Marchand_Emmanuel_1_notebook_032024

March 24, 2024

PROJET 4 DATA ANALYST

Réalisez une étude de santé publique avec R ou Python

1 OBJECTIF DE CE NOTEBOOK

Bienvenue dans l'outil plébiscité par les analystes de données Jupyter.

Il s'agit d'un outil permettant de mixer et d'alterner codes, textes et graphique.

Cet outil est formidable pour plusieurs raisons:

- il permet de tester des lignes de codes au fur et à mesure de votre rédaction, de constater immédiatement le résultat d'un instruction, de la corriger si nécessaire.
- De rédiger du texte pour expliquer l'approche suivie ou les résultats d'une analyse et de le mettre en forme grâce à du code html ou plus simple avec **Markdown**
- d'agrémenter de graphiques

Pour vous aider dans vos premiers pas à l'usage de Jupyter et de Python, nous avons rédigé ce notebook en vous indiquant les instructions à suivre.

Il vous suffit pour cela de saisir le code Python répondant à l'instruction donnée.

Vous verrez de temps à autre le code Python répondant à une instruction donnée mais cela est fait pour vous aider à comprendre la nature du travail qui vous est demandée.

Et garder à l'esprit, qu'il n'y a pas de solution unique pour résoudre un problème et qu'il y a autant de résolutions de problèmes que de développeurs ;)…

Note jeremy Est ce qu'il faut faire le calcul de la sous nutrition sur les pays qu'on a ? Est ce qu'il faut faire des graphiques ? Rajouter le soja La liste des céréales est difficile a trouver …

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

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

1.2 - Chargement des fichiers Excel

[2]:
```
#Importation du fichier population.csv
population = pd.read_csv('population.csv')
```

```
#Importation du fichier dispo_alimentaire.csv
dispo_alimentaire = pd.read_csv('dispo_alimentaire.csv')

#Importation du fichier aide_alimentaire.csv
aide_alimentaire = pd.read_csv('aide_alimentaire.csv')

#Importation du fichier sous_nutrition.csv
sous_nutrition = pd.read_csv('sous_nutrition.csv')
```

Etape 2 - Analyse exploratoire des fichiers

2.1 - Analyse exploratoire du fichier population

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

```
Le tableau comporte 1416 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

[5]:
```
#Consulter le nombre de colonnes
print("Nombre de colonnes dans le fichier population : ", population.shape[1])

#La nature des données dans chacune des colonnes
print(" ")
print("La nature des données du fichier popuation est la suivante :")
print(population.dtypes)

#Le nombre de valeurs présentes dans chacune des colonnes
print(" ")
print("Le nombre de lignes non nulles de chaque colonne est : ")
population.count()
```

```
Nombre de colonnes dans le fichier population :  3

La nature des données du fichier popuation est la suivante :
Zone       object
Année       int64
Valeur    float64
dtype: object

Le nombre de lignes non nulles de chaque colonne est :
```

[5]:
```
Zone      1416
Année     1416
Valeur    1416
dtype: int64
```

[6]:
```
#Affichage les 5 premières lignes de la table
display(population.head())
```

```
          Zone  Année     Valeur
0  Afghanistan   2013  32269.589
1  Afghanistan   2014  33370.794
2  Afghanistan   2015  34413.603
3  Afghanistan   2016  35383.032
4  Afghanistan   2017  36296.113
```

[7]:
```
#Nous allons harmoniser les unités. Pour cela, nous avons décidé de multiplier
 ↪la population par 1000
#Multiplication de la colonne valeur par 1000
population['Valeur'] = population['Valeur'] * 1000
```

[8]:
```
#changement du nom de la colonne Valeur par Population
population.rename(columns={'Valeur':'Population'}, inplace=True)
```

[9]:
```
#Affichage les 5 premières lignes de la table pour voir les modifications
population.head()
```

[9]:
```
          Zone  Année  Population
0  Afghanistan   2013  32269589.0
1  Afghanistan   2014  33370794.0
2  Afghanistan   2015  34413603.0
3  Afghanistan   2016  35383032.0
4  Afghanistan   2017  36296113.0
```

2.2 - Analyse exploratoire du fichier disponibilité alimentaire

[10]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".
 ↪format(dispo_alimentaire.shape[0]))
print("Le tableau comporte {} colonne(s)".format(dispo_alimentaire.shape[1]))
```

```
Le tableau comporte 15605 observation(s) ou article(s)
Le tableau comporte 18 colonne(s)
```

[11]:
```
#Consulter le nombre de colonnes
print(dispo_alimentaire.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15605 entries, 0 to 15604
Data columns (total 18 columns):
 #   Column                                                    Non-Null
Count  Dtype
---  ------
--------------  -----
 0   Zone                                                      15605 non-
```

```
null  object
 1   Produit                                                 15605 non-
null  object
 2   Origine                                                 15605 non-
null  object
 3   Aliments pour animaux                                   2720 non-
null   float64
 4   Autres Utilisations                                     5496 non-
null   float64
 5   Disponibilité alimentaire (Kcal/personne/jour)          14241 non-
null  float64
 6   Disponibilité alimentaire en quantité (kg/personne/an)  14015 non-
null  float64
 7   Disponibilité de matière grasse en quantité (g/personne/jour)  11794 non-
null  float64
 8   Disponibilité de protéines en quantité (g/personne/jour)  11561 non-
null  float64
 9   Disponibilité intérieure                                15382 non-
null  float64
 10  Exportations - Quantité                                 12226 non-
null  float64
 11  Importations - Quantité                                 14852 non-
null  float64
 12  Nourriture                                              14015 non-
null  float64
 13  Pertes                                                  4278 non-
null   float64
 14  Production                                              9180 non-
null   float64
 15  Semences                                                2091 non-
null   float64
 16  Traitement                                              2292 non-
null   float64
 17  Variation de stock                                      6776 non-
null   float64
dtypes: float64(15), object(3)
memory usage: 2.1+ MB
None
```

[12]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

[12]:

	Zone	Produit	Origine	Aliments pour animaux	\
0	Afghanistan	Abats Comestible	animale	NaN	
1	Afghanistan	Agrumes, Autres	vegetale	NaN	
2	Afghanistan	Aliments pour enfants	vegetale	NaN	
3	Afghanistan	Ananas	vegetale	NaN	

```
4  Afghanistan                Bananes  vegetale                      NaN

   Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
0                  NaN                                             5.0
1                  NaN                                             1.0
2                  NaN                                             1.0
3                  NaN                                             0.0
4                  NaN                                             4.0

   Disponibilité alimentaire en quantité (kg/personne/an)  \
0                                               1.72
1                                               1.29
2                                               0.06
3                                               0.00
4                                               2.70

   Disponibilité de matière grasse en quantité (g/personne/jour)  \
0                                               0.20
1                                               0.01
2                                               0.01
3                                                NaN
4                                               0.02

   Disponibilité de protéines en quantité (g/personne/jour)  \
0                                               0.77
1                                               0.02
2                                               0.03
3                                                NaN
4                                               0.05

   Disponibilité intérieure  Exportations - Quantité  Importations - Quantité  \
0                      53.0                       NaN                      NaN
1                      41.0                       2.0                     40.0
2                       2.0                       NaN                      2.0
3                       0.0                       NaN                      0.0
4                      82.0                       NaN                     82.0

   Nourriture  Pertes  Production  Semences  Traitement  Variation de stock
0        53.0     NaN        53.0       NaN         NaN                 NaN
1        39.0     2.0         3.0       NaN         NaN                 NaN
2         2.0     NaN         NaN       NaN         NaN                 NaN
3         0.0     NaN         NaN       NaN         NaN                 NaN
4        82.0     NaN         NaN       NaN         NaN                 NaN
```

[13]:
```
#remplacement des NaN dans le dataset par des 0
#on utilise la méthode .replace et la fonction numpy NAN pour détecter les NAN
import numpy as np
```

```
dispo_alimentaire.replace(np.NAN, 0,inplace=True)
dispo_alimentaire.head()
```

[13]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	Nourriture	Pertes	Production	Semences	Traitement	Variation de stock
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.77	53.0	0.0	0.0						
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02	41.0	2.0	40.0						
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.03	2.0	0.0	2.0						
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00	0.0	0.0	0.0						
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.05	82.0	0.0	82.0						

0	53.0	0.0	53.0	0.0	0.0	0.0
1	39.0	2.0	3.0	0.0	0.0	0.0
2	2.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	82.0	0.0	0.0	0.0	0.0	0.0

[14]:
```
dispo_alimentaire.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15605 entries, 0 to 15604
Data columns (total 18 columns):
 #   Column                                                Non-Null
Count  Dtype
---  ------
--------------  -----
 0   Zone                                                  15605 non-
null  object
 1   Produit                                               15605 non-
null  object
 2   Origine                                               15605 non-
null  object
 3   Aliments pour animaux                                 15605 non-
null  float64
 4   Autres Utilisations                                   15605 non-
null  float64
 5   Disponibilité alimentaire (Kcal/personne/jour)        15605 non-
null  float64
 6   Disponibilité alimentaire en quantité (kg/personne/an)  15605 non-
null  float64
 7   Disponibilité de matière grasse en quantité (g/personne/jour)  15605 non-
null  float64
 8   Disponibilité de protéines en quantité (g/personne/jour)  15605 non-
null  float64
 9   Disponibilité intérieure                              15605 non-
null  float64
 10  Exportations - Quantité                               15605 non-
null  float64
 11  Importations - Quantité                               15605 non-
null  float64
 12  Nourriture                                            15605 non-
null  float64
 13  Pertes                                                15605 non-
null  float64
 14  Production                                            15605 non-
null  float64
 15  Semences                                              15605 non-
null  float64
 16  Traitement                                            15605 non-
```

```
null  float64
 17  Variation de stock                                    15605 non-
null  float64
dtypes: float64(15), object(3)
memory usage: 2.1+ MB
```

[15]:
```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
liste_colonnes_enKg = ['Aliments pour animaux', 'Autres Utilisations',␣
 ↪'Disponibilité intérieure' ,
                       'Exportations - Quantité' , 'Importations - Quantité' ,␣
 ↪'Nourriture' ,
                       'Pertes' , 'Production' , 'Semences' , 'Traitement' ,␣
 ↪'Variation de stock']
for elements in liste_colonnes_enKg :
  dispo_alimentaire[elements] = dispo_alimentaire[elements] * 1000000
```

[16]:
```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

[16]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02

```
   Disponibilité de protéines en quantité (g/personne/jour)  \
0                                                0.77
1                                                0.02
2                                                0.03
3                                                0.00
4                                                0.05

   Disponibilité intérieure  Exportations - Quantité  Importations - Quantité  \
0                53000000.0                      0.0                      0.0
1                41000000.0                2000000.0                40000000.0
2                 2000000.0                      0.0                 2000000.0
3                       0.0                      0.0                       0.0
4                82000000.0                      0.0                82000000.0

   Nourriture     Pertes  Production  Semences  Traitement  Variation de stock
0  53000000.0        0.0  53000000.0       0.0         0.0                 0.0
1  39000000.0  2000000.0   3000000.0       0.0         0.0                 0.0
2   2000000.0        0.0         0.0       0.0         0.0                 0.0
3         0.0        0.0         0.0       0.0         0.0                 0.0
4  82000000.0        0.0         0.0       0.0         0.0                 0.0
```

2.3 - Analyse exploratoire du fichier aide alimentaire

[17]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".
  ↪format(aide_alimentaire.shape[0]))
print("Le tableau comporte {} colonne(s)".format(aide_alimentaire.shape[1]))
```

```
Le tableau comporte 1475 observation(s) ou article(s)
Le tableau comporte 4 colonne(s)
```

[18]:
```
#Consulter le nombre de colonnes
print(aide_alimentaire.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1475 entries, 0 to 1474
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Pays bénéficiaire  1475 non-null   object
 1   Année              1475 non-null   int64
 2   Produit            1475 non-null   object
 3   Valeur             1475 non-null   int64
dtypes: int64(2), object(2)
memory usage: 46.2+ KB
None
```

[19]:
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

[19]:
	Pays bénéficiaire	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682
1	Afghanistan	2014	Autres non-céréales	335
2	Afghanistan	2013	Blé et Farin	39224
3	Afghanistan	2014	Blé et Farin	15160
4	Afghanistan	2013	Céréales	40504

[20]:
```
#changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire.rename(columns={'Pays bénéficiaire':'Zone'}, inplace=True)
```

[21]:
```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000␣
 ↪pour avoir des kg
aide_alimentaire['Valeur'] = aide_alimentaire['Valeur'] * 1000
```

[22]:
```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

[22]:
	Zone	Année	Produit	Valeur
0	Afghanistan	2013	Autres non-céréales	682000
1	Afghanistan	2014	Autres non-céréales	335000
2	Afghanistan	2013	Blé et Farin	39224000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

2.4 - Analyse exploratoire du fichier sous nutrition

[23]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".
 ↪format(sous_nutrition.shape[0]))
print("Le tableau comporte {} colonne(s)".format(sous_nutrition.shape[1]))
```

```
Le tableau comporte 1218 observation(s) ou article(s)
Le tableau comporte 3 colonne(s)
```

[24]:
```
#Consulter le nombre de colonnes
print(sous_nutrition.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1218 entries, 0 to 1217
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Zone    1218 non-null   object
 1   Année   1218 non-null   object
 2   Valeur  624 non-null    object
dtypes: object(3)
memory usage: 28.7+ KB
None
```

[25]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

[25]:

	Zone	Année	Valeur
0	Afghanistan	2012-2014	8.6
1	Afghanistan	2013-2015	8.8
2	Afghanistan	2014-2016	8.9
3	Afghanistan	2015-2017	9.7
4	Afghanistan	2016-2018	10.5

[26]:
```
#Conversion de la colonne sous nutrition en numérique
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'], errors =␣
↪'coerce')
```

[27]:
```
#Conversion de la colonne (avec l'argument errors=coerce qui permet de␣
↪convertir automatiquement les lignes qui ne sont pas des nombres en NaN)
#Puis remplacement des NaN en 0
sous_nutrition.replace(np.NAN, 0,inplace=True)
```

[28]:
```
#changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition.rename(columns={'Valeur':'sous_nutrition'}, inplace=True)
```

[30]:
```
#Multiplication de la colonne sous_nutrition par 1000000
sous_nutrition['sous_nutrition'] = sous_nutrition['sous_nutrition'] * 1000000
```

[31]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

[31]:

	Zone	Année	sous_nutrition
0	Afghanistan	2012-2014	8.600000e+12
1	Afghanistan	2013-2015	8.800000e+12
2	Afghanistan	2014-2016	8.900000e+12
3	Afghanistan	2015-2017	9.700000e+12
4	Afghanistan	2016-2018	1.050000e+13

3.1 - Proportion de personnes en sous nutrition

Etape 3 - Analyse des données

[32]:
```
# Il faut tout d'abord faire une jointure entre la table population et la table␣
↪sous nutrition, en ciblant l'année 2017
population_2017 = population.loc[population['Année'] == 2017, :]
sous_nutrition_2017 = sous_nutrition.loc[sous_nutrition['Année'] ==␣
↪'2016-2018', :]

# jointure de type externe pour conserver toutes les informations des deux␣
↪tables
```

```
merge_pop_sousnut =  pd.merge(population_2017, sous_nutrition_2017,␣
 ↪how='outer', on='Zone')
```

[33]:
```
#Affichage du dataset
display(merge_pop_sousnut)
```

```
                                      Zone  Année_x  Population    Année_y  \
0                                Afghanistan     2017  36296113.0  2016-2018
1                             Afrique du Sud     2017  57009756.0  2016-2018
2                                    Albanie     2017   2884169.0  2016-2018
3                                    Algérie     2017  41389189.0  2016-2018
4                                  Allemagne     2017  82658409.0  2016-2018
..                                       …       …         …          …
231  Venezuela (République bolivarienne du)     2017  29402484.0  2016-2018
232                                Viet Nam     2017  94600648.0  2016-2018
233                                   Yémen     2017  27834819.0  2016-2018
234                                  Zambie     2017  16853599.0  2016-2018
235                                Zimbabwe     2017  14236595.0  2016-2018

     sous_nutrition
0      1.050000e+13
1      3.100000e+12
2      1.000000e+11
3      1.300000e+12
4      0.000000e+00
..              …
231    8.000000e+12
232    6.500000e+12
233    0.000000e+00
234    0.000000e+00
235    0.000000e+00

[236 rows x 5 columns]
```

[34]:
```
#Calcul et affichage du nombre de personnes en état de sous nutrition

# nombre de personnes exprimé en Millions
nb_sous_nutrition = round(sous_nutrition_2017['sous_nutrition'].sum() /␣
 ↪1000000, 2)
print("Nombre de personnes en sous nutrition en 2017 : ", nb_sous_nutrition, "␣
 ↪Millions de personnes")

# pourcentage  de personnes
prop_sous_nutrition = round(sous_nutrition_2017['sous_nutrition'].sum() * 100 /␣
 ↪population_2017['Population'].sum(), 2)
print("La proportion de personnes en sous nutrition dans le monde est ",␣
 ↪prop_sous_nutrition, " %")
```

```
Nombre de personnes en sous nutrition en 2017 :  535700000.0  Millions de
personnes
La proportion de personnes en sous nutrition dans le monde est  7097118.2  %
```

3.2 - Nombre théorique de personne qui pourrait être nourries

[ ]:
```
#Combien mange en moyenne un être humain ? Source =>
```

Selon l'Organisation des Nations Unies pour l'alimentation et l'agriculture (FAO), la quantité moyenne de nourriture consommée par une personne dans le monde est d'environ **2 700 calories par jour**. Cependant, cette moyenne varie considérablement d'un pays à l'autre et dépend également de facteurs tels que le niveau de revenu et l'accès à la nourriture.

[35]:
```
#On commence par faire une jointure entre le data frame population et␣
 ↪Dispo_alimentaire afin d'ajouter dans ce dernier la population
# jointure de type gauche pour compléter le fichier dispo par les données de␣
 ↪population de 2017

merge_dispo_pop =  pd.merge(dispo_alimentaire,␣
 ↪population_2017[['Zone','Population']], how='left', on='Zone')
```

[36]:
```
#Affichage du nouveau dataframe
display(merge_dispo_pop)
```

```
              Zone                Produit   Origine  Aliments pour animaux  \
0       Afghanistan       Abats Comestible   animale                    0.0
1       Afghanistan        Agrumes, Autres  vegetale                    0.0
2       Afghanistan  Aliments pour enfants  vegetale                    0.0
3       Afghanistan                Ananas  vegetale                    0.0
4       Afghanistan                Bananes  vegetale                    0.0
…             …                     …       …                        …
15600  Îles Salomon       Viande de Suides   animale                    0.0
15601  Îles Salomon    Viande de Volailles   animale                    0.0
15602  Îles Salomon          Viande, Autre   animale                    0.0
15603  Îles Salomon                    Vin  vegetale                    0.0
15604  Îles Salomon         Épices, Autres  vegetale                    0.0

       Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
0                      0.0                                             5.0
1                      0.0                                             1.0
2                      0.0                                             1.0
3                      0.0                                             0.0
4                      0.0                                             4.0
…                      …                                               …
15600                  0.0                                            45.0
15601                  0.0                                            11.0
15602                  0.0                                             0.0
15603                  0.0                                             0.0
15604                  0.0                                             4.0
```

```
       Disponibilité alimentaire en quantité (kg/personne/an)  \
0                                                       1.72
1                                                       1.29
2                                                       0.06
3                                                       0.00
4                                                       2.70
…                                                       …
15600                                                   4.70
15601                                                   3.34
15602                                                   0.06
15603                                                   0.07
15604                                                   0.48

       Disponibilité de matière grasse en quantité (g/personne/jour)  \
0                                                       0.20
1                                                       0.01
2                                                       0.01
3                                                       0.00
4                                                       0.02
…                                                       …
15600                                                   4.28
15601                                                   0.69
15602                                                   0.00
15603                                                   0.00
15604                                                   0.21

       Disponibilité de protéines en quantité (g/personne/jour)  \
0                                                       0.77
1                                                       0.02
2                                                       0.03
3                                                       0.00
4                                                       0.05
…                                                       …
15600                                                   1.41
15601                                                   1.14
15602                                                   0.04
15603                                                   0.00
15604                                                   0.15

       Disponibilité intérieure  Exportations - Quantité  \
0                    53000000.0                       0.0
1                    41000000.0                 2000000.0
2                     2000000.0                       0.0
3                           0.0                       0.0
4                    82000000.0                       0.0
…                           …                         …
15600                 3000000.0                       0.0
```

```
15601                  2000000.0                      0.0
15602                        0.0                      0.0
15603                        0.0                      0.0
15604                        0.0                      0.0

       Importations - Quantité  Nourriture     Pertes  Production  Semences  \
0                          0.0  53000000.0        0.0  53000000.0       0.0
1                   40000000.0  39000000.0  2000000.0   3000000.0       0.0
2                    2000000.0   2000000.0        0.0         0.0       0.0
3                          0.0         0.0        0.0         0.0       0.0
4                   82000000.0  82000000.0        0.0         0.0       0.0
…                          …         …          …          …        …
15600                      0.0   3000000.0        0.0   2000000.0       0.0
15601                2000000.0   2000000.0        0.0         0.0       0.0
15602                      0.0         0.0        0.0         0.0       0.0
15603                      0.0         0.0        0.0         0.0       0.0
15604                      0.0         0.0        0.0         0.0       0.0

       Traitement  Variation de stock  Population
0             0.0                 0.0  36296113.0
1             0.0                 0.0  36296113.0
2             0.0                 0.0  36296113.0
3             0.0                 0.0  36296113.0
4             0.0                 0.0  36296113.0
…           …                   …          …
15600         0.0                 0.0    636039.0
15601         0.0                 0.0    636039.0
15602         0.0                 0.0    636039.0
15603         0.0                 0.0    636039.0
15604         0.0                 0.0    636039.0

[15605 rows x 19 columns]
```

[ ]:
```
merge_dispo_pop.info()
```

[37]:
```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
# Pour exprimer dispo_kal en Kcal il faut lui appliquer la population et␣
 ↪l'annualité

merge_dispo_pop['dispo_kcal'] = merge_dispo_pop['Disponibilité alimentaire␣
 ↪(Kcal/personne/jour)'] * merge_dispo_pop['Population'] * 365
```

[38]:
```
#Calcul du nombre d'humains pouvant être nourris
#  1 - calculer la dispo mondiale en kcal
#  2 - calculer combien de personnes peuvent être nourries par an
#  3 - calcluer la proportion de personnes qui peuvent être nourries
```

```
#étape 1 : calcul de la dispo kcal mondiale en 2017

dispo_kcal_2017 = merge_dispo_pop['dispo_kcal'].sum()
print ('Le total de kcalories disponibles en 2017   : ',␣
 ↪format(dispo_kcal_2017, ','), ' kcal' )
```

```
Le total de kcalories disponibles en 2017   :  7,635,429,388,975,815.0  kcal
```

[40]:
```
#étape 2 : combien de personnes peuvent êtres nourries par an

nb_humains_nourris = dispo_kcal_2017 / (2700*365)
print ('Le nb humains pouvant être nourris en 2017   : ',␣
 ↪format(nb_humains_nourris,','), ' personnes' )
```

```
Le nb humains pouvant être nourris en 2017   :  7,747,772,084.196667  personnes
```

[41]:
```
#étape 3 : calcul de la proportion

pop_mondiale_2017 = population_2017['Population'].sum()

prop_pop_nourris = round(nb_humains_nourris * 100 / pop_mondiale_2017, 2)
print("La proportion de personnes qui pourraient être nourries est de  ",␣
 ↪prop_pop_nourris, " %")

pop_nourris = np.abs(pop_mondiale_2017 - nb_humains_nourris)
print("Ces ", np.abs(round(100 - prop_pop_nourris, 2)) , " %", "représentent",␣
 ↪format(pop_nourris, ','), " personnes")
```

```
La proportion de personnes qui pourraient être nourries est de   102.64  %
Ces  2.64  % représentent 199,637,973.19666672  personnes
```

3.3 - Nombre théorique de personne qui pourrait être nourrie avec les produits végétaux

[42]:
```
#Transfert des données avec les végétaux dans un nouveau dataframe
dispo_veget = merge_dispo_pop.loc[merge_dispo_pop['Origine'] == 'vegetale', :]
```

[43]:
```
#Calcul du nombre de kcal disponible pour les végétaux
total_dispo_veget = dispo_veget['dispo_kcal'].sum()
print ('Le total de kcalories végétales disponibles en 2017   : ',␣
 ↪format(total_dispo_veget, ','), ' kcal par jour' )
```

```
Le total de kcalories végétales disponibles en 2017   :  6,300,178,937,197,865.0
kcal par jour
```

[44]:
```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
nb_humains_nourris_veget = total_dispo_veget / (2700*365)
print ('Le nb humains pouvant être nourris par les produits végétaux  en 2017  ␣
 ↪: ', format(nb_humains_nourris_veget,','), ' humains' )
```

```
prop_pop_nourris_veget = round(nb_humains_nourris_veget * 100 /␣
 ↪population_2017['Population'].sum(), 2)
print("La proportion de personnes pouvant être nourries avec les produits␣
 ↪végétaux est de ", prop_pop_nourris_veget, " %")
```

```
Le nb humains pouvant être nourris par les produits végétaux  en 2017   :
6,392,875,633.889259  humains
La proportion de personnes pouvant être nourries avec les produits végétaux est
de  84.69  %
```

3.4 - Utilisation de la disponibilité intérieure

[45]:
```
#Calcul de la disponibilité totale
dispo_int = merge_dispo_pop['Disponibilité intérieure'].sum()
print ('La disponibilité intérieure mondiale est : ', format(dispo_int,','),␣
 ↪'Kg' )
```

```
La disponibilité intérieure mondiale est :  9,848,994,000,000.0 Kg
```

[46]:
```
#création d'une boucle for pour afficher les différentes valeurs en fonction␣
 ↪des colonnes aliments pour animaux, pertes, nourritures,
# on boucle sur chaque colonne du dataframe merge_dispo_pop
# on applique le pourcentage de la dispo intérieure uniquement
#    sur les colonnes représentatives

nepastraiter = ['Zone','Produit','Origine', 'Disponibilité alimentaire (Kcal/
 ↪personne/jour)',
                'Disponibilité alimentaire en quantité (kg/personne/an)',
                'Disponibilité de matière grasse en quantité (g/personne/jour)',
                'Disponibilité de protéines en quantité (g/personne/jour)',
                'Disponibilité intérieure', 'Exportations - Quantité',␣
 ↪'Importations - Quantité',
                'Production', 'Variation de stock',
                 'Population', 'dispo_kcal']

for colonne in merge_dispo_pop.columns:
    if colonne in nepastraiter:
      continue
    valeur = (merge_dispo_pop[colonne].sum() * 100) / dispo_int
    print( "Proportion de ",  colonne, " : ", round(valeur,2), "%")
```

```
Proportion de  Aliments pour animaux  :  13.24 %
Proportion de  Autres Utilisations  :  8.78 %
Proportion de  Nourriture  :  49.51 %
Proportion de  Pertes  :  4.61 %
Proportion de  Semences  :  1.57 %
Proportion de  Traitement  :  22.38 %
```

3.5 - Utilisation des céréales

[47]:
```
#Création d'une liste avec toutes les variables

# etape 1 - on liste les différents produits qui sont dans le fichier
display(dispo_veget['Produit'].unique())

# etape 2 - on créé une liste manuellement en ne retenant que les produits de␣
 ↪type céréales
liste_cereales_old = ['Blé', 'Riz (Eq Blanchi)', 'Orge', 'Maïs', 'Seigle',
                  'Avoine', 'Millet', 'Sorgho', 'Céréales', 'Autres']

liste_cereales = ['Blé', 'Riz (Eq Blanchi)', 'Orge', 'Maïs', 'Seigle',
                  'Avoine', 'Millet', 'Sorgho', 'Céréales, Autres']
```

```
array(['Agrumes, Autres', 'Aliments pour enfants', 'Ananas', 'Bananes',
       'Bière', 'Blé', 'Boissons Alcooliques', 'Café',
       'Coco (Incl Coprah)', 'Céréales, Autres', 'Dattes',
       'Edulcorants Autres', 'Feve de Cacao', 'Fruits, Autres',
       'Graines de coton', 'Graines de tournesol',
       'Huil Plantes Oleif Autr', 'Huile Graines de Coton',
       "Huile d'Arachide", "Huile d'Olive", 'Huile de Colza&Moutarde',
       'Huile de Palme', 'Huile de Soja', 'Huile de Sésame',
       'Huile de Tournesol', 'Légumes, Autres', 'Légumineuses Autres',
       'Maïs', 'Miel', 'Millet', 'Miscellanees', 'Noix', 'Olives',
       'Oranges, Mandarines', 'Orge', 'Plantes Oleiferes, Autre',
       'Poivre', 'Pommes', 'Pommes de Terre', 'Raisin',
       'Riz (Eq Blanchi)', 'Sucre Eq Brut', 'Sucre, betterave',
       'Sucre, canne', 'Sésame', 'Thé', 'Tomates', 'Vin',
       'Épices, Autres', 'Alcool, non Comestible',
       'Arachides Decortiquees', 'Avoine', 'Bananes plantains',
       'Boissons Fermentés', 'Citrons & Limes', 'Girofles',
       'Graines Colza/Moutarde', 'Haricots', 'Huile de Coco',
       'Huile de Germe de Maïs', 'Huile de Palmistes', 'Ignames',
       'Manioc', 'Oignons', 'Palmistes', 'Pamplemousse', 'Patates douces',
       'Piments', 'Pois', 'Racines nda', 'Seigle', 'Soja', 'Sorgho',
       'Huile de Son de Riz', 'Sucre non centrifugé'], dtype=object)
```

[48]:
```
#Création d'un dataframe avec les informations uniquement pour ces céréales

# etape 1  - on utilise un masque booléen à partir de la liste de céréales
masque = dispo_veget['Produit'].isin(liste_cereales)

# etpae 2 - on rempli le nouveau dataframe en utilisant le masque booléen
utilisation_cereales = dispo_veget.loc[masque]

display(utilisation_cereales)
```

```
              Zone           Produit   Origine  Aliments pour animaux  \
7       Afghanistan              Blé  vegetale                    0.0
```

```
12      Afghanistan  Céréales, Autres  vegetale                   0.0
32      Afghanistan              Maïs  vegetale           200000000.0
34      Afghanistan             Millet  vegetale                   0.0
40      Afghanistan              Orge  vegetale           360000000.0
…             …                  …       …                      …
15545  Îles Salomon  Céréales, Autres  vegetale                   0.0
15568  Îles Salomon              Maïs  vegetale                   0.0
15575  Îles Salomon              Orge  vegetale                   0.0
15591  Îles Salomon  Riz (Eq Blanchi)  vegetale                   0.0
15593  Îles Salomon            Sorgho  vegetale                   0.0

       Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
7                      0.0                                           1369.0
12                     0.0                                              0.0
32                     0.0                                             21.0
34                     0.0                                              3.0
40                     0.0                                             26.0
…                      …                                                …
15545                  0.0                                              0.0
15568                  0.0                                              1.0
15575                  0.0                                              0.0
15591           12000000.0                                            623.0
15593                  0.0                                              0.0

       Disponibilité alimentaire en quantité (kg/personne/an)  \
7                                                      160.23
12                                                       0.00
32                                                       2.50
34                                                       0.40
40                                                       2.92
…                                                           …
15545                                                    0.00
15568                                                    0.15
15575                                                    0.07
15591                                                   63.76
15593                                                    0.00

       Disponibilité de matière grasse en quantité (g/personne/jour)  \
7                                                       4.69
12                                                      0.00
32                                                      0.30
34                                                      0.02
40                                                      0.24
…                                                          …
15545                                                   0.00
15568                                                   0.01
15575                                                   0.00
15591                                                   1.36
```

```
15593                                                0.00

       Disponibilité de protéines en quantité (g/personne/jour)  \
7                                                    36.91
12                                                    0.00
32                                                    0.56
34                                                    0.08
40                                                    0.79
…                                                     …
15545                                                 0.00
15568                                                 0.03
15575                                                 0.01
15591                                                10.90
15593                                                 0.00

       Disponibilité intérieure  Exportations - Quantité  \
7                  5.992000e+09                       0.0
12                 0.000000e+00                       0.0
32                 3.130000e+08                       0.0
34                 1.300000e+07                       0.0
40                 5.240000e+08                       0.0
…                         …                          …
15545              0.000000e+00                       0.0
15568              0.000000e+00                       0.0
15575              1.000000e+06                       0.0
15591              4.900000e+07                       0.0
15593              0.000000e+00                       0.0

       Importations - Quantité    Nourriture       Pertes    Production  \
7                 1.173000e+09  4.895000e+09  775000000.0  5.169000e+09
12                0.000000e+00  0.000000e+00          0.0  0.000000e+00
32                1.000000e+06  7.600000e+07   31000000.0  3.120000e+08
34                0.000000e+00  1.200000e+07    1000000.0  1.300000e+07
40                1.000000e+07  8.900000e+07   52000000.0  5.140000e+08
…                         …           …           …           …
15545             0.000000e+00  0.000000e+00          0.0  0.000000e+00
15568             0.000000e+00  0.000000e+00          0.0  0.000000e+00
15575             1.000000e+06  0.000000e+00          0.0  0.000000e+00
15591             4.700000e+07  3.600000e+07    1000000.0  3.000000e+06
15593             0.000000e+00  0.000000e+00          0.0  0.000000e+00

          Semences  Traitement  Variation de stock  Population    dispo_kcal
7      322000000.0         0.0        -350000000.0  36296113.0  1.813662e+13
12             0.0         0.0                 0.0  36296113.0  0.000000e+00
32       5000000.0         0.0                 0.0  36296113.0  2.782097e+11
34             0.0         0.0                 0.0  36296113.0  3.974424e+10
40      22000000.0         0.0                 0.0  36296113.0  3.444501e+11
…              …           …                   …           …           …
```

```
15545          0.0          0.0                 0.0    636039.0  0.000000e+00
15568          0.0          0.0                 0.0    636039.0  2.321542e+08
15575          0.0    1000000.0                 0.0    636039.0  0.000000e+00
15591          0.0          0.0                 0.0    636039.0  1.446321e+11
15593          0.0          0.0                 0.0    636039.0  0.000000e+00

[1497 rows x 20 columns]
```

[49]:
```
#Affichage de la proportion d'alimentation animale - CORRECTION

prop_anim = (utilisation_cereales['Aliments pour animaux'].sum() * 100) /␣
 ↪utilisation_cereales['Disponibilité intérieure'].sum()
print( "Proportion d'alimentation animale : ", round(prop_anim,2), "%")

prop_nourriture = (utilisation_cereales['Nourriture'].sum() * 100) /␣
 ↪utilisation_cereales['Disponibilité intérieure'].sum()
print( "Proportion d'alimentation humaine : ", round(prop_nourriture,2), "%")
```

```
Proportion d'alimentation animale :  36.29 %
Proportion d'alimentation humaine :  42.75 %
```

[50]:
```
# répartition de l'utilisation des créréales pour l'humain par céréale
repart_cereale_humain = utilisation_cereales.groupby('Produit')['Nourriture'].
 ↪sum().reset_index()
repart_cereale_humain['Nourriture (en tones)'] =␣
 ↪repart_cereale_humain['Nourriture'] / 1000
display(repart_cereale_humain[['Produit','Nourriture (en tones)']])
```

```
            Produit  Nourriture (en tones)
0            Avoine              3903000.0
1               Blé            457824000.0
2  Céréales, Autres              5324000.0
3              Maïs            125184000.0
4            Millet             23040000.0
5              Orge              6794000.0
6  Riz (Eq Blanchi)            377286000.0
7            Seigle              5502000.0
8            Sorgho             24153000.0
```

[51]:
```
# répartition de l'utilisation des créréales pour l'animal par céréale
repart_cereale_humain = utilisation_cereales.groupby('Produit')['Aliments pour␣
 ↪animaux'].sum().reset_index()
repart_cereale_humain['Aliments pour animaux (en tones)'] =␣
 ↪repart_cereale_humain['Aliments pour animaux'] / 1000
display(repart_cereale_humain[['Produit','Aliments pour animaux (en tones)']])
```

```
            Produit  Aliments pour animaux (en tones)
0            Avoine                          16251000.0
1               Blé                         129668000.0
```

```
2  Céréales, Autres                      19035000.0
3              Maïs                     546116000.0
4            Millet                       3306000.0
5              Orge                      92658000.0
6  Riz (Eq Blanchi)                      33594000.0
7            Seigle                       8099000.0
8            Sorgho                      24808000.0
```

3.6 - Pays avec la proportion de personnes sous-alimentée la plus forte en 2017

[52]:
```
#Création de la colonne proportion par pays
merge_pop_sousnut['prop_sous_nutri'] =␣
 ↪round((merge_pop_sousnut['sous_nutrition'] * 100) /␣
 ↪merge_pop_sousnut['Population'], 2)
display(merge_pop_sousnut)
```

```
                                    Zone  Année_x  Population    Année_y  \
0                              Afghanistan     2017  36296113.0  2016-2018
1                            Afrique du Sud     2017  57009756.0  2016-2018
2                                   Albanie     2017   2884169.0  2016-2018
3                                   Algérie     2017  41389189.0  2016-2018
4                                 Allemagne     2017  82658409.0  2016-2018
..                                      …       …         …         …
231  Venezuela (République bolivarienne du)     2017  29402484.0  2016-2018
232                               Viet Nam     2017  94600648.0  2016-2018
233                                  Yémen     2017  27834819.0  2016-2018
234                                 Zambie     2017  16853599.0  2016-2018
235                               Zimbabwe     2017  14236595.0  2016-2018

     sous_nutrition  prop_sous_nutri
0      1.050000e+13      28928717.52
1      3.100000e+12       5437665.79
2      1.000000e+11       3467203.20
3      1.300000e+12       3140916.82
4      0.000000e+00             0.00
..              …                …
231    8.000000e+12      27208585.51
232    6.500000e+12       6870988.88
233    0.000000e+00             0.00
234    0.000000e+00             0.00
235    0.000000e+00             0.00

[236 rows x 6 columns]
```

[53]:
```
#affichage après trie des 10 où la proportion de personnes en état de sous␣
 ↪nutrition est la plus forte en 2017

# Tri du dataframe par ordre décroissant de la nouvelle colonne
```

```
df_trie = merge_pop_sousnut.sort_values("prop_sous_nutri", ascending=False)
df_trie[['Zone', 'prop_sous_nutri']].head(10)
```

[53]:

	Zone	prop_sous_nutri
87	Haïti	48259182.04
181	République populaire démocratique de Corée	47188684.94
128	Madagascar	41062924.36
122	Libéria	38279742.40
119	Lesotho	38249437.97
216	Tchad	37957606.41
186	Rwanda	35055618.66
145	Mozambique	32810897.74
219	Timor-Leste	32173531.16
0	Afghanistan	28928717.52

3.7 - Pays qui ont le plus bénéficié d'aide alimentaire depuis 2013

[55]:
```
#calcul du total de l'aide alimentaire par pay

# 1 - il faut filtrer le dataframe sur l'année supérieure ou égale à 2013
aide_alim_depuis2013 = aide_alimentaire.loc[aide_alimentaire['Année'] >= 2013, :
 ↪]

# 2 - sur ce datframe on fait un groupby pour faire la somme des aides pour␣
 ↪chaque pays
# Attention le resulat d'un groupby est un série c'est pouquoi il faire un␣
 ↪reset.index() pour transformer le résultat en un dataframe
aide_alim_dep2013_par_pays =  aide_alim_depuis2013.groupby('Zone')['Valeur'].
 ↪sum().reset_index()

# 3 - On conertit le Kg en tone
aide_alim_dep2013_par_pays['Valeur'] = aide_alim_dep2013_par_pays['Valeur'] /␣
 ↪1000

display(aide_alim_dep2013_par_pays)
```

	Zone	Valeur
0	Afghanistan	185452.0
1	Algérie	81114.0
2	Angola	5014.0
3	Bangladesh	348188.0
4	Bhoutan	2666.0
..	…	…
71	Zambie	3026.0
72	Zimbabwe	62570.0
73	Égypte	1122.0
74	Équateur	1362.0
75	Éthiopie	1381294.0

[76 rows x 2 columns]

[56]:
```
#affichage après trie des 10 pays qui ont bénéficié le plus de l'aide␣
 ↪alimentaire

# 3 Tri du dataframe par ordre décroissant de l'aide octroyée
aide_alim_dep2013_par_pays.sort_values("Valeur", ascending=False).head(10)
```

[56]:

	Zone	Valeur
50	République arabe syrienne	1858943.0
75	Éthiopie	1381294.0
70	Yémen	1206484.0
61	Soudan du Sud	695248.0
60	Soudan	669784.0
30	Kenya	552836.0
3	Bangladesh	348188.0
59	Somalie	292678.0
53	République démocratique du Congo	288502.0
43	Niger	276344.0

3.8 - Evolution des 5 pays qui ont le plus bénéficiés de l'aide alimentaire entre 2013 et 2016

[57]:
```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis␣
 ↪groupby sur zone et année

# 1 - il faut filtrer le dataframe sur la période 2013 à 2016
aide_alim_de2013_a2016 = aide_alimentaire.loc[(aide_alimentaire['Année'] >=␣
 ↪2013) & (aide_alimentaire['Année'] <= 2016), :]

# 2 - on créé un dataframe qui aura la zone, l'année et le montant des aides
#     pour cela on fait un groupy et un  reset_index() pour pouvoir travailler␣
 ↪avec le Dataframe
evol_aide_de2013_a2016 =  aide_alim_de2013_a2016.
 ↪groupby(['Zone','Année'])['Valeur'].sum().reset_index()
```

[58]:
```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de␣
 ↪l'aide alimentaire
# 3 - On caclule la somme des aides reçus par chaque pays  entre 2013 et 2016
#     On trie le dataframe par ordre décroissant
#     On copie dans une liste avec .tolist() les cinq premiers

top5 = aide_alim_de2013_a2016.groupby('Zone')['Valeur'].sum().reset_index()
top5 = top5.sort_values("Valeur", ascending=False)

liste = top5['Zone'].head(5).tolist()
print(liste)
```

```
['République arabe syrienne', 'Éthiopie', 'Yémen', 'Soudan du Sud', 'Soudan']
```

[59]:
```
#On filtre sur le dataframe avec notre liste
# 1 - on utilise un masque booléen à partir de la liste des top5
masque_top5 = evol_aide_de2013_a2016['Zone'].isin(liste)

# 2 - on rempli le nouveau dataframe en utilisant le masque booléen
top5_benef_aide_alim = evol_aide_de2013_a2016[masque_top5].reset_index()

# 3 - on convertit les Kg en tone
top5_benef_aide_alim['Valeur'] = top5_benef_aide_alim['Valeur'] / 1000
```

[60]:
```
# Affichage des pays avec l'aide alimentaire par année
display(top5_benef_aide_alim)
```

	index	Zone	Année	Valeur
0	157	République arabe syrienne	2013	563566.0
1	158	République arabe syrienne	2014	651870.0
2	159	République arabe syrienne	2015	524949.0
3	160	République arabe syrienne	2016	118558.0
4	189	Soudan	2013	330230.0
5	190	Soudan	2014	321904.0
6	191	Soudan	2015	17650.0
7	192	Soudan du Sud	2013	196330.0
8	193	Soudan du Sud	2014	450610.0
9	194	Soudan du Sud	2015	48308.0
10	214	Yémen	2013	264764.0
11	215	Yémen	2014	103840.0
12	216	Yémen	2015	372306.0
13	217	Yémen	2016	465574.0
14	225	Éthiopie	2013	591404.0
15	226	Éthiopie	2014	586624.0
16	227	Éthiopie	2015	203266.0

3.9 - Pays avec le moins de disponibilité par habitant

[61]:
```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo_kcal_pays = merge_dispo_pop.groupby(['Zone'])['Disponibilité alimentaire␣
↪(Kcal/personne/jour)'].sum().reset_index()
```

[62]:
```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
dispo_kcal_pays.sort_values("Disponibilité alimentaire (Kcal/personne/jour)").
↪head(10)
```

[62]:

	Zone	\
128	République centrafricaine	
166	Zambie	
91	Madagascar	
0	Afghanistan	

```
65                                      Haïti
133  République populaire démocratique de Corée
151                                      Tchad
167                                   Zimbabwe
114                                    Ouganda
154                                 Timor-Leste

     Disponibilité alimentaire (Kcal/personne/jour)
128                                          1879.0
166                                          1924.0
91                                           2056.0
0                                            2087.0
65                                           2089.0
133                                          2093.0
151                                          2109.0
167                                          2113.0
114                                          2126.0
154                                          2129.0
```

3.10 - Pays avec le plus de disponibilité par habitant

[63]:
```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
dispo_kcal_pays.sort_values("Disponibilité alimentaire (Kcal/personne/jour)",␣
↪ascending=False).head(10)
```

[63]:
```
                     Zone  Disponibilité alimentaire (Kcal/personne/jour)
11                Autriche                                          3770.0
16                Belgique                                          3737.0
159                Turquie                                          3708.0
171  États-Unis d'Amérique                                          3682.0
74                  Israël                                          3610.0
72                 Irlande                                          3602.0
75                  Italie                                          3578.0
89              Luxembourg                                          3540.0
168                 Égypte                                          3518.0
4                Allemagne                                          3503.0
```

3.11 - Exemple de la Thaïlande pour le Manioc

[64]:
```
#création d'un dataframe avec uniquement la Thaïlande
dispopop_thailande_manioc = merge_dispo_pop.loc[(merge_dispo_pop['Zone'] ==␣
↪'Thaïlande') & (merge_dispo_pop['Produit'] == 'Manioc'), :]
dispopop_thailande_manioc
```

[64]:
```
            Zone Produit   Origine  Aliments pour animaux  \
13809  Thaïlande  Manioc  vegetale           1.800000e+09

       Autres Utilisations  Disponibilité alimentaire (Kcal/personne/jour)  \
```

```
13809        2.081000e+09                                                40.0

       Disponibilité alimentaire en quantité (kg/personne/an)  \
13809                                                13.0

       Disponibilité de matière grasse en quantité (g/personne/jour)  \
13809                                                0.05

       Disponibilité de protéines en quantité (g/personne/jour)  \
13809                                                0.14

       Disponibilité intérieure  Exportations - Quantité  \
13809              6.264000e+09             2.521400e+10

       Importations - Quantité   Nourriture        Pertes    Production  \
13809             1.250000e+09  871000000.0  1.511000e+09  3.022800e+10

       Semences  Traitement  Variation de stock  Population    dispo_kcal
13809       0.0         0.0                 0.0  69209810.0  1.010463e+12
```

[65]:
```
#Calcul de la sous nutrition en Thaïlande

sousnut_thailande = merge_pop_sousnut.loc[merge_pop_sousnut['Zone']==␣
 ↪'Thaïlande']
pourcent_sousnut_thai = (sousnut_thailande['sous_nutrition'].sum() * 100) /␣
 ↪sousnut_thailande['Population'].sum()
print("Le pourcentage de population Thaïlandaise en sous mutrition en 2017 : ",␣
 ↪round(pourcent_sousnut_thai,2), " %")
```

```
Le pourcentage de population Thaïlandaise en sous mutrition en 2017 :
8958267.62  %
```

[66]:
```
# On calcule la proportion exportée en fonction de la proportion
exportations = dispopop_thailande_manioc['Exportations - Quantité'].sum()
#dispo_alim_qté = (dispopop_thailande_manioc['Disponibilité alimentaire en␣
 ↪quantité (kg/personne/an)'].sum()) * 365 *␣
 ↪dispopop_thailande_manioc['Population'].sum()
production_thai = dispopop_thailande_manioc['Production'].sum()

pourcent_exp = (exportations * 100) / production_thai
print("La part des exportations en Thailande par rapport à leur production  :␣
 ↪", round(pourcent_exp,2), " %")
```

```
La part des exportations en Thailande par rapport à leur production  :  83.41  %
```

Etape 4 - Analyses complémentaires - PERTES ALIMENTAIRES

[ ]:
```
#Rajouter en dessous toutes les analyses complémtaires suite à la demande de␣
 ↪mélanie :
#"et toutes les infos que tu trouverais utiles pour mettre en relief les pays␣
 ↪qui semblent être
#le plus en difficulté au niveau alimentaire"
```

[67]:
```
# On fait un groupby pour faire la somme des pertes pour chaque pays
# Attention le resulat d'un groupby est un série c'est pouquoi il faire un␣
 ↪reset.index() pour transformer le résultat en un dataframe
pertes_pays =  dispo_alimentaire.groupby('Zone')['Pertes'].sum().reset_index()

# on convertit le Kg en Million de Tone
pertes_pays['Pertes'] = round(pertes_pays['Pertes'] / 1000000000,2)

# on change le nom de la colonne Pertes
pertes_pays2 = pertes_pays.rename(columns={'Pertes': 'Pertes (en Millions de␣
 ↪tones)'})

#Affichage des 10 pays qui ont le plus de pertes
pertes_pays2.sort_values("Pertes (en Millions de tones)", ascending=False).
 ↪head(10)
```

[67]:

	Zone	Pertes (en Millions de tones)
36	Chine, continentale	89.58
23	Brésil	75.91
68	Inde	55.93
108	Nigéria	19.85
69	Indonésie	13.08
159	Turquie	12.04
100	Mexique	8.29
168	Égypte	7.61
57	Ghana	7.44
171	États-Unis d'Amérique	7.16

[68]:
```
# calcul des pertes totales 2017 en les exprimant tones
pertes_total_tones =  dispo_alimentaire['Pertes'].sum() / 1000

# affichage en Millions de tones
print("En 2017 les pertes représentent ", round((pertes_total_tones /␣
 ↪1000000),2), " Millions de tones de déchets alimentaires")

#caclul de la production en tones
production_total_tones  = dispo_alimentaire['Production'].sum() / 1000

pourcent_perte = (pertes_total_tones * 100) / production_total_tones
print("Cela représente ", round(pourcent_perte,2)," % de la production mondiale␣
 ↪de 2017")
```

```
En 2017 les pertes représentent  453.7  Millions de tones de déchets
alimentaires
Cela représente  4.53  % de la production mondiale de 2017
```

[1]:
```
#Exportation au format PDF
!pip install nbconvert
```

```
Requirement already satisfied: nbconvert in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (6.5.4)
Requirement already satisfied: lxml in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(4.9.3)
Requirement already satisfied: beautifulsoup4 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(4.12.2)
Requirement already satisfied: bleach in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(4.1.0)
Requirement already satisfied: defusedxml in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(0.4)
Requirement already satisfied: jinja2>=3.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(3.1.2)
Requirement already satisfied: jupyter-core>=4.7 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(5.3.0)
Requirement already satisfied: jupyterlab-pygments in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(0.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(2.1.1)
Requirement already satisfied: mistune<2,>=0.8.1 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(0.5.13)
Requirement already satisfied: nbformat>=5.1 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(5.9.2)
Requirement already satisfied: packaging in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(23.1)
```

```
Requirement already satisfied: pandocfilters>=1.4.1 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(1.5.0)
Requirement already satisfied: pygments>=2.4.1 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(2.15.1)
Requirement already satisfied: tinycss2 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(1.2.1)
Requirement already satisfied: traitlets>=5.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from nbconvert)
(5.7.1)
Requirement already satisfied: platformdirs>=2.5 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from jupyter-
core>=4.7->nbconvert) (3.10.0)
Requirement already satisfied: pywin32>=300 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from jupyter-
core>=4.7->nbconvert) (305.1)
Requirement already satisfied: jupyter-client>=6.1.5 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
nbclient>=0.5.0->nbconvert) (7.4.9)
Requirement already satisfied: nest-asyncio in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
nbclient>=0.5.0->nbconvert) (1.5.6)
Requirement already satisfied: fastjsonschema in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
nbformat>=5.1->nbconvert) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
nbformat>=5.1->nbconvert) (4.17.3)
Requirement already satisfied: soupsieve>1.2 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
beautifulsoup4->nbconvert) (2.4)
Requirement already satisfied: six>=1.9.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
bleach->nbconvert) (1.16.0)
Requirement already satisfied: webencodings in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
bleach->nbconvert) (0.5.1)
Requirement already satisfied: attrs>=17.4.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (22.1.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.1->nbconvert) (0.18.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient>=0.5.0->nbconvert) (2.8.2)
```

```
Requirement already satisfied: pyzmq>=23.0 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient>=0.5.0->nbconvert) (23.2.0)
Requirement already satisfied: tornado>=6.2 in
c:\users\emmanuelm\appdata\local\anaconda3\lib\site-packages (from jupyter-
client>=6.1.5->nbclient>=0.5.0->nbconvert) (6.3.2)
```

[ ]: