

# 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 ieremv 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

```
#Importation de la librairie Pandas
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
drive.mount('/content/drive')
```

 Mounted at /content/drive

### 1.2 - Chargement des fichiers Excel

```
#Importation du fichier population.csv
path='drive/MyDrive/datas/'
population = pd.read_csv(path+'population.csv')
```

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

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

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

```
population.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1416 entries, 0 to 1415
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  -
 0   Zone    1416 non-null   object
 1   Année   1416 non-null   int64
 2   Valeur  1416 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 33.3+ KB
```

```
#Affichage les 5 premières lignes de la table
print(" Les 5 premières lignes de la table 'population' : \n")
population.head()
```

```
Les 5 premières lignes de la table 'population' :
```

	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
```

Étapes suivantes :

[Générer du code avec population](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

```
#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"] *= 1000
```

```
#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	Valeur
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

Étapes suivantes :

[Générer du code avec population](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

## 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
print(f"Nombre de colonne : {dispo_alimentaire.shape[1]}")
```

```
Nombre de colonne : 18
```

```
#Affichage les 5 premières lignes de la table
print("Les 5 premières lignes :")
dispo_alimentaire.head()
```

```
Les 5 premières lignes :
```

	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)
0	Afghanistan	Abats Comestible	animale	NaN	NaN	5.0	1.72	0.20	0.77
1	Afghanistan	Agrumes, Autres	vegetale	NaN	NaN	1.0	1.29	0.01	0.02
		...							

2	Afghanistan	Aliments pour enfants	vegetale	NaN	NaN	1.0	0.06	0.01	0.03
3	Afghanistan	Ananas	vegetale	NaN	NaN	0.0	0.00	NaN	NaN
4	Afghanistan	Bananes	vegetale	NaN	NaN	4.0	2.70	0.02	0.05

Étapes suivantes :

[Générer du code avec dispo\\_alimentaire](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

```
#remplacement des NaN dans le dataset par des 0
dispo_alimentaire=dispo_alimentaire.fillna(0)
```

```
colonnes_a_convertir = [
    "Disponibilité intérieure",
    "Exportations - Quantité",
    "Importations - Quantité",
    "Nourriture",
    "Pertes",
    "Production",
    "Semences",
    "Traitement",
    "Variation de stock",
    "Aliments pour animaux"
]
```

```
dispo_alimentaire.loc[:, colonnes_a_convertir] *= 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)	Disponibilité alimentaire en quantité (kg/personne/ an)	Disponibilité de matière grasse en quantité (g/ personne/ jour)	Disponibilité de protéines en quantité (g/personne/ jour)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.77
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.03
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.05

Étapes suivantes :

[Générer du code avec dispo\\_alimentaire](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

## 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
print(f"Nombre de colonne : {aide_alimentaire.shape[1]}")
```

Nombre de colonne : 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	



Étapes suivantes :

[Générer du code avec aide\\_alimentaire](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

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

```
#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	



2	Afghanistan	2013	Blé et Farin	15160000
3	Afghanistan	2014	Blé et Farin	15160000
4	Afghanistan	2013	Céréales	40504000

Étapes suivantes :

[Générer du code avec aide\\_alimentaire](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

## 2.3 - Analyse exploratoire du fichier sous nutrition

#Afficher les dimensions du dataset

```
print(f"Le tableau comporte {sous_nutrition.shape[0]} observation(s) ou article(s)")
print(f"Le tableau comporte {sous_nutrition.shape[1]} colonne(s)")
```

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

#Consulter le nombre de colonnes

```
print(f"Nombre de colonne {sous_nutrition.shape[1]}")
```

```
Nombre de colonne 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

Étapes suivantes :

[Générer du code avec sous\\_nutrition](#)[Afficher les graphiques recommandés](#)[New interactive sheet](#)

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



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

	Zone	Année	sous_nutrition
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
...	...	...	...
1213	Zimbabwe	2013-2015	0.0
1214	Zimbabwe	2014-2016	0.0
1215	Zimbabwe	2015-2017	0.0
1216	Zimbabwe	2016-2018	0.0
1217	Zimbabwe	2017-2019	0.0

[1218 rows x 3 columns]

```
#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	

Étapes suivantes :

[Générer du code avec sous\\_nutrition](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

## ▼ Etape 3 - Analyses

### 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



```
population['Année'] = population['Année'].astype(str)
sous_nutrition['Année'] = sous_nutrition['Année'].astype(str)
```

```
# Filtrer les données pour l'année 2017 dans la colonne 'Année' de sous_nutrition
sous_nutrition_2017 = sous_nutrition[sous_nutrition["Année"]=="2016-2018"]
```

```
# Filtrer les données pour l'année 2017 dans la population
population_2017 = population[population["Année"] == '2017']
```

```
# Jointure sur la colonne 'Zone'
pop_sous_nutrition_2017 = pd.merge(population_2017, sous_nutrition_2017, left_on=["Zone"], right_on=["Zone"])
```

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

	Zone	Année_x	Valeur	Année_y	sous_nutrition	
0	Afghanistan	2017	36296113.0	2016-2018	10500000.0	
1	Afrique du Sud	2017	57009756.0	2016-2018	3100000.0	
2	Albanie	2017	2884169.0	2016-2018	100000.0	
3	Algérie	2017	41389189.0	2016-2018	1300000.0	
4	Allemagne	2017	82658409.0	2016-2018	0.0	

Étapes  
suivantes :

[Générer du code avec pop\\_sous\\_nutrition\\_2017](#)
[Afficher les graphiques recommandés](#)
[New interactive sheet](#)

```
#Calcul et affichage du nombre de personnes en état de sous nutrition
pop_sous_nutrition_somme=pop_sous_nutrition_2017["sous_nutrition"].sum()
print(f"En 2017, le nombre de personne en état de sous nutition est de: {round(pop_sous_nutrition_somme):,}.")
```

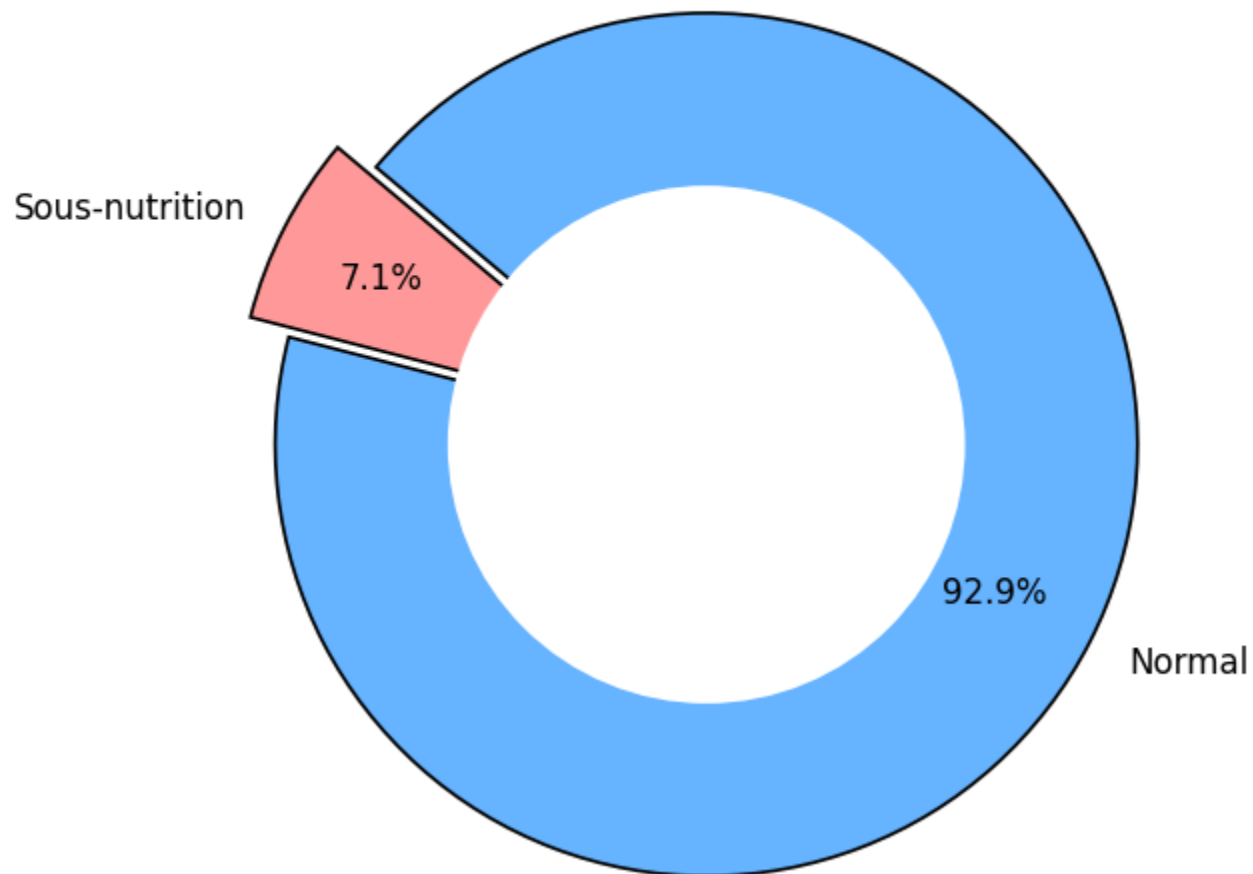
En 2017, le nombre de personne en état de sous nutition est de: 535,700,000.

```
# Calcul des proportions
population_mondiale_2017 = pop_sous_nutrition_2017['Valeur'].sum()
proportion_personne_sous_nutrition = (pop_sous_nutrition_somme / population_mondiale_2017) * 100
proportion_normale = 100 - proportion_personne_sous_nutrition
```

```
# Graphique
labels = ["Sous-nutrition", "Normal"]
sizes = [pop_sous_nutrition_somme, population_mondiale_2017 - pop_sous_nutrition_somme]
colors = ["#ff9999", "#66b3ff"]
explode = (0.1, 0)
plt.figure(figsize=(7, 7))
wedges, texts, autotexts = plt.pie(
```

```
wedges, texts, autotexts = plt.pie(\n    sizes, labels=labels, autopct='%1.1f%%', colors=colors, explode=explode,\n    startangle=140, wedgeprops={"edgecolor": "black", "linewidth": 1.2},\n    textprops={"fontsize": 12}, pctdistance=0.75\n)\n\ncentre_circle = plt.Circle((0, 0), 0.60, fc='white')\nplt.gca().add_artist(centre_circle)\nplt.title("Proportion de personnes en état de sous-nutrition dans le monde en 2017", fontsize=12, fontweight="bold")\nplt.show()
```

### Proportion de personnes 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 ? Source => <https://agriculture.gouv.fr/infographie-la-consommation-alimentaire>

```
print("Une femme a besoin en moyenne de 2000 kcal par jour, tandis qu'un homme a besoin de 2600 kcal par jour.\n"
      "En supposant que la population mondiale est composée d'hommes et de femmes répartis de manière égale :\n"
      "On a besoin de 2300 kcal par jour et par humain en moyenne.")
conso_nourriture_kcl_jour = 2300
conso_nourriture_kcl_an = 2300 * 365
```

Une femme a besoin en moyenne de 2000 kcal par jour, tandis qu'un homme a besoin de 2600 kcal par jour.  
 En supposant que la population mondiale est composée d'hommes et de femmes répartis de manière égale :  
 On a besoin de 2300 kcal par jour et par humain en moyenne.

#On commence par faire une jointure entre le data frame population et Dispo\_alimentaire afin d'ajouter dans ce dernier l

```
pop_dispo_alimentaire_2017 = dispo_alimentaire.merge(population.loc[population['Année'] == '2017'], on='Zone')
```

#Affichage du nouveau dataframe

```
pop_dispo_alimentaire_2017.head()
```

	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)
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.77
		Agaricus							

1	Afghanistan	Agroaliments, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.03
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.05

Étapes  
suivantes :

[Générer du code avec pop\\_dispo\\_alimentaire\\_2017](#)
[Afficher les graphiques recommandés](#)
[New interactive sheet](#)

#Création de la colonne dispo\_kcal avec calcul des kcal disponibles mondialement

```
pop_dispo_alimentaire_2017["dispo_kcal"] = pop_dispo_alimentaire_2017['Disponibilité alimentaire (Kcal/personne/jour)'] * p
print(f'La disponibilité mondial en kcal est de {round(pop_dispo_alimentaire_2017["dispo_kcal"].sum()):,}')
```

La disponibilité mondial en kcal est de 7,635,429,388,975,815

#Calcul du nombre d'humains pouvant être nourris

```
nb_humains_nourris = pop_dispo_alimentaire_2017['dispo_kcal'].sum() / conso_nourriture_kcl_an
print(f"En théorie, la production calorique disponible pourrait nourrir environ {round(nb_humains_nourris):,} personnes, s
```

En théorie, la production calorique disponible pourrait nourrir environ 9,095,210,708 personnes, soit 120.57% de la  
Ainsi, l'ensemble de la population mondiale pourrait être correctement alimenté, compte tenu d'un surplus calorique

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

#Transfert des données avec les végétaux dans un nouveau dataframe

```
pop_dispo_alimentaire_2017_vegetal = pop_dispo_alimentaire_2017[pop_dispo_alimentaire_2017['Origine']=='vegetale']
pop_dispo_alimentaire_2017_vegetal.head(20)
```

	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 protéine en quantité (g/personne/ jour)
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.0
6	Afghanistan	Bière	vegetale	0.0	0.0	0.0	0.09	0.00	0.0
7	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	160.23	4.69	36.9
8	Afghanistan	Boissons Alcooliques	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
9	Afghanistan	Café	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
10	Afghanistan	Coco (Incl Coprah)	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
12	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
13	Afghanistan	Dattes	vegetale	0.0	0.0	0.0	0.02	0.00	0.0
14	Afghanistan	Edulcorants Autres	vegetale	0.0	0.0	2.0	0.53	0.00	0.0
15	Afghanistan	Feve de Cacao	vegetale	0.0	0.0	0.0	0.02	0.04	0.0
16	Afghanistan	Fruits, .	vegetale	0.0	0.0	9.0	7.29	0.06	0.0



		Autres							
17	Afghanistan	Graines de coton	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
18	Afghanistan	Graines de tournesol	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
20	Afghanistan	Huil Plantes Oleif Austr	vegetale	0.0	359.0	2.0	0.08	0.21	0.0
21	Afghanistan	Huile Graines de Coton	vegetale	0.0	1.0	3.0	0.12	0.34	0.0
22	Afghanistan	Huile d'Arachide	vegetale	0.0	0.0	0.0	0.00	0.00	0.0
23	Afghanistan	Huile d'Olive	vegetale	0.0	0.0	0.0	0.02	0.05	0.0

20 rows × 21 columns

#Calcul du nombre de kcal disponible pour les végétaux

```
kcal_dispo_vegetales=pop_dispo_alimentaire_2017_vegetal["dispo_kcal"].sum()
print(f"Les végétaux produits comptent {round(kcal_dispo_vegetales):,} kcal disponibles.")
```

Les végétaux produits comptent 6,300,178,937,197,865 kcal disponibles.

#Calcul du nombre d'humains pouvant être nourris avec les végétaux

```
humains_nourris_vegetaux=kcal_dispo_vegetales/conso_nourriture_kcl_an
```

```
print(f"Les calories issues des végétaux permettraient de nourrir {round(humains_nourris_vegetaux):,} personnes, soit {(
```

Les calories issues des végétaux permettraient de nourrir 7,504,680,092 personnes, soit 99.5 % de la population mond

Ainsi, presque l'ensemble de l'humanité de cette année-là pourrait être alimenté exclusivement par les ressources v

### 3.4 - Utilisation de la disponibilité intérieure

```
#Calcul de la disponibilité totale : Disponibilite totale
Disponibilite_interieure_totale=pop_dispo_alimentaire_2017['Disponibilité intérieure'].sum()
print(f"La disponibilité intérieure totale est de {round(Disponibilite_interieure_totale):,} kcal")
```

La disponibilité intérieure totale est de 9,733,927,000,000 kcal

```
#création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animaux, pertes,
# Donner la part de l'alimentation humaine, animale , perdue(est ce qu'on peut calculer les autres?)
```

```
# Liste des catégories à afficher
liste_col = ["Aliments pour animaux", "Pertes", "Nourriture","Semences","Traitement","Autres Utilisations"]
proportions = []
labels = []
```

```
for col in liste_col :
    proportion=(dispo_alimentaire[col].sum() / Disponibilite_interieure_totale ) * 100
    print(f'Les {col} représente {round(proportion,2)}% dans la disponibilité intérieur de nourriture')
```

```
Les Aliments pour animaux représente 13.4% dans la disponibilité intérieur de nourriture
Les Pertes représente 4.66% dans la disponibilité intérieur de nourriture
Les Nourriture représente 50.1% dans la disponibilité intérieur de nourriture
Les Semences représente 1.59% dans la disponibilité intérieur de nourriture
Les Traitement représente 22.65% dans la disponibilité intérieur de nourriture
Les Autres Utilisations représente 0.0% dans la disponibilité intérieur de nourriture
```

### 3.5 - Utilisation des céréales

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

```
#Création d'un dataframe avec les informations uniquement pour ces céréales
pop_dispo_alimentaire_2017_cereales= pop_dispo_alimentaire_2017[pop_dispo_alimentaire_2017['Produit'].isin(liste_cereale
pop_dispo_alimentaire_2017_cereales.head()
```

	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)	Disponibili de protéin en quanti (g/personn jou
7	Afghanistan	Blé	vegetale	0.0	0.0	1369.0	160.23	4.69	36
12	Afghanistan	Céréales, Autres	vegetale	0.0	0.0	0.0	0.00	0.00	0
32	Afghanistan	Maïs	vegetale	200000000.0	0.0	21.0	2.50	0.30	0
34	Afghanistan	Millet	vegetale	0.0	0.0	3.0	0.40	0.02	0
40	Afghanistan	Orge	vegetale	360000000.0	0.0	26.0	2.92	0.24	0

5 rows × 21 columns

```
#Affichage de la proportion d'alimentation animale
proportion_part_alimentation_animale= ( pop_dispo_alimentaire_2017_cereales["Aliments pour animaux"].sum() / pop_dispo_a
print(f"La part des céréales utilisées pour l'alimentation des animaux est de {round(proportion_part_alimentation_animal
```

La part des céréales utilisées pour l'alimentation des animaux est de 36.14%.

```
proportion_part_alimentation_humaine= ( pop_dispo_alimentaire_2017_cereales["Nourriture"].sum() / pop_dispo_alimentaire_
print(f"La part des céréales utilisées pour l'alimentation humaine est de {round(proportion_part_alimentation_humaine,2)
```

La part des céréales utilisées pour l'alimentation humaine est de 42.91%.

Double-cliquez (ou appuyez sur Entrée) pour modifier

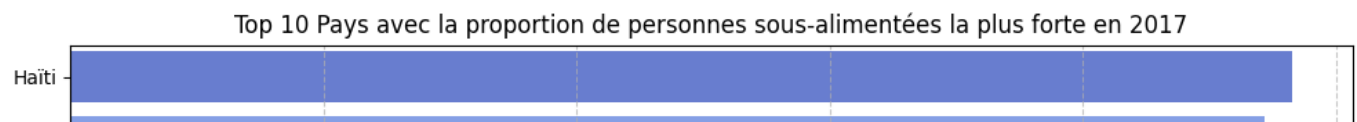
### 3.6 - Pays avec la proportion de personnes sous-alimentée la plus forte en 2017

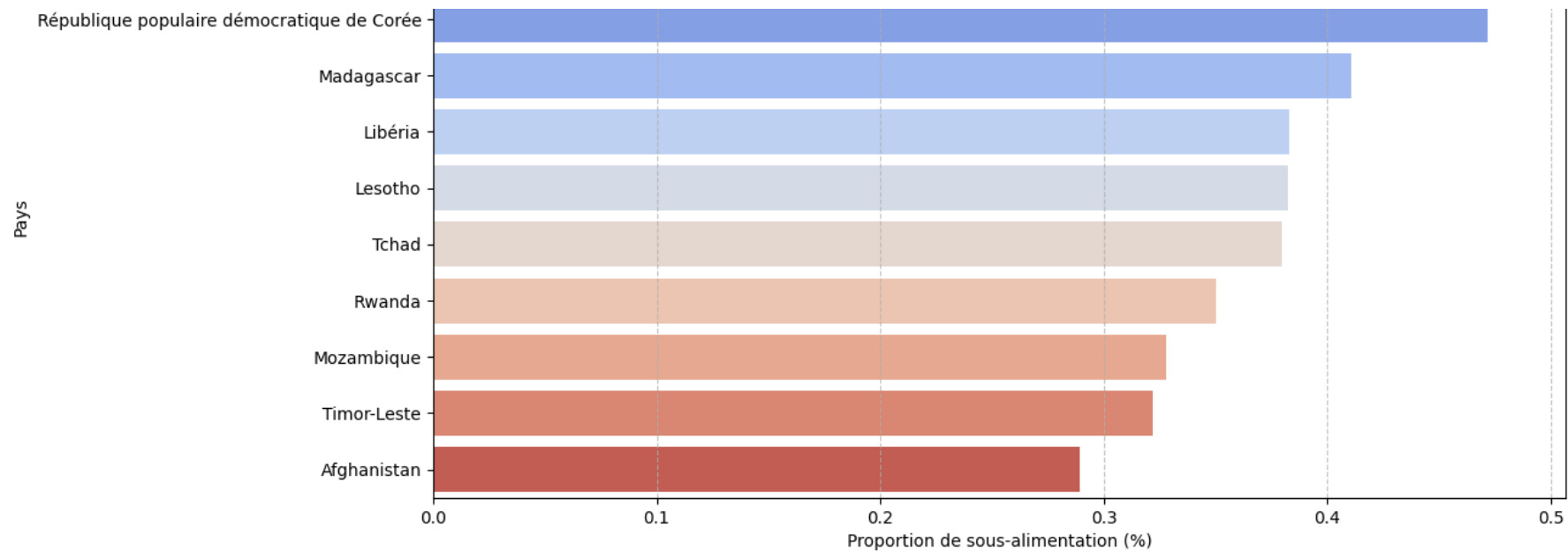
```
#Création de la colonne proportion par pays
pop_sous_nutrition_2017["proportion_par_pays"] = pop_sous_nutrition_2017["sous_nutrition"] / pop_sous_nutrition_2017["Val"]

# Tri des pays ayant la plus forte proportion de sous-alimentation en 2017
pop_sous_nutrition_top10 = pop_sous_nutrition_2017.sort_values(by="proportion_par_pays", ascending=False).head(10)



# Graphique
plt.figure(figsize=(12, 6))
sns.barplot(
    data=pop_sous_nutrition_top10,
    x="proportion_par_pays",
    y="Zone",
    hue="Zone",
    palette="coolwarm"
)

plt.xlabel("Proportion de sous-alimentation (%)")
plt.ylabel("Pays")
plt.title("Top 10 Pays avec la proportion de personnes sous-alimentées la plus forte en 2017")
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.show()
```





```
pop_sous_nutrition_top10[["Zone","sous_nutrition","proportion_par_pays"]]
```

	Zone	sous_nutrition	proportion_par_pays	
78	Haïti	5300000.0	0.482592	
157	République populaire démocratique de Corée	12000000.0	0.471887	
108	Madagascar	10500000.0	0.410629	
103	Libéria	1800000.0	0.382797	
100	Lesotho	800000.0	0.382494	
183	Tchad	5700000.0	0.379576	

<b>161</b>	Rwanda	4200000.0	0.350556
<b>121</b>	Mozambique	9400000.0	0.328109
<b>186</b>	Timor-Leste	400000.0	0.321735
<b>0</b>	Afghanistan	10500000.0	0.289287

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

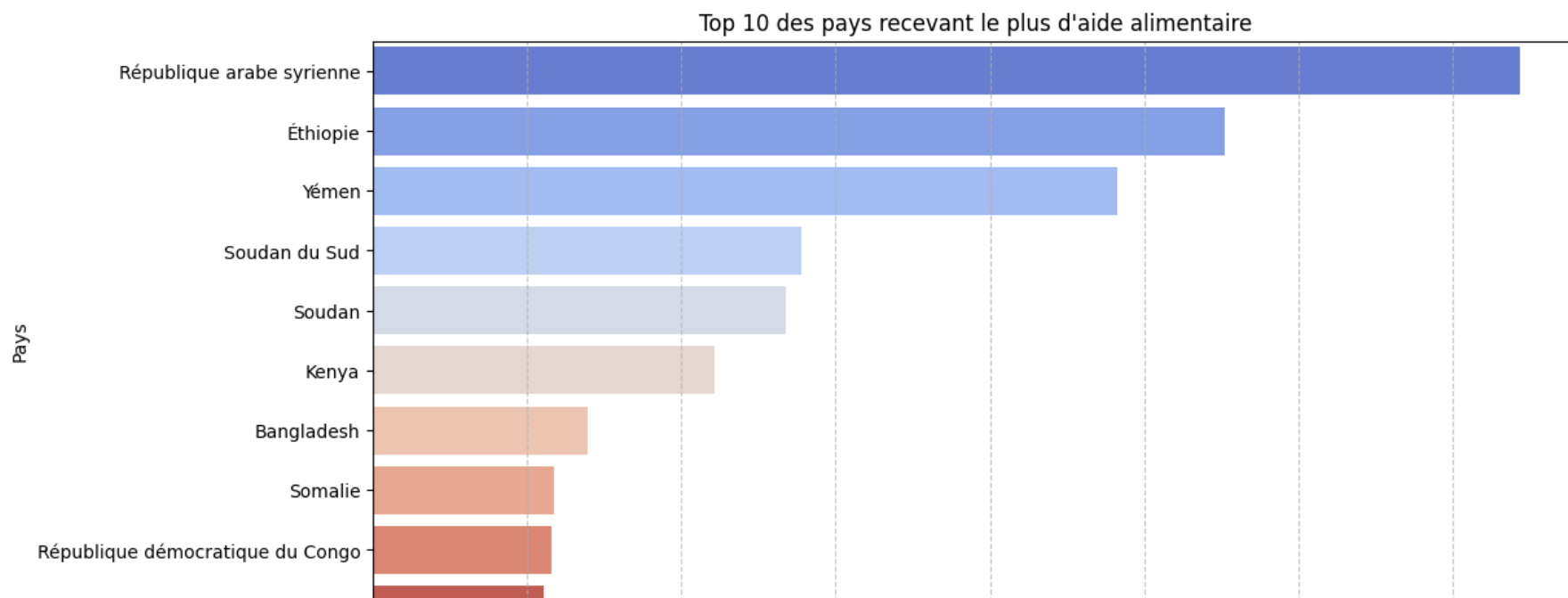
#calcul du total de l'aide alimentaire par pays

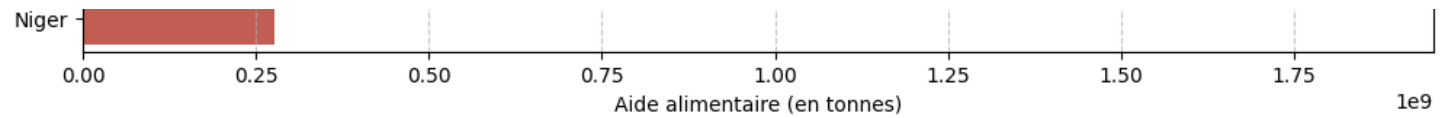
```
aide_alimentaire_par_pays=aide_alimentaire.groupby('Zone',as_index=False).sum()
aide_alimentaire_top10 = aide_alimentaire_par_pays.sort_values('Valeur', ascending=False).head(10)
aide_alimentaire_top10[["Zone","Valeur"]]
```

	Zone	Valeur	
<b>50</b>	République arabe syrienne	1858943000	
<b>75</b>	Éthiopie	1381294000	
<b>70</b>	Yémen	1206484000	
<b>61</b>	Soudan du Sud	695248000	
<b>60</b>	Soudan	669784000	
<b>30</b>	Kenya	552836000	
<b>3</b>	Bangladesh	348188000	
<b>59</b>	Somalie	292678000	
<b>53</b>	République démocratique du Congo	288502000	
<b>43</b>	Niger	276344000	

```
#Graphique
plt.figure(figsize=(12, 6))
sns.barplot(
    data=aide_alimentaire_top10,
    x="Valeur",
    y="Zone",
    hue="Zone",
    dodge=False,
    palette="coolwarm",
    legend=False
)

plt.xlabel("Aide alimentaire (en tonnes)")
plt.ylabel("Pays")
plt.title("Top 10 des pays recevant le plus d'aide alimentaire")
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.show()
```





### 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 puis groupby sur zone et année
aide_alimentaire_par_pays_2=aide_alimentaire[['Zone', 'Année', 'Valeur']].groupby(['Zone', 'Année'],as_index=False)[ 'Valeur'
aide_alimentaire_par_pays_2.head(20)
```

	Zone	Année	Valeur	
0	Afghanistan	2013	128238000	
1	Afghanistan	2014	57214000	
2	Algérie	2013	35234000	
3	Algérie	2014	18980000	
4	Algérie	2015	17424000	
5	Algérie	2016	9476000	
6	Angola	2013	5000000	
7	Angola	2014	14000	
8	Bangladesh	2013	131018000	
9	Bangladesh	2014	194628000	



10	Bangladesh	2015	22542000
11	Bhoutan	2013	1724000
12	Bhoutan	2014	146000
13	Bhoutan	2015	578000
14	Bhoutan	2016	218000
15	Bolivie (État plurinational de)	2014	6000
16	Burkina Faso	2013	18620000
17	Burkina Faso	2014	22938000
18	Burkina Faso	2015	23182000
19	Burkina Faso	2016	72000

Étapes  
suivantes :

[Générer du code avec aide\\_alimentaire\\_par\\_pays\\_2](#)

[Afficher les graphiques recommandés](#)

[New interactive sheet](#)

#Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de l'aide alimentaire

```
pays_top_cinq_aide_alimentaire_liste=aide_alimentaire_par_pays.sort_values('Valeur', ascending=False)[['Zone', 'Valeur']]
pays_top_cinq_aide_alimentaire_liste
```

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

#On filtre sur le dataframe avec notre liste

```
pays_top_cinq_aide_alimentaire=aide_alimentaire_par_pays_2[aide_alimentaire_par_pays_2['Zone'].isin(pays_top_cinq_aide_a
```

# Affichage des pays avec l'aide alimentaire par année

```
pays_top_cinq_aide_alimentaire
```

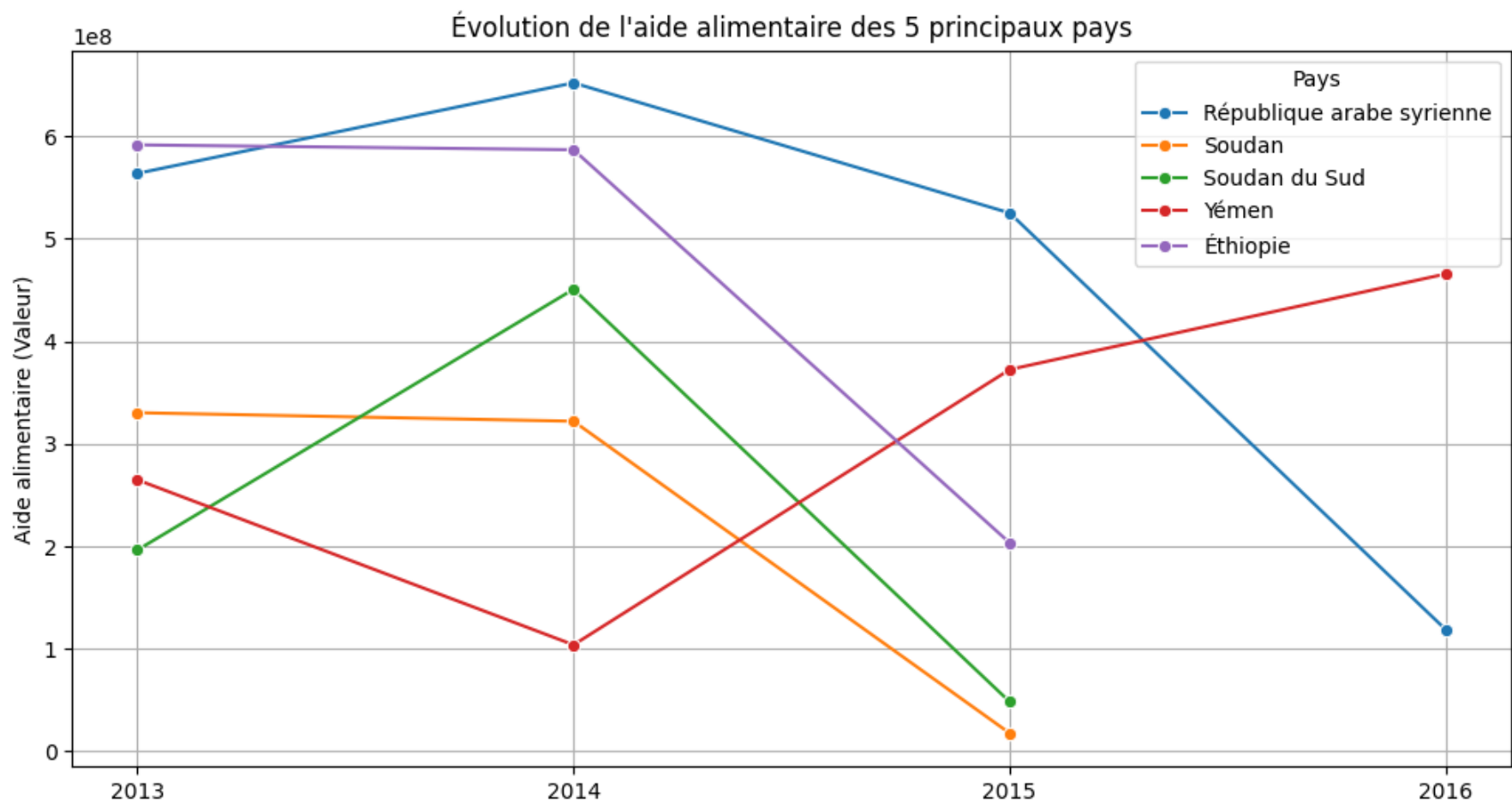
```
plt.figure(figsize=(12, 6))
```

```
sns.lineplot(data=pays_top_cinq_aide_alimentaire, x='Année', y='Valeur', hue='Zone', marker='o')

annees_uniques = sorted(pays_top_cinq_aide_alimentaire['Année'].unique())
plt.xticks(ticks=annees_uniques)

plt.title("Évolution de l'aide alimentaire des 5 principaux pays")
plt.xlabel("Année")
plt.ylabel("Aide alimentaire (Valeur)")
plt.legend(title="Pays")
plt.grid(True)

plt.show()
```



Annee

### 3.9 - Pays avec le moins de disponibilité par habitant

```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne  
dispo_alimentaire.groupby("Zone")['Disponibilité alimentaire (Kcal/personne/jour)'].sum().sort_values(ascending=True).he
```

Disponibilité alimentaire (Kcal/personne/jour)	
Zone	
République centrafricaine	1879.0
Zambie	1924.0
Madagascar	2056.0
Afghanistan	2087.0
Haïti	2089.0
République populaire démocratique de Corée	2093.0
Tchad	2109.0
Zimbabwe	2113.0
Ouganda	2126.0
Timor-Leste	2129.0

dtype: float64

### 3.10 - Pays avec le plus de disponibilité par habitant

```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
```

```
dispo_alimentaire_top10 = dispo_alimentaire.groupby("Zone")['Disponibilité alimentaire (Kcal/personne/jour)']\
    .sum().sort_values(ascending=False).head(10)
```

```
zones = dispo_alimentaire_top10.index
valeurs = dispo_alimentaire_top10.values
```

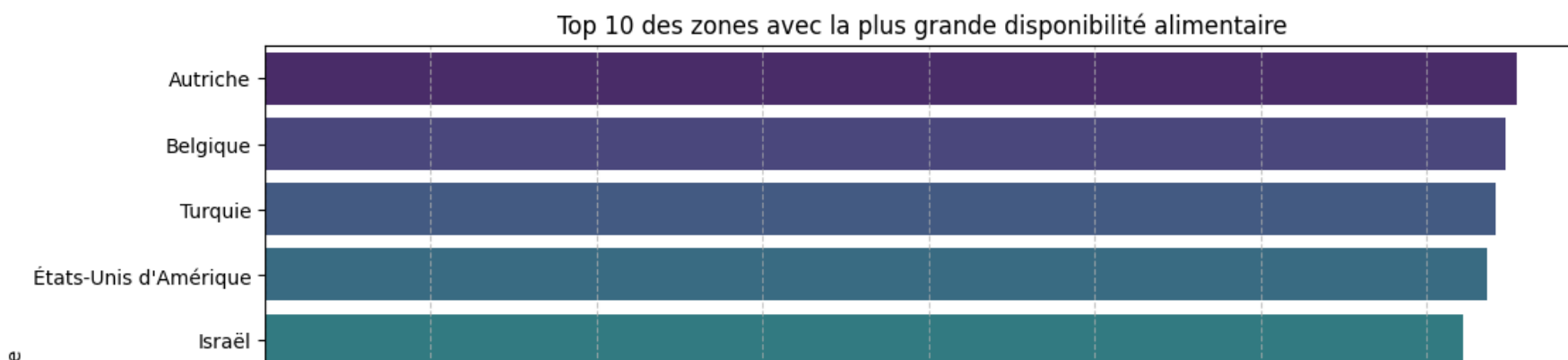
```
plt.figure(figsize=(12, 6))
sns.barplot(
    x=valeurs,
    y=zones,
    palette="viridis"
)
```

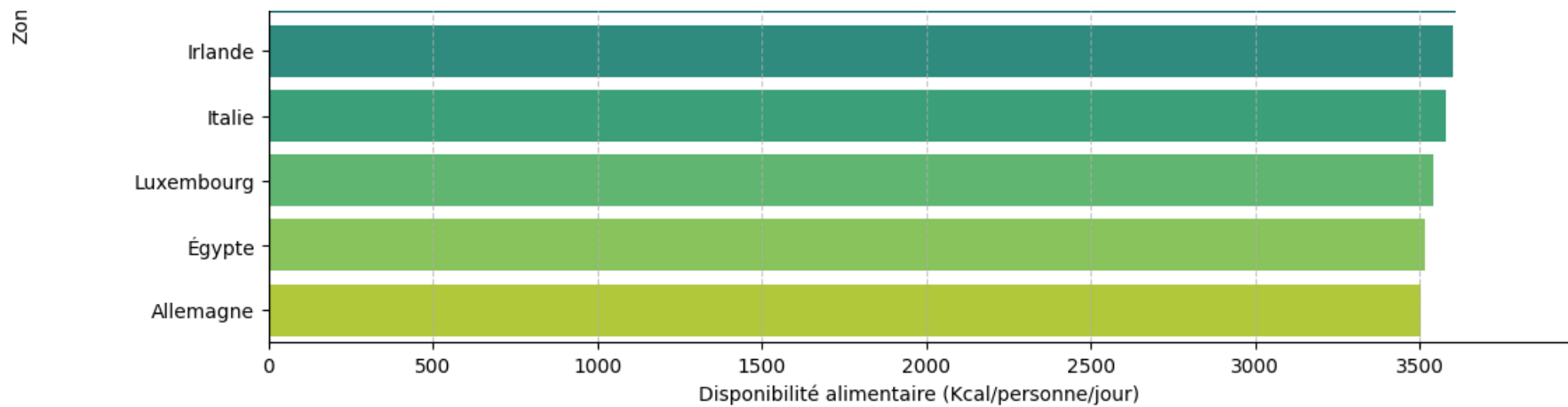
```
plt.xlabel("Disponibilité alimentaire (Kcal/personne/jour)")
plt.ylabel("Zone")
plt.title("Top 10 des zones avec la plus grande disponibilité alimentaire")
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.show()
```

<ipython-input-57-c45867796028>:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `

```
sns.barplot(
```





### ✓ 3.101 -Pays avec le moins de disponibilité par **habitant**

```
dispo_alimentaire_top10less = dispo_alimentaire.groupby("Zone")['Disponibilité alimentaire (Kcal/personne/jour)'].sum().
```

```
zones=dispo_alimentaire_top10less.index
valeurs = dispo_alimentaire_top10less.values
```

```
plt.figure(figsize=(12, 6))
sns.barplot(
    x=valeurs,
    y=zones,
    palette="mako"
)
```

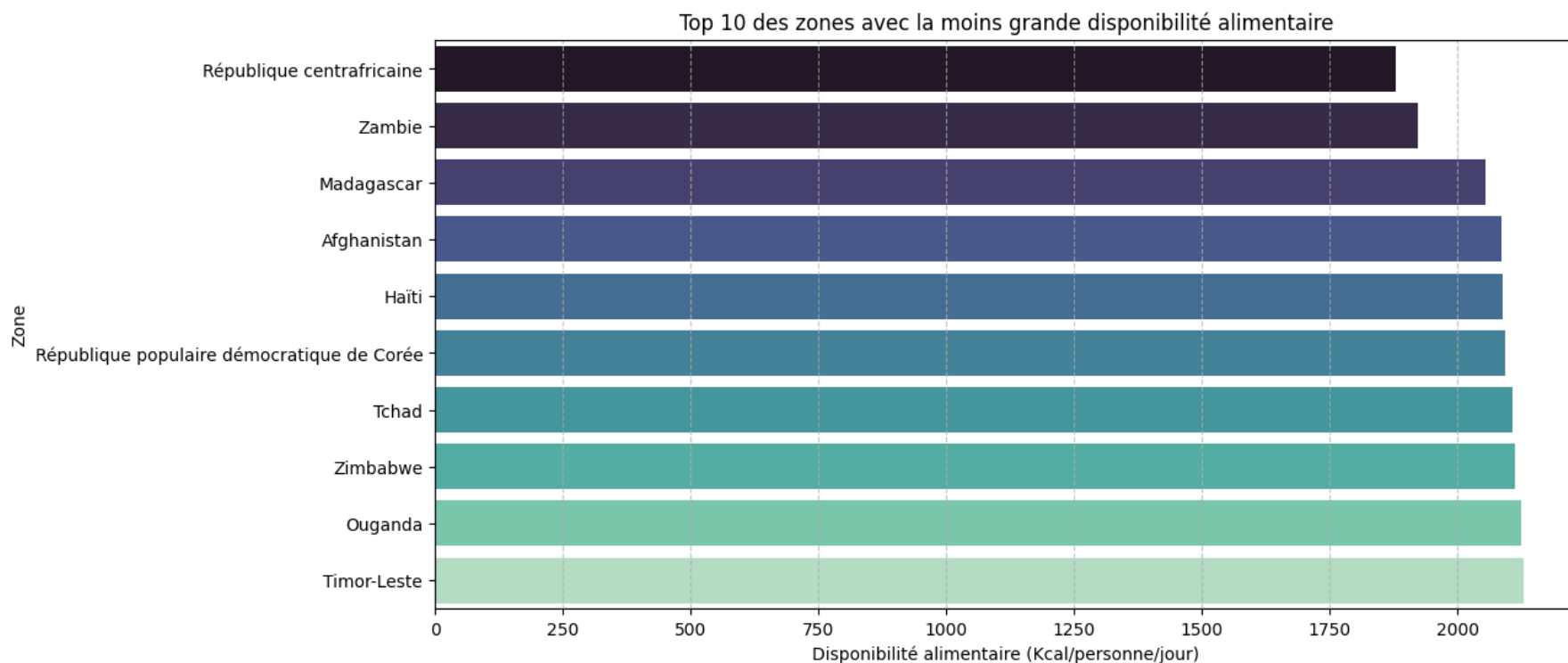
```
plt.xlabel("Disponibilité alimentaire (Kcal/personne/jour)")
```

```
plt.xlabel("Disponibilité alimentaire (Kcal/personne/jour) ")
plt.ylabel("Zone")
plt.title("Top 10 des zones avec la moins grande disponibilité alimentaire")
plt.grid(axis="x", linestyle="--", alpha=0.7)
plt.show()
```

<ipython-input-58-e03bec51ab26>:8: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `

sns.barplot(



### 3.11 - Exemple de la Thaïlande pour le Manioc

```
#création d'un dataframe avec uniquement la Thaïlande
pop_sous_nutrition_Thai=pop_sous_nutrition_2017 [(pop_sous_nutrition_2017["Zone"] == "Thaïlande")]
pop_sous_nutrition_Thai.head()
```

	Zone	Année_x	Valeur	Année_y	sous_nutrition	proportion_par_pays
<b>185</b>	Thaïlande	2017	69209810.0	2016-2018	6200000.0	0.089583



```
#Calcul de la sous nutrition en Thaïlande
```

```
sous_nutrition_2017_Thai = pop_sous_nutrition_Thai["sous_nutrition"].iloc[0]
population_2017_Thai=pop_sous_nutrition_Thai["Valeur"].iloc[0]
print(f"Sur une population de {round(population_2017_Thai):,} habitants, nous trouvons {round(sous_nutrition_2017_Thai):,} individus en sous nutrition."
print(f"Ainsi {round( (sous_nutrition_2017_Thai / population_2017_Thai) * 100 ,2)}% de la population Thaïlandaise est to
```

Sur une population de 69,209,810 habitants, nous trouvons 6,200,000 d'individu en état de sous nutrition.  
Ainsi 8.96% de la population Thaïlandaise est touchée.

```
dispo_alimentaire_Manioc=dispo_alimentaire[dispo_alimentaire["Produit"]=="Manioc"]
```

```
dispo_alimentaire_Thai=dispo_alimentaire[dispo_alimentaire["Zone"]=="Thaïlande"]
dispo_alimentaire_thai_par_habitant= dispo_alimentaire_Thai["Disponibilité intérieure"] / population_2017_Thai
```

```
dispo_alimentaire_thai_par_habitant.head()
```

Disponibilité intérieure	
13759	1.069213
13760	0.115591
13761	5.172677
13762	0.173386
13763	11.298976

**dtype:** float64

```
dispo_alimentaire_Thai_Manioc=dispo_alimentaire_Thai[dispo_alimentaire_Thai["Produit"]=="Manioc"]
ratio_exportation_manioc = (dispo_alimentaire_Thai_Manioc["Exportations - Quantité"] / dispo_alimentaire_Thai_Manioc["Pr
```

```
ratio_exportation_manioc = ratio_exportation_manioc.values[0]
```

```
print(f"La Thaïlande exporte {ratio_exportation_manioc:.2f} % de sa production de manioc,ce qui témoigne de son orientat
```

La Thaïlande exporte 83.41 % de sa production de manioc,ce qui témoigne de son orientation vers le marché internatio  
Cela suggère que la culture du manioc dans le pays est principalement destinée à l'exportation plutôt qu'à la consom

## Etape 6 - Analyse complémentaires

### ✓ 6.1 - Population pouvant être nourris avec les pertes

```
Pertes_2017=pop_dispo_alimentaire_2017['Pertes'].sum()
population_nourries_pertes=round(Pertes_2017/conso_nourriture_kcl_an,2)
print(f"Les pertes pourraient nourrir {round(population_nourries_pertes):,} personnes.")
print(f"Cela représente {round((population_nourries_pertes/population_mondiale_2017)*100,2)}% de la population mondiale"
```

```
Les pertes pourraient nourrir 538 753 personnes
```



Les pertes pourraient nourrir 558,755 personnes.  
Cela représente 0.01% de la population mondiale