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'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

In [1]:
```
#Importation de la librairie Pandas
import pandas as pd
#Importation de la librairie Seaborn
import seaborn as sns
#Importation de la librairie Matplotlib
import matplotlib.pyplot as plt
```

1.2 - Chargement des fichiers Excel

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

#Importation du fichier dispo_alimentaire.csv
```

```
dispo = pd.read_csv('dispo_alimentaire.csv')

#Importation du fichier aide_alimentaire.csv
aide = 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

In [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)
```

In [4]:
```
#Consulter le nombre de colonnes
#La nature des données dans chacune des colonnes
print('La nature des données dans chaque colonnes :\n',population.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
print('Nombre de valeurs présentes dans chacune des colonnes :\n',population.value_counts())
```

```
La nature des données dans chaque colonnes :
 Zone       object
Année       int64
Valeur    float64
dtype: object
Nombre de valeurs présentes dans chacune des colonnes :
 Zone                        Année  Valeur
Afghanistan                 2013   32269.589    1
République dominicaine      2014   10165.183    1
République arabe syrienne   2014   18710.711    1
                            2013   19578.461    1
Rwanda                      2018   12301.970    1
                                                ..
Guyana                      2015   767.432      1
                            2014   763.380      1
                            2013   759.285      1
Guinée-Bissau               2018   1874.303     1
Îles Wallis-et-Futuna       2018   11.661       1
Length: 1416, dtype: int64
```

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

Out[5]:

	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

In [6]:
```
#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
```

In [7]:
```
#changement du nom de la colonne Valeur par Population
population=population.rename(columns={'Valeur': 'Population'})
```

In [8]:
```
#Affichage les 5 premières lignes de la table pour voir les modifications
population.head()
```

Out[8]:

	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

In [9]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(dispo.shape[0]))
print("Le tableau comporte {} colonne(s)".format(dispo.shape[1]))
```

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

In [10]:
```
#Consulter le nombre de colonnes
#La nature des données dans chacune des colonnes
print('La nature des données dans chaque colonnes :\n',dispo.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
print('Nombre de valeurs présentes dans chacune des colonnes :\n',dispo.info())
```

```
La nature des données dans chaque colonnes :
 Zone                                                 object
Produit                                               object
Origine                                               object
Aliments pour animaux                                float64
Autres Utilisations                                  float64
Disponibilité alimentaire (Kcal/personne/jour)       float64
Disponibilité alimentaire en quantité (kg/personne/an)  float64
Disponibilité de matière grasse en quantité (g/personne/jour)  float64
Disponibilité de protéines en quantité (g/personne/jour)  float64
Disponibilité intérieure                             float64
Exportations - Quantité                              float64
Importations - Quantité                              float64
Nourriture                                           float64
Pertes                                               float64
Production                                           float64
Semences                                             float64
Traitement                                           float64
Variation de stock                                   float64
dtype: object
<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
Nombre de valeurs présentes dans chacune des colonnes :
 None
```

In [11]:
```
#Affichage les 5 premières lignes de la table
dispo.head()
```

Out[11]:

	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)	Dis i
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	

In [12]:
```
#remplacement des NaN dans le dataset par des 0
dispo = dispo.fillna(0)
```

In [13]:
```
#multiplication de toutes les lignes contenant des milliers de tonnes en Kg
dispo.iloc[:,9:]=dispo.iloc[:,9:]*1000000
dispo.iloc[:,[3,4]]=dispo.iloc[:,[3,4]]*1000000
```

In [14]:
```
#Affichage les 5 premières lignes de la table
dispo.head()
```

Out[14]:

	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)	Dis i
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.77	5
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02	4
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	8

2.3 - Analyse exploratoire du fichier aide alimentaire

In [15]:
```
#Afficher les dimensions du dataset
print("Le tableau comporte {} observation(s) ou article(s)".format(aide.shape[0]))
print("Le tableau comporte {} colonne(s)".format(aide.shape[1]))
```

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

In [16]:
```
#Consulter le nombre de colonnes
#La nature des données dans chacune des colonnes
print('La nature des données dans chaque colonnes :\n',aide.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
print('Nombre de valeurs présentes dans chacune des colonnes :\n',aide.info())
```

```
La nature des données dans chaque colonnes :
 Pays bénéficiaire    object
Année                 int64
Produit              object
Valeur                int64
dtype: object
<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
Nombre de valeurs présentes dans chacune des colonnes :
 None
```

In [17]:
```
#Affichage les 5 premières lignes de la table
aide.head()
```

Out[17]:

	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

In [18]:
```
#changement du nom de la colonne Pays bénéficiaire par Zone et valeur par Aide_alimentaire
aide=aide.rename(columns={'Pays bénéficiaire': 'Zone','Valeur':'Aide_alimentaire'})
```

In [19]:
```
#Multiplication de la colonne Aide_alimentaire qui contient des tonnes par 1000 pour avoir des kg
aide['Aide_alimentaire']=aide['Aide_alimentaire']*1000
```

In [20]:
```
#Affichage les 5 premières lignes de la table
aide.head()
```

Out[20]:

	Zone	Année	Produit	Aide_alimentaire
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

In [21]:
```
#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)
```

In [22]:
```
#Consulter le nombre de colonnes
#La nature des données dans chacune des colonnes
print('La nature des données dans chaque colonnes :\n',sous_nutrition.dtypes)
#Le nombre de valeurs présentes dans chacune des colonnes
print('Nombre de valeurs présentes dans chacune des colonnes :\n',sous_nutrition.value_counts())
```

```
La nature des données dans chaque colonnes :
 Zone      object
Année     object
Valeur    object
dtype: object
Nombre de valeurs présentes dans chacune des colonnes :
 Zone          Année      Valeur
Afghanistan   2012-2014  8.6       1
Ouzbékistan   2017-2019  0.8       1
Oman          2016-2018  0.4       1
              2017-2019  0.4       1
Ouzbékistan   2012-2014  2         1
                                   ..
Guatemala     2017-2019  2.8       1
Guyana        2012-2014  <0.1      1
              2013-2015  <0.1      1
              2014-2016  <0.1      1
Îles Salomon  2017-2019  <0.1      1
Length: 624, dtype: int64
```

In [23]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

Out[23]:

	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

In [24]:
```
#Conversion de la colonne sous nutrition en numérique
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'],errors='coerce')
```

In [25]:
```
#Conversion de la colonne (avec l'argument errors=coerce qui permet de convertir automatiquement les lignes qui ne s
#Puis remplacement des NaN en 0
sous_nutrition = sous_nutrition.fillna(0)
```

In [26]:
```
#changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition=sous_nutrition.rename(columns={'Valeur':'sous_nutrition'})
```

In [27]:
```
#Multiplication de la colonne sous_nutrition par 1000000
sous_nutrition['sous_nutrition']=sous_nutrition['sous_nutrition']*1000000
```

In [28]:
```
#Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

Out[28]:

	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

In [29]:
```
# Il faut tout d'abord faire une jointure entre la table population et la table sous nutrition, en ciblant l'année
personnes_sous_nutrition=pd.merge(sous_nutrition.loc[sous_nutrition['Année']=='2016-2018',['Zone','sous_nutrition']
```

In [30]:
```
#Affichage du dataset
display(personnes_sous_nutrition)
```

	Zone	sous_nutrition	Année	Population
0	Afghanistan	10500000.0	2017	36296113.0
1	Afrique du Sud	3100000.0	2017	57009756.0
2	Albanie	100000.0	2017	2884169.0
3	Algérie	1300000.0	2017	41389189.0
4	Allemagne	0.0	2017	82658409.0
...	...	...	...	...
198	Venezuela (République bolivarienne du)	8000000.0	2017	29402484.0
199	Viet Nam	6500000.0	2017	94600648.0
200	Yémen	0.0	2017	27834819.0
201	Zambie	0.0	2017	16853599.0
202	Zimbabwe	0.0	2017	14236595.0

203 rows × 4 columns

In [31]:
```
#Calcul et affichage du nombre de personnes en état de sous nutrition
mask = personnes_sous_nutrition.groupby('Année').sum()
#insertion des chiffres dans des variables
sous_nutrition_totale=mask.iloc[0,0]
population_totale=mask.iloc[0,1]
print('Nombre de personnes en sous nutrition dans le monde : ',sous_nutrition_totale)
print('Proportion de la population mondiale en sous nutrition dans le monde : ',round(100*sous_nutrition_totale/po
```

```
Nombre de personnes en sous nutrition dans le monde :  535700000.0
Proportion de la population mondiale en sous nutrition dans le monde :  7.1 %
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\4020741850.py:2: FutureWarning: The default value of numeric_only
in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify num
eric_only or select only columns which should be valid for the function.
  mask = personnes_sous_nutrition.groupby('Année').sum()
```

In [32]:
```
#Recherche d'un équivalent en termes de population d'un pays avec un focus sur 2017
#Et insertion dans un dataframe
equivalent_sous_nutrition=population.loc[(population['Population']<=(sous_nutrition_totale))&(population['Année']==
#Affichage du dataframe
```

```
display(equivalent_sous_nutrition)

#Vérification le la population correspondante
#insertion dans une variable
diff_population=abs(equivalent_sous_nutrition.loc["États-Unis d'Amérique",'Population']+equivalent_sous_nutrition.
print('Différence de population entre Etats-Unis+Pakistan avec la population totale en sous-nutrititon',diff_popula
```

Zone	Année	Population
États-Unis d'Amérique	2017	325084756.0
Indonésie	2017	264650963.0
Pakistan	2017	207906209.0
Brésil	2017	207833823.0
Nigéria	2017	190873244.0

```
Différence de population entre Etats-Unis+Pakistan avec la population totale en sous-nutrititon 2709035.0 habitant
s ( 0.04 % de la population mondiale).
```

3.2 - Nombre théorique de personne qui pourrait être nourries

In [33]:
```
#Combien mange en moyenne un être humain ? 2200kcal Source => https://www.data.gouv.fr/fr/reuses/calculateur-de-c
```

In [34]:
```
#On commence par faire une jointure entre le data frame population et Dispo_alimentaire afin d'ajouter dans ce de
nb_personnes_nourrissable=pd.merge(population,dispo,on='Zone')
nb_personnes_nourrissable=nb_personnes_nourrissable.loc[nb_personnes_nourrissable['Année']==2017,:]
```

In [35]:
```
#Affichage du nouveau dataframe
display(nb_personnes_nourrissable)
```

	Zone	Année	Population	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponil matière gr c (g/personn
240	Afghanistan	2017	36296113.0	Abats Comestible	animale	0.0	0.0	5.0	1.72	
241	Afghanistan	2017	36296113.0	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	
242	Afghanistan	2017	36296113.0	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	
243	Afghanistan	2017	36296113.0	Ananas	vegetale	0.0	0.0	0.0	0.00	
244	Afghanistan	2017	36296113.0	Bananes	vegetale	0.0	0.0	4.0	2.70	
...	...	...	...	...	...	...	...	...	...	
92399	Zimbabwe	2017	14236595.0	Viande de Suides	animale	0.0	0.0	24.0	2.65	
92400	Zimbabwe	2017	14236595.0	Viande de Volailles	animale	0.0	0.0	17.0	4.97	
92401	Zimbabwe	2017	14236595.0	Viande, Autre	animale	0.0	1000000.0	7.0	2.29	
92402	Zimbabwe	2017	14236595.0	Vin	vegetale	0.0	0.0	1.0	0.27	
92403	Zimbabwe	2017	14236595.0	Épices, Autres	vegetale	0.0	0.0	1.0	0.06	

15416 rows × 20 columns

In [36]:
```
#Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement
nb_personnes_nourrissable['dispo_kcal']=nb_personnes_nourrissable['Population']*nb_personnes_nourrissable['Dispon
```

In [37]:
```
#Calcul du nombre d'humains pouvant être nourris (retrait des colonnes aliments pour animaux et autres utilisatio
nourrissable=round(nb_personnes_nourrissable['dispo_kcal'].sum()/2200,0)
print('Un total de ',nourrissable,' personnes peuvent être nourries grâce à la disponnibilité alimentaire mondial
```

```
Un total de  9508629376.0  personnes peuvent être nourries grâce à la disponnibilité alimentaire mondiale.
```

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

In [38]:
```
#Transfert des données avec les végétaux dans un nouveau dataframe
nb_personnes_nourrissable_vegetaux=nb_personnes_nourrissable.loc[nb_personnes_nourrissable['Origine']=='vegetale
```

In [39]:
```
#Calcul du nombre de kcal disponible pour les végétaux
kcal_vegetaux=nb_personnes_nourrissable_vegetaux['dispo_kcal'].sum()
```

In [40]:
```
#Calcul du nombre d'humains pouvant être nourris avec les végétaux
nourrissable_vegetaux=round(kcal_vegetaux/2200,0)
print('Un total de ',nourrissable_vegetaux,' personnes peuvent être nourries grâce à la disponnibilité alimentai
```

```
Un total de  7845801914.0  personnes peuvent être nourries grâce à la disponnibilité alimentaire végétale mondi
ale.
```

3.4 - Utilisation de la disponibilité intérieure

In [41]:
```
#Calcul de la disponibilité totale
dispo_totale=dispo.groupby('Zone').sum()
```

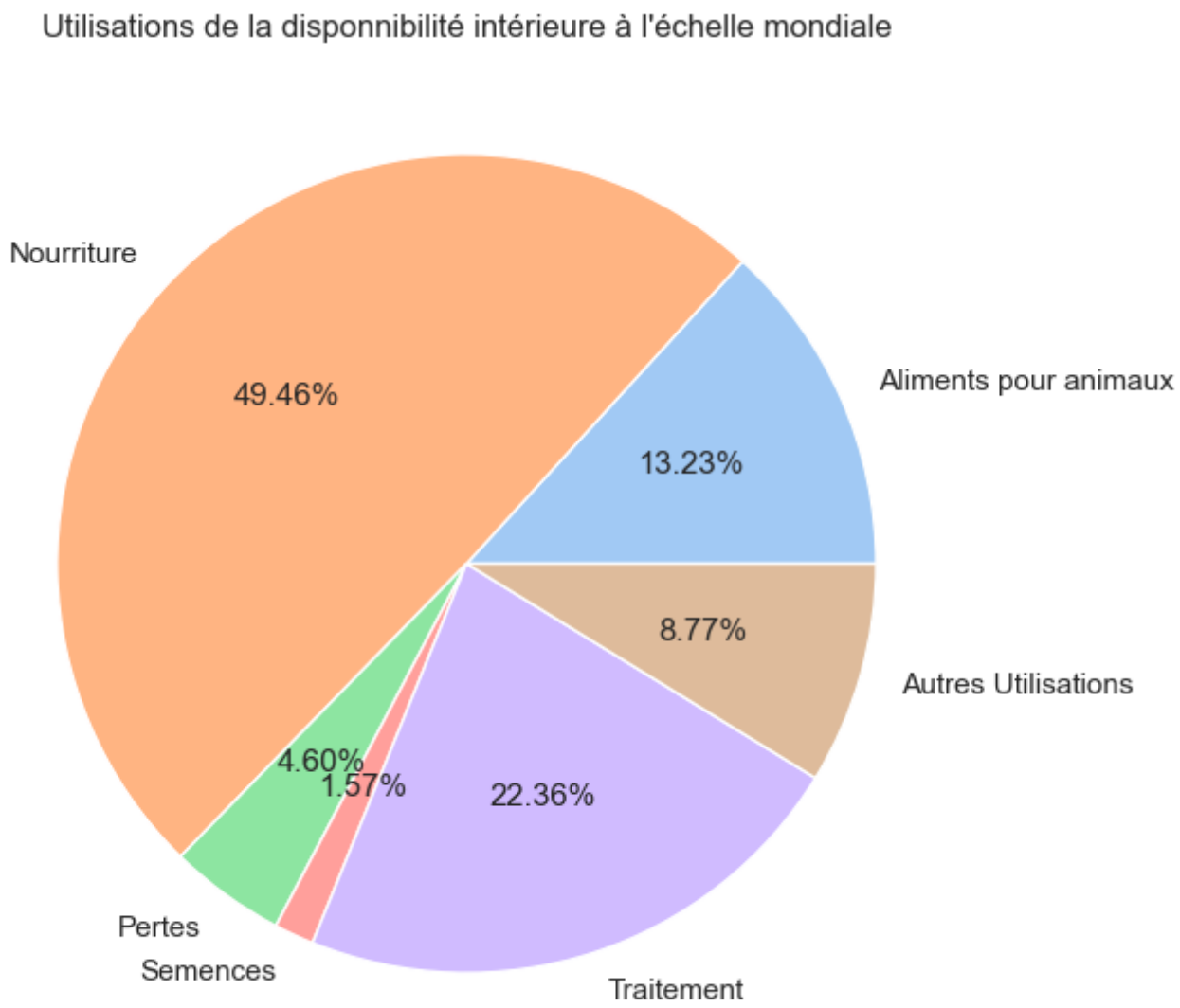
```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\3999837428.py:2: FutureWarning: The default value of numeric_
only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either sp
ecify numeric_only or select only columns which should be valid for the function.
  dispo_totale=dispo.groupby('Zone').sum()
```

In [42]:
```
#création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes aliments pour animau
print('Les différents utilisations de la disponnibilité intérieure : \n')
```

```
for i in ['Aliments pour animaux','Pertes','Nourriture']:
    print(i,' : ',dispo_totale[i].sum(),'\n')
```

```
Les différents utilisations de la disponnibilité intérieure :

Aliments pour animaux  :  1304245000000.0

Pertes  :  453698000000.0

Nourriture  :  4876258000000.0
```

In [43]:
```
#Graphique rteprésentant la répartition de l'utilisation de la disponnibilité intérieure mondiale

utilisation_dispo=dispo_totale.sum().reset_index().iloc[[0,9,10,12,13,1],:]
utilisation_dispo=utilisation_dispo.rename(columns={0:'quantité en milliards de tonnes','index':'type de conso
utilisation_dispo['quantité en milliards de tonnes']=utilisation_dispo['quantité en milliards de tonnes']/1000
#Taille fenêtre
plt.figure(figsize=(10, 7))
#Palette de couleurs
sns.set_theme(style='whitegrid', palette='pastel')
#Graphique
plt.pie(x=utilisation_dispo['quantité en milliards de tonnes'],labels=utilisation_dispo['type de consommation'

plt.title('Utilisations de la disponnibilité intérieure à l\'échelle mondiale')
plt.show()
```



3.5 - Utilisation des céréales

In [44]:
```
#Création d'une liste avec toutes les variables
dispo_produit=dispo.groupby('Produit').sum().reset_index()
```

```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\4125156610.py:2: FutureWarning: The default value of numeri
c_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Eithe
r specify numeric_only or select only columns which should be valid for the function.
  dispo_produit=dispo.groupby('Produit').sum().reset_index()
```

In [45]:
```
#Création d'un dataframe avec les informations uniquement pour ces céréales
#['Orge','Avoine','Millet','Soja','Sorgho','Riz (Eq Blanchi)','Maïs','Blé','Céréales, Autres']
liste_cereales=['Orge','Avoine','Millet','Sorgho','Riz (Eq Blanchi)','Maïs','Blé','Céréales, Autres','Seigle'
dispo_cereale=dispo_produit[dispo_produit['Produit'].isin(liste_cereales)].iloc[:,[0,1,7,10]]
```

In [46]:
```
#Affichage de la proportion d'alimentation animale par céréale
dispo_cereale['proportion_animale']=100*dispo_cereale['Aliments pour animaux']/dispo_cereale['Disponibilité i
dispo_cereale['proportion_nourriture']=100*dispo_cereale['Nourriture']/dispo_cereale['Disponibilité intérieur
dispo_cereale['proportion_autres']=100-dispo_cereale['proportion_animale']-dispo_cereale['proportion_nourritu
dispo_cereale
```

Out[46]:

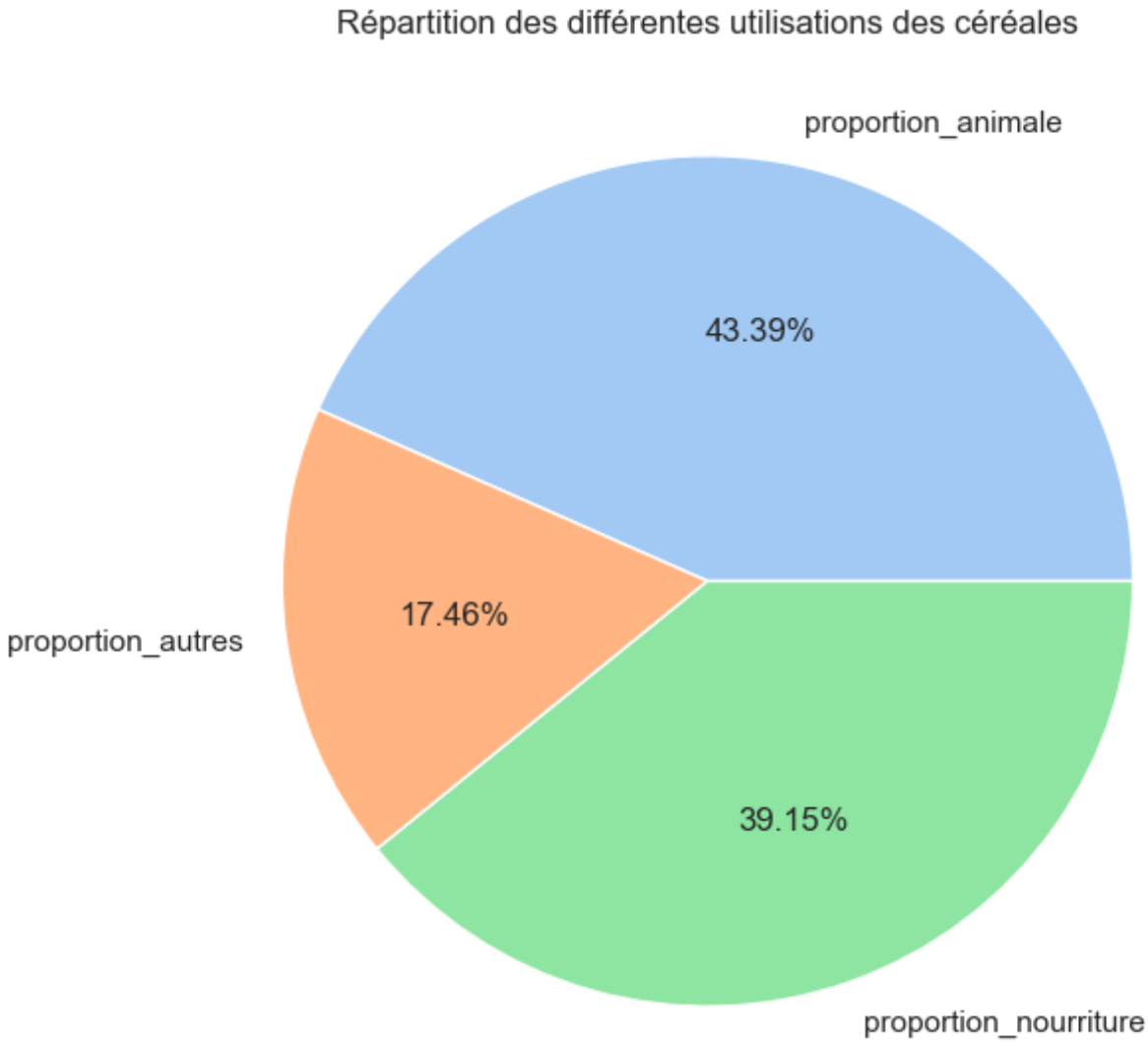
	Produit	Aliments pour animaux	Disponibilité intérieure	Nourriture	proportion_animale	proportion_nourriture	proportion_autres
7	Avoine	1.625100e+10	2.340700e+10	3.903000e+09	69.427949	16.674499	13.897552
12	Blé	1.296680e+11	6.794980e+11	4.578240e+11	19.082911	67.376799	13.540290
21	Céréales, Autres	1.903500e+10	2.748500e+10	5.324000e+09	69.255958	19.370566	11.373476
52	Maïs	5.461160e+11	9.557990e+11	1.251840e+11	57.137118	13.097314	29.765568
54	Millet	3.306000e+09	2.991100e+10	2.304000e+10	11.052790	77.028518	11.918692
62	Orge	9.265800e+10	1.404390e+11	6.794000e+09	65.977399	4.837688	29.184913
79	Riz (Eq Blanchi)	3.359400e+10	4.756560e+11	3.772860e+11	7.062667	79.319088	13.618245
80	Seigle	8.099000e+09	1.656700e+10	5.502000e+09	48.886340	33.210599	17.903060
82	Sorgho	2.480800e+10	5.823700e+10	2.415300e+10	42.598348	41.473634	15.928018

In [47]:
```
#Affichage de la proportion d'alimentation animale
print('La proportion de céréales utilisée pour nourrir les animaux est de :',dispo_cereale['proportion_animal
#Affichage de la proportion d'alimentation humaine
print('La proportion de céréales utilisée pour nourrir les humains est de :',dispo_cereale['proportion_nourri
```

```
La proportion de céréales utilisée pour nourrir les animaux est de : 43.3868311793665 %
La proportion de céréales utilisée pour nourrir les humains est de : 39.15430046611264 %
```

In [48]:
```
#Diagramme représentant la répartition des différentes utilisations des céréales
repartition_cereale=dispo_cereale.melt(id_vars='Produit',value_vars=['proportion_animale','proportion_nourrit
#Taille fenêtre
plt.figure(figsize=(10, 7))
plt.pie(x=repartition_cereale['value'],labels=repartition_cereale['variable'],autopct='%.2f%%')
plt.title('Répartition des différentes utilisations des céréales')
plt.show()
```

```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\3872229433.py:2: FutureWarning: The default value of numeri
c_only in DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Eith
er specify numeric_only or select only columns which should be valid for the function.
  repartition_cereale=dispo_cereale.melt(id_vars='Produit',value_vars=['proportion_animale','proportion_nour
riture','proportion_autres']).groupby('variable').mean().reset_index()
```



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

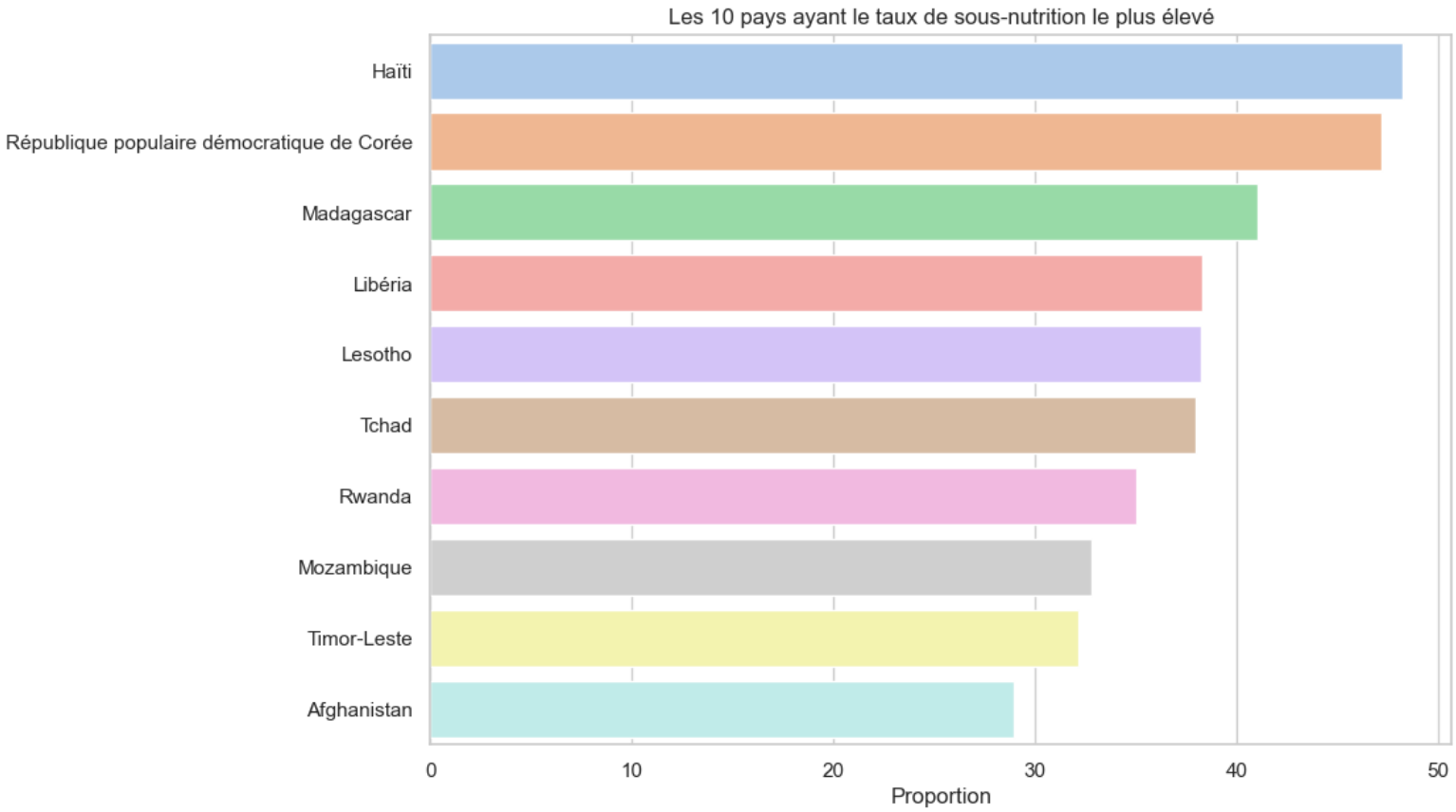
```
</div>
```

In [49]:
```
#Création de la colonne proportion par pays
personnes_sous_nutrition['Proportion']=100*personnes_sous_nutrition['sous_nutrition']/personnes_sous_nutrit
```

In [50]:
```
#affichage après tri des 10 pires pays
top_sous_nutrition=personnes_sous_nutrition.sort_values('Proportion',ascending = False).head(10)
display(top_sous_nutrition)
```

	Zone	sous_nutrition	Année	Population	Proportion
78	Haïti	5300000.0	2017	10982366.0	48.259182
157	République populaire démocratique de Corée	12000000.0	2017	25429825.0	47.188685
108	Madagascar	10500000.0	2017	25570512.0	41.062924
103	Libéria	1800000.0	2017	4702226.0	38.279742
100	Lesotho	800000.0	2017	2091534.0	38.249438
183	Tchad	5700000.0	2017	15016753.0	37.957606
161	Rwanda	4200000.0	2017	11980961.0	35.055619
121	Mozambique	9400000.0	2017	28649018.0	32.810898
186	Timor-Leste	400000.0	2017	1243258.0	32.173531
0	Afghanistan	10500000.0	2017	36296113.0	28.928718

In [51]:
```
#graphique représentant les 10 pays ayant le taux de sous-nutrition le plus élevé
#Taille fenêtre
plt.figure(figsize=(10, 7))
sns.barplot(data=top_sous_nutrition,y='Zone',x='Proportion')
plt.ylabel(None)
plt.title('Les 10 pays ayant le taux de sous-nutrition le plus élevé')
plt.show()
```



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

In [52]:
```
#calcul du total de l'aide alimentaire par pays
total_aide=aide.groupby('Zone').sum()
#ajout de la proportion de personnes en sous-nutrition
total_aide=pd.merge(total_aide,personnes_sous_nutrition.loc[:,['Zone','Proportion']],on='Zone')
total_aide=total_aide.rename(columns={'Proportion':'proportion_sous_nutrition'})
```

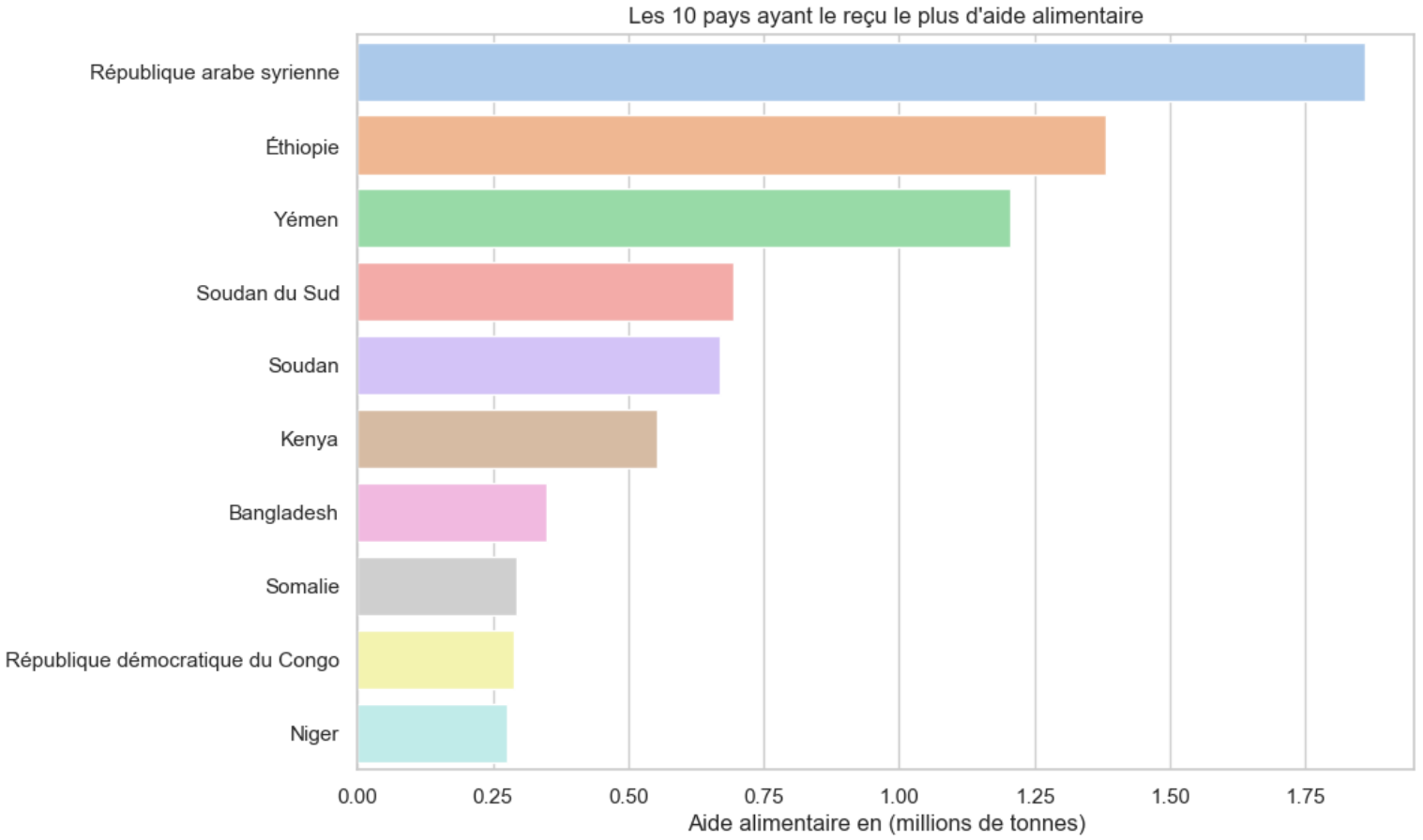
```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\2435640617.py:2: FutureWarning: The default value of nume
ric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. E
ither specify numeric_only or select only columns which should be valid for the function.
  total_aide=aide.groupby('Zone').sum()
```

In [53]:
```
#affichage après tri des 10 pays qui ont bénéficié le plus de l'aide alimentaire
top_aide=total_aide.sort_values('Aide_alimentaire',ascending = False).head(10)
top_aide=top_aide.drop(columns='Année')
display(top_aide)
```

	Zone	Aide_alimentaire	proportion_sous_nutrition
50	République arabe syrienne	1858943000	0.000000
74	Éthiopie	1381294000	19.830841
69	Yémen	1206484000	0.000000
61	Soudan du Sud	695248000	0.000000
60	Soudan	669784000	12.250879
30	Kenya	552836000	23.695200
3	Bangladesh	348188000	13.463972
59	Somalie	292678000	0.000000
53	République démocratique du Congo	288502000	0.000000
43	Niger	276344000	0.000000

In [54]:
```
#graphique représentant les 10 pays ayant reçu le plus d'aide alimentaire
#Manipulations sur le Dataframe pour le graphique
#Changement d'unité pour l'aide alimentaire
top_aide=top_aide.rename(columns={'Aide_alimentaire':'Aide alimentaire en (millions de tonnes)'})
top_aide['Aide alimentaire en (millions de tonnes)']=top_aide['Aide alimentaire en (millions de tonnes)']/

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=top_aide,y='Zone',x='Aide alimentaire en (millions de tonnes)')
plt.title('Les 10 pays ayant le reçu le plus d\'aide alimentaire')
```

```
plt.ylabel(None)
plt.show()
```



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

In [55]:
```
#Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année
evolution_aide=aide.iloc[:,[0,1,3]]
```

```
evolution_aide=evolution_aide.groupby(['Zone','Année']).sum().reset_index()
```

In [56]:
```
#Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de l'aide alimentaire
top_cinq_aide=evolution_aide.groupby('Zone').sum().sort_values('Aide_alimentaire',ascending = False).rese
```

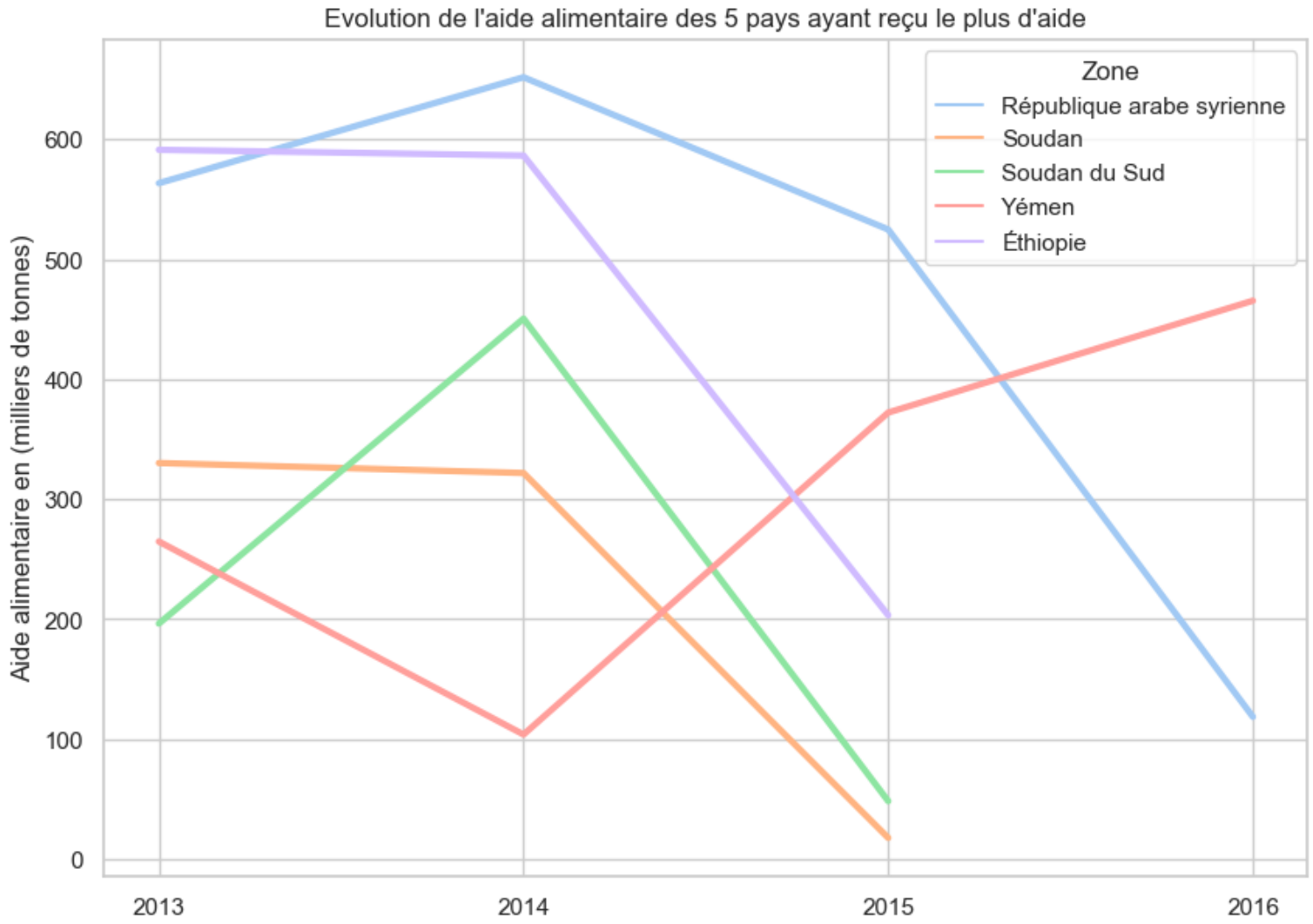
In [57]:
```
#On filtre sur le dataframe avec notre liste
evolution_aide_top_cinq=evolution_aide[evolution_aide['Zone'].isin(top_cinq_aide)]
```

In [58]:
```
# Affichage des pays avec l'aide alimentaire par année
evolution_aide_top_cinq
```

Out[58]:

	Zone	Année	Aide_alimentaire
157	République arabe syrienne	2013	563566000
158	République arabe syrienne	2014	651870000
159	République arabe syrienne	2015	524949000
160	République arabe syrienne	2016	118558000
189	Soudan	2013	330230000
190	Soudan	2014	321904000
191	Soudan	2015	17650000
192	Soudan du Sud	2013	196330000
193	Soudan du Sud	2014	450610000
194	Soudan du Sud	2015	48308000
214	Yémen	2013	264764000
215	Yémen	2014	103840000
216	Yémen	2015	372306000
217	Yémen	2016	465574000
225	Éthiopie	2013	591404000
226	Éthiopie	2014	586624000
227	Éthiopie	2015	203266000

In [59]:
```
#graphique représentant l'évolution d'aide alimentaiore reçue par année pour les 5 pays qui en ont reçu l
#Manipulation des données pour l'échelle du graphique
#Changement d'unité pour l'aide alimentaire
evolution_aide_top_cinq=evolution_aide_top_cinq.rename(columns={'Aide_alimentaire':'Aide alimentaire en (
evolution_aide_top_cinq['Aide alimentaire en (milliers de tonnes)']=evolution_aide_top_cinq['Aide aliment

#Taille fenêtre
plt.figure(figsize=(10, 7))
sns.lineplot(data=evolution_aide_top_cinq,x='Année',y='Aide alimentaire en (milliers de tonnes)',hue='Zor
plt.title("Evolution de l'aide alimentaire des 5 pays ayant reçu le plus d'aide")
plt.xlabel(None)
plt.xticks([2013,2014,2015,2016])
plt.show()
```

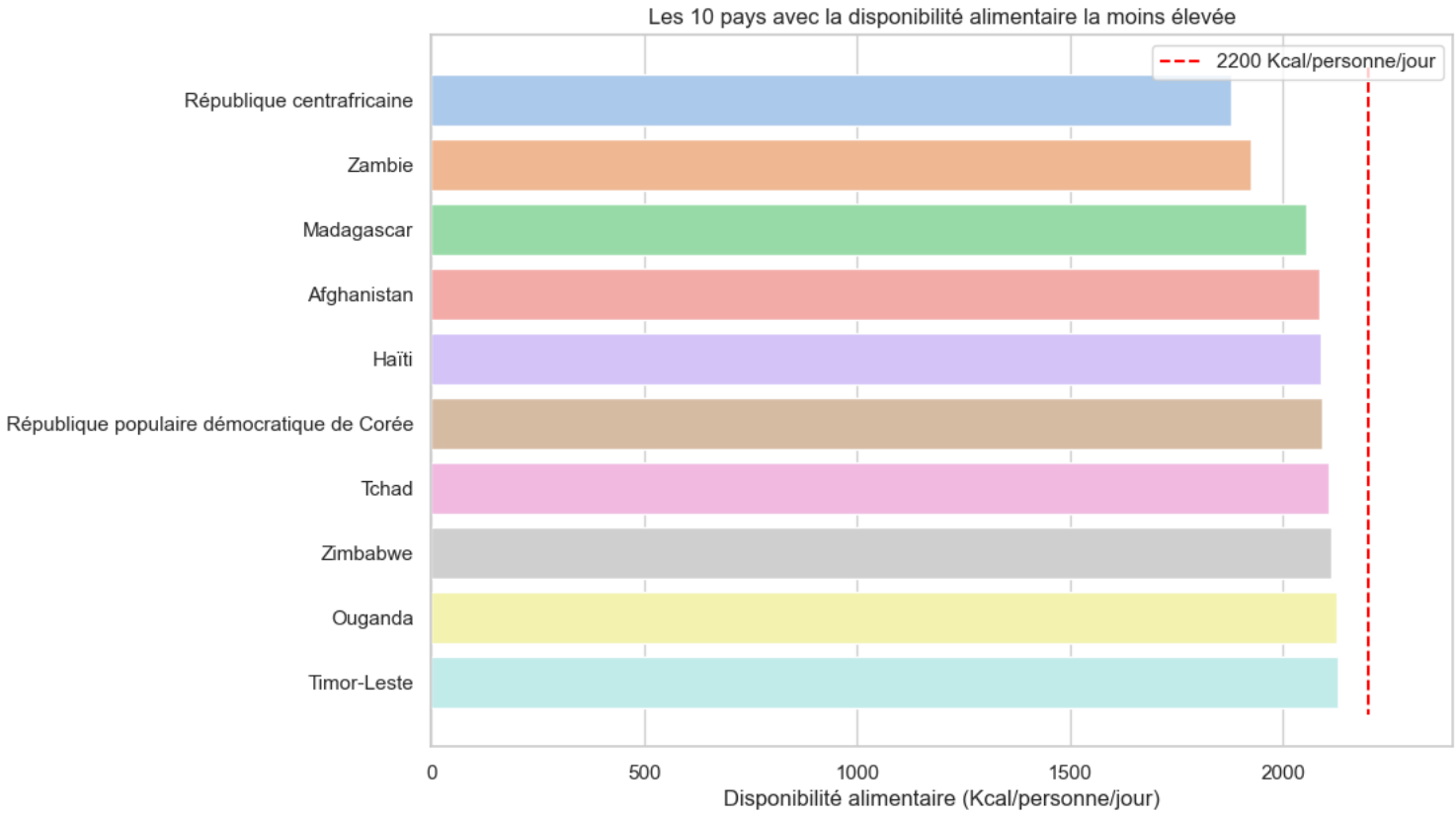


3.9 - Pays avec le moins de disponibilité par habitant

In [60]:
```
#Calcul de la disponibilité en kcal par personne par jour par pays
dispo_par_pays=dispo.groupby('Zone').sum().reset_index().loc[:,['Zone','Disponibilité alimentaire (Kcal
```

```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\2309743926.py:2: FutureWarning: The default value of n
umeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to Fa
lse. Either specify numeric_only or select only columns which should be valid for the function.
  dispo_par_pays=dispo.groupby('Zone').sum().reset_index().loc[:,['Zone','Disponibilité alimentaire (Kc
al/personne/jour)']]
```

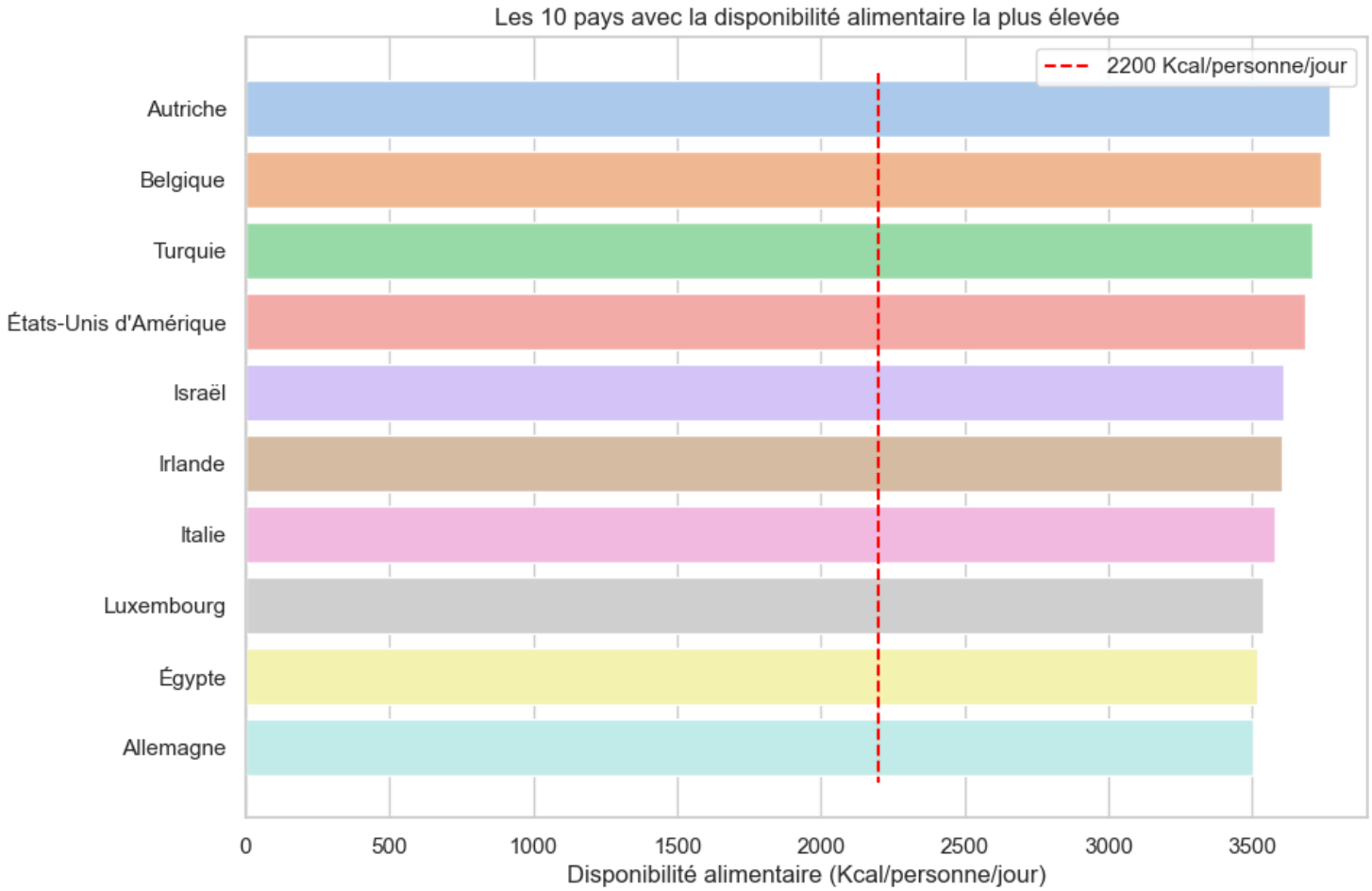
In [61]:
```
#Affichage des 10 pays qui ont le moins de dispo alimentaire par personne

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=dispo_par_pays.sort_values('Disponibilité alimentaire (Kcal/personne/jour)',ascending=
plt.title('Les 10 pays avec la disponibilité alimentaire la moins élevée')
plt.ylabel(None)
plt.xlim(0,2400)

#Création d'une barre verticale représentant les besoins énergétiques d'un adulte
plt.vlines([2200],-0.5,9.5,linestyles='dashed', colors='red', label="2200 Kcal/personne/jour")
plt.legend()

plt.show()
```



3.10 - Pays avec le plus de disponibilité par habitant

In [62]:
```
#Affichage des 10 pays qui ont le plus de dispo alimentaire par personne

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=dispo_par_pays.sort_values('Disponibilité alimentaire (Kcal/personne/jour)',ascending
plt.title('Les 10 pays avec la disponibilité alimentaire la plus élevée')
plt.ylabel(None)
```

```
plt.xlim(0,3900)

#Création d'une barre verticale représentant les besoins énergétiques d'un adulte
plt.vlines([2200],-0.5,9.5,linestyles='dashed', colors='red', label="2200 Kcal/personne/jour")
plt.legend()

plt.show()
```



3.11 - Exemple de la Thaïlande pour le Manioc

```
</div>
```

In [63]:
```
#création d'un dataframe avec uniquement la Thaïlande
thailande=dispo.loc[dispo['Zone']=='Thaïlande']
```

In [64]:
```
#Calcul de la sous nutrition en Thaïlande
thailande=pd.merge(thailande,sous_nutrition.loc[sous_nutrition['Année']=='2016-2018',['Zone','sous_nu
#indexation du df par produit
thailande=thailande.set_index('Produit')
```

In [65]:
```
# On calcule la proportion exportée en fonction de la production totale
#calcul du pourcentage
proportion_manioc=round((100*thailande.loc['Manioc','Exportations - Quantité']/thailande.loc['Manioc

print('La proportion de manioc exportée en Thaïlande est de ',proportion_manioc,'%.')
```

```
La proportion de manioc exportée en Thaïlande est de  83.41 %.
```

In [66]:
```
#La part de personnes en état de sous-nutrition pour ce pays
#Dataframe contenant la population de la Thaïlande pour 2017
A=population.loc[(population['Zone']=='Thaïlande') & (population['Année']==2017),['Zone','Population

#Dataframe contenant la sous-nutrition en Thaïlande
B=sous_nutrition.loc[(sous_nutrition['Année']=='2016-2018') & (sous_nutrition['Zone']=='Thaïlande'),

#Jointure des deux dataframes
sous_nutrition_thailande=pd.merge(A,B,on='Zone')

print('La proportion de Thaïlandais en sous nutrition est de ',round(100*sous_nutrition_thailande.il
```

```
La proportion de Thaïlandais en sous nutrition est de  8.96 %.
```

In [67]:
```
#Calcul de la disponnibilité par habitant
thailande_total=thailande.groupby('Zone').sum()
print('Disponibilité alimentaire (Kcal/personne/jour) pour la Thïlande : ',thailande_total.loc['Thaï
```

```
Disponibilité alimentaire (Kcal/personne/jour) pour la Thïlande :  2785.0
```

```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\3828784223.py:2: FutureWarning: The default value o
f numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default
to False. Either specify numeric_only or select only columns which should be valid for the function.
  thailande_total=thailande.groupby('Zone').sum()
```

In [68]:
```
#Calcul dela proportion de production allouée au manioc
proprotion_production_manioc=round(100*thailande.loc['Manioc','Production']/thailande['Production'].

print('Proportion de production allouée au manioc : ',proprotion_production_manioc,'%')
```

```
Proportion de production allouée au manioc :  14.98 %
```

Etape 6 - Analyse complémentaires

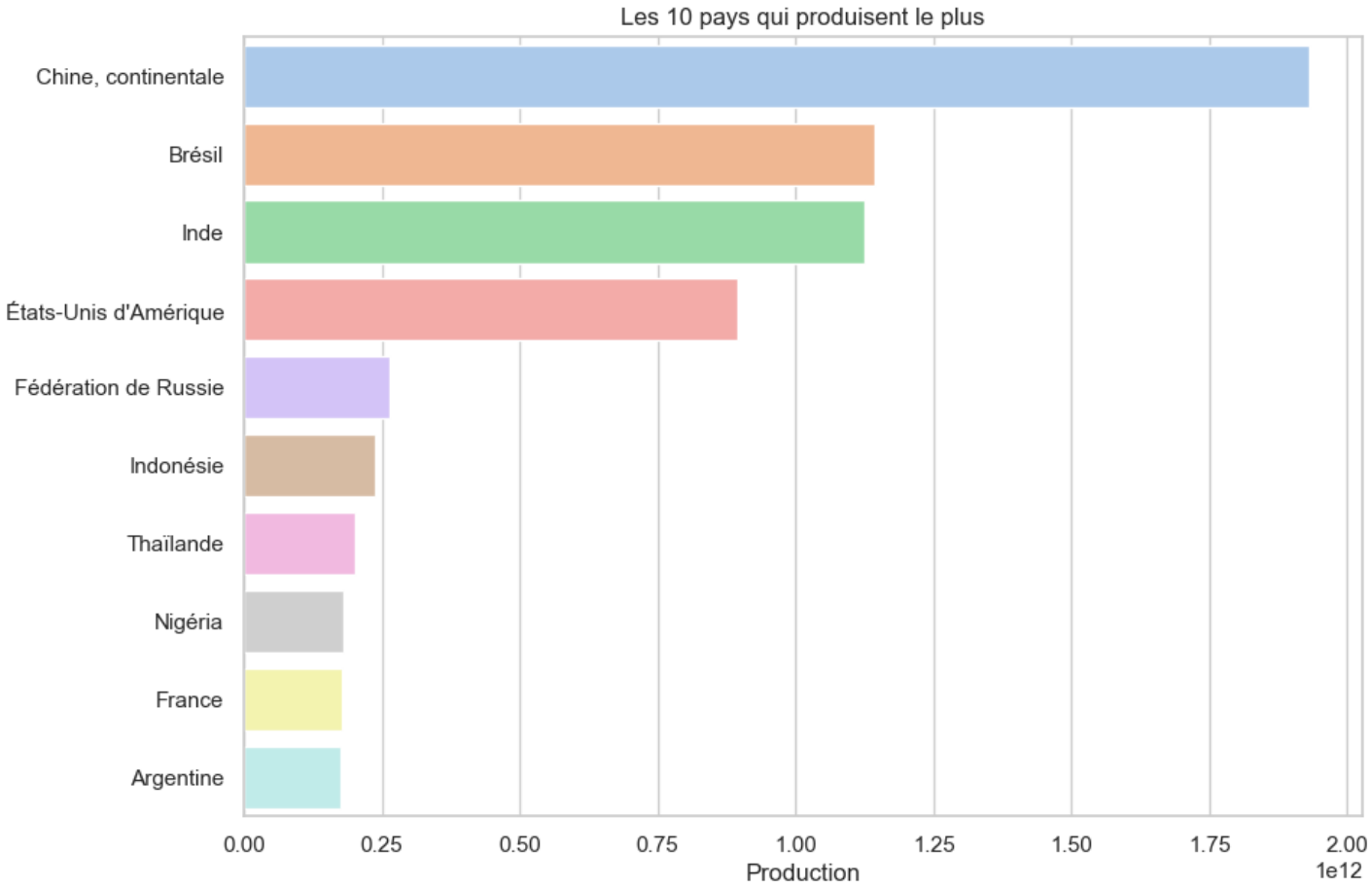
In [69]:
```
#Rajouter en dessous toutes les analyses complémtaires suite à la demande de mélanie :
#"et toutes les infos que tu trouverais utiles pour mettre en relief les pays qui semblent être
#le plus en difficulté au niveau alimentaire"

#liste des pays qui exportent le plus
#  -> étude dessus (limitation de la nourreiture pour animaux par exemple)
```

In [70]:
```
#dataframe avec les caractéristiques de chaque pays
analyse=dispo.groupby('Zone').sum().reset_index()

#Ajout de la population et de la proportion de personnes en sous nutrition depuis le DF du 3.6 (top
analyse=pd.merge(analyse,personnes_sous_nutrition,on='Zone').drop(columns='Année')

#renommage de la colonne 'Proportion'
analyse=analyse.rename(columns={'Proportion':'Proportion_sous_nutrition'})

#suppression des colonnes en trop
analyse=analyse.drop(columns=['Disponibilité alimentaire en quantité (kg/personne/an)','Disponibili
```

```
C:\Users\adrie\AppData\Local\Temp\ipykernel_1944\499561103.py:2: FutureWarning: The default value o
f numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will defaul
t to False. Either specify numeric_only or select only columns which should be valid for the functi
on.
  analyse=dispo.groupby('Zone').sum().reset_index()
```

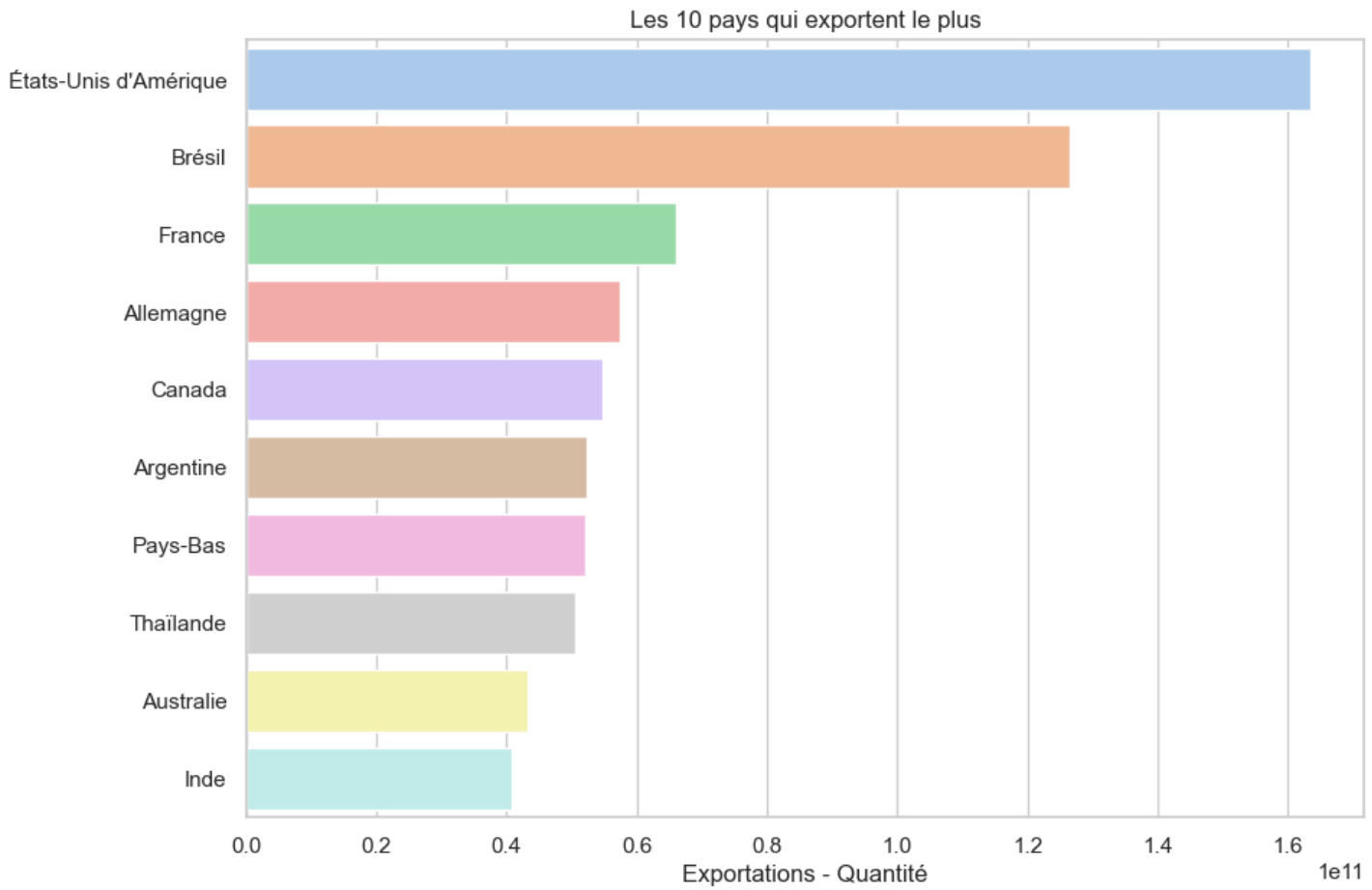
In [71]:
```
#liste des pays qui produisent le plus
#Taille fenêtre
plt.figure(figsize=(10, 7))
```

```
sns.barplot(data=analyse.sort_values('Production',ascending=False).head(10),y='Zone',x='Production'
plt.title('Les 10 pays qui produisent le plus')
plt.ylabel(None)
plt.show()
```

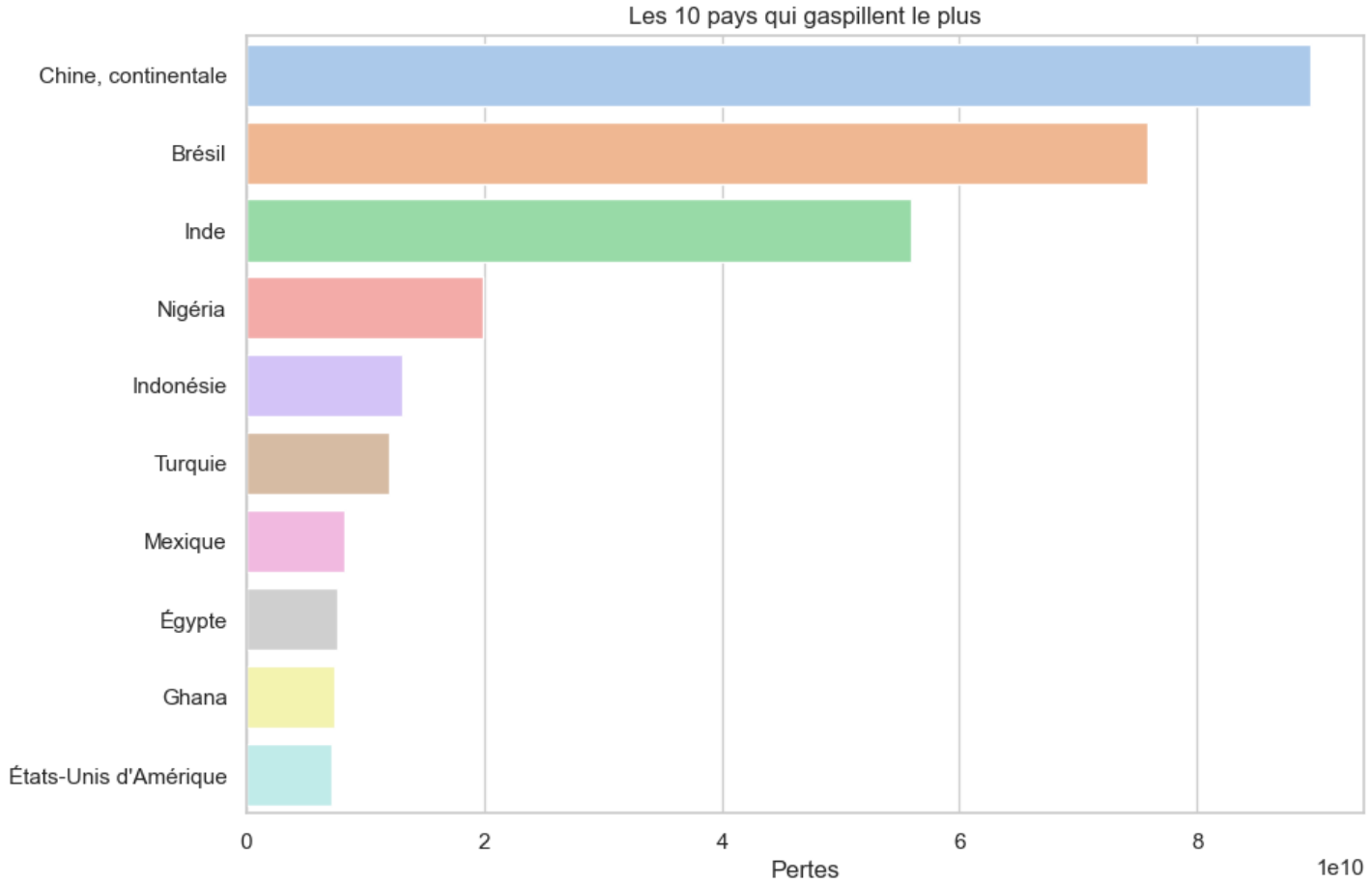


In [72]:
```
#liste des pays qui exportent le plus

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=analyse.sort_values('Exportations - Quantité',ascending=False).head(10),y='Zone',x
plt.title('Les 10 pays qui exportent le plus')
```

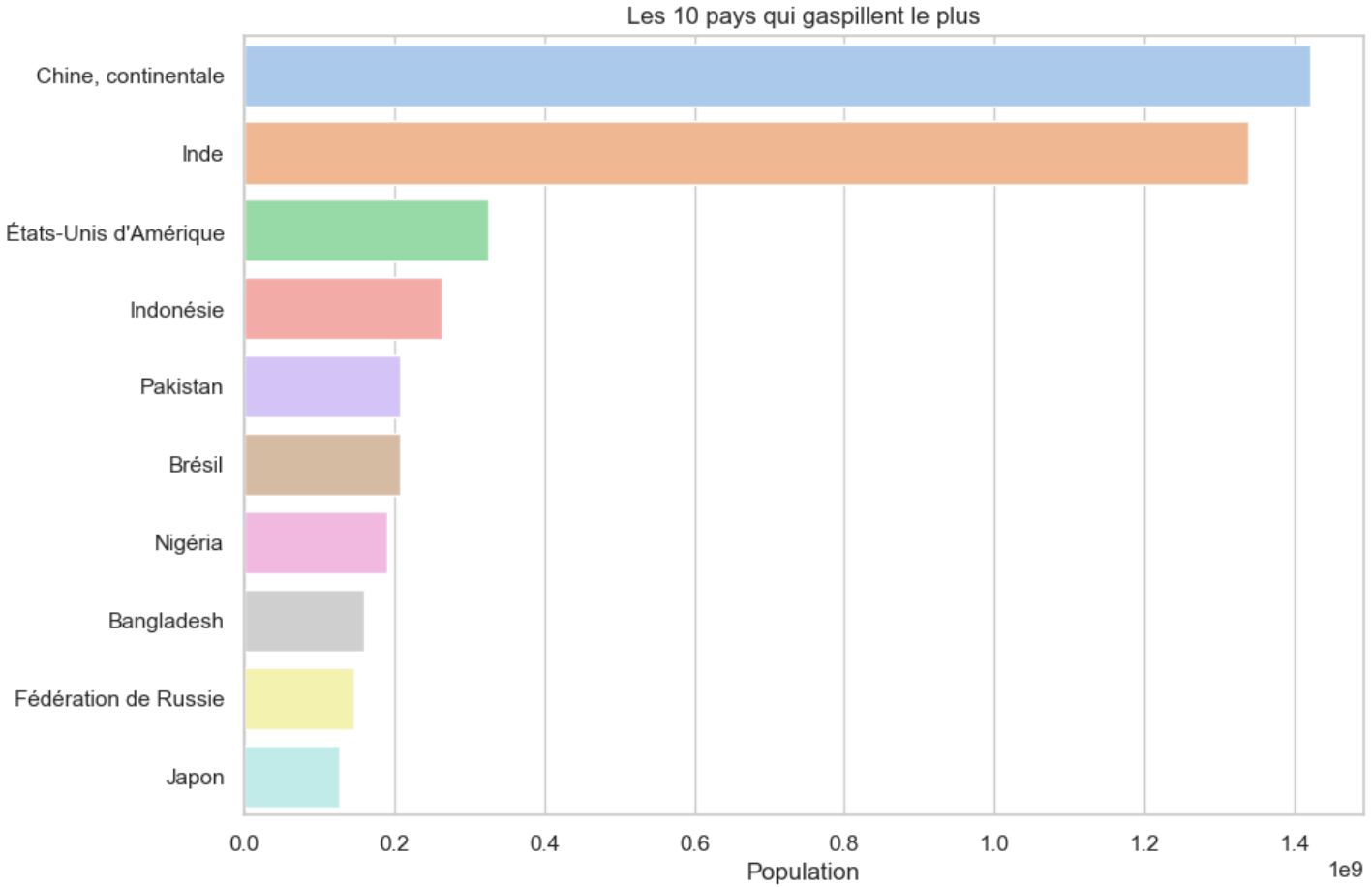
```
plt.ylabel(None)
plt.show()
```



In [73]:
```
#liste des pays qui gasillent le plus

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=analyse.sort_values('Pertes',ascending=False).head(10),y='Zone',x='Pertes')
plt.title('Les 10 pays qui gaspillent le plus')
plt.ylabel(None)
plt.show()
```



In [74]:
```
#liste des pays avec la population plus élevée

#Taille fenêtre
plt.figure(figsize=(10, 7))

sns.barplot(data=analyse.sort_values('Population',ascending=False).head(10),y='Zone',x='Population'
plt.title('Les 10 pays qui gaspillent le plus')
plt.ylabel(None)
plt.show()
```



In [75]:
```
#Les Etats-Unis apparraissent dans chacun des indicateurs : population, exportation, production
#Questions : Et si les Etats-Unis produisent moins de nourriture animale, quels seraient les effets
eu=analyse.loc[analyse['Zone']=="États-Unis d'Amérique"]
#Affichage du dataframe concernant uniquement les Etats-Unis
display(eu)
```

	Zone	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	
169	États-Unis d'Amérique	1.484320e+11	1.546990e+11	3682.0	7.779920e+11	1.635240e+11	8.188700e+10	7.1620

In [76]:
```
#Combien de tonnes représentent 10% de la production des Etats-Unis ?
#selection de la donnée dans le dataframe eu
nourriture_animaux_eu=eu.iloc[0,1]

print('Si les Etats-Unis diminuaient le production de nourriture d\'origine animale de 10%, cela re
```

```
Si les Etats-Unis diminuaient le production de nourriture d'origine animale de 10%, cela représente
rait  14.84  milliards de tonnes de céréales.
```

In [77]:
```
#Recherche d'un équivalent au gain de céréales qui en déoculerait
analyse.loc[analyse['Production']<=(eu.iloc[0,1])].sort_values('Production',ascending=False).head()
```

Out[77]:

	Zone	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	
157	Turquie	1.775000e+10	3.006000e+09	3708.0	1.313770e+11	1.459300e+10	1.213600e+10	1.20360
31	Canada	2.613700e+10	4.450000e+09	3499.0	7.956200e+10	5.477100e+10	1.942200e+10	2.11800
162	Viet Nam	1.135600e+10	3.815000e+09	2744.0	9.513400e+10	2.263800e+10	9.217000e+09	6.74300
120	Philippines	6.379000e+09	9.387000e+09	2568.0	1.088850e+11	7.897000e+09	8.778000e+09	2.90100
10	Australie	9.236000e+09	3.032000e+09	3278.0	6.699200e+10	4.318400e+10	5.879000e+09	5.20000

In [78]:
```
#Cas de la Chine, le pays avec le plus de pertes.
#Calcul de la quantité de pertes de la Chine
perte_chine=analyse.loc[analyse['Zone']=='Chine, continentale'].iloc[0,7]

print('La perte de nourriture de la chine est de',perte_chine/1000000000,'milliards de tonnes. Soit
```

```
La perte de nourriture de la chine est de 89.575 milliards de tonnes. Soit l'équivalent d'un peu pl
us de la disponibilité intérieure du Canada.
```

In [79]:
```
#Recherche de l'équivalent disponnibilité intérieure de la perte de la chine
analyse.loc[analyse['Production']<=(perte_chine)].sort_values('Production',ascending=False).head()
```

Out[79]:

	Zone	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité intérieure	Exportations - Quantité	Importations - Quantité	
121	Pologne	1.972500e+10	2.761000e+09	3450.0	7.843100e+10	1.960500e+10	1.122400e+10	2.7100
75	Italie	1.614400e+10	3.228000e+09	3578.0	9.740500e+10	2.364300e+10	3.869200e+10	1.8610
70	Iran (République islamique d')	1.168100e+10	1.287000e+09	3089.0	9.204400e+10	3.202000e+09	1.626900e+10	5.4500
38	Colombie	6.111000e+09	3.917000e+09	2800.0	7.758300e+10	3.599000e+09	8.754000e+09	2.0400
14	Bangladesh	3.335000e+09	2.015000e+09	2453.0	7.275700e+10	3.280000e+08	1.001300e+10	4.0800

In [ ]:
```

```