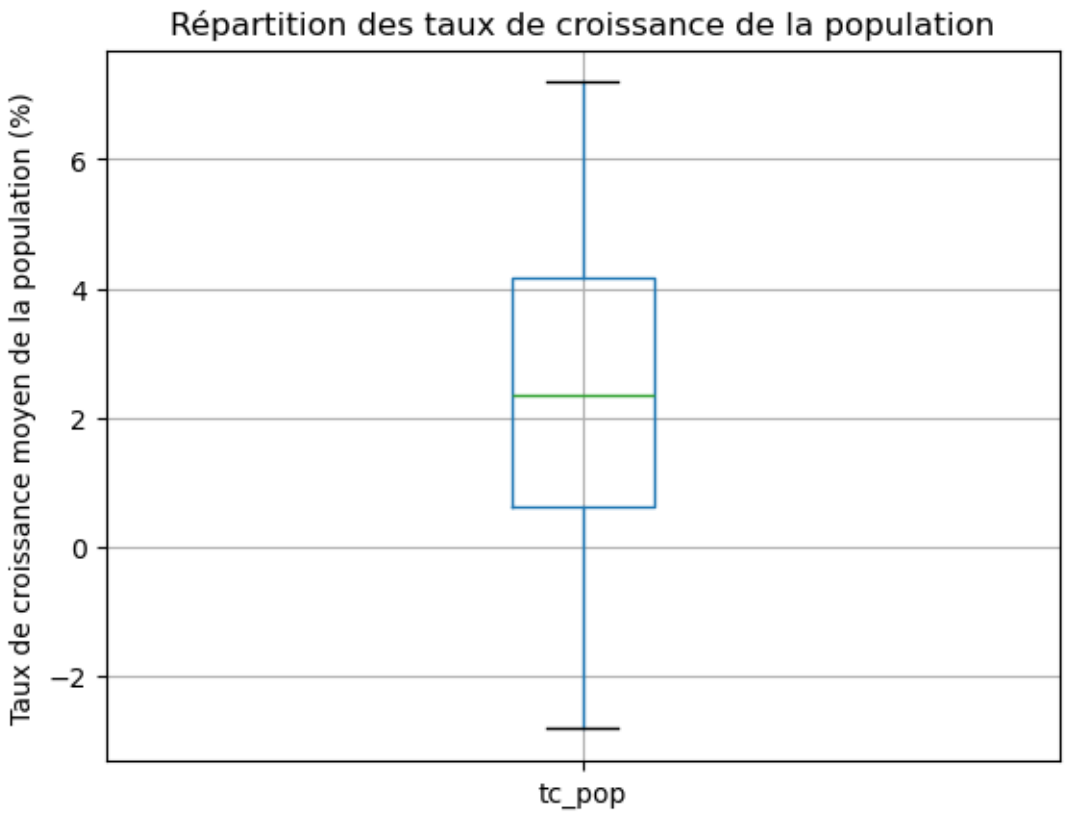



```
Statistique du test de Shapiro-Wilk : 0.9945310220072522
Valeur p : 0.7259128405637243
```

AVec cette transformation, les données de prix suivent une loi normale.

Variable tc_pop (taux de croissance de la population)

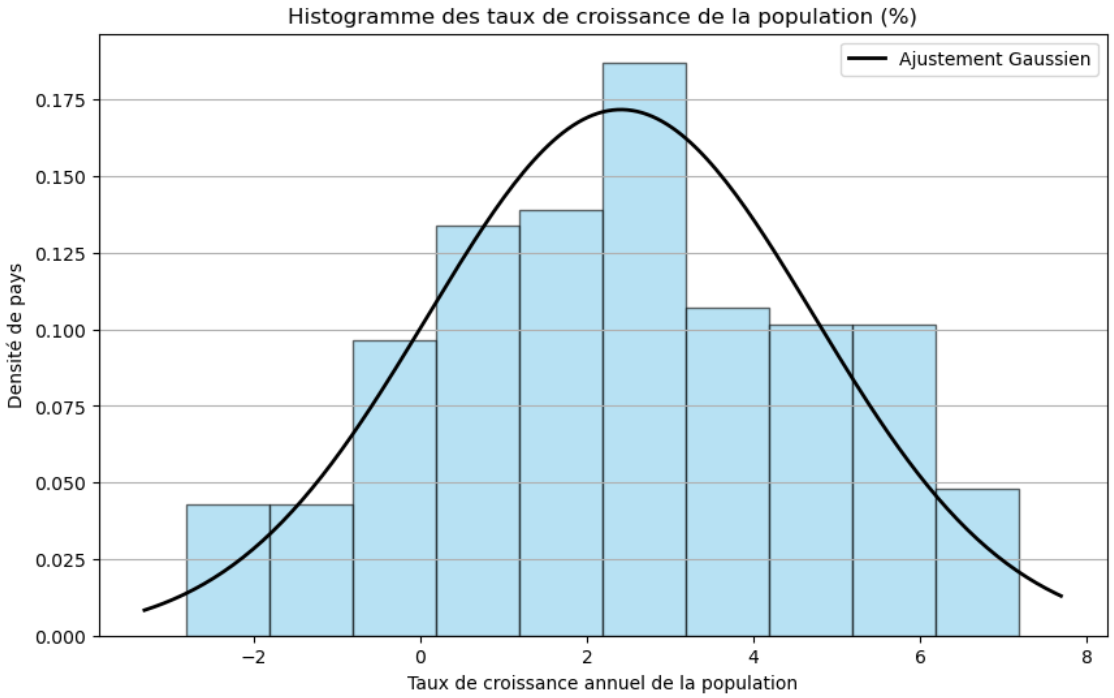
[155]:
```
df_work.boxplot(column=["tc_pop"], showfliers=True)
plt.ylabel("Taux de croissance moyen de la population (%)")
plt.title("Répartition des taux de croissance de la population")
plt.show()
```



[156]:
```
data = df_work["tc_pop"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
```

```
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des taux de croissance de la population (%)")
plt.xlabel('Taux de croissance annuel de la population')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 2.41
Écart type : 2.33
Z_min : -2.24
Z_max : 2.05
```

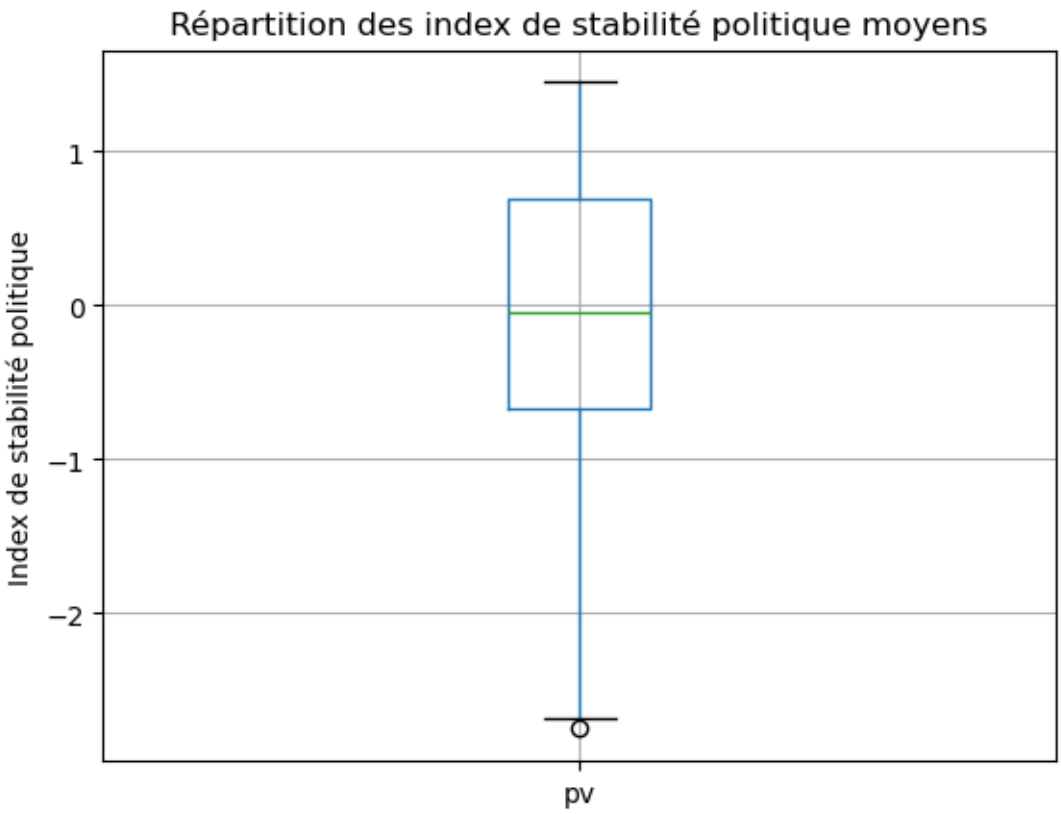



```
Statistique du test de Shapiro-Wilk : 0.9844128035140616
Valeur p : 0.035842856234601456
```

On peut même accepter l'hypothèse que les taux de croissance de la population suivent une loi normale.

Variable pv (stabilité politique)

[157]:
```
df_tmp.boxplot(column=["pv"], showfliers=True)
plt.ylabel("Index de stabilité politique")
plt.title("Répartition des index de stabilité politique moyens")
plt.show()
```



[158]:
```
display((df_tmp.loc[df_tmp["pv"]<-2.5][["ISO3","Zone","pv"]]).
  ↪sort_values("pv",ascending=True))
```

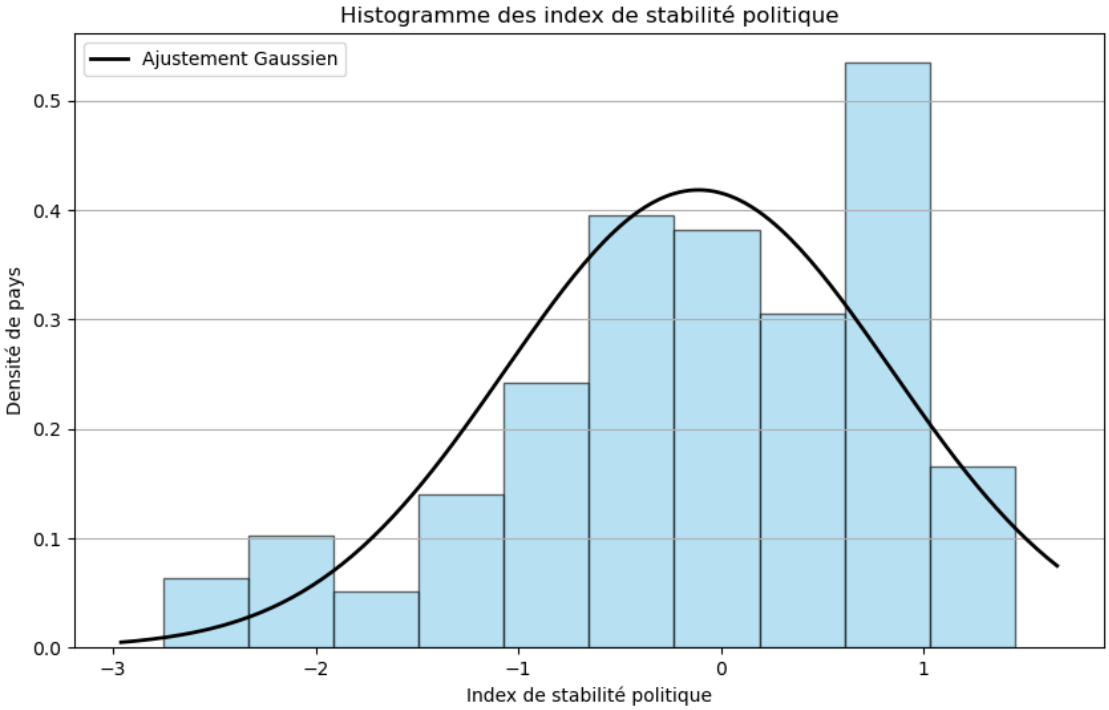
	ISO3	Zone	pv
159	SYR	République arabe syrienne	-2.75
183	YEM	Yémen	-2.68
0	AFG	Afghanistan	-2.60

Les deux zones (Syrie et Yémen) avec un index de stabilité politique pv atypiquement bas ont connu des épisodes de guerre civile sur la période.

[159]:
```
data = df_work["pv"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")
```

```
# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des index de stabilité politique")
plt.xlabel('Index de stabilité politique')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

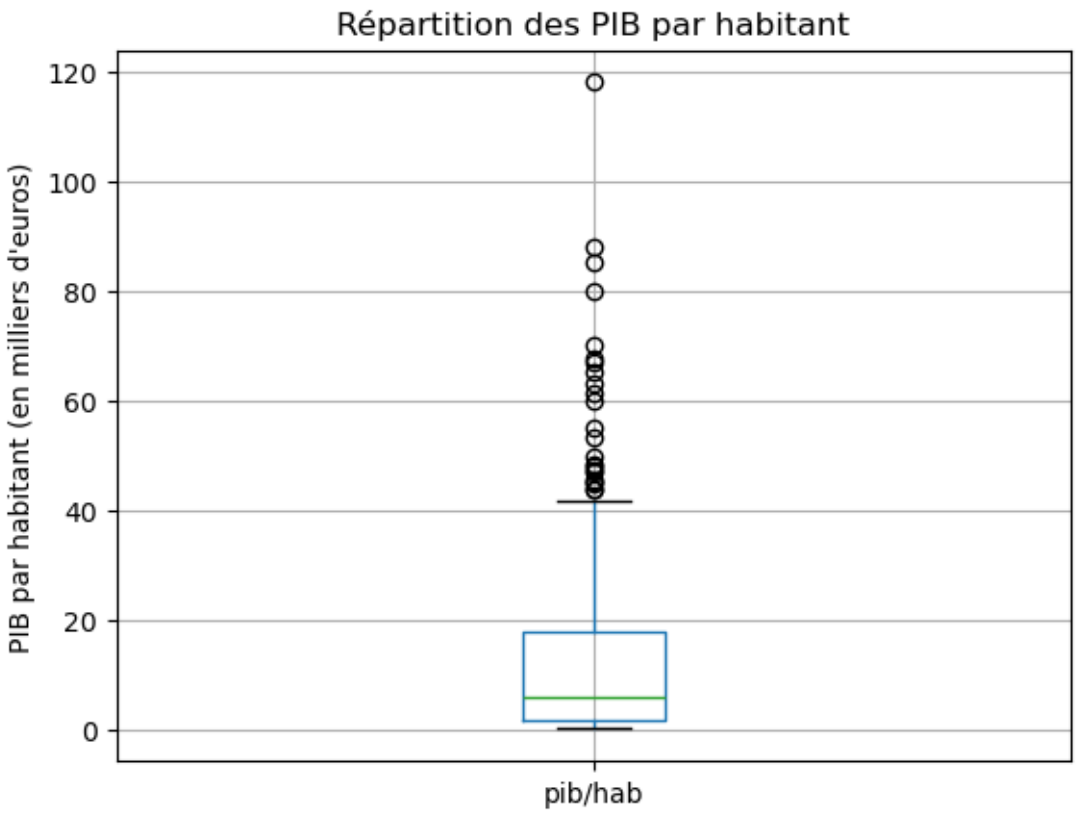
```
Moyenne : -0.11
Écart type : 0.96
Z_min : -2.76
Z_max : 1.63
```




```
Statistique du test de Shapiro-Wilk : 0.9530602791037242
Valeur p : 7.490725790086021e-06
```

Variable pib/hab

[160]:
```
df_tmp["pib/hab"] = df_work["pib/hab"]/1e3
df_tmp.boxplot(column=["pib/hab"], showfliers=True)
plt.ylabel("PIB par habitant (en milliers d'euros)")
plt.title("Répartition des PIB par habitant")
plt.show()
```



[161]:
```
display((df_tmp.loc[df_tmp["pib/hab"]>40][["ISO3","Zone","pib/hab"]]).
  ↪sort_values("pib/hab",ascending=True))
```

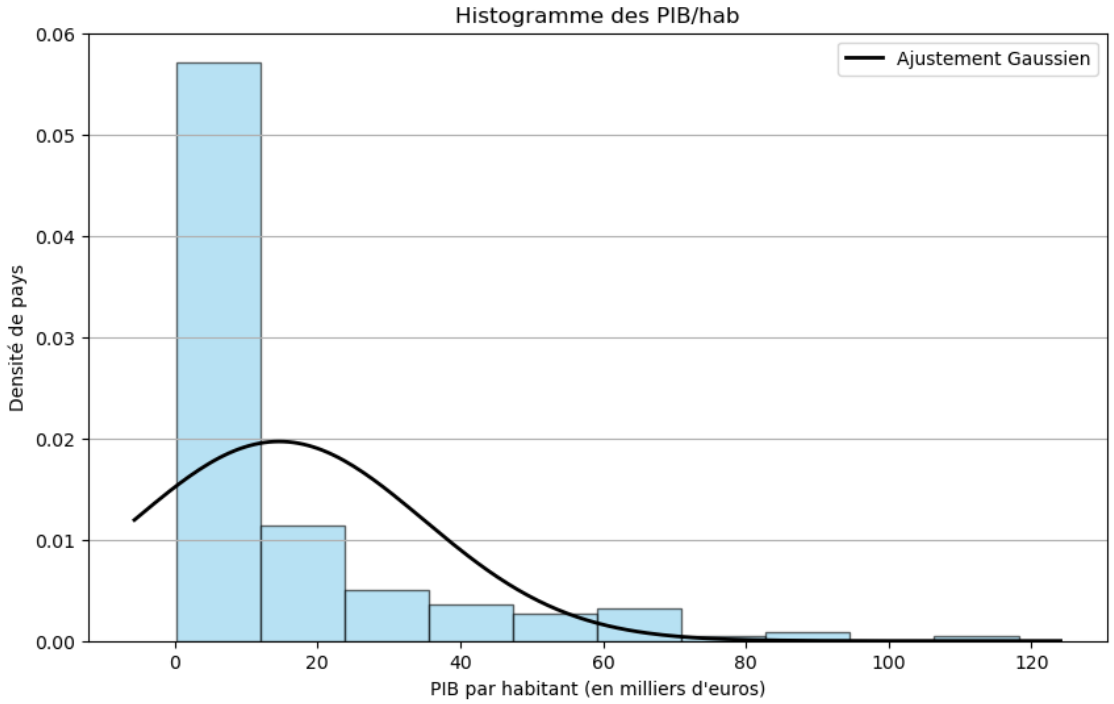
	ISO3	Zone	pib/hab
56	FRA	France	41.84280
127	NZL	Nouvelle-Zélande	43.92500
58	GBR	Royaume-Uni de Grande-Bretagne et d'Irlande du…	43.94335
3	ARE	Émirats arabes unis	44.93594
69	HKG	Chine - RAS de Hong-Kong	45.49066
80	ISR	Israël	45.83571
11	BEL	Belgique	47.23278
43	DEU	Allemagne	47.59429
28	CAN	Canada	48.30016
54	FIN	Finlande	48.62474
8	AUT	Autriche	49.92282
123	NLD	Pays-Bas (Royaume des)	53.33101
156	SWE	Suède	55.20474
7	AUS	Australie	60.14311
45	DNK	Danemark	61.54279
176	USA	États-Unis d'Amérique	63.11720
79	ISL	Islande	65.34883

```
145  SGP                                  Singapour   67.06025
103  MAC                      Chine - RAS de Macao   67.69473
138  QAT                                    Qatar   70.27649
76   IRL                                  Irlande   80.12525
124  NOR                                  Norvège   85.48789
29   CHE                                   Suisse   88.00213
101  LUX                                Luxembourg  118.16818
```

[162]:
```python
data = (df_work["pib/hab"]/1000).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des PIB/hab")
plt.xlabel("PIB par habitant (en milliers d'euros)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()
```

```
# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 14.65
Écart type : 20.32
Z_min : -0.71
Z_max : 5.1
```

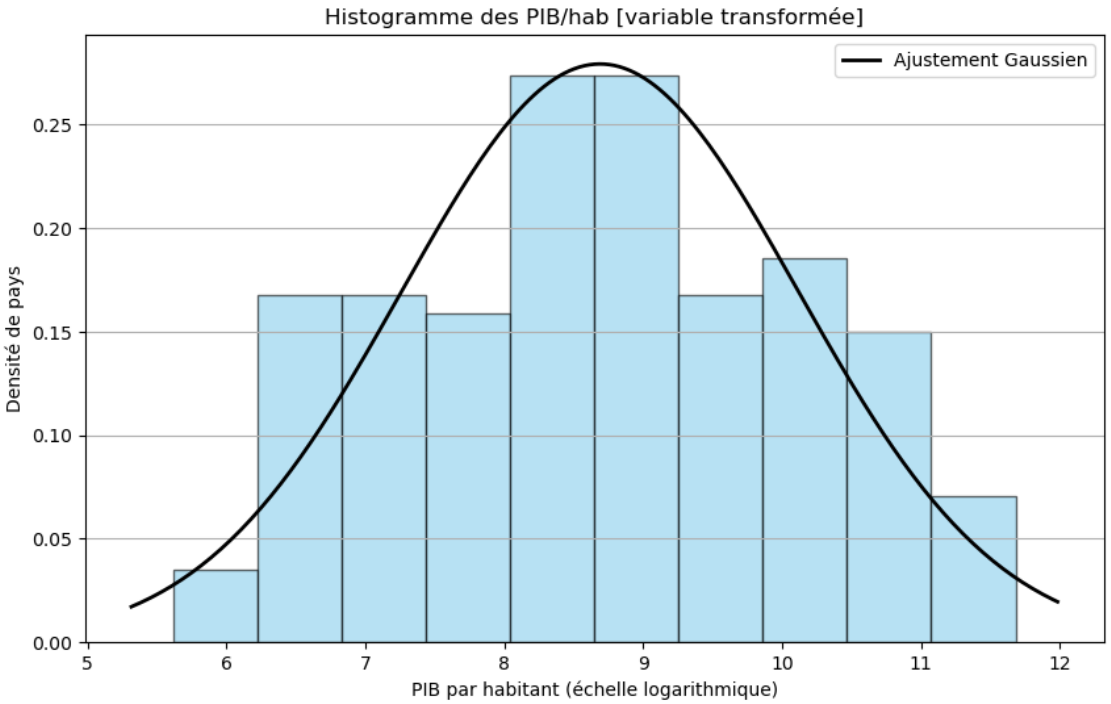



```
Statistique du test de Shapiro-Wilk : 0.7005524291796824
Valeur p : 4.823131561518574e-18
```

[163]:
```
df_tmp["pib/hab"] = df_work["pib/hab"].apply(lambda x: np.log(1+x/1))
data = df_tmp["pib/hab"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
```

```
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des PIB/hab [variable transformée]")
plt.xlabel("PIB par habitant (échelle logarithmique)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

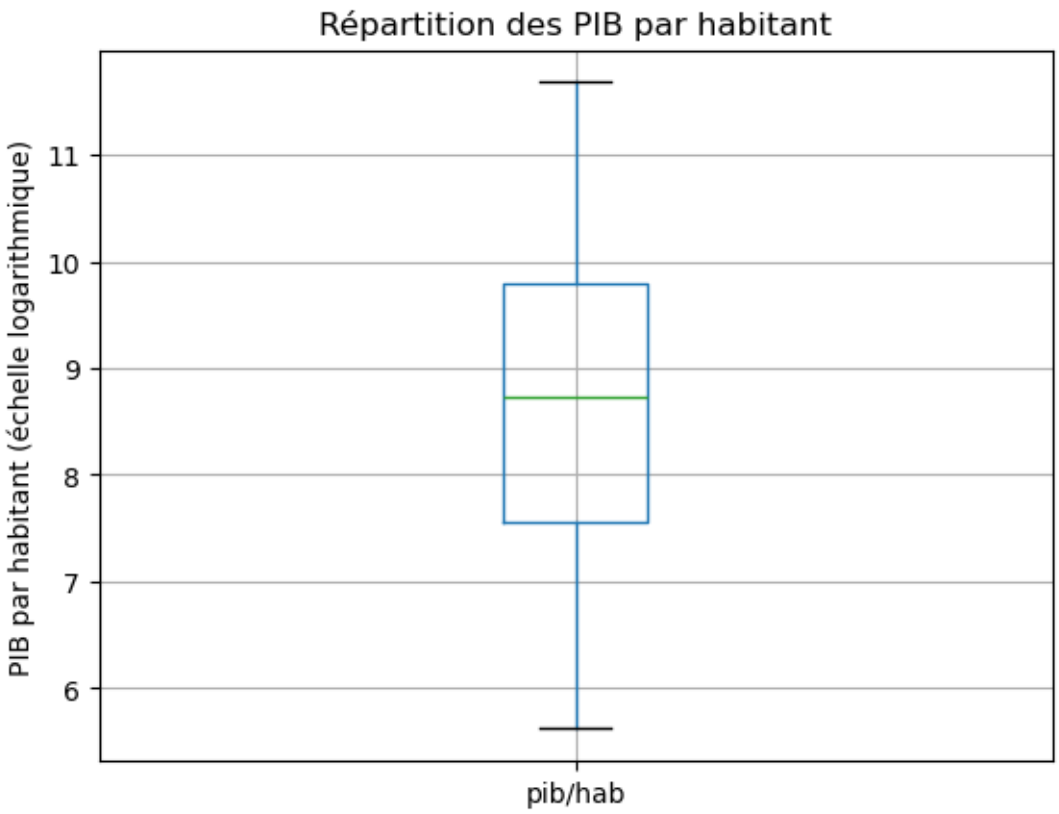
```
Moyenne : 8.69
Écart type : 1.43
Z_min : -2.14
Z_max : 2.09
```




```
Statistique du test de Shapiro-Wilk : 0.977774320784365
Valeur p : 0.0044657247324070585
```

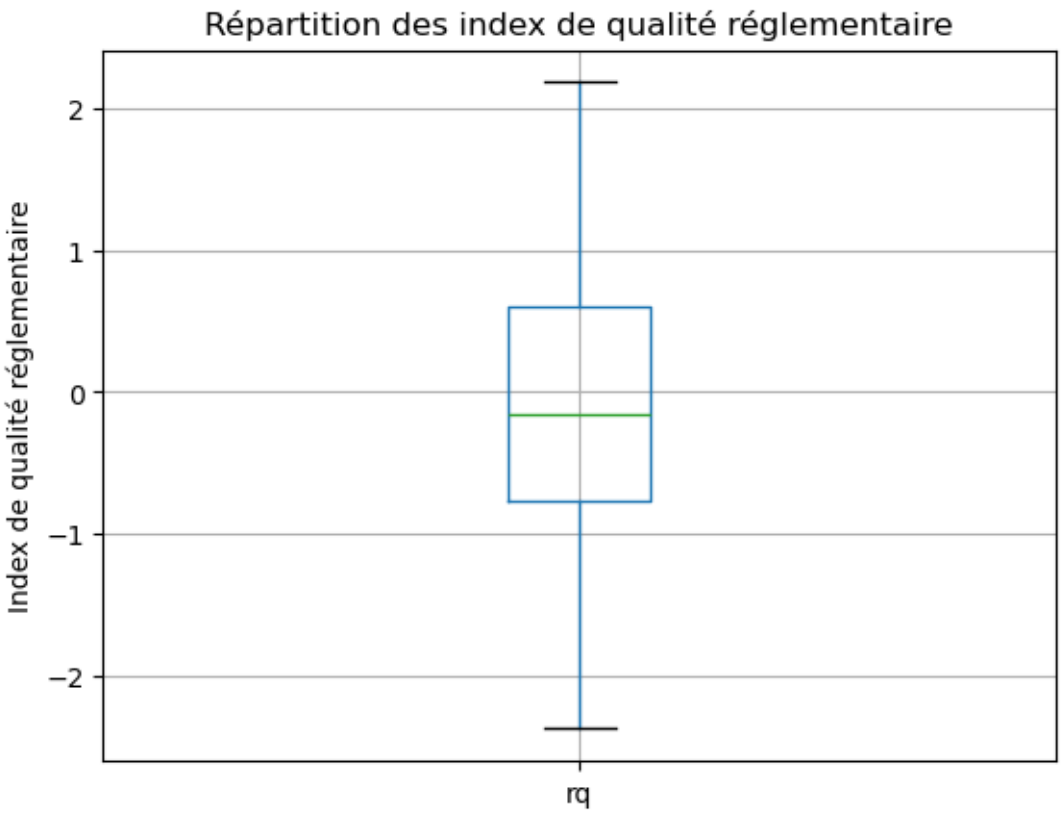
Appliquer le logarithme naturel améliore grandemnt la régularité de la répartition des données.

[164]:
```
df_tmp.boxplot(column=["pib/hab"], showfliers=True)
plt.ylabel("PIB par habitant (échelle logarithmique)")
plt.title("Répartition des PIB par habitant")
plt.show()
```



Variable rq (qualité réglementaire)

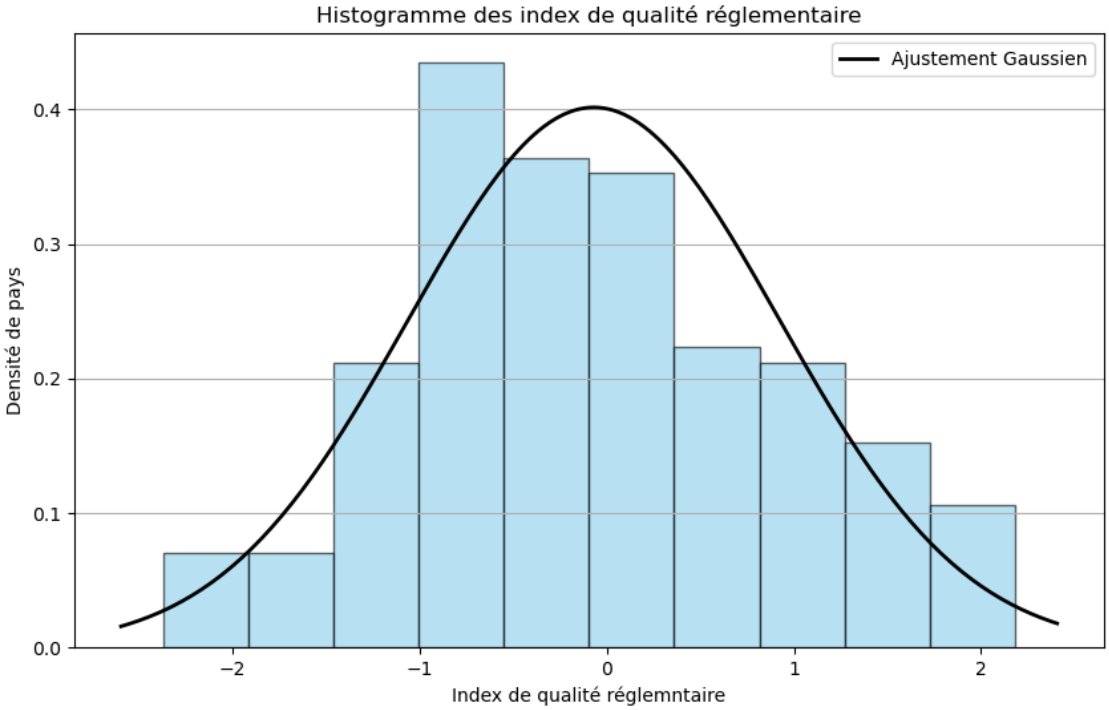
[165]:
```
df_tmp.boxplot(column=["rq"], showfliers=True)
plt.ylabel("Index de qualité réglementaire")
plt.title("Répartition des index de qualité réglementaire")
plt.show()
```



[166]:
```
data = df_work["rq"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
```

```
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des index de qualité réglementaire")
plt.xlabel('Index de qualité réglemntaire')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : -0.07
Écart type : 1.0
Z_min : -2.31
Z_max : 2.26
```

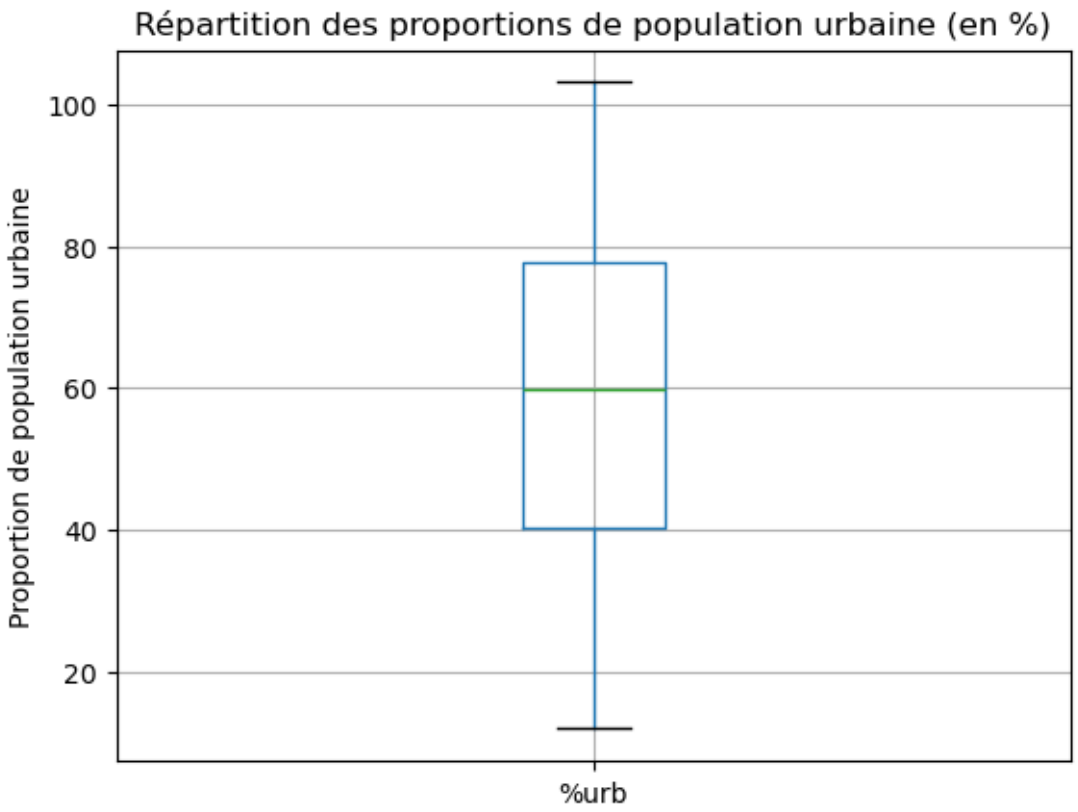



```
Statistique du test de Shapiro-Wilk : 0.9825276629975126
Valeur p : 0.01956105605822986
```

La distribution des données initiales pour cette variable est plutôt bonne.

Variable %urb

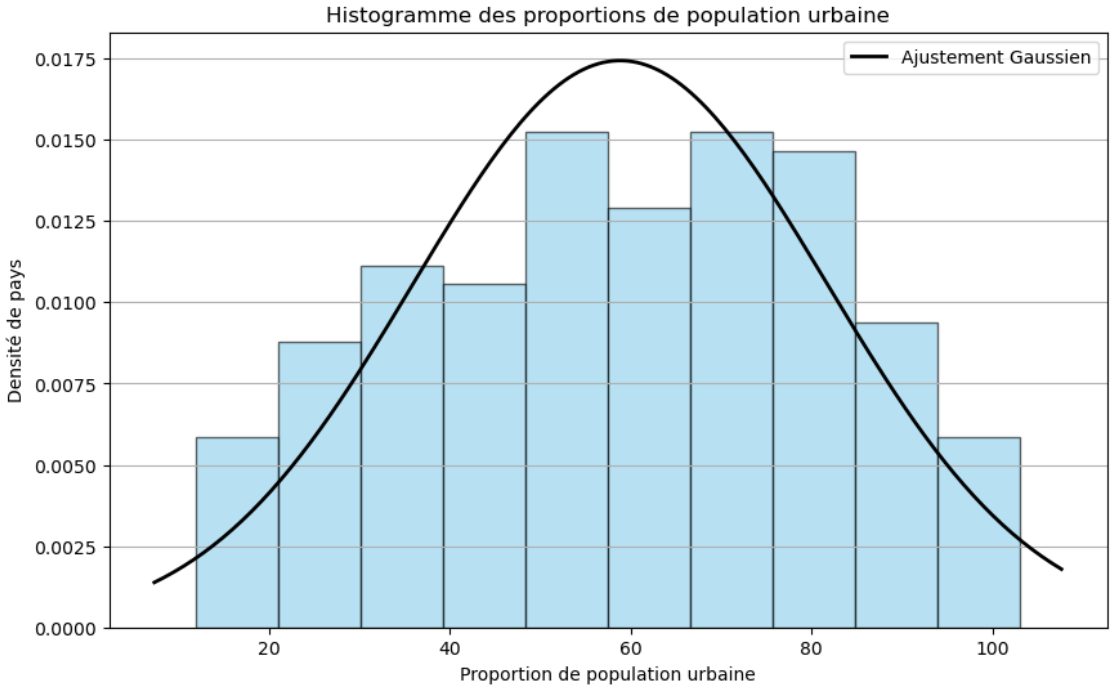
[167]:
```
df_tmp.boxplot(column=["%urb"], showfliers=True)
plt.ylabel("Proportion de population urbaine")
plt.title("Répartition des proportions de population urbaine (en %)")
plt.show()
```



[168]:
```
data = df_work["%urb"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
```

```
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des proportions de population urbaine")
plt.xlabel('Proportion de population urbaine')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

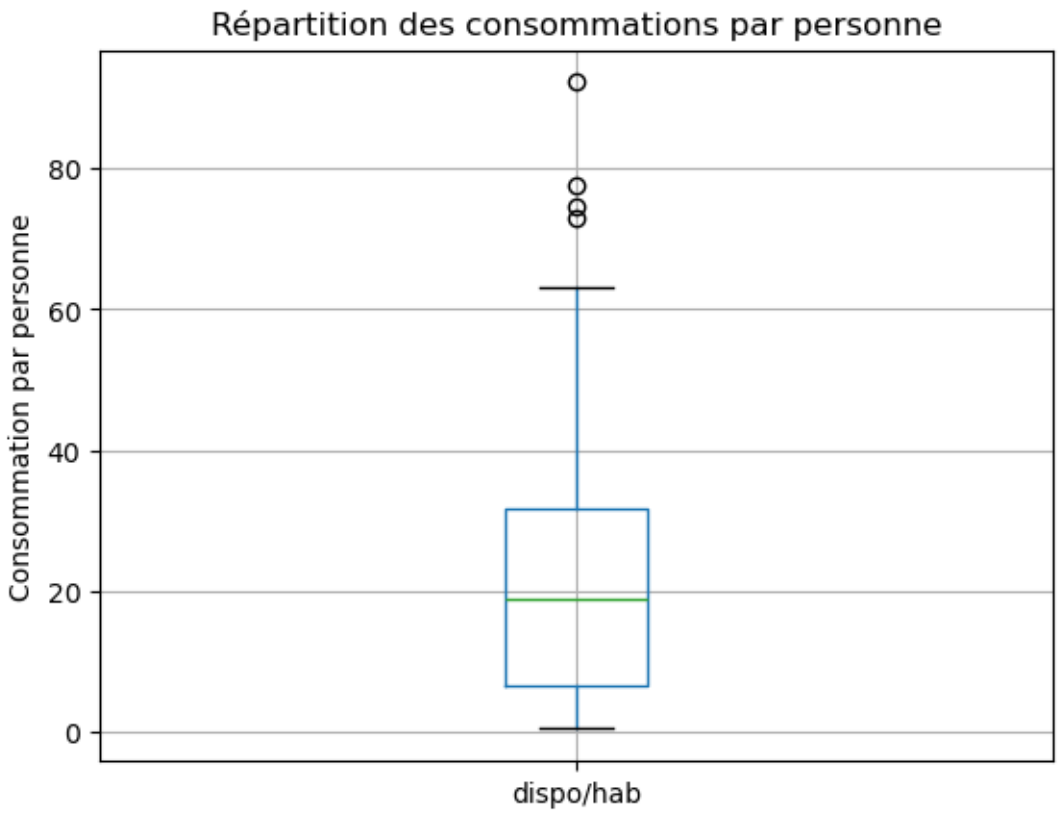
```
Moyenne : 58.85
Écart type : 22.94
Z_min : -2.05
Z_max : 1.93
```




```
Statistique du test de Shapiro-Wilk : 0.974418595836969
Valeur p : 0.0016564619660838372
```

Variable dispo/hab (consommation par personne)

[169]:
```
df_tmp.boxplot(column=["dispo/hab"], showfliers=True)
plt.ylabel("Consommation par personne")
plt.title("Répartition des consommations par personne")
plt.show()
```



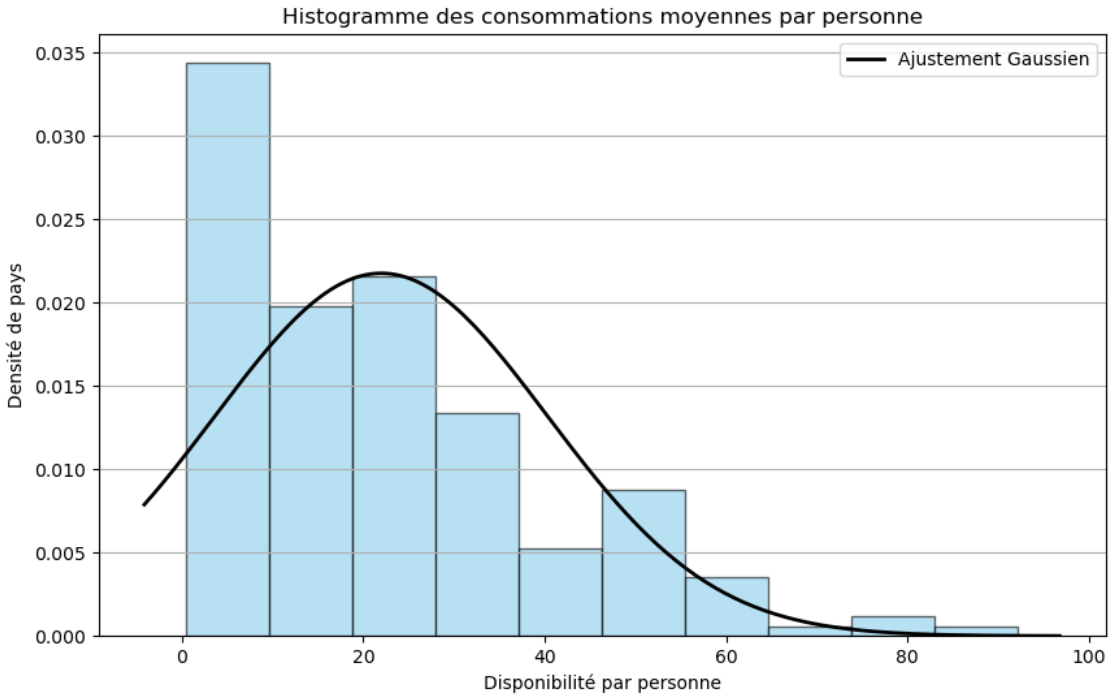
[170]:
```
display((df_tmp.loc[df_tmp["dispo/hab"]>60][["ISO3","Zone","dispo/hab"]]).
  ↪sort_values("dispo/hab",ascending=True))
```

	ISO3	Zone	dispo/hab
80	ISR	Israël	62.10
24	BRN	Brunéi Darussalam	62.16
6	ATG	Antigua-et-Barbuda	63.19
90	KNA	Saint-Kitts-et-Nevis	72.85
178	VCT	Saint-Vincent-et-les Grenadines	74.72
182	WSM	Samoa	77.46
166	TON	Tonga	92.21

[171]:
```
data = df_work["dispo/hab"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
```

```
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des consommations moyennes par personne")
plt.xlabel('Disponibilité par personne')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 21.98
Écart type : 18.38
Z_min : -1.17
Z_max : 3.82
```

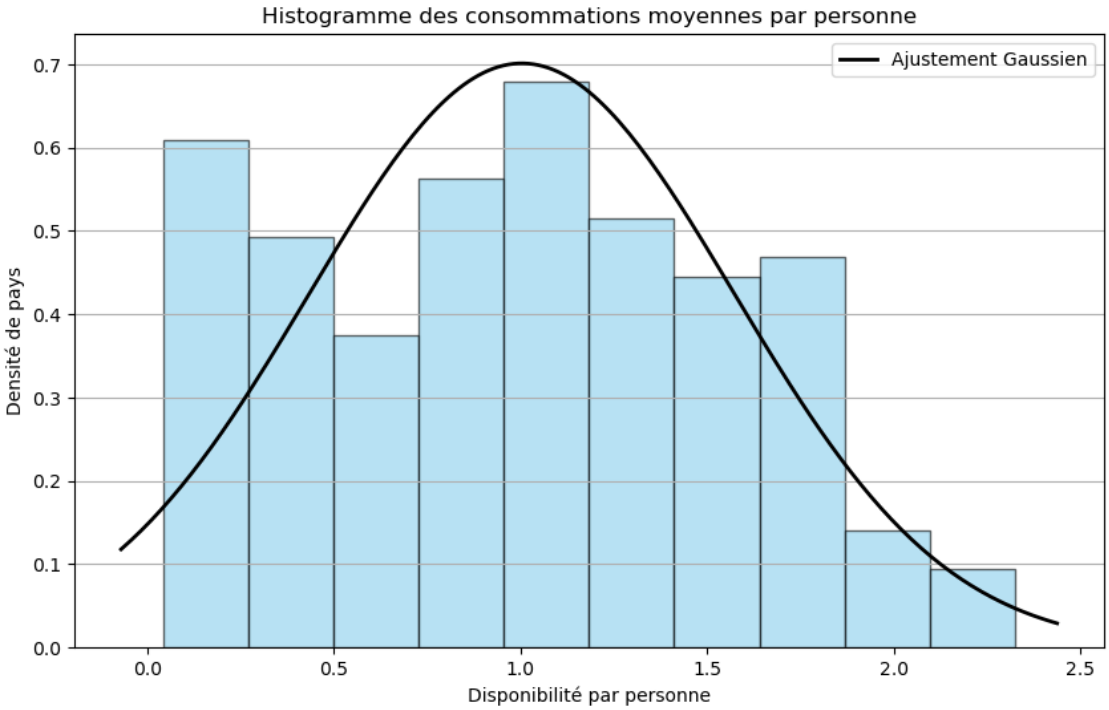



```
Statistique du test de Shapiro-Wilk : 0.907004097854673
Valeur p : 1.8545016398942342e-09
```

[172]:
```
df_tmp["dispo/hab"] = df_work["dispo/hab"].apply(lambda x: np.log(1+x/10))
data = df_tmp["dispo/hab"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)
```

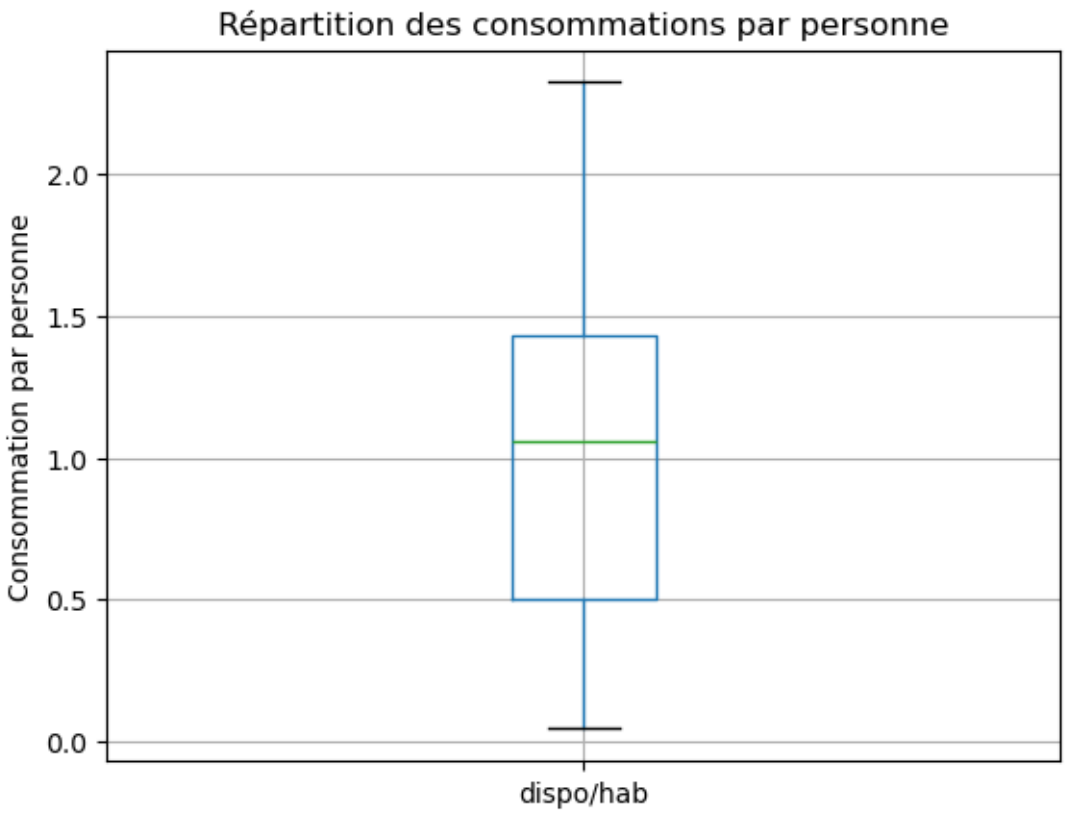
```python
# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des consommations moyennes par personne")
plt.xlabel('Disponibilité par personne')
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 1.0
Écart type : 0.57
Z_min : -1.68
Z_max : 2.32
```



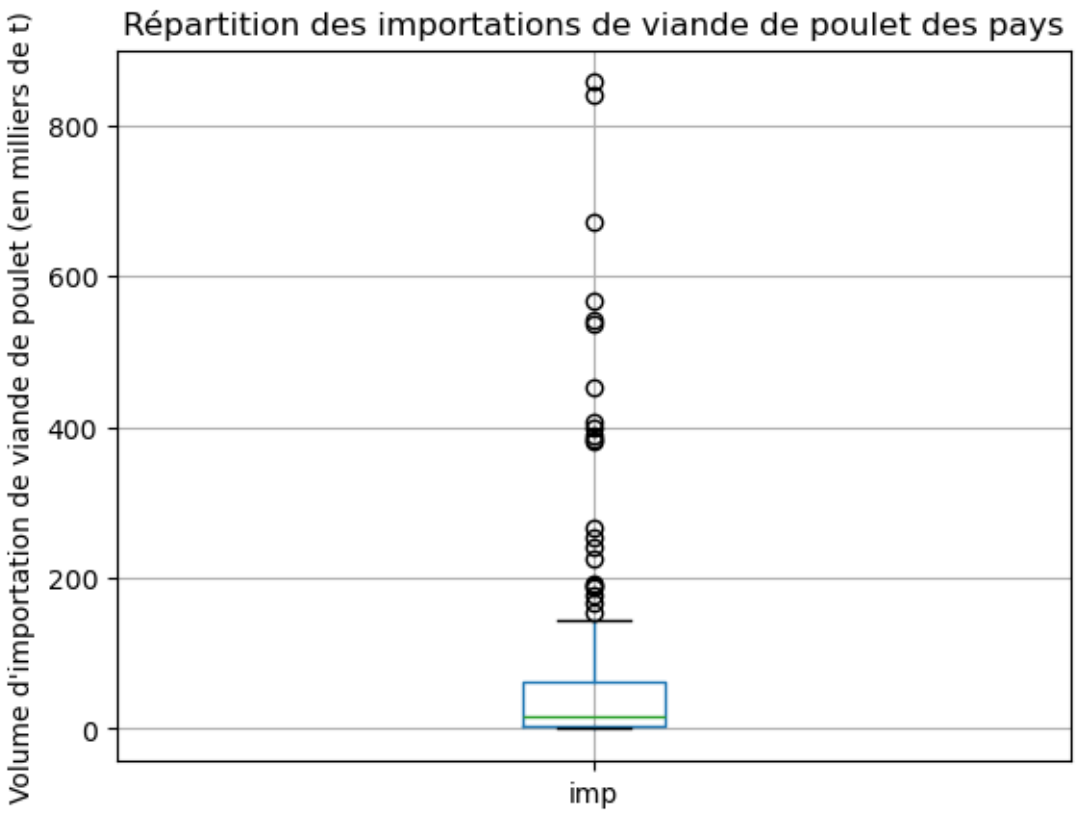

```
Statistique du test de Shapiro-Wilk : 0.9690094993321735
Valeur p : 0.00036648627957519645
```

[173]:
```
df_tmp.boxplot(column=["dispo/hab"], showfliers=True)
plt.ylabel("Consommation par personne")
plt.title("Répartition des consommations par personne")
plt.show()
```



Variable imp (importations de viande de poulet)

[174]:
```
df_tmp["imp"] = df_work["imp"]/1e3
df_tmp.boxplot(column=["imp"], showfliers=True)
plt.ylabel("Volume d'importation de viande de poulet (en milliers de t)")
plt.title("Répartition des importations de viande de poulet des pays")
plt.show()
```



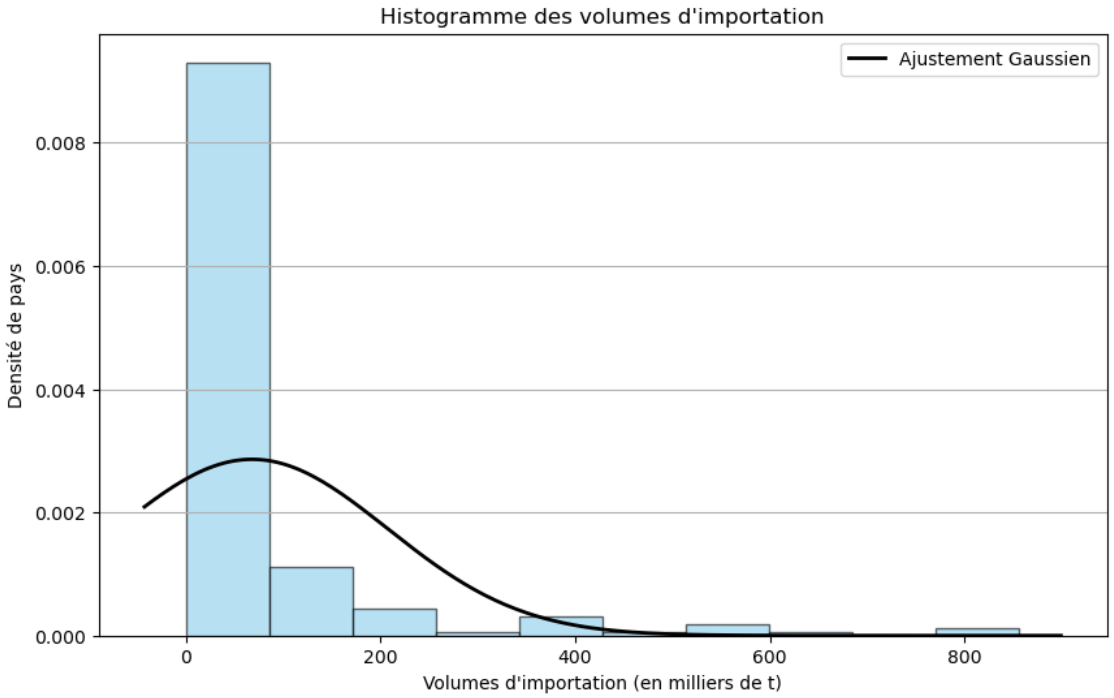
[175]:
```
display((df_tmp.loc[df_tmp["imp"]>200][["ISO3","Zone","imp"]]).
  ↪sort_values("imp",ascending=False))
```

	ISO3	Zone	imp
107	MEX	Mexique	856.59991
31	CHN	Chine (continentale)	840.85157
142	SAU	Arabie saoudite	672.51184
69	HKG	Chine - RAS de Hong-Kong	568.16751
84	JPN	Japon	541.21428
3	ARE	Émirats arabes unis	536.01567
43	DEU	Allemagne	452.59041
123	NLD	Pays-Bas (Royaume des)	406.36530
78	IRQ	Iraq	399.46931
56	FRA	France	387.47048
184	ZAF	Afrique du Sud	384.10689
58	GBR	Royaume-Uni de Grande-Bretagne et d'Irlande du…	379.82165
1	AGO	Angola	265.84372
40	CUB	Cuba	253.97501
140	RUS	Fédération de Russie	239.79952
132	PHL	Philippines	224.77593

[176]:
```
data = (df_work["imp"]/1e3).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des volumes d'importation")
plt.xlabel("Volumes d'importation (en milliers de t)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 67.73
Écart type : 139.84
```

```
Z_min : -0.48
Z_max : 5.64
```



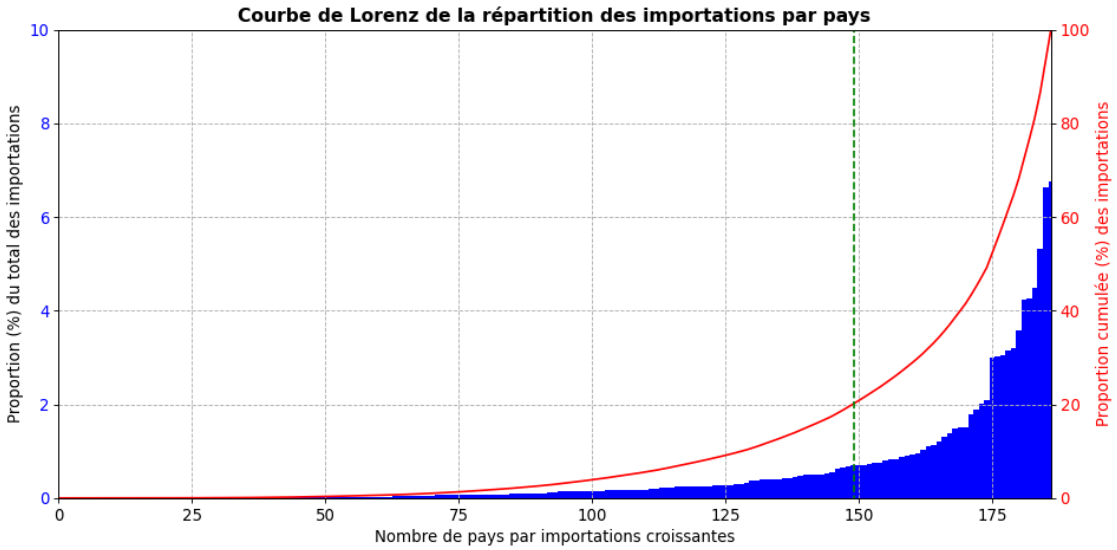

```
Statistique du test de Shapiro-Wilk : 0.5197849032943722
Valeur p : 2.2062138396531333e-22
```

Cette distribution est particulièrement problématique car la plupart des valeurs sont concentrées entre 0 et 200.

[177]:
```
# Courbe de Lorenz des importations

data = (df_work["imp"]/1e3).sort_values(ascending=True)

data_prop = (100 / data.sum()) * data
data_csum = data_prop.cumsum()

#Calcul du 80/20
lim8020 = len(data_csum.loc[data_csum.values>=20])
print("Nombre minimal de pays représentant 80% des importations :",lim8020)
print("Soit, en proportion du nombre de pays :", round(100 * lim8020 /␣
 ↪len(data),1), "%\n")

#Courbe de Lorenz
data.reset_index(inplace=True,drop=True)
```

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

p=ax1.bar(x=data.index.values, height=data_prop, width=1, color="blue")

ax1.set_xlabel("Nombre de pays par importations croissantes", fontsize=12)
ax1.set_xlim([0,len(data)-1])
ax1.set_ylabel("Proportion (%) du total des importations", fontsize=12)
ax1.set_ylim([0,10])
#ax1.set_yticks(np.linspace(0,1,11))
ax1.tick_params(axis='x', labelsize=12)
ax1.tick_params(axis='y', labelcolor='blue', labelsize=12)
plt.grid(visible=True, axis='both', linestyle='--')

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

ax2.plot(data.index.values,data_csum,color='red')

ax2.set_ylabel("Proportion cumulée (%) des importations", color='red',␣
 ↪fontsize=12)
ax2.set_ylim([0,100])
ax2.set_yticks(np.linspace(0,100,6))
ax2.tick_params(axis='y', labelcolor='red', labelsize=12)
plt.title("Courbe de Lorenz de la répartition des importations par pays",␣
 ↪fontsize=14, fontweight='heavy')
fig.tight_layout() # Pour affichage optimal du 2nd axe vertical à droite

# Limite 80/20
plt.plot([len(data)-lim8020]*13, np.linspace(-10,110,13), color='green',␣
 ↪linestyle='dashed')

plt.savefig("Lorenz_imp.png")
plt.show()

# Remarque : l'aire sous la courbe rouge (intégrale) est calculée par la␣
 ↪méthode des rectangles
# Bien penser à normaliser en divisant par le nombre de références
print("\nCoefficient de Gini pour la répartition des importations :",round(1-0.
 ↪02*(data_csum).sum()/(len(data)),3),"\n")
```

```
Nombre minimal de pays représentant 80% des importations : 38
Soit, en proportion du nombre de pays : 20.3 %
```

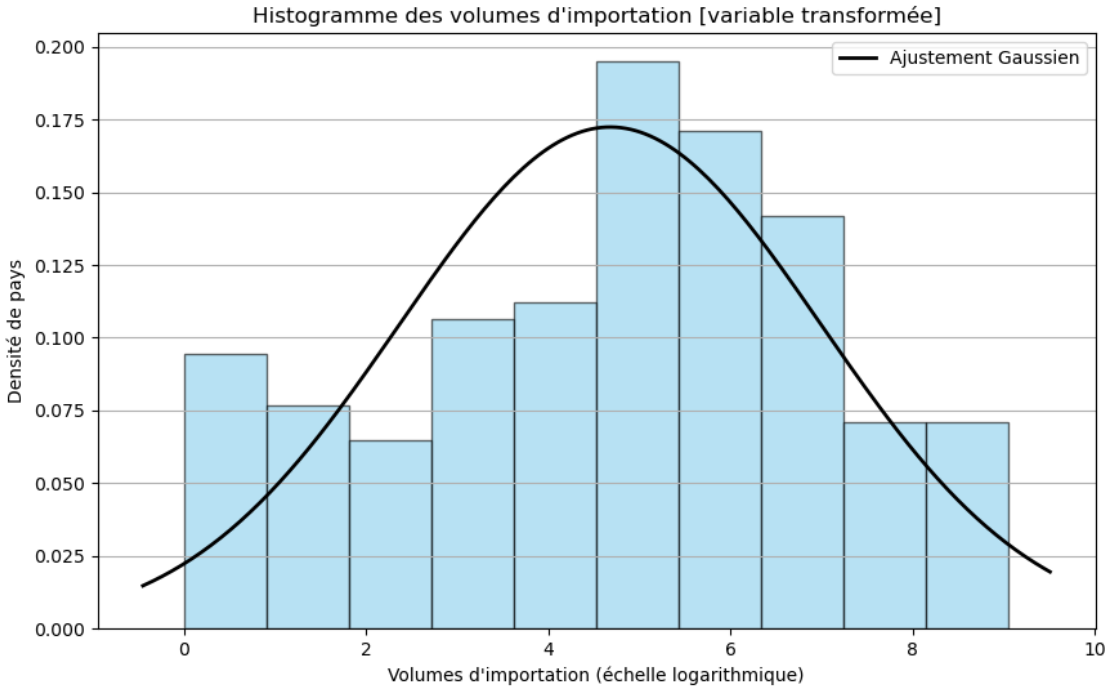



```
Coefficient de Gini pour la répartition des importations : 0.758
```

[178]:
```
df_tmp["imp"] = df_work["imp"].apply(lambda x: np.log(1+x/100))
data = df_tmp["imp"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
```

```python
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des volumes d'importation [variable transformée]")
plt.xlabel("Volumes d'importation (échelle logarithmique)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

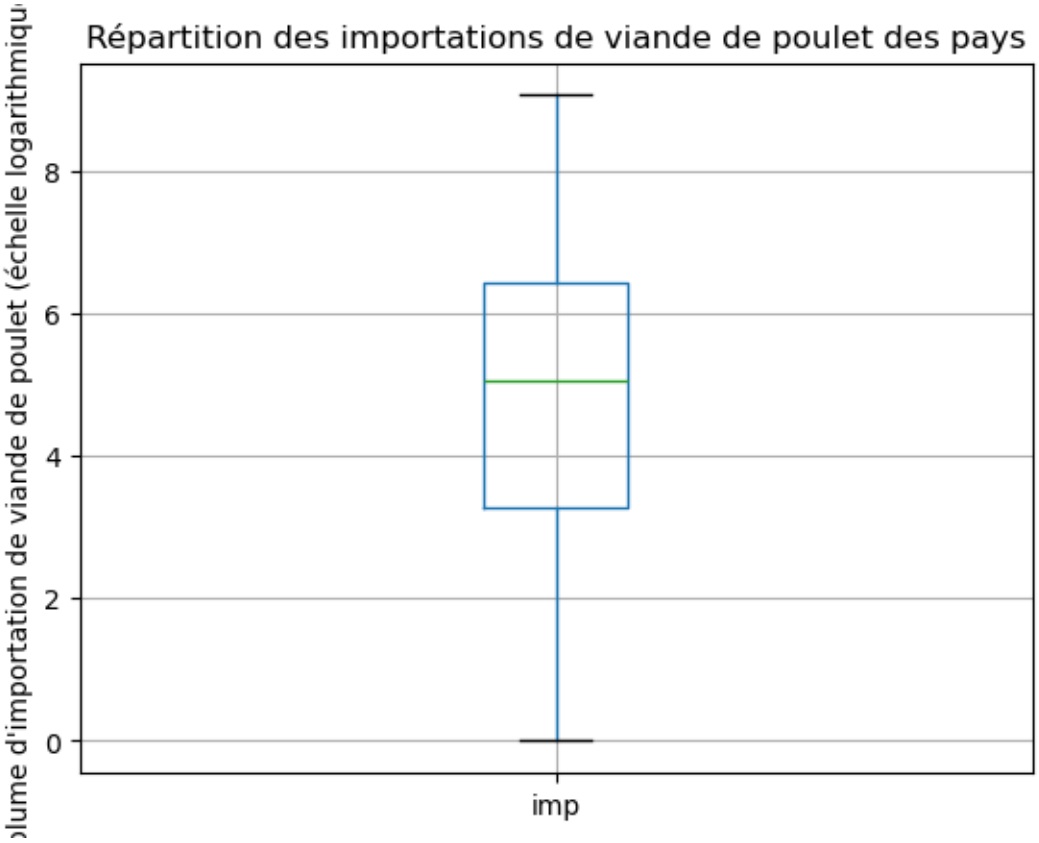
```
Moyenne : 4.68
Écart type : 2.32
Z_min : -2.02
Z_max : 1.89
```



```
Statistique du test de Shapiro-Wilk : 0.9676534222364468
Valeur p : 0.00025529031123804914
```

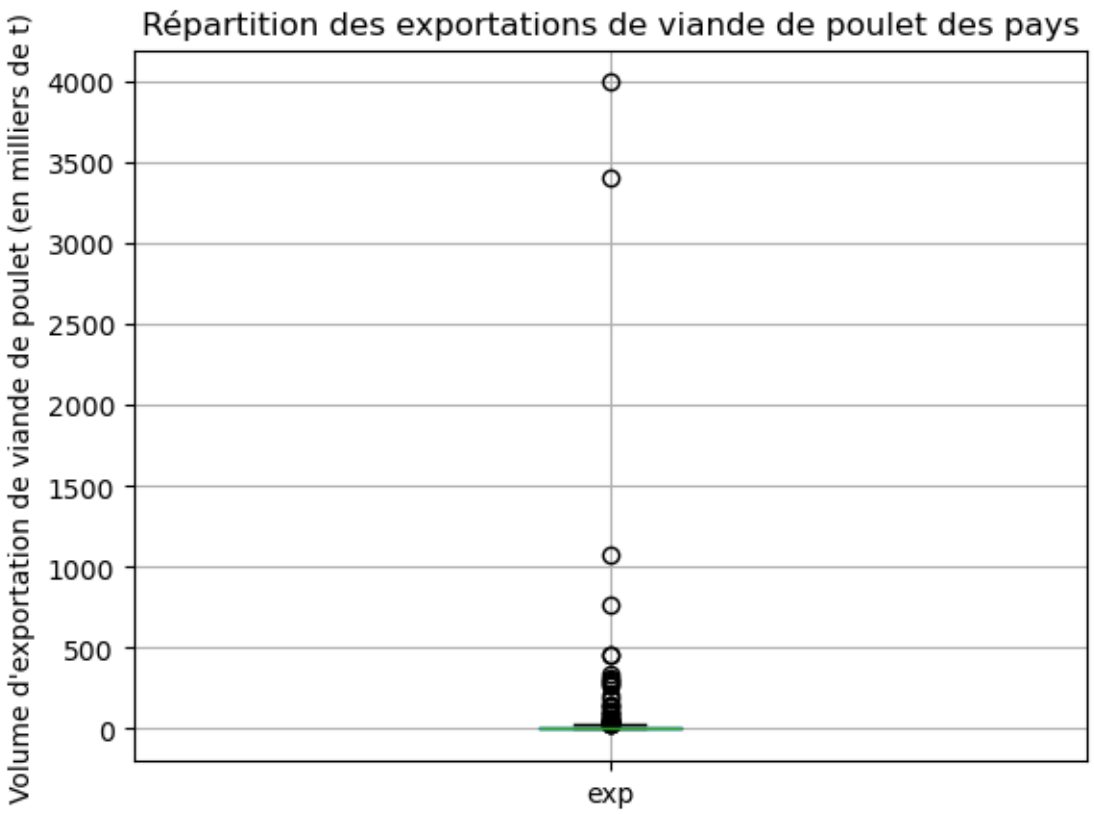
La normalité de la répartition des données s'est considérablement améliorée.

[179]:
```
df_tmp.boxplot(column=["imp"], showfliers=True)
plt.ylabel("Volume d'importation de viande de poulet (échelle logarithmique)")
plt.title("Répartition des importations de viande de poulet des pays")
plt.show()
```



Variable exp (exportations de viande de poulet)

[180]:
```
df_tmp["exp"] = df_work["exp"]/1e3
df_tmp.boxplot(column=["exp"], showfliers=True)
plt.ylabel("Volume d'exportation de viande de poulet (en milliers de t)")
plt.title("Répartition des exportations de viande de poulet des pays")
plt.show()
```



[181]:
```
display((df_tmp.loc[df_tmp["exp"]>47][["ISO3","Zone","exp"]]).
 ↪sort_values("exp",ascending=False))
```

	ISO3	Zone	exp
22	BRA	Brésil	3996.50411
176	USA	États-Unis d'Amérique	3409.22242
123	NLD	Pays-Bas (Royaume des)	1076.73051
134	POL	Pologne	764.39878
11	BEL	Belgique	456.88798
169	TUR	Türkiye	455.82972
69	HKG	Chine - RAS de Hong-Kong	340.02805
174	UKR	Ukraine	314.13514
43	DEU	Allemagne	300.70978
56	FRA	France	289.98212
58	GBR	Royaume-Uni de Grande-Bretagne et d'Irlande du…	274.53115
162	THA	Thaïlande	273.06451
4	ARG	Argentine	209.06485
31	CHN	Chine (continentale)	186.77757
51	ESP	Espagne	149.38738
19	BLR	Bélarus	147.60018
140	RUS	Fédération de Russie	136.60865

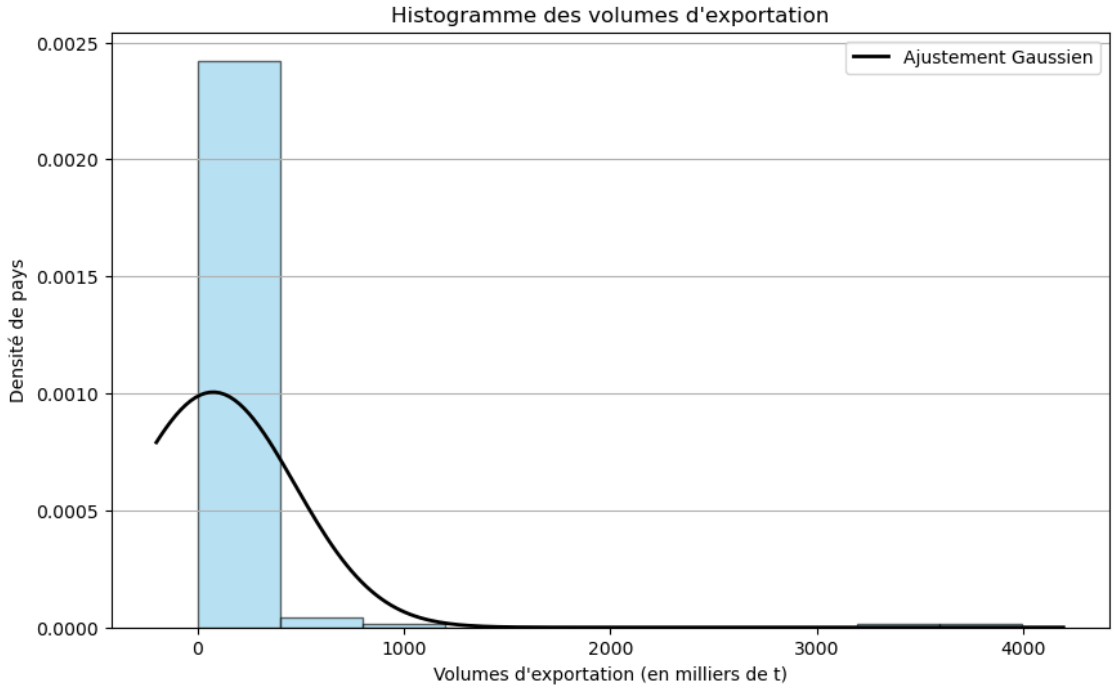
```
73    HUN                                  Hongrie   107.89986
30    CHL                                    Chili   104.24013
28    CAN                                   Canada    93.61389
81    ITA                                   Italie    91.65381
139   ROU                                 Roumanie    66.89674
45    DNK                                 Danemark    64.08048
76    IRL                                  Irlande    54.90718
184   ZAF                           Afrique du Sud    51.24995
3     ARE                      Émirats arabes unis    47.77740
```

Les pays qui exportent le plus de poulet sont en fait les pays les plus développés.

[182]:
```python
data = (df_work["exp"]/1e3).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des volumes d'exportation")
plt.xlabel("Volumes d'exportation (en milliers de t)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')
```

```
# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 75.41
Écart type : 397.86
Z_min : -0.19
Z_max : 9.86
```



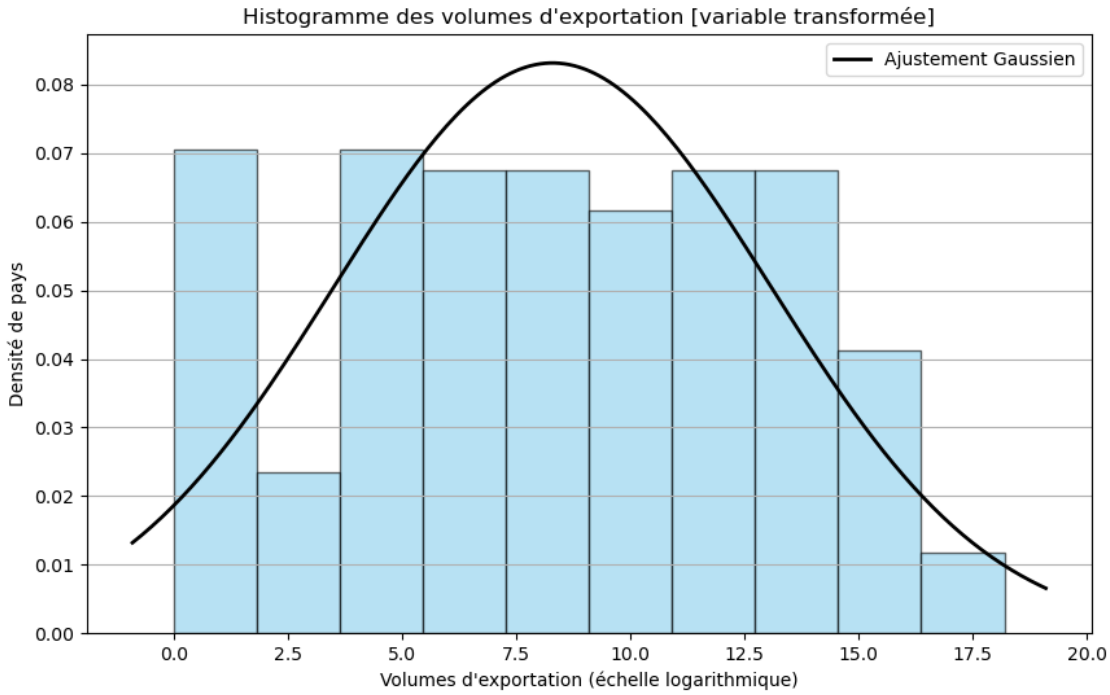

```
Statistique du test de Shapiro-Wilk : 0.1772218473573124
Valeur p : 6.038602374656577e-28
```

Un ajustement logarithmique est nécessaire ! Les données sont complètement écrasées par les outliers hauts (z=9).

[183]:
```
df_tmp["exp"] = df_work["exp"].apply(lambda x: np.log(1+x/0.05))
data = df_tmp["exp"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
```

```
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des volumes d'exportation [variable transformée]")
plt.xlabel("Volumes d'exportation (échelle logarithmique)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

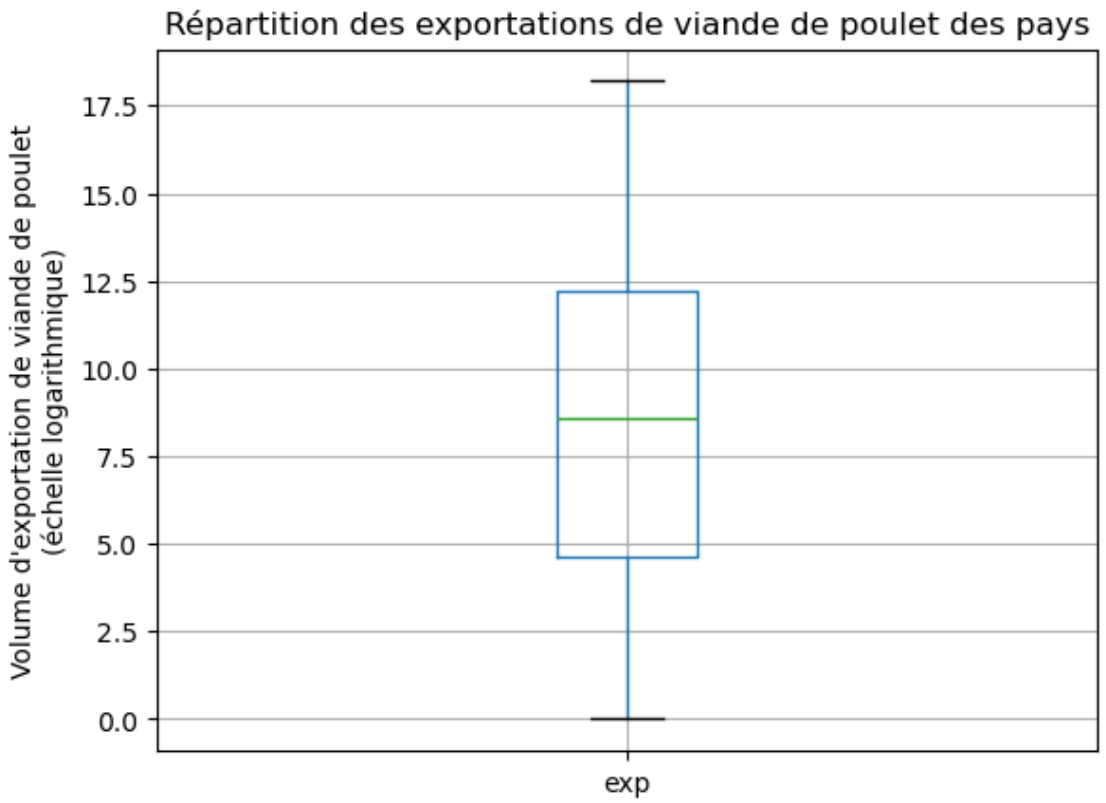
```
Moyenne : 8.3
Écart type : 4.81
Z_min : -1.72
Z_max : 2.06
```




```
Statistique du test de Shapiro-Wilk : 0.9648547622943681
Valeur p : 0.00012346667652636195
```

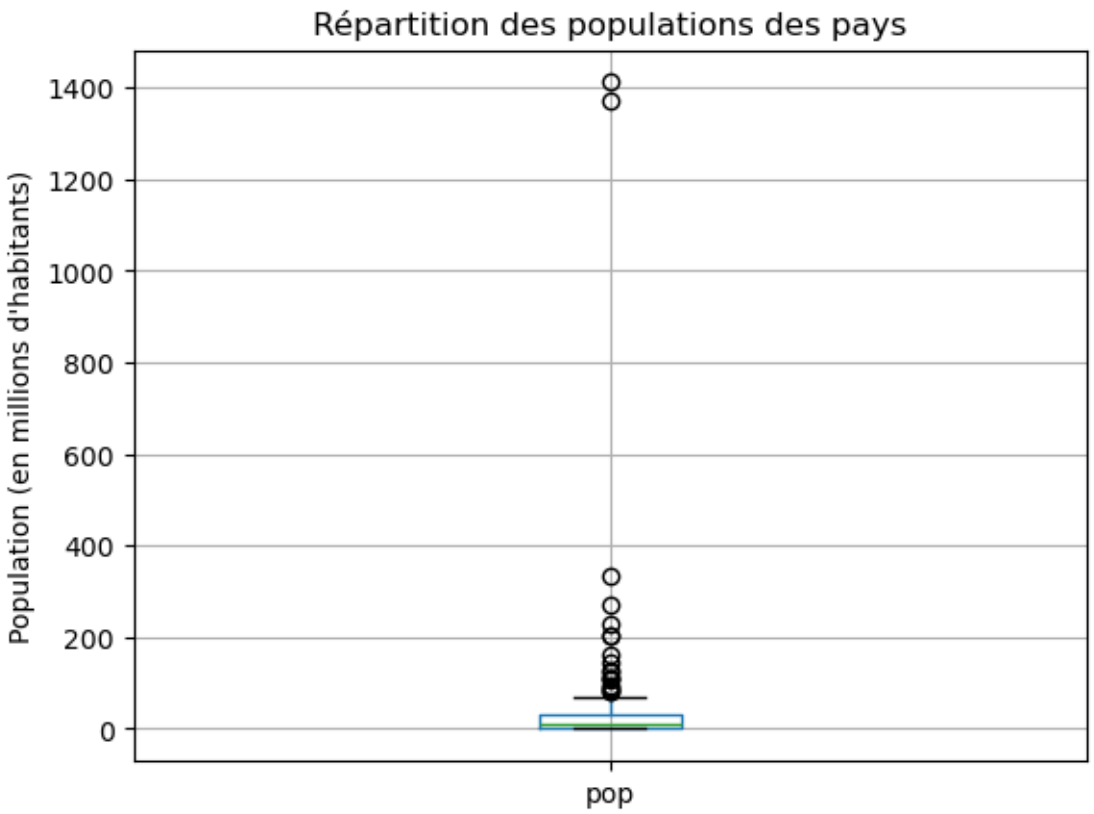
La qualité de la distribution de la variable s'est considérablement améliorée !

[184]:
```
df_tmp.boxplot(column=["exp"], showfliers=True)
plt.ylabel("Volume d'exportation de viande de poulet\n(échelle logarithmique)")
plt.title("Répartition des exportations de viande de poulet des pays")
plt.show()
```



Variable pop (population)

[185]:
```
df_tmp["pop"] = df_work["pop"]/1e6
df_tmp.boxplot(column=["pop"], showfliers=True)
plt.ylabel("Population (en millions d'habitants)")
plt.title("Répartition des populations des pays")
plt.show()
```



[186]:
```
display((df_tmp.loc[df_tmp["pop"]>76][["ISO3","Zone","pop"]]).
  ↪sort_values("pop",ascending=False))
```

	ISO3	Zone	pop
31	CHN	Chine (continentale)	1411.121702
75	IND	Inde	1371.248788
176	USA	États-Unis d'Amérique	333.472024
74	IDN	Indonésie	269.321630
129	PAK	Pakistan	228.132440
22	BRA	Brésil	205.580138
121	NGA	Nigéria	204.663656
14	BGD	Bangladesh	163.647568
140	RUS	Fédération de Russie	145.726719
84	JPN	Japon	126.516002
107	MEX	Mexique	124.362575
53	ETH	Éthiopie	113.030901
132	PHL	Philippines	109.119963
49	EGY	Égypte	105.314691
180	VNM	Viet Nam	95.882842
34	COD	République démocratique du Congo	90.235336
77	IRN	Iran (République islamique d')	85.714237

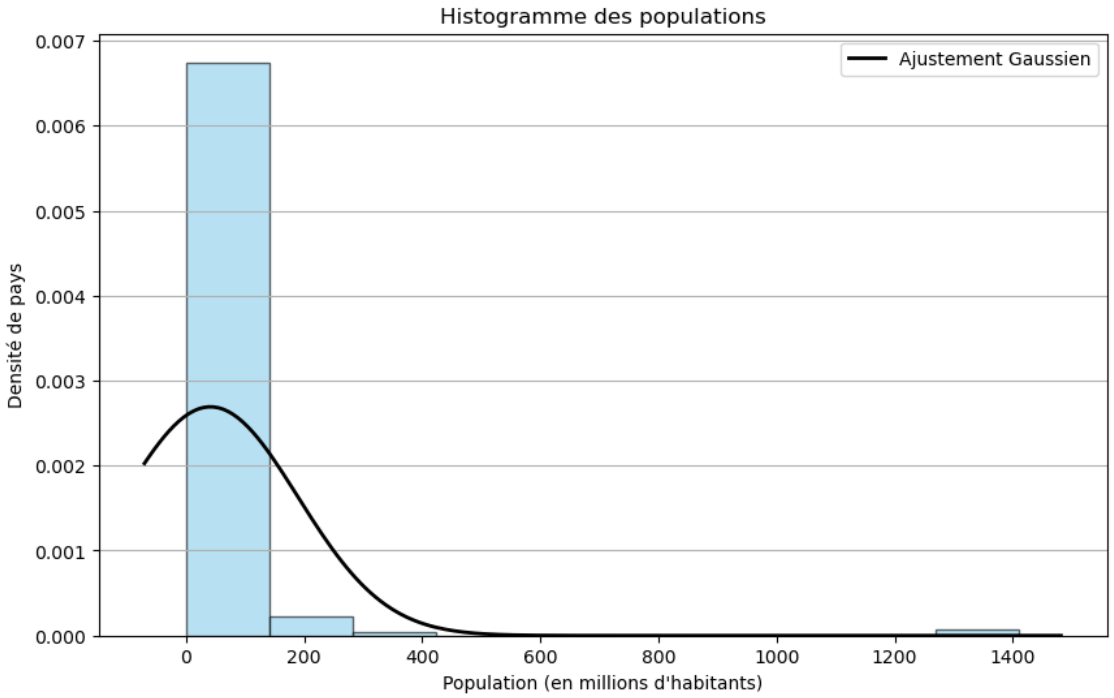
```
169  TUR                          Türkiye    83.279066
43   DEU                        Allemagne    83.022478
```

Les pays les plus puissants au monde sont très souvent très peupelés.

[187]:
```python
data = (df_work["pop"]/1e6).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des populations")
plt.xlabel("Population (en millions d'habitants)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
```

```
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 41.16
Écart type : 148.57
Z_min : -0.28
Z_max : 9.22
```

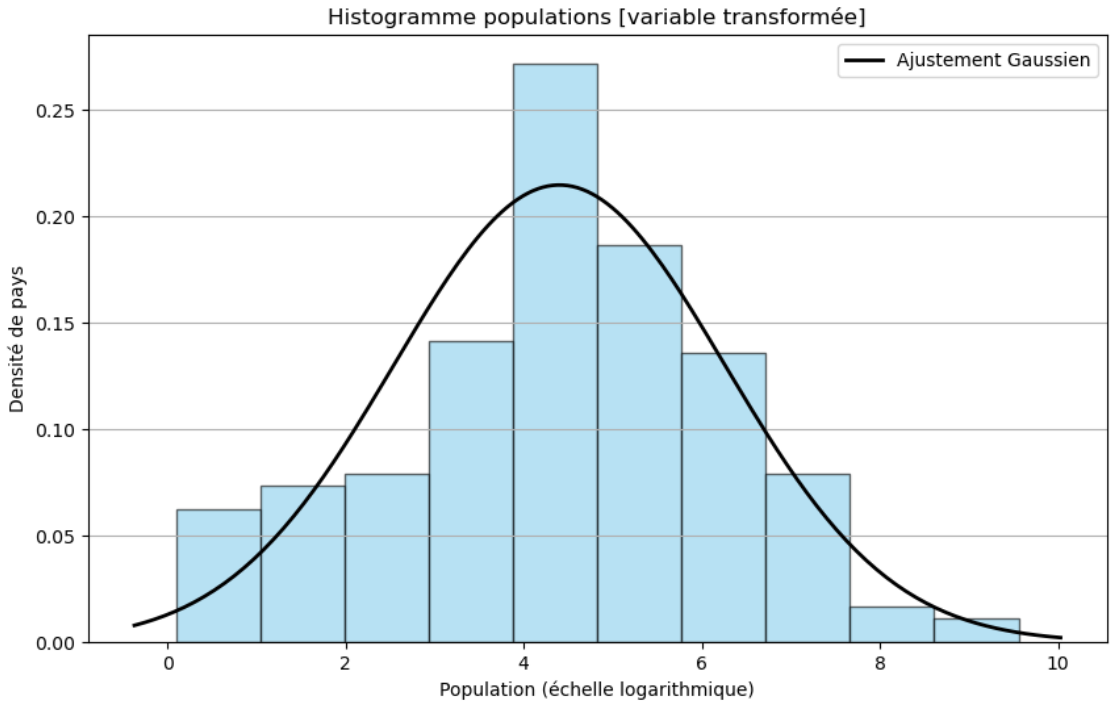



```
Statistique du test de Shapiro-Wilk : 0.2345946092783272
Valeur p : 3.6796027875356016e-27
```

[188]:
```
df_tmp["pop"] = df_work["pop"].apply(lambda x: np.log(1+x/1e5))
data = df_tmp["pop"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
```

```
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme populations [variable transformée]")
plt.xlabel("Population (échelle logarithmique)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

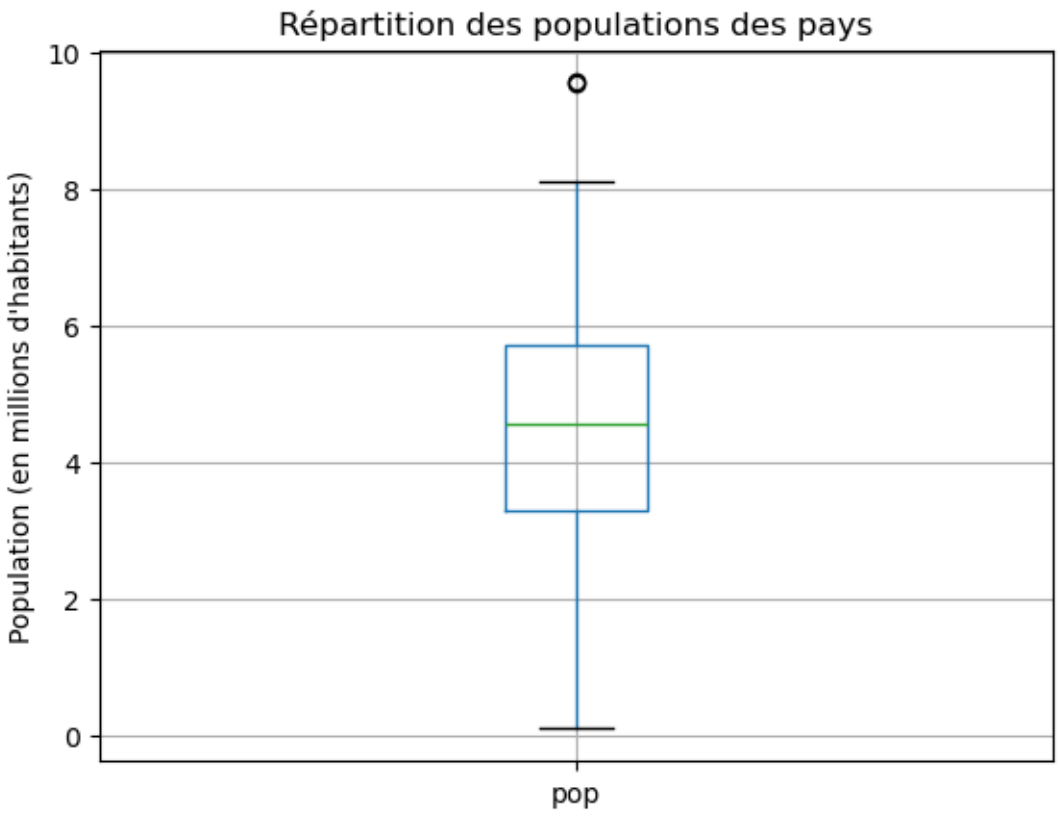
```
Moyenne : 4.4
Écart type : 1.86
Z_min : -2.31
Z_max : 2.77
```




```
Statistique du test de Shapiro-Wilk : 0.9873436317568136
Valeur p : 0.09283482493766287
```

la transformation est spectaculaire : on n'est pas si loin d'une distribution normale.

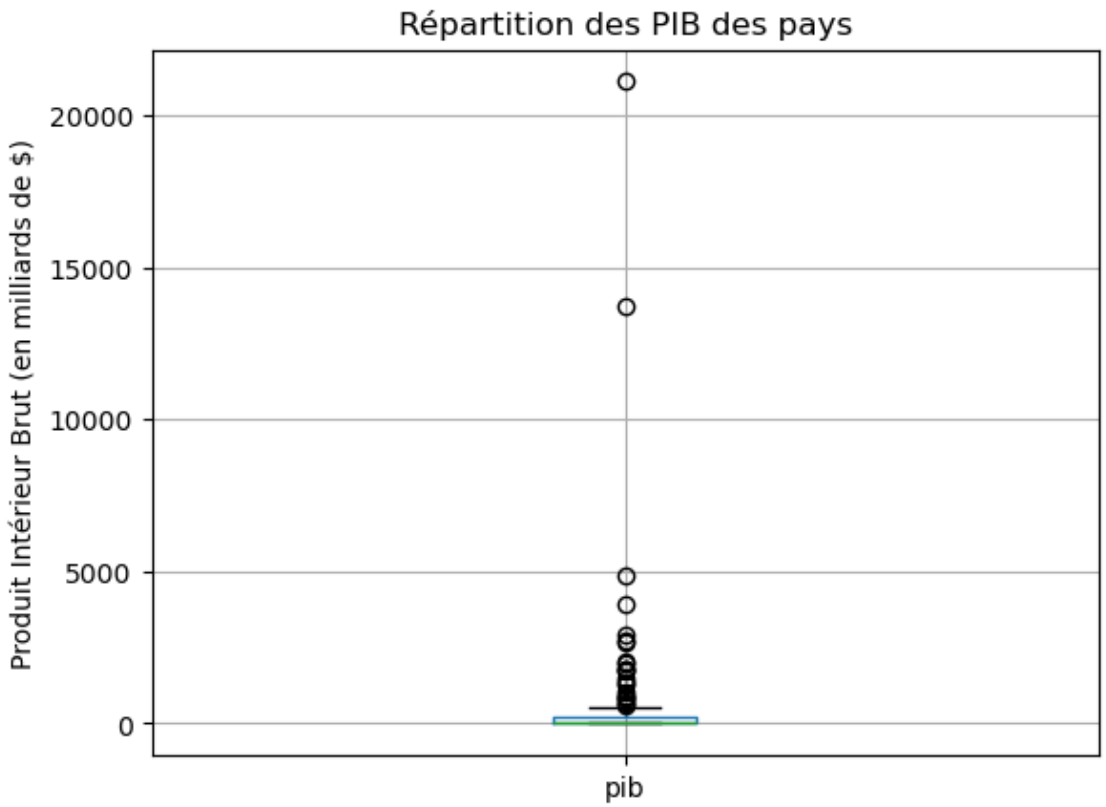
[189]:
```
df_tmp.boxplot(column=["pop"], showfliers=True)
plt.ylabel("Population (en millions d'habitants)")
plt.title("Répartition des populations des pays")
plt.show()
```



On a 2 outliers que sont la Chine et l'Inde.

Variable pib

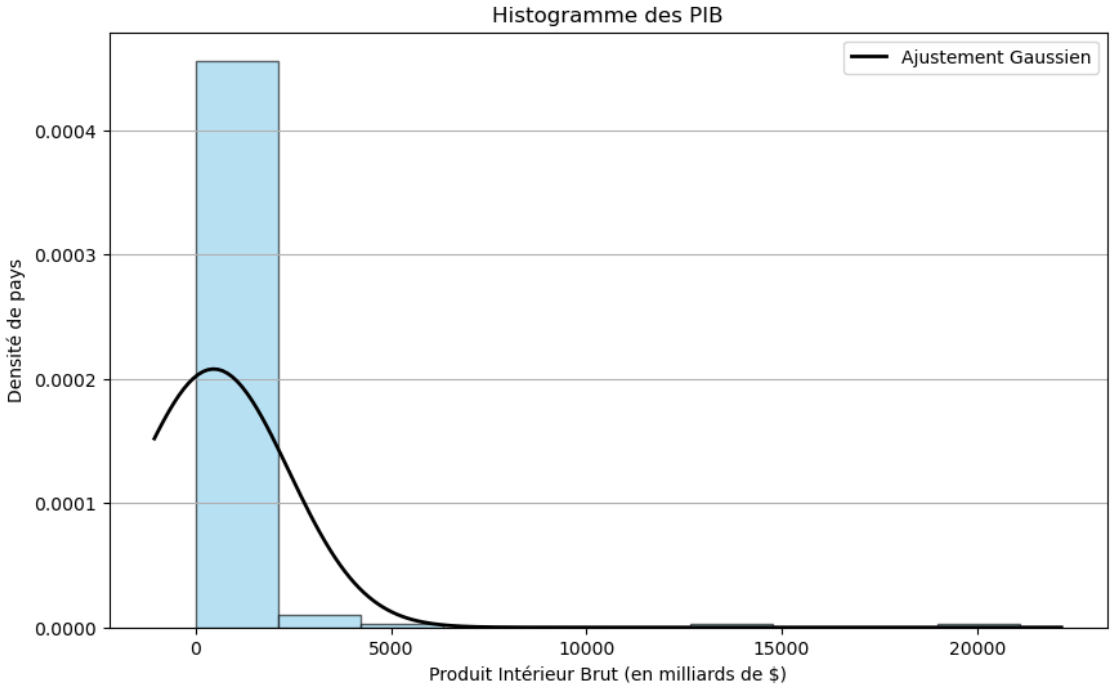
[190]:
```
df_tmp["pib"] = df_work["pib"]/1e9
df_tmp.boxplot(column=["pib"], showfliers=True)
plt.ylabel("Produit Intérieur Brut (en milliards de $)")
plt.title("Répartition des PIB des pays")
plt.show()
```



[191]:
```
data = (df_work["pib"]/1e9).sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
```

```python
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des PIB")
plt.xlabel("Produit Intérieur Brut (en milliards de $)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

```
Moyenne : 464.92
Écart type : 1923.89
Z_min : -0.24
Z_max : 10.73
```

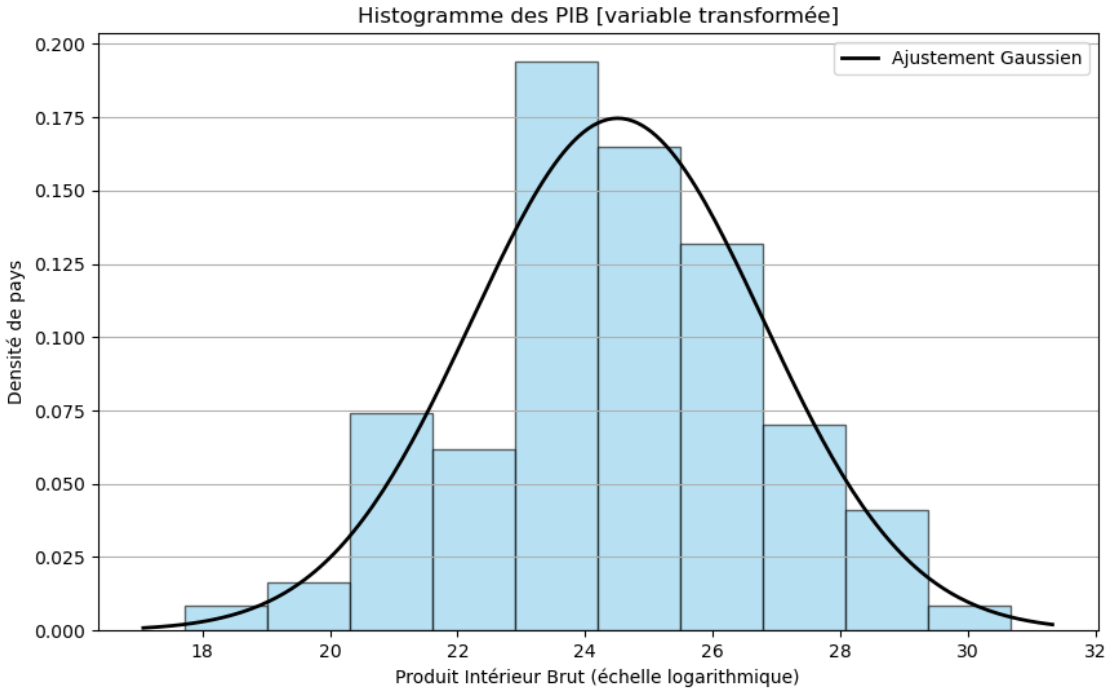



```
Statistique du test de Shapiro-Wilk : 0.22258511049428475
Valeur p : 2.498902610924662e-27
```

[192]:
```
df_tmp["pib"] = df_work["pib"].apply(lambda x: np.log(1+x))
data = df_tmp["pib"].sort_values()

# Estimer la moyenne et l'écart type
mn = data.mean()
sd = data.std(ddof=1)    # Estimateur sans biais

# Afficher les valeurs calculées
print(f"Moyenne : {round(mn,2)}")
print(f"Écart type : {round(sd,2)}")
print(f"Z_min : {round((data.min()-mn)/sd,2)}")
print(f"Z_max : {round((data.max()-mn)/sd,2)}")

# Tracer l'histogramme avec une courbe gaussienne
plt.figure(figsize=(10, 6))

# Histogramme
#plt.hist(data, bins=np.linspace(15,100,18), density=True, color='skyblue',␣
 ↪edgecolor='black', alpha=0.6)
plt.hist(data, density=True, color='skyblue', edgecolor='black', alpha=0.6)

# Ajustement gaussien
```

```
mu, std = st.norm.fit(data)
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, data.count())
p = st.norm.pdf(x, mu, std)

# Tracer la courbe gaussienne
plt.plot(x, p, 'k', linewidth=2, label='Ajustement Gaussien')

# Ajouter les titres et légendes en français
plt.title("Histogramme des PIB [variable transformée]")
plt.xlabel("Produit Intérieur Brut (échelle logarithmique)")
plt.ylabel('Densité de pays')
plt.legend()
plt.grid(axis='y')

# Afficher le graphique
plt.show()

# Test de Shapiro-Wilk
stat, p_value = st.shapiro(data)

# Afficher les résultats
print(f"Statistique du test de Shapiro-Wilk : {stat}")
print(f"Valeur p : {p_value}")
```

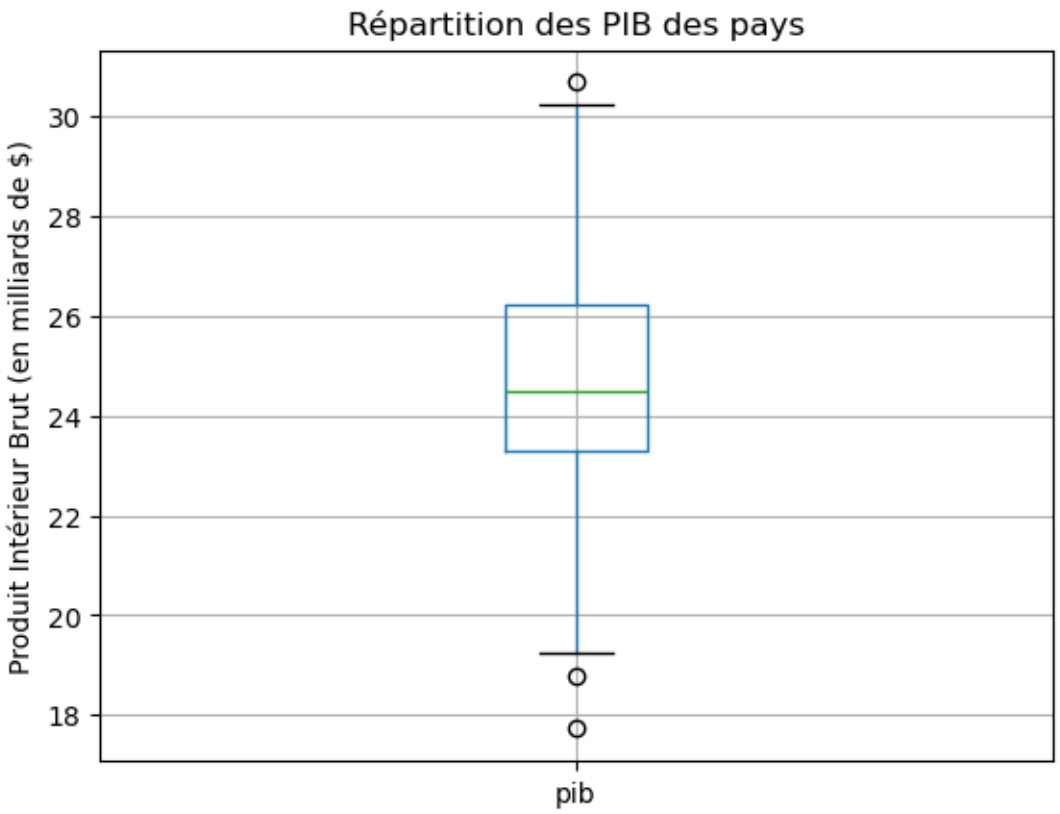
```
Moyenne : 24.51
Écart type : 2.29
Z_min : -2.97
Z_max : 2.69
```




```
Statistique du test de Shapiro-Wilk : 0.9951624872418018
Valeur p : 0.8110657559402463
```

la transformation est spectaculaire : on peut même accepter l'hypothèse avec un fort niveau de confiance que les logarithmes des PIB des pays du monde suivent une loi normale.

[193]:
```
df_tmp.boxplot(column=["pib"], showfliers=True)
plt.ylabel("Produit Intérieur Brut (en milliards de $)")
plt.title("Répartition des PIB des pays")
plt.show()
```



5.2 - Sélection des pays pour analyse

[194]:
```
set_pays = set(df_work["ISO3"].unique())
sr_pays_sel = pd.Series(list(set_pays - outzones), name="ISO3")
df_sel = df_work.merge(sr_pays_sel, how='inner', on="ISO3")
display(df_sel)
```

```
    ISO3                 Zone          pop           pib        exp  \
0    AFG          Afghanistan  36758062.91  1.812068e+10      52.98
1    AGO               Angola  31358489.00  1.031760e+11      72.78
2    ALB              Albanie   2876923.27  1.525616e+10       4.91
3    ARE  Émirats arabes unis   9259546.36  4.155386e+11   47777.40
4    ARG            Argentine  44395603.91  5.591068e+11  209064.85
..   …                    …           …             …        …
182  WSM                Samoa    207637.27  8.705741e+08      34.87
183  YEM                Yémen  34180357.18  1.888517e+10      15.84
184  ZAF       Afrique du Sud  58886231.00  3.789851e+11   51249.95
185  ZMB               Zambie  18006950.64  2.428533e+10    3800.64
186  ZWE             Zimbabwe  15097522.91  2.272259e+10       0.00

           imp        prod    cc    pv    rq  …   pib/hab  dispo/hab  \
0     30146.73    27156.60 -1.38 -2.60 -1.23  …    499.65       1.59
```

```
1    265843.72    44621.98 -1.11 -0.45 -0.85  …   3393.95      10.06
2     24500.61    14010.37 -0.53  0.24  0.22  …   5315.70      13.42
3    536015.67    51563.00  1.13  0.68  0.99  …  44935.94      58.23
4      4318.53  2137418.79 -0.34  0.00 -0.58  …  12614.01      43.46
..         …         …   …   …   …  …       …        …
182   15739.14      395.99  0.52  1.12 -0.20  …   4191.05      77.46
183  100922.62   183741.98 -1.61 -2.68 -1.53  …    580.50       8.35
184  384106.89  1783399.45 -0.12 -0.34  0.05  …   6444.51      35.95
185   17087.18    48216.33 -0.53  0.11 -0.55  …   1359.56       3.40
186    5285.46    93027.04 -1.31 -0.80 -1.57  …   1499.38       6.48

     tc_pop  tc_urb  tc_pib  tc_pib/hab  tc_dispo  tc_dispo/hab    prix  \
0      5.43    6.82   -4.57       -9.89    -10.25        -15.73  1270.0
1      6.76    8.55  -11.12      -18.20     -2.35         -9.35  2693.0
2     -0.63    3.28   12.20       12.91     11.34         12.14   880.9
3      5.18    3.21    4.58       -1.01      5.93          1.29  1582.0
4      1.36    2.13   -1.91       -3.35      6.00          4.69  1611.0
..      …     …      …          …         …            …      …
182    1.84   -0.78    2.86        1.02      4.03          2.25   941.5
183    5.87    8.00  -27.96      -33.07      4.59         -1.56  1371.0
184    2.86    3.89    1.16       -1.75      2.47         -0.33   752.4
185    5.92    8.41    0.81       -5.50      9.67          3.83  1416.0
186    3.08    4.40    7.95        4.80      9.03          6.02   657.0

        dist
0     5590.4
1     6510.3
2     1603.5
3     5249.5
4    11072.2
..       …
182  16011.9
183   5317.3
184   9353.6
185   7604.8
186   7949.7

[187 rows x 22 columns]
```

[195]:
```
df_sel = df_sel[["ISO3","Zone","pop","tc_pop","%urb","pib",\
                 "tc_pib","pib/hab","imp","exp","dispo/hab",\
                 "tc_dispo/hab","prix","pv","rq","dist"]].copy()
display(df_sel)
```

```
    ISO3                Zone          pop  tc_pop   %urb           pib  \
0    AFG         Afghanistan  36758062.91    5.43  25.26  1.812068e+10
1    AGO              Angola  31358489.00    6.76  64.48  1.031760e+11
2    ALB             Albanie   2876923.27   -0.63  61.43  1.525616e+10
```

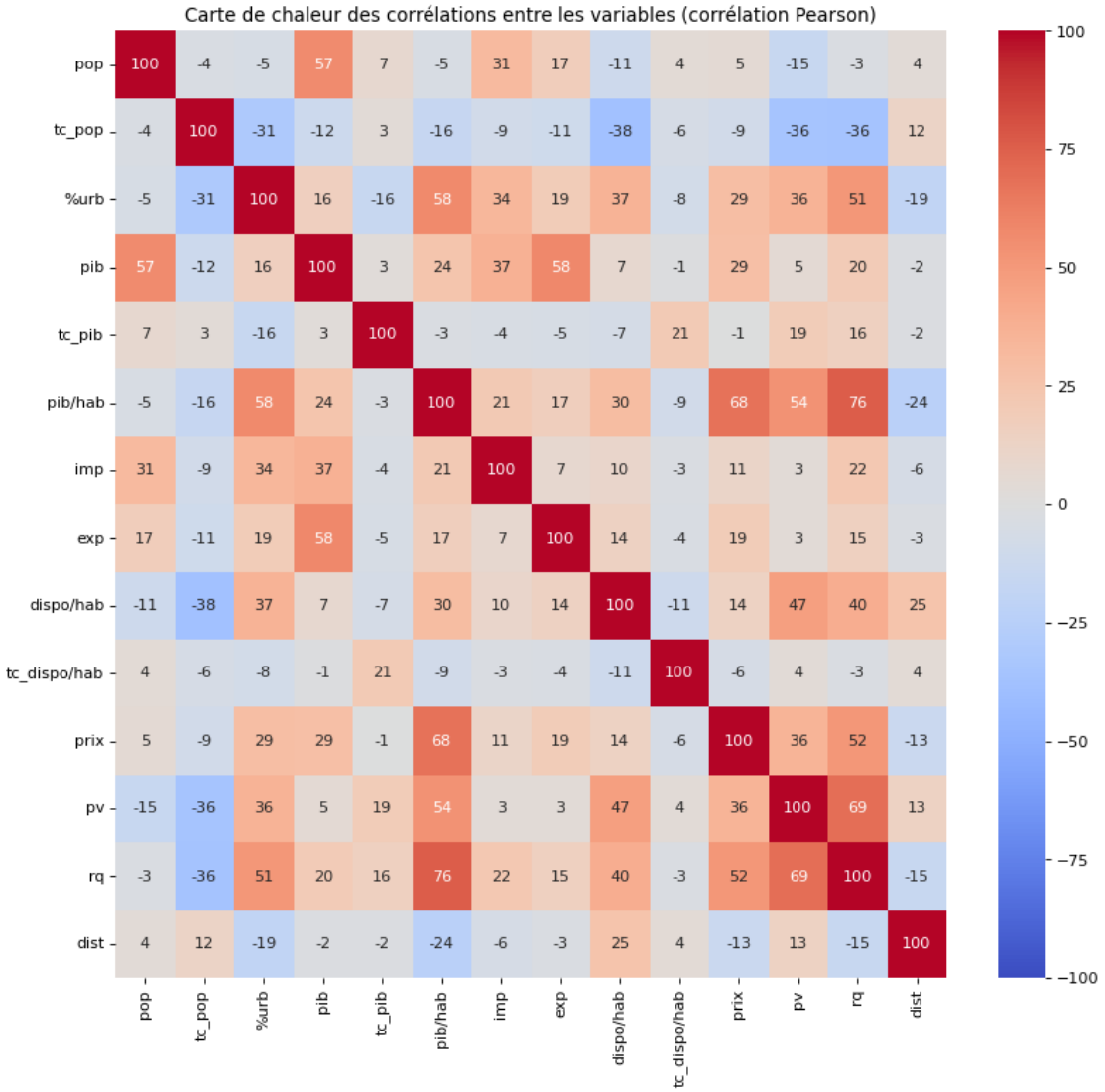
```
3    ARE  Émirats arabes unis   9259546.36    5.18  89.57  4.155386e+11
4    ARG            Argentine  44395603.91    1.36  92.41  5.591068e+11
..   …                    …          …     …     …          …
182  WSM                Samoa    207637.27    1.84  17.45  8.705741e+08
183  YEM                Yémen  34180357.18    5.87  31.04  1.888517e+10
184  ZAF       Afrique du Sud  58886231.00    2.86  64.58  3.789851e+11
185  ZMB               Zambie  18006950.64    5.92  42.75  2.428533e+10
186  ZWE             Zimbabwe  15097522.91    3.08  36.22  2.272259e+10

     tc_pib   pib/hab        imp        exp  dispo/hab  tc_dispo/hab    prix  \
0     -4.57    499.65   30146.73      52.98       1.59        -15.73  1270.0
1    -11.12   3393.95  265843.72      72.78      10.06         -9.35  2693.0
2     12.20   5315.70   24500.61       4.91      13.42         12.14   880.9
3      4.58  44935.94  536015.67   47777.40      58.23          1.29  1582.0
4     -1.91  12614.01    4318.53  209064.85      43.46          4.69  1611.0
..      …       …          …         …         …              …     …
182    2.86   4191.05   15739.14      34.87      77.46          2.25   941.5
183  -27.96    580.50  100922.62      15.84       8.35         -1.56  1371.0
184    1.16   6444.51  384106.89   51249.95      35.95         -0.33   752.4
185    0.81   1359.56   17087.18    3800.64       3.40          3.83  1416.0
186    7.95   1499.38    5285.46       0.00       6.48          6.02   657.0

       pv    rq     dist
0   -2.60 -1.23   5590.4
1   -0.45 -0.85   6510.3
2    0.24  0.22   1603.5
3    0.68  0.99   5249.5
4    0.00 -0.58  11072.2
..    …   …       …
182  1.12 -0.20  16011.9
183 -2.68 -1.53   5317.3
184 -0.34  0.05   9353.6
185  0.11 -0.55   7604.8
186 -0.80 -1.57   7949.7

[187 rows x 16 columns]
```

[196]:
```
#Le nombre de valeurs présentes dans chacune des colonnes
for c in list(df_sel):
    print("\nColonne", c, "- Nombre de valeurs NaN :", ((df_sel[c]).isna()).
 ↪sum())
    print("Colonne", c, "- Nombre de valeurs non-vides :", df_sel.shape[0] -␣
 ↪((df_sel[c]).isna()).sum())

print("\nNombre de zones dans les données :", (df_sel["ISO3"]).nunique())
```

```
Colonne ISO3 - Nombre de valeurs NaN : 0
```

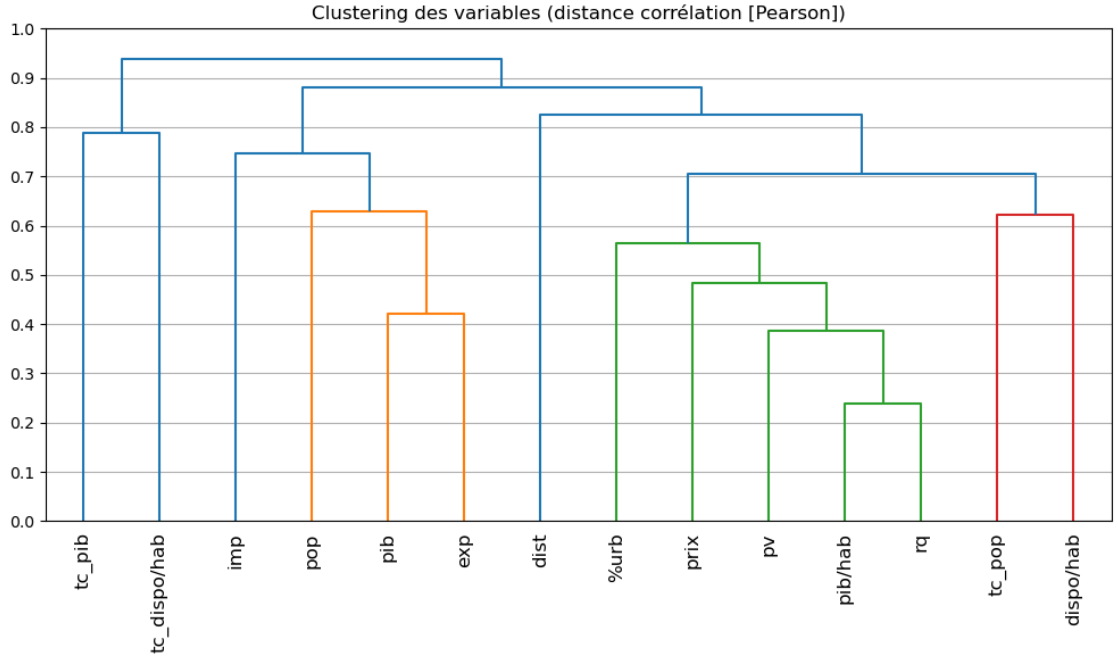
```
Colonne ISO3 - Nombre de valeurs non-vides : 187

Colonne Zone - Nombre de valeurs NaN : 0
Colonne Zone - Nombre de valeurs non-vides : 187

Colonne pop - Nombre de valeurs NaN : 0
Colonne pop - Nombre de valeurs non-vides : 187

Colonne tc_pop - Nombre de valeurs NaN : 0
Colonne tc_pop - Nombre de valeurs non-vides : 187

Colonne %urb - Nombre de valeurs NaN : 0
Colonne %urb - Nombre de valeurs non-vides : 187

Colonne pib - Nombre de valeurs NaN : 0
Colonne pib - Nombre de valeurs non-vides : 187

Colonne tc_pib - Nombre de valeurs NaN : 0
Colonne tc_pib - Nombre de valeurs non-vides : 187

Colonne pib/hab - Nombre de valeurs NaN : 0
Colonne pib/hab - Nombre de valeurs non-vides : 187

Colonne imp - Nombre de valeurs NaN : 0
Colonne imp - Nombre de valeurs non-vides : 187

Colonne exp - Nombre de valeurs NaN : 0
Colonne exp - Nombre de valeurs non-vides : 187

Colonne dispo/hab - Nombre de valeurs NaN : 0
Colonne dispo/hab - Nombre de valeurs non-vides : 187

Colonne tc_dispo/hab - Nombre de valeurs NaN : 0
Colonne tc_dispo/hab - Nombre de valeurs non-vides : 187

Colonne prix - Nombre de valeurs NaN : 0
Colonne prix - Nombre de valeurs non-vides : 187

Colonne pv - Nombre de valeurs NaN : 0
Colonne pv - Nombre de valeurs non-vides : 187

Colonne rq - Nombre de valeurs NaN : 0
Colonne rq - Nombre de valeurs non-vides : 187

Colonne dist - Nombre de valeurs NaN : 0
Colonne dist - Nombre de valeurs non-vides : 187

Nombre de zones dans les données : 187
```

[197]:
```
fig = plt.figure(figsize=(12,11), dpi=80)

sb.heatmap(100*df_sel[list(df_sel.columns)[2:]].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()
```

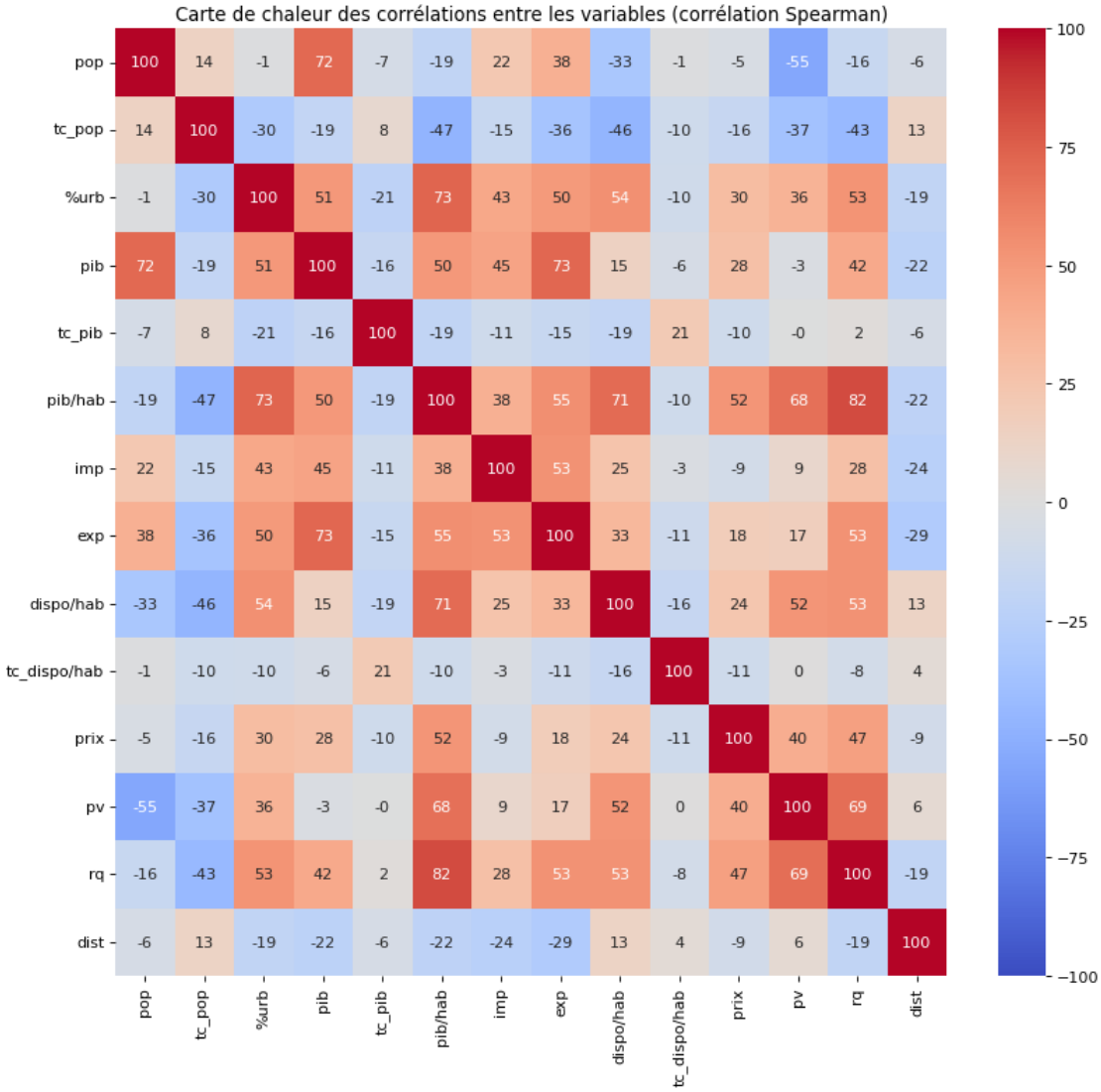


[198]:
```
vcorr = (df_sel[list(df_sel.columns)[2:]].corr(method='pearson')).abs()

# 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_sel.columns)[2:], 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_variableslog_Pea.png")
plt.show()
```

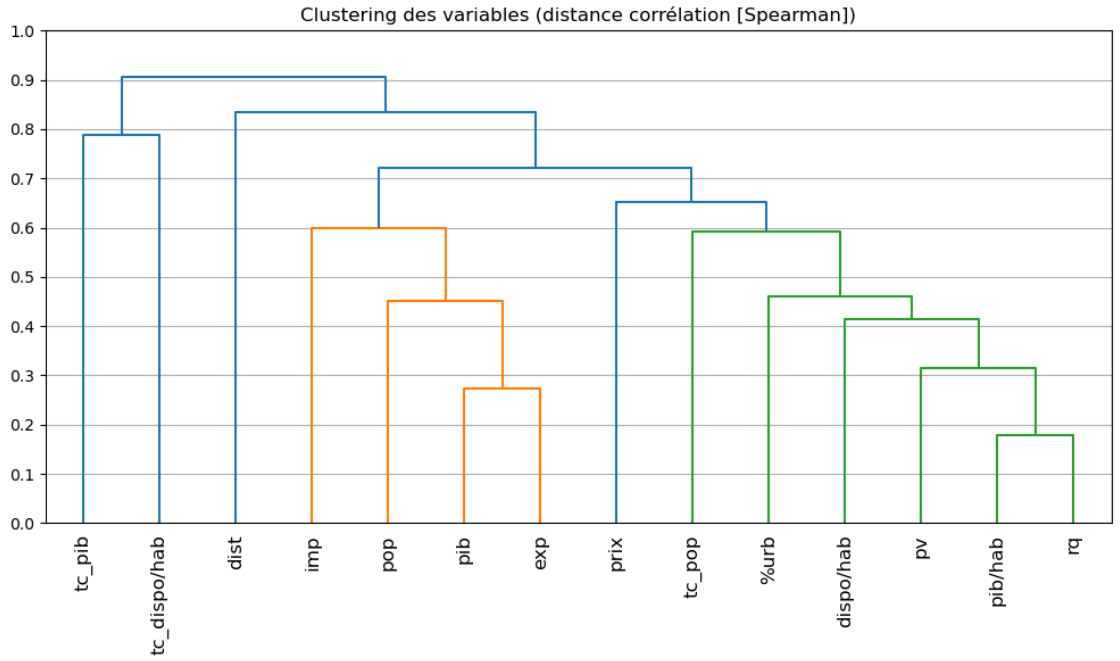


[199]:
```
fig = plt.figure(figsize=(12,11), dpi=80)

sb.heatmap(100*df_sel[list(df_sel.columns)[2:]].corr(method='spearman'),\
           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␣
 ↪Spearman)")
#plt.savefig("Heatmap.png")
plt.show()
```



[200]:
```
vcorr = (df_sel[list(df_sel.columns)[2:]].corr(method='spearman')).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_sel.columns)[2:], leaf_rotation=90)
plt.yticks(np.linspace(0,1,11))
```

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



Partie 6 - Génération des fichiers

[201]:
```
# Données sans feature engineering pour tous les pays (avec valeurs manquantes)
df_data_viz.to_csv("world_data.csv", encoding='utf-8')
# Séries temporelles avec feature engineering pour le maximum de pays sans␣
 ↪valeur manquante
df_data.to_csv("time_data.csv", encoding='utf-8')
# Données complètes avec feature engineering pour les pays cibles (187 pays)
df_sel.to_csv("work_data.csv", encoding='utf-8')
```

[ ]:
```
```