

PROJET 4 DATA ANALYST

Réalisez une étude de santé publique avec R ou Python

✓ 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'une 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 à trouver ...

Etape 1 - Importation des librairies et chargement des fichiers

1.1 - Importation des librairies

```
#Importation des librairies Pandas & Numpy & Matplotlib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.2 - Chargement des fichiers Excel

```
#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

```
#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)
```

```
#Consulter le nombre de colonnes
len(population.axes[1])
#La nature des données dans chacune des colonnes
population.dtypes
#Le nombre de valeurs présentes dans chacune des colonnes
population.count()
```

```
Zone          1416
Année         1416
Population    1416
dtype: int64
```

```
#Affichage les 5 premières lignes de la table
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

```
#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'] = round(population['Valeur'] * 1000, 0)
```

```
#changement du nom de la colonne Valeur par Population
population = population.rename(columns={'Valeur': 'Population'})
```

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

	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

```
#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)
```

```
#Consulter le nombre de colonnes
len(dispo_alimentaire.axes[1])
```

```
18
```

```
#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

```

Zone    Produit    Origine    Aliments pour animaux    Autres Utilisations    Disponibilité alimentaire (Kcal/personne/jour)

0  Afghanistan    Abats Comestible    animale    NaN    NaN    5.0

1  Afghanistan    Agrumes, Autres    vegetale    NaN    NaN    1.0

2  Afghanistan    Aliments pour enfants    vegetale    NaN    NaN    1.0

#remplacement des NaN dans le dataset par des 0
dispo_alimentaire.fillna(0, inplace=True)

4  Afghanistan    Bananes    vegetale    NaN    NaN    4.0

#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
dispo_alimentaire.iloc[:, 3:5] = dispo_alimentaire.iloc[:, 3:5] * 1000000
dispo_alimentaire.iloc[:, 9:] = dispo_alimentaire.iloc[:, 9:] * 1000000

#Affichage les 5 premières lignes de la table
dispo_alimentaire.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0

2.3 - Analyse exploratoire du fichier aide alimentaire

```

#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)

#Consulter le nombre de colonnes
len(aide_alimentaire.axes[1])

4

#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

	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

```

#changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire = aide_alimentaire.rename(columns={'Pays bénéficiaire': 'Zone'})

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

```
#Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

	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	

2.3 - Analyse exploratoire du fichier sous nutrition


```
#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)
```

```
#Consulter le nombre de colonnes
len(sous_nutrition.axes[1])
```

```
3
```

```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

	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	



```
#Conversion de la colonne sous nutrition en numérique
#Les valeurs <0.1 ont été remplacées par des NaN
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'], errors='coerce')
```

```
#Conversion de la colonne (avec l'argument errors=coerce qui permet de convertir automatiquement les lignes qui ne sont pas
#Puis remplacement des NaN en 0
sous_nutrition.fillna(0, inplace=True)
```

```
#changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition = sous_nutrition.rename(columns={'Valeur': 'sous_nutrition'})
```

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

```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

	Zone	Année	sous_nutrition	
0	Afghanistan	2012-2014	8600000.0	
1	Afghanistan	2013-2015	8800000.0	
2	Afghanistan	2014-2016	8900000.0	
3	Afghanistan	2015-2017	9700000.0	
4	Afghanistan	2016-2018	10500000.0	

3.1 - Proportion de personnes en sous nutrition

```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'année 2017
# obtenir le data frame du nombre de personnes en sous_nutrition par pays pour l'année 2016-2018
sous_nutrition_2017 = sous_nutrition.loc[sous_nutrition['Année'] == '2016-2018', :].reset_index(drop=True)
sous_nutrition_2017 = sous_nutrition_2017.drop('Année', axis=1)
# obtenir le data frame de la population par pays en 2017
population_2017 = population.loc[population['Année'] == 2017, :].reset_index(drop=True)
# vérifier nombre de lignes par df ()
population_2017.shape[0] #236
sous_nutrition_2017.shape[0] #203
# on fusionne les deux tables
pop_sous_nutrition_2017 = pd.merge(population_2017, sous_nutrition_2017, on='Zone', how='right')
# remplacement des NaN par des 0
pop_sous_nutrition_2017.fillna(0, inplace=True)
```

```
#Affichage du dataset
pop_sous_nutrition_2017.head()
```

	Zone	Année	Population	sous_nutrition
0	Afghanistan	2017	36296113.0	10500000.0
1	Afrique du Sud	2017	57009756.0	3100000.0
2	Albanie	2017	2884169.0	100000.0
3	Algérie	2017	41389189.0	1300000.0
4	Allemagne	2017	82658409.0	0.0

```
#Calcul et affichage du nombre de personnes en état de sous nutrition
print("{} millions de personnes étaient en état de sous nutrition dans le monde en 2017.".format(round(pop_sous_nutrition_2017.sous_nutrition.sum()/1000000)))
```

```
#Calcul et affichage du taux de personnes en état de sous nutrition
print("{} % des humains étaient en état de sous nutrition dans le monde en 2017.".format(round((pop_sous_nutrition_2017.sous_nutrition.sum()/pop_sous_nutrition_2017.Population.sum())*100)))
```

535.7 millions de personnes étaient en état de sous nutrition dans le monde en 2017.
7.1 % des humains étaient en état de sous nutrition dans le monde en 2017.

3.2 - Nombre théorique de personne qui pourrait être nourries

```
#Combien mange en moyenne un être humain ? moyenne de 2100 (femmes) et 2600 (hommes) Source => https://www.anses.fr/en/systeme
# deuxième source pour consommation moyenne effective : https://www.fao.org/economic/the-statistics-division-ess/chartroom-a
conso_calories_jour = 2350
print("Un être humain consomme en moyenne {} calories par jour.".format(conso_calories_jour))
```

Un être humain consomme en moyenne 2350 calories par jour.

```
#On commence par faire une jointure entre le data frame population et Dispo_alimentaire afin d'ajouter dans ce dernier la pc
#On crée un df population pour 2017
population_2017 = population.loc[population['Année'] == 2017, :].reset_index(drop=True)
population_2017 = population_2017.drop("Année", axis='columns')
#On fait la jointure avec ce nouveau df
dispo_alimentaire_2017 = pd.merge(population_2017, dispo_alimentaire, on="Zone", how='right')
dispo_alimentaire_2017.fillna(0, inplace=True)
```

```
#Affichage du nouveau dataframe
dispo_alimentaire_2017.head()
```

	Zone	Population	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disa (Kcal/pers
0	Afghanistan	36296113.0	Abats Comestible	animale	0.0	0.0	
1	Afghanistan	36296113.0	Agrumes, Autres	vegetale	0.0	0.0	
2	Afghanistan	36296113.0	Aliments pour enfants	vegetale	0.0	0.0	
3	Afghanistan	36296113.0	Ananas	vegetale	0.0	0.0	
4	Afghanistan	36296113.0	Bananes	vegetale	0.0	0.0	

```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
dispo_alimentaire_monde = 0
for i in dispo_alimentaire_2017.index:
    dispo_alimentaire_monde += dispo_alimentaire_2017['Disponibilité alimentaire (Kcal/personne/jour)'][i] * dispo_alimentaire_2017['Population'][i]

print("Le monde a produit {} billions de Kcal par jour en 2017.".format(round(dispo_alimentaire_monde/1000000000000,2)))

    Le monde a produit 20.92 billions de Kcal par jour en 2017.

#Calcul du nombre d'humains pouvant être nourris
population_monde = 0
for i in population.index:
    if population['Année'][i] == 2017:
        population_monde += population['Population'][i]
print('Il y avait {} milliards d\'êtres humains sur Terre en 2017.'.format(round(population_monde/1000000000,2)))

print('En 2017, la planète produisait suffisamment de ressources pour nourrir {} milliards d\'êtres humains, soit {} % de la population mondiale.'.format(round(population_monde/1000000000,2), round(dispo_alimentaire_monde/(population_monde*1000000000),2)))

    Il y avait 7.55 milliards d'êtres humains sur Terre en 2017.
    En 2017, la planète produisait suffisamment de ressources pour nourrir 8.9 milliards d'êtres humains, soit 117.93 % de la population mondiale.
```

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

```
#Transfert des données avec les végétaux dans un nouveau dataframe
dispo_alimentaire_2017.head()
```

	Zone	Population	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité (Kcal/pers/jour)
0	Afghanistan	36296113.0	Abats Comestible	animale	0.0	0.0	0.0
1	Afghanistan	36296113.0	Agrumes, Autres	vegetale	0.0	0.0	0.0
2	Afghanistan	36296113.0	Aliments pour enfants	vegetale	0.0	0.0	0.0
3	Afghanistan	36296113.0	Ananas	vegetale	0.0	0.0	0.0
4	Afghanistan	36296113.0	Bananes	vegetale	0.0	0.0	0.0

```
#Calcul du nombre de kcal disponible pour les végétaux
dispo_alimentaire_veg = 0
for i in dispo_alimentaire_2017.index:
    if dispo_alimentaire_2017['Origine'][i] == "vegetale":
        dispo_alimentaire_veg += dispo_alimentaire_2017['Disponibilité alimentaire (Kcal/personne/jour)'][i] * dispo_alimentaire_2017['Population'][i]

print("Le monde a produit {} billions de Kcal par jour avec les végétaux en 2017.".format(round(dispo_alimentaire_veg/1000000000000,2)))

    Le monde a produit 17.26 billions de Kcal par jour avec les végétaux en 2017.

#Calcul du nombre d'humains pouvant être nourris avec les végétaux
print('Il y avait {} milliards d\'êtres humains sur Terre en 2017.'.format(round(population_monde/1000000000,2)))

print('En 2017, la planète produisait suffisamment de ressources d\'origine végétale pour nourrir {} milliards d\'êtres humains, soit {} % de la population mondiale.'.format(round(population_monde/1000000000,2), round(dispo_alimentaire_veg/(population_monde*1000000000),2)))

    Il y avait 7.55 milliards d'êtres humains sur Terre en 2017.
    En 2017, la planète produisait suffisamment de ressources d'origine végétale pour nourrir 7.35 milliards d'êtres humains, soit 96.82 % de la population mondiale.
```

3.4 - Utilisation de la disponibilité intérieure

```
#Calcul de la disponibilité totale
#disponibilités totales = Production + importations + baisses des stocks
#disponibilité intérieure = Production + importations - exportations + variations des stocks (baisse ou augmentation)
# = Semences + Pertes + Aliments pour animaux + Nourriture + Traitement + Autres utilisations
dispo_alimentaire['Disponibilité totale'] = 0
for i in dispo_alimentaire.index:
    if dispo_alimentaire['Variation de stock'][i] < 0:
        dispo_alimentaire['Disponibilité totale'][i] = dispo_alimentaire['Importations - Quantité'][i] + dispo_alimentaire['Pertes - Quantité'][i]
    else:
        dispo_alimentaire['Disponibilité totale'][i] = dispo_alimentaire['Importations - Quantité'][i] + dispo_alimentaire['Production + Variations des stocks'][i]

dispo_interieure_totale = dispo_alimentaire.iloc[:,9].sum()
dispo_alimentaire.head()
```

dispo_alimentaire_cereales

```
<ipython-input-42-7fd94c1265e8>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/s>
 dispo_alimentaire['Disponibilité totale'][i] = dispo_alimentaire['Importati

```
<ipython-input-42-7fd94c1265e8>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

See the caveats in the documentation: <https://pandas.pydata.org/pandas-docs/s>
 dispo_alimentaire['Disponibilité totale'][i] = dispo_alimentaire['Importati

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0

```
#création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux, pertes, nou
aliments = ['Aliments pour animaux', 'Pertes', 'Nourriture', 'Semences', 'Traitement', 'Autres Utilisations']
print('La disponibilité intérieure mondiale en 2017 était d\'environ {} milliards de tonnes.'.format(round(dispo_interieure_
for i in aliments:
    print('En 2017, à l\'échelle mondiale, la proportion du segment \'' + i + '\' par rapport à la disponibilité intérieure était c
```

La disponibilité intérieure mondiale en 2017 était d'environ 9.85 milliards de tonnes.

En 2017, à l'échelle mondiale, la proportion du segment 'aliments pour animaux' par rapport à la disponibilité intérieur

En 2017, à l'échelle mondiale, la proportion du segment 'pertes' par rapport à la disponibilité intérieure était de 4.61

En 2017, à l'échelle mondiale, la proportion du segment 'nourriture' par rapport à la disponibilité intérieure était de 1.

En 2017, à l'échelle mondiale, la proportion du segment 'semences' par rapport à la disponibilité intérieure était de 1.

En 2017, à l'échelle mondiale, la proportion du segment 'traitement' par rapport à la disponibilité intérieure était de

En 2017, à l'échelle mondiale, la proportion du segment 'autres utilisations' par rapport à la disponibilité intérieure

3.5 - Utilisation des céréales

```
#Création d'une liste avec toutes les variables
```

```
liste_brute = ['blé', 'orge', 'maïs', 'riz (eq blanchi)', 'seigle', 'avoine', 'millet', 'sorgho', 'céréales, autres']
```

```
liste_cereales = []
```

```
for i in dispo_alimentaire.index:
```

```
    if dispo_alimentaire['Produit'][i].lower() in liste_brute and dispo_alimentaire['Produit'][i] not in liste_cereales:
        liste_cereales.append(dispo_alimentaire['Produit'][i])
```

```
liste_cereales
```

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

```
#Création d'un dataframe avec les informations uniquement pour ces céréales
```

```
dispo_alimentaire_cereales = dispo_alimentaire.loc[dispo_alimentaire['Produit'].isin(liste_cereales), :].reset_index(drop=True)
dispo_alimentaire_cereales.head()
```

```

Zone  Produit  Origine  Aliments  Autres  Disponibilité
pour  Utilisations  alimentaire
animaux  (Kcal/personne/jour)

#Affichage de la proportion d'alimentation humaine (selon disponibilité intérieure)
#disponibilité intérieure = Semences + Pertes + Aliments pour animaux + Nourriture + Traitement + Autres utilisations
alimentation_humaine_cereale = round((dispo_alimentaire_cereales.iloc[:,12].sum()) / (dispo_alimentaire_cereales.iloc[:,9].sum()))
print('{} % des céréales disponibles dans le monde étaient destinées aux humains en 2017.'.format(alimentation_humaine_cereale))

42.75 % des céréales disponibles dans le monde étaient destinées aux humains en 2017.

3  Afghanistan  Millet  vegetale  0.0  0.0  3.0

#Affichage de la proportion d'alimentation animale (selon disponibilité intérieure)
alimentation_animaux_cereale = round((dispo_alimentaire_cereales.iloc[:,3].sum()) / (dispo_alimentaire_cereales.iloc[:,9].sum()))
print('{} % des céréales disponibles dans le monde étaient destinées aux animaux en 2017.'.format(alimentation_animaux_cereale))
print('Les {} % restants non consommés par les animaux ou les humains étaient destinés aux semences, au traitement, aux autres')

36.29 % des céréales disponibles dans le monde étaient destinées aux animaux en 2017.
Les 20.96 % restants non consommés par les animaux ou les humains étaient destinés aux semences, au traitement, aux autres

```

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

```

#Création de la colonne proportion par pays
# ajout d'une colonne taux de sous_nutrition
pop_sous_nutrition_2017['taux'] = round((pop_sous_nutrition_2017['sous_nutrition'] / pop_sous_nutrition_2017['Population']))
pop_sous_nutrition_2017.head()

```

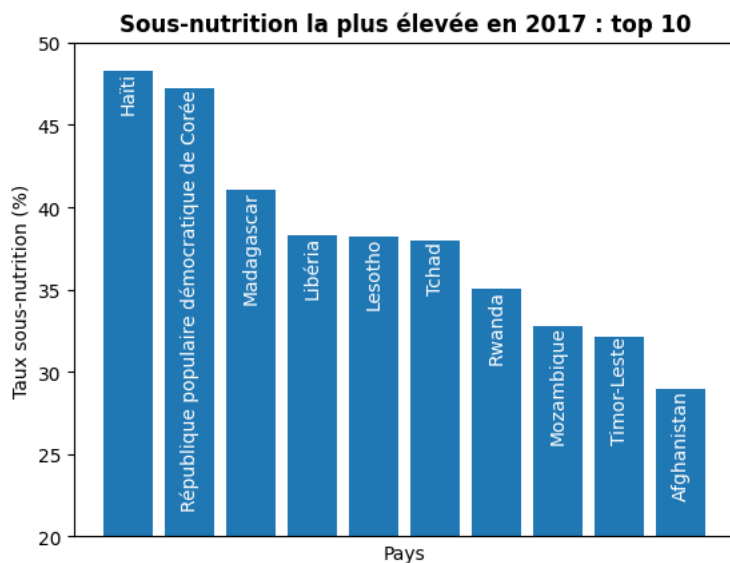
	Zone	Année	Population	sous_nutrition	taux
0	Afghanistan	2017	36296113.0	10500000.0	28.93
1	Afrique du Sud	2017	57009756.0	3100000.0	5.44
2	Albanie	2017	2884169.0	100000.0	3.47
3	Algérie	2017	41389189.0	1300000.0	3.14
4	Allemagne	2017	82658409.0	0.0	0.00

```

#affichage après tri des 10 pires pays
#création df avec top 10 pire pays uniquement
top_10_pays_sous_nutrition = pop_sous_nutrition_2017.sort_values('taux', ascending=False).iloc[:10,:].reset_index(drop=True)
#création diagramme à barres
plt.bar(height=top_10_pays_sous_nutrition['taux'], x=top_10_pays_sous_nutrition['Zone'])
plt.title('Sous-nutrition la plus élevée en 2017 : top 10', fontweight='bold')
plt.xlabel('Pays')
plt.xticks([])
plt.ylabel('Taux sous-nutrition (%)')
for index, value in enumerate(top_10_pays_sous_nutrition['taux']):
    plt.text(index, value, str(top_10_pays_sous_nutrition['Zone'][index]), ha='center', va='top', rotation='vertical', color='white')
plt.ylim([20, 50])

```

(20.0, 50.0)

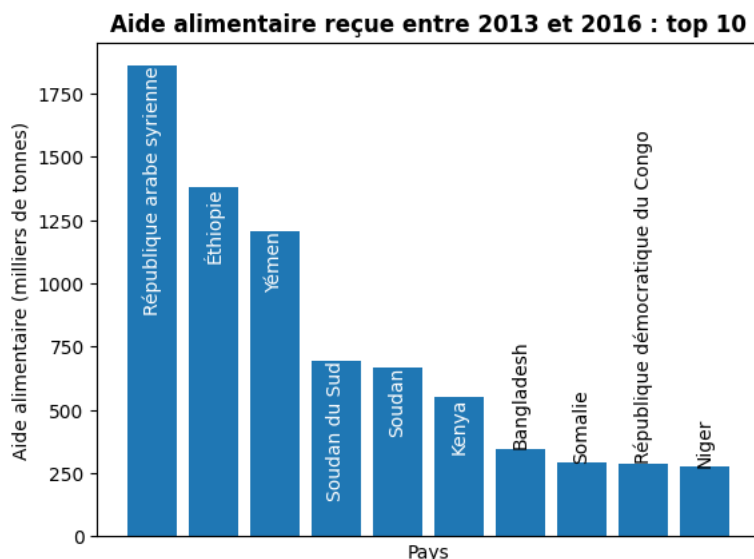


3.7 - Pays qui ont le plus bénéficié d'aide alimentaire depuis 2013


```
#Division de la colonne Aide_alimentaire qui contient des kg par 1,000,000 pour avoir des milliers de tonnes
aide_alimentaire['Valeur'] = aide_alimentaire['Valeur'] / 1000000
#calcul du total de l'aide alimentaire par pays
aide_alimentaire_totale = aide_alimentaire.groupby('Zone')['Valeur'].sum().sort_values(ascending=False)
aide_alimentaire_totale = aide_alimentaire_totale.reset_index()
aide_alimentaire_totale.head()
```

	Zone	Valeur
0	République arabe syrienne	1858.943
1	Éthiopie	1381.294
2	Yémen	1206.484
3	Soudan du Sud	695.248
4	Soudan	669.784

```
#affichage après tri des 10 pays qui ont bénéficié le plus de l'aide alimentaire
#création df avec top 10 pire pays uniquement
top_10_pays_aide_alimentaire = aide_alimentaire_totale.iloc[:10,:].reset_index(drop=True)
#création diagramme à barres
plt.bar(height=top_10_pays_aide_alimentaire['Valeur'], x=top_10_pays_aide_alimentaire['Zone'])
plt.title('Aide alimentaire reçue entre 2013 et 2016 : top 10', fontweight='bold')
plt.xlabel('Pays')
plt.ylabel('Aide alimentaire (milliers de tonnes)')
plt.xticks([])
for index, value in enumerate(top_10_pays_aide_alimentaire['Valeur'][:6]):
    plt.text(index, value, str(top_10_pays_aide_alimentaire['Zone'][index]), ha='center', va='top', rotation='vertical', col
for index, value in enumerate(top_10_pays_aide_alimentaire['Valeur'][:6]):
    plt.text(index+6, value, str(top_10_pays_aide_alimentaire['Zone'][index+6]), ha='center', va='bottom', rotation='vertical')
```



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

```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire (en milliers de tonnes) puis groupby sur zone et année
aide_alimentaire_par_annee = aide_alimentaire.pivot_table(index='Année', columns='Zone', values='Valeur', aggfunc='sum')
aide_alimentaire_par_annee = aide_alimentaire_par_annee.reset_index()
aide_alimentaire_par_annee.fillna(0, inplace=True)
aide_alimentaire_par_annee.head()
```

Zone	Année	Afghanistan	Algérie	Angola	Bangladesh	Bhoutan	Bolivie (État plurinational de)
0	2013	128.238	35.234	5.000	131.018	1.724	0.000
1	2014	57.214	18.980	0.014	194.628	0.146	0.006
2	2015	0.000	17.424	0.000	22.542	0.578	0.000
3	2016	0.000	9.476	0.000	0.000	0.218	0.000

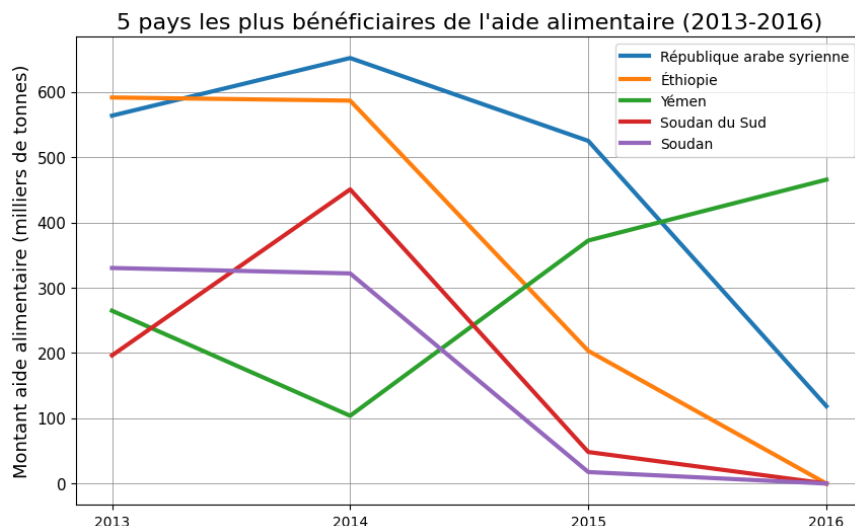
4 rows x 77 columns

```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficié de l'aide alimentaire (milliers de tonnes ?)
top_5_aide_alimentaire = aide_alimentaire_totale.iloc[:5,:].reset_index(drop=True)
liste_5_pays = top_5_aide_alimentaire['Zone'].tolist()
liste_5_pays.append('Année')
```

```
#On filtre sur le dataframe avec notre liste
aide_alimentaire_par_annee_5 = aide_alimentaire_par_annee.filter(items=liste_5_pays)
aide_alimentaire_par_annee_5.head()
```

Zone	République arabe syrienne	Éthiopie	Yémen	Soudan du Sud	Soudan	Année
0	563.566	591.404	264.764	196.330	330.230	2013
1	651.870	586.624	103.840	450.610	321.904	2014
2	524.949	203.266	372.306	48.308	17.650	2015
3	118.558	0.000	465.574	0.000	0.000	2016

```
# Affichage des pays avec l'aide alimentaire par année
plt.figure(figsize=(10, 6))
for i in liste_5_pays[5]:
    plt.plot(aide_alimentaire_par_annee_5['Année'], aide_alimentaire_par_annee_5[i], label=i, linewidth=3)
plt.legend(loc='upper right')
plt.ylabel('Montant aide alimentaire (milliers de tonnes)', fontsize=13)
plt.yticks(fontsize=11)
plt.title("5 pays les plus bénéficiaires de l'aide alimentaire (2013-2016)", fontsize=16)
plt.grid(color='gray', linestyle='-', linewidth=0.5)
plt.xticks(np.arange(2013, 2017, step=1))
plt.show()
```

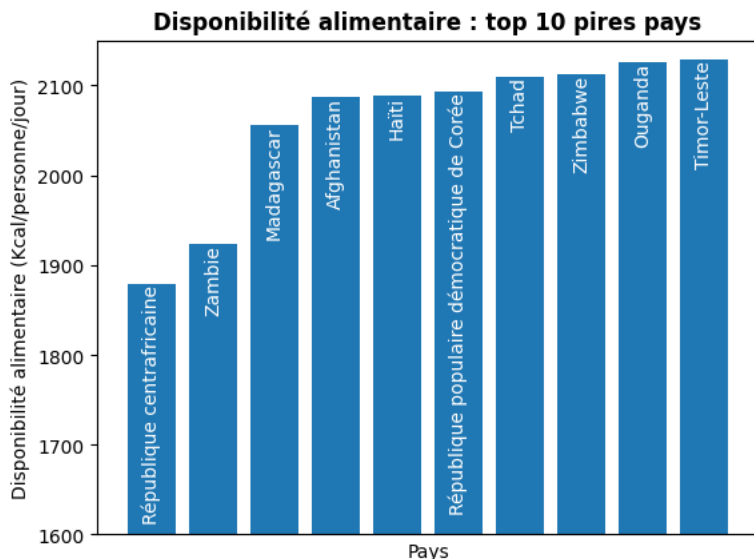


3.9 - Pays avec le moins de disponibilité par habitant

```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo_alimentaire_par_jour = dispo_alimentaire.groupby('Zone')['Disponibilité alimentaire (Kcal/personne/jour)'].sum().reset_index()
dispo_alimentaire_par_jour.head()
```

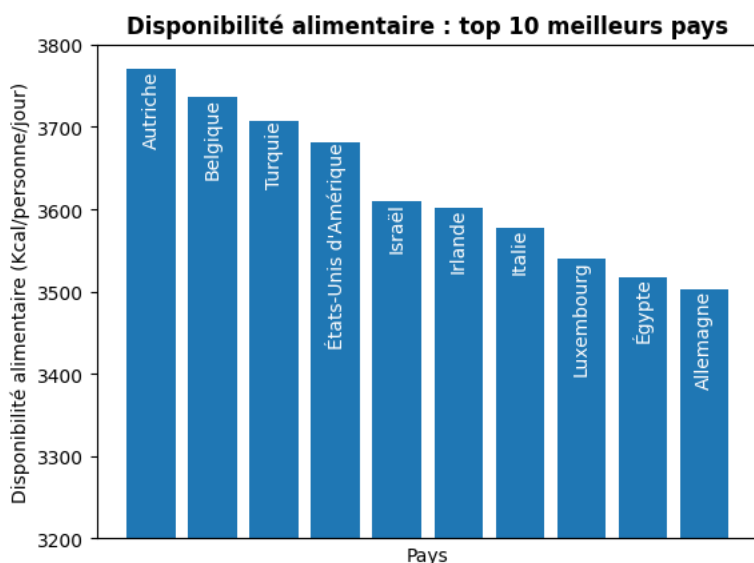
Zone	Disponibilité alimentaire (Kcal/personne/jour)
0 Afghanistan	2087.0
1 Afrique du Sud	3020.0
2 Albanie	3188.0
3 Algérie	3293.0
4 Allemagne	3503.0

```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
#création df avec top 10 pire pays uniquement
bottom_10_dispo_alimentaire = dispo_alimentaire_par_jour.sort_values('Disponibilité alimentaire (Kcal/personne/jour)',ascend)
#création diagramme à barres
plt.bar(height=bottom_10_dispo_alimentaire['Disponibilité alimentaire (Kcal/personne/jour)'], x=bottom_10_dispo_alimentaire['Zone'])
plt.title('Disponibilité alimentaire : top 10 pires pays', fontweight='bold')
plt.xlabel('Pays')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)')
plt.ylim([1600, 2150])
plt.xticks([])
for index, value in enumerate(bottom_10_dispo_alimentaire['Disponibilité alimentaire (Kcal/personne/jour)']):
    plt.text(index, value, str(bottom_10_dispo_alimentaire['Zone'][index]), ha='center', va='top', rotation='vertical', colc
```



3.10 - Pays avec le plus de disponibilité par habitant

```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
#création df avec top 10 meilleurs pays uniquement
top_10_dispo_alimentaire = dispo_alimentaire_par_jour.sort_values('Disponibilité alimentaire (Kcal/personne/jour)',ascending)
#création diagramme à barres
plt.bar(height=top_10_dispo_alimentaire['Disponibilité alimentaire (Kcal/personne/jour)'], x=top_10_dispo_alimentaire['Zone'])
plt.title('Disponibilité alimentaire : top 10 meilleurs pays', fontweight='bold')
plt.xlabel('Pays')
plt.ylabel('Disponibilité alimentaire (Kcal/personne/jour)')
plt.xticks(rotation='vertical')
plt.ylim([3200, 3800])
plt.xticks([])
for index, value in enumerate(top_10_dispo_alimentaire['Disponibilité alimentaire (Kcal/personne/jour)']):
    plt.text(index, value, str(top_10_dispo_alimentaire['Zone'][index]), ha='center', va='top', rotation='vertical', color='
```



3.11 - Exemple de la Thaïlande pour le Manioc

```
#création d'un dataframe avec uniquement la Thaïlande
dispo_alimentaire_thai = dispo_alimentaire.loc[dispo_alimentaire['Zone'] == 'Thaïlande', :].reset_index(drop=True)
sous_nutrition_thai = sous_nutrition.loc[sous_nutrition['Zone'] == 'Thaïlande', :].reset_index(drop=True)
population_thai = population.loc[population['Zone'] == 'Thaïlande', :].reset_index(drop=True)
aide_alimentaire_thai = aide_alimentaire.loc[aide_alimentaire['Zone'] == 'Thaïlande', :].reset_index(drop=True)
dispo_alimentaire_thai.head()
```

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)
0	Thaïlande	Abats Comestible	animale	0.0	0.0	3.0
1	Thaïlande	Agrumes, Autres	vegetale	0.0	0.0	0.0
2	Thaïlande	Alcool, non Comestible	vegetale	0.0	358000000.0	0.0
3	Thaïlande	Aliments pour enfants	vegetale	0.0	0.0	2.0
4	Thaïlande	Ananas	vegetale	0.0	0.0	10.0

```
#Calcul de la sous nutrition en Thaïlande
sous_nutrition_thai['taux'] = round((sous_nutrition_thai['sous_nutrition'] / population_thai['Population'])*100,2)
print('En 2017, {} % de la population thaïlandaise était en état de sous-nutrition.'.format(sous_nutrition_thai['taux'][4]))
```

En 2017, 8.96 % de la population thaïlandaise était en état de sous-nutrition.

```
# On calcule la proportion exportée (de manioc ?) en fonction de la proportion
dispo_alimentaire_thai_manioc = dispo_alimentaire_thai.loc[dispo_alimentaire_thai['Produit'] == 'Manioc', :]
exports_manioc = dispo_alimentaire_thai_manioc.loc[:, 'Exportations - Quantité'].sum()
imports_manioc = dispo_alimentaire_thai_manioc.loc[:, 'Importations - Quantité'].sum()
prod_manioc = dispo_alimentaire_thai_manioc.loc[:, 'Production'].sum()
dispo_habitant_manioc_thai = round(dispo_alimentaire_thai_manioc.loc[:, 'Disponibilité alimentaire (Kcal/personne/jour)'].sum() / population_thai['Population'].sum(), 2)
dispo_interieure_manioc_thai = round(dispo_alimentaire_thai_manioc.loc[:, 'Disponibilité intérieure'].sum() / population_thai['Population'].sum(), 2)
dispo_alimentaire_totale_thai = round(dispo_alimentaire_thai.iloc[:, 5].sum() / population_thai['Population'].sum(), 2)
conso_manioc_animaux = round((dispo_alimentaire_thai_manioc.iloc[:, 3].sum() / (dispo_alimentaire_thai.iloc[:, 3].sum() * 100)) * 100, 2)

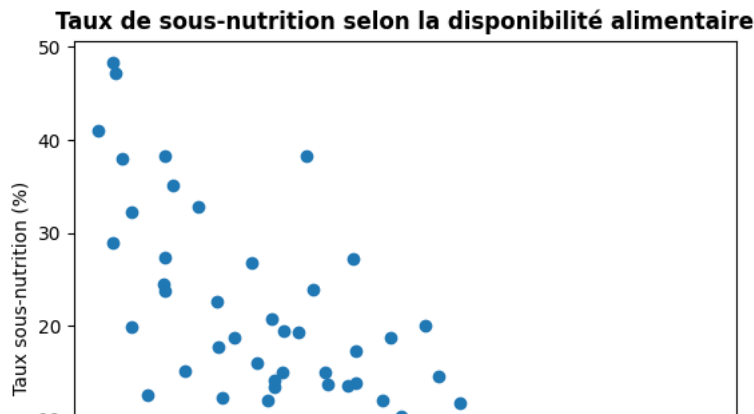
print('En 2017, la production de manioc représentait {} % de la production totale (tous aliments) dans le pays.'.format(round(exports_manioc / prod_manioc * 100, 2)))
print('En 2017, la disponibilité intérieure du manioc représentait {} % de la DI totale (tous aliments) dans le pays.'.format(round(dispo_interieure_manioc_thai * 100, 2)))
print('En 2017, {} % de la production de manioc en Thaïlande a été exportée.'.format(round(exports_manioc / prod_manioc * 100, 2)))
print('En 2017, {} % de la production de manioc en Thaïlande a été importée.'.format(round(imports_manioc / prod_manioc * 100, 2)))
print('En 2017, la disponibilité alimentaire du manioc en Thaïlande était de {} Kcal/personne/jour, alors que la disponibilité alimentaire totale était de {} Kcal/personne/jour.'.format(dispo_habitant_manioc_thai * 1000, dispo_alimentaire_totale_thai * 1000))
print('En 2017, {} % des aliments pour animaux étaient du manioc.'.format(conso_manioc_animaux))
```

En 2017, la production de manioc représentait 14.98 % de la production totale (tous aliments) dans le pays.
 En 2017, la disponibilité intérieure du manioc représentait 3.96 % de la DI totale (tous aliments) dans le pays.
 En 2017, 83.41 % de la production de manioc en Thaïlande a été exportée.
 En 2017, 4.14 % de la production de manioc en Thaïlande a été importée.
 En 2017, la disponibilité alimentaire du manioc en Thaïlande était de 40 Kcal/personne/jour, alors que la disponibilité alimentaire totale était de 1000 Kcal/personne/jour.
 En 2017, 18.83 % des aliments pour animaux étaient du manioc.

Etape 6 - Analyse complémentaires

```
#Rajouter en dessous toutes les analyses complémentaires 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"
# cas2 : étudier corrélation disponibilité alimentaire / taux sous-nutrition (échelle:monde)
dispo_alimentaire_sous_nutrition = pd.merge(dispo_alimentaire_par_jour, pop_sous_nutrition_2017, on='Zone', how='outer')
dispo_alimentaire_sous_nutrition.fillna(0, inplace=True)
dispo_alimentaire_sous_nutrition.head()
plt.scatter(dispo_alimentaire_sous_nutrition['Disponibilité alimentaire (Kcal/personne/jour)'], dispo_alimentaire_sous_nutrition['Taux sous-nutrition (%)'])
plt.xlim([2000, 3500])
plt.title('Taux de sous-nutrition selon la disponibilité alimentaire', fontweight='bold')
plt.xlabel('Disponibilité alimentaire (Kcal/personne/jour)')
plt.ylabel('Taux sous-nutrition (%)')
```

Text(0, 0.5, 'Taux sous-nutrition (%)')



#cas2 le blé en Afghanistan

```
dispo_alimentaire_afghanistan = dispo_alimentaire.loc[dispo_alimentaire['Zone'] == 'Afghanistan', :].reset_index(drop=True)
dispo_alimentaire_afghanistan_ble = dispo_alimentaire_afghanistan.loc[dispo_alimentaire_afghanistan['Produit'] == 'Blé', :]
prod_ble = dispo_alimentaire_afghanistan_ble.loc[:, 'Production'].sum()
exports_ble = dispo_alimentaire_afghanistan_ble.loc[:, 'Exportations - Quantité'].sum()
imports_ble = dispo_alimentaire_afghanistan_ble.loc[:, 'Importations - Quantité'].sum()
dispo_habitant_afghanistan_ble = round(dispo_alimentaire_afghanistan_ble.loc[:, 'Disponibilité alimentaire (Kcal/personne/jour)'].sum() / dispo_alimentaire_afghanistan_ble.loc[:, 'Population'].sum(), 2)
dispo_interieure_afghanistan_ble = round(dispo_alimentaire_afghanistan_ble.loc[:, 'Disponibilité intérieure'].sum() / dispo_alimentaire_afghanistan_ble.loc[:, 'Population'].sum(), 2)
dispo_alimentaire_totale_afghanistan = round(dispo_alimentaire_afghanistan.iloc[:, 5].sum() / dispo_alimentaire_afghanistan.iloc[:, 5].sum(), 2)
conso_ble_animaux = round((dispo_alimentaire_afghanistan_ble.iloc[:, 3].sum() / (dispo_alimentaire_afghanistan.iloc[:, 3].sum() + dispo_alimentaire_totale_afghanistan * dispo_alimentaire_afghanistan.iloc[:, 5].sum())) * 100, 2)
print('En 2017, la production de blé représentait {} % de la production totale (tous aliments) dans tout le pays.'.format(round((prod_ble / dispo_alimentaire_totale_afghanistan) * 100, 2)))
print('En 2017, la disponibilité intérieure du blé représentait {} % de la DI totale (tous aliments) dans le pays.'.format(round((dispo_interieure_afghanistan_ble / dispo_habitant_afghanistan_ble) * 100, 2)))
print('En 2017, {} % de la production de blé en Afghanistan a été exportée.'.format(round((exports_ble / prod_ble) * 100, 2)))
print('En 2017, {} % de la production de blé en Afghanistan a été importée.'.format(round((imports_ble / prod_ble) * 100, 2)))
print('En 2017, la disponibilité alimentaire du blé en Afghanistan était de {} Kcal/personne/jour, alors que la disponibilité alimentaire totale était de {} Kcal/personne/jour.'.format(dispo_habitant_afghanistan_ble, dispo_habitant_afghanistan))
print('En 2017, {} % des aliments pour animaux étaient du blé.'.format(conso_ble_animaux))
```

En 2017, la production de blé représentait 46.27 % de la production totale (tous aliments) dans tout le pays.

En 2017, la disponibilité intérieure du blé représentait 44.34 % de la DI totale (tous aliments) dans le pays.

En 2017, 0.0 % de la production de blé en Afghanistan a été exportée.

En 2017, 22.69 % de la production de blé en Afghanistan a été importée.

En 2017, la disponibilité alimentaire du blé en Afghanistan était de 1369 Kcal/personne/jour, alors que la disponibilité alimentaire totale était de 1369 Kcal/personne/jour.

En 2017, 0.0 % des aliments pour animaux étaient du blé.