PROJET 4 DATA ANALYST

Réalisez une étude de santé publique avec R ou Python

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
```

In [2]:
```
# Importation de la librairie Numpy
import numpy as np
```

In [3]:
```
# Importation de la librairie Matplotlib
import matplotlib.pyplot as plt
```

In [4]:
```
# Importation de la librairie Seaborn
import seaborn as sns
```

1.2 - Chargement des fichiers Excel

In [5]:
```
# Importation du fichier dispo_alimentaire.csv
dispo_alimentaire = pd.read_csv('dispo_alimentaire.csv')

# Importation du fichier population.csv
population = pd.read_csv('population.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

In [6]:
```
# 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 [7]:
```
# Consulter le nombre de colonnes
nb_colonne = len(population.columns)
display(nb_colonne)
```

```
3
```

In [8]:
```
# La nature des données dans chacune des colonnes
print(population.dtypes)
```

```
Zone       object
Année       int64
Valeur    float64
dtype: object
```

In [9]:
```
# Le nombre de valeurs présentes dans chacune des colonnes
print(("Nombre de zones :",population['Zone'].nunique()),
      ("Nombre d'Années :",population['Année'].nunique()),
      ("Nombre de Valeurs :",population['Valeur'].nunique())
     )
```

```
('Nombre de zones :', 236) ("Nombre d'Années :", 6) ('Nombre de Valeurs :', 1413)
```

In [10]:
```
# Affichage les 5 premières lignes de la table
population.head()
```

Out[10]:

	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 [11]:
```
# Multiplication de la colonne valeur par 1000
population['Valeur'] = population['Valeur']* 1_000
```

In [12]:
```
# changement du nom de la colonne Valeur par Population
population.rename(columns = {"Valeur":"Population"},
                  inplace = "TRUE")
```

In [13]:
```
# Affichage les 5 premières lignes de la table pour voir les modifications
population.head()
```

Out[13]:

	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 [14]:
```
# 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)
```

In [15]:
```
# Consulter le nombre de colonnes
nb_colonne1 = len(dispo_alimentaire.columns)

display(nb_colonne1)
```

```
18
```

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

Out[16]:

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

In [17]:
```
# remplacement des NaN dans le dataset par des 0
dispo_alimentaire = dispo_alimentaire.fillna(0)
```

In [18]:
```
# multiplication de toutes les lignes contenant des milliers de tonnes en Kg
liste = ['Aliments pour animaux','Autres Utilisations',
         'Disponibilité intérieure','Exportations - Quantité',
         'Importations - Quantité','Nourriture','Pertes',
         'Production','Semences','Traitement','Variation de stock']

for var in liste :
    dispo_alimentaire[var] = dispo_alimentaire[var]*1_000
```

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

Out[19]:

	Zone	Produit	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Disponibilité intérieure
0	Afghanistan	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	0.77	53000.0
1	Afghanistan	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	0.02	41000.0
2	Afghanistan	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	0.03	2000.0
3	Afghanistan	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	0.00	0.0
4	Afghanistan	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	0.05	82000.0

2.3 - Analyse exploratoire du fichier aide alimentaire

In [20]:
```
# 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)
```

In [21]:
```
# Consulter le nombre de colonnes
nb_colonne2 = len(aide_alimentaire.columns)

display(nb_colonne2)
```

```
4
```

In [22]:
```
# La nature des données dans chacune des colonnes
aide_alimentaire.dtypes
```

Out[22]:
```
Pays bénéficiaire    object
Année                 int64
Produit              object
Valeur                int64
dtype: object
```

In [23]:
```
# changement du nom de la colonne Pays bénéficiaire par Zone
aide_alimentaire.rename(columns = {"Pays bénéficiaire":"Zone"},
                        inplace = "TRUE")
```

In [24]:
```
# Multiplication de la colonne Aide_alimentaire qui
# contient des tonnes par 1000 pour avoir des kg
aide_alimentaire['Valeur'] = aide_alimentaire['Valeur']*1_000
```

In [25]:
```
# Affichage les 5 premières lignes de la table
aide_alimentaire.head()
```

Out[25]:

	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

In [26]:
```
# 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 [27]:
```
# Consulter le nombre de colonnes
nb_colonne3 = len(sous_nutrition.columns)

display(nb_colonne3)
```

```
3
```

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

Out[28]:

	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 [29]:
```
# Conversion de la colonne sous nutrition en numérique
# sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'])
# J'ai commenté la ligne ci-dessus qui renvoie une erreur
# et empêche de relancer toutes les cellules d'un coup
```

In [30]:
```
# Conversion de la colonne (avec l'argument errors=coerce qui permet de convertir
# automatiquement les lignes qui ne sont pas des nombres en NaN)
sous_nutrition['Valeur'] = pd.to_numeric(sous_nutrition['Valeur'],
                                         errors = 'coerce')
```

In [31]:
```
# Puis remplacement des NaN en 0
sous_nutrition['Valeur'] = sous_nutrition['Valeur'].fillna(0)
```

In [32]:
```
sous_nutrition.head()
```

Out[32]:

	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 [33]:
```
# changement du nom de la colonne Valeur par sous_nutrition
sous_nutrition.rename(columns = {"Valeur":"sous_nutrition"},
                      inplace = "TRUE")
```

In [34]:
```
# Multiplication de la colonne sous_nutrition par 1000000
sous_nutrition['sous_nutrition'] = sous_nutrition['sous_nutrition']*1_000_000
```

In [35]:
```
# Afficher les 5 premières lignes de la table
sous_nutrition.head()
```

Out[35]:

	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 [36]:
```
# Il faut tout d'abord faire une jointure entre la table
#population et la table sous nutrition, en ciblant l'année 2017
```

```
data_pop_sous_nutrition_2017 = pd.merge(population.loc[population['Année']==2017,
                                                       ['Zone','Population']],
                                        sous_nutrition.loc[sous_nutrition['Année'] == "2016-2018",
                                                           ['Zone', 'sous_nutrition']],
                                        on = ('Zone'),
                                        how = "left")
```

In [37]:
```
# Affichage du dataset
display(data_pop_sous_nutrition_2017)
```

	Zone	Population	sous_nutrition
0	Afghanistan	36296113.0	10500000.0
1	Afrique du Sud	57009756.0	3100000.0
2	Albanie	2884169.0	100000.0
3	Algérie	41389189.0	1300000.0
4	Allemagne	82658409.0	0.0
...	...	...	...
231	Venezuela (République bolivarienne du)	29402484.0	8000000.0
232	Viet Nam	94600648.0	6500000.0
233	Yémen	27834819.0	0.0
234	Zambie	16853599.0	0.0
235	Zimbabwe	14236595.0	0.0

236 rows × 3 columns

In [38]:
```
# Calcul et affichage du nombre de personnes en état de sous nutrition
# Calcul nombre de personnes souffrant de sous-nutrition dans le monde
pop_sous_nutrition = data_pop_sous_nutrition_2017['sous_nutrition'].sum()

print('{} millions de personnes souffraient de sous-nutrition dans le monde en 2017.'
      .format(pop_sous_nutrition/1_000_000))
```

```
535.7 millions de personnes souffraient de sous-nutrition dans le monde en 2017.
```

In [39]:
```
# Calcul de la population mondiale
pop_totale = data_pop_sous_nutrition_2017['Population'].sum()
```

```
print('En 2017, nous étions {} milliards de personnes dans le monde.'
      .format(round((pop_totale)/1_000_000_000),2))
```

```
En 2017, nous étions 8 milliards de personnes dans le monde.
```

In [40]:
```
# Pourcentage du nombre de personnes souffrant de sous-nutrition dans le monde
pop_tot_sn_pourcent = (pop_sous_nutrition)/(pop_totale)*100

print ('En 2017, {} % de la population mondiale était en sous-nutrition.'
       .format(round(pop_tot_sn_pourcent)))
```

```
En 2017, 7 % de la population mondiale était en sous-nutrition.
```

3.2 - Nombre théorique de personne qui pourrait être nourries

In [41]:
```
# Combien mange en moyenne un être humain ? Source =>
print('Un homme doit consommer environ 2 500 kilocalories (kcal) par jour,'
      'contre 2 000 kcal par jour pour une femme.')

print('Source : Organisation mondiale de la Santé')
```

```
Un homme doit consommer environ 2 500 kilocalories (kcal) par jour,contre 2 000 kcal par jour pour une femme.
Source : Organisation mondiale de la Santé
```

In [42]:
```
# On commence par faire une jointure entre le data frame population et
# Dispo_alimentaire afin d'ajouter dans ce dernier la population
join_population_dispo_2017 = pd.merge(population.loc[population['Année'] == 2017],
                                      dispo_alimentaire,
                                      on = ('Zone')
                                     )
```

In [43]:
```
# Affichage du nouveau dataframe
display(join_population_dispo_2017)
```

	Zone	Année	Population	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)	(g
0	Afghanistan	2017	36296113.0	Abats Comestible	animale	0.0	0.0	5.0	1.72	0.20	
1	Afghanistan	2017	36296113.0	Agrumes, Autres	vegetale	0.0	0.0	1.0	1.29	0.01	
2	Afghanistan	2017	36296113.0	Aliments pour enfants	vegetale	0.0	0.0	1.0	0.06	0.01	
3	Afghanistan	2017	36296113.0	Ananas	vegetale	0.0	0.0	0.0	0.00	0.00	
4	Afghanistan	2017	36296113.0	Bananes	vegetale	0.0	0.0	4.0	2.70	0.02	
...	...	...	...	...	...	...	...	...	...	...	
15411	Zimbabwe	2017	14236595.0	Viande de Suides	animale	0.0	0.0	24.0	2.65	2.25	
15412	Zimbabwe	2017	14236595.0	Viande de Volailles	animale	0.0	0.0	17.0	4.97	1.05	
15413	Zimbabwe	2017	14236595.0	Viande, Autre	animale	0.0	1000.0	7.0	2.29	0.21	
15414	Zimbabwe	2017	14236595.0	Vin	vegetale	0.0	0.0	1.0	0.27	0.00	
15415	Zimbabwe	2017	14236595.0	Épices, Autres	vegetale	0.0	0.0	1.0	0.06	0.02	

15416 rows × 20 columns

In [44]:
```
# Création de la colonne dispo_kcal avec calcul des kcal disponibles mondialement

join_population_dispo_2017 = join_population_dispo_2017.assign(dispo_kcal =
                             (join_population_dispo_2017['Population']*
                              join_population_dispo_2017['Disponibilité alimentaire (Kcal/personne/jour)'])
                                                        )
```

In [45]:
```
# Calcul de la dispo_kcal mondiale
dispo_kcal_mondiale = join_population_dispo_2017['dispo_kcal'].sum()
```

In [46]:
```
print('Tout types de produits confondus, les kcal journalières disponibles '
      'mondialement étaient de {} trillions de kcal en 2017.'
      .format(round(((dispo_kcal_mondiale)/1_000_000_000_000),0)))
```

```
Tout types de produits confondus, les kcal journalières disponibles mondialement étaient de 21.0 trillions de kcal en 201
7.
```

In [47]:
```
# Calcul du nombre d'humains pouvant être nourris, résultat exprimé en kcal.
# 2250 = moyenne de la valeur journalière recommandée par l'OMS
calcul = dispo_kcal_mondiale/2250

calcul_prop_pop_alimentee = round((round((calcul/1_000_000_000),2)
                                   /round((pop_totale/1_000_000_000),2)*100),0)

calcul_nb_pop_alimentee = round(round((calcul/1_000_000_000),2)
                                -round((pop_totale/1_000_000_000),2),2)

print('En 2017, la disponibilité alimentaire journalière '
      'mondiale permettait de fournir {} kcal par être humain.'
      .format(round((dispo_kcal_mondiale/pop_totale),0)))

print('Sachant que la valeur journalière est de 2500 kcal '
      'pour un homme et 2000 kcal pour une femme,'
      'alors chaque être humain aurait pu être nourris.')

print('La disponibilité journalière mondiale en kcal aurait '
      'permis de nourrir {} milliards d\'êtres humains.'
      .format(round((calcul/1_000_000_000),2)),
      'Sachant qu\'en 2017 nous étions {} milliards d\'êtres humains, '
      'la dispo alimentaire aurait permis de nourrir la population mondiale.'
      .format(round((pop_totale/1_000_000_000),2)))

print('Avec la nourriture disponible mondialement en 2017, {} '
      '% d\'êtres humains auraient pu être nourris, soit'
      .format(calcul_prop_pop_alimentee),
      '{} milliard de personnes supplémentaires.'
      .format(calcul_nb_pop_alimentee))
```

```
En 2017, la disponibilité alimentaire journalière mondiale permettait de fournir 2771.0 kcal par être humain.
Sachant que la valeur journalière est de 2500 kcal pour un homme et 2000 kcal pour une femme,alors chaque être humain aura
it pu être nourris.
La disponibilité journalière mondiale en kcal aurait permis de nourrir 9.3 milliards d'êtres humains. Sachant qu'en 2017 n
ous étions 7.55 milliards d'êtres humains, la dispo alimentaire aurait permis de nourrir la population mondiale.
Avec la nourriture disponible mondialement en 2017, 123.0 % d'êtres humains auraient pu être nourris, soit 1.75 milliard d
e personnes supplémentaires.
```

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

In [48]:
```
# Transfert des données avec les végétaux dans un nouveau dataframe
dispo_alimentaire_vegetale = pd.DataFrame(join_population_dispo_2017.loc
                                          [join_population_dispo_2017['Origine'] == "vegetale"]
                                         )
```

In [49]:
```
# Calcul du nombre de kcal disponible pour les végétaux
dispo_kcal_mondiale_vege = dispo_alimentaire_vegetale['dispo_kcal'].sum()
print('La disponibilité alimentaire journalière d\'origine végétale '
      'était de {} trillions de kcal au niveau mondial.'
      .format(round((dispo_kcal_mondiale_vege/1_000_000_000_000),0)))
```

```
La disponibilité alimentaire journalière d'origine végétale était de 17.0 trillions de kcal au niveau mondial.
```

In [50]:
```
# Calcul du nombre d'humains pouvant être nourris avec des végétaux, résultat exprimé en kcal.
# 2250 = moyenne de la valeur journalière recommandée par l'OMS
calcul1 = dispo_kcal_mondiale_vege/2250

calcul_prop_pop_alimentee_vege = round((round((calcul1/1_000_000_000),2)
                                        /round((pop_totale/1_000_000_000),2)*100),0)

calcul_nb_pop_alimentee_vege = round(round((calcul1/1_000_000),2)
                                     -round((pop_totale/1_000_000),2),2)

print('En 2017, la disponibilité alimentaire journalière mondiale ayant pour origine des '
      'végétaux permettait de fournir {} kcal par être humain.'
      .format(round((dispo_kcal_mondiale_vege/pop_totale),0)))

print('Sachant que la valeur journalière est de 2500 kcal pour un homme et 2000 kcal pour une femme,'
      'alors chaque être humain pourrait être nourris de façon '
```

```
      'quotidienne uniquement avec l\'alimentation végétale.')

print('En 2017 la disponibilité végétale journalière mondiale '
      'permettait l\'accès à la nourriture à {} milliards d\'êtres humains.'
      .format(round((calcul1/1000000000),2)),
      'Sachant qu\'en 2017 nous étions {} milliards d\'êtres humains,'
      ' alors nous aurions pu nourrir toute la planète '
      'avec une alimentation strictement végétale.'
      .format(round((pop_totale/1000000000),2)))

print('Avec la nourriture disponible mondialement en 2017,', calcul_prop_pop_alimentee_vege,
      '% d\êtres humains auraient pu être nourris, soit', calcul_nb_pop_alimentee_vege,
      ' millions de personnes supplémentaire.' )
```

```
En 2017, la disponibilité alimentaire journalière mondiale ayant pour origine des végétaux permettait de fournir 2287.0 k
cal par être humain.
Sachant que la valeur journalière est de 2500 kcal pour un homme et 2000 kcal pour une femme,alors chaque être humain pou
rrait être nourris de façon quotidienne uniquement avec l'alimentation végétale.
En 2017 la disponibilité végétale journalière mondiale permettait l'accès à la nourriture à 7.67 milliards d'êtres humain
s. Sachant qu'en 2017 nous étions 7.55 milliards d'êtres humains, alors nous aurions pu nourrir toute la planète avec une
alimentation strictement végétale.
Avec la nourriture disponible mondialement en 2017, 102.0 % d\êtres humains auraient pu être nourris, soit 123.32  millio
ns de personnes supplémentaire.
```

3.4 - Utilisation de la disponibilité intérieure

In [51]:
```
# Calcul de la disponibilité totale
dispo_int_totale = dispo_alimentaire['Disponibilité intérieure'].sum()

print('La disponibilité intérieure totale est de {} gigatonnes.'
      .format(round((dispo_int_totale/1_000_000_000),1)))
```

```
La disponibilité intérieure totale est de 9.8 gigatonnes.
```

In [52]:
```
# Création d'une boucle for pour afficher les différentes valeurs en fonction des colonnes
# aliments pour animaux, pertes, nourritures,
liste_dispo = ['Aliments pour animaux','Pertes','Nourriture',
               'Autres Utilisations','Semences','Traitement']

for var in liste_dispo :
```

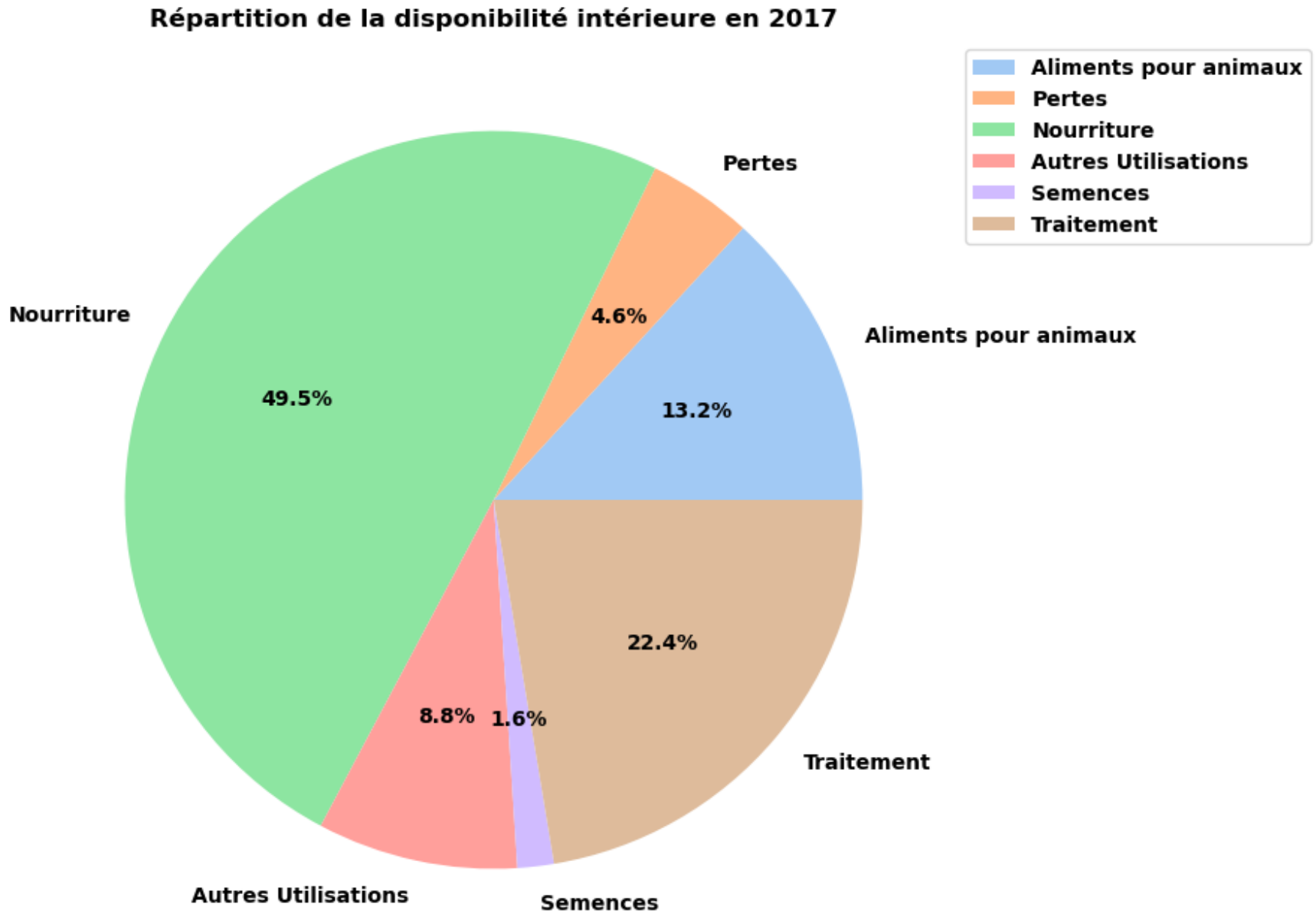
```
    print(var,': \n',' --- Valeurs:',round(((dispo_alimentaire[var])
                                        .sum()/1000_000_000),1), 'mégatonnes \n',
          ' --- Proportion :',round((dispo_alimentaire[var]
                                    .sum()*100)/dispo_int_totale,1),'%')
```

```
Aliments pour animaux :
  --- Valeurs: 1.3 mégatonnes
  --- Proportion : 13.2 %
Pertes :
  --- Valeurs: 0.5 mégatonnes
  --- Proportion : 4.6 %
Nourriture :
  --- Valeurs: 4.9 mégatonnes
  --- Proportion : 49.5 %
Autres Utilisations :
  --- Valeurs: 0.9 mégatonnes
  --- Proportion : 8.8 %
Semences :
  --- Valeurs: 0.2 mégatonnes
  --- Proportion : 1.6 %
Traitement :
  --- Valeurs: 2.2 mégatonnes
  --- Proportion : 22.4 %
```

In [53]:
```
# FIG : Répartition de la dispo intérieure - Diagramme circulaire
# Stocker les données a exploiter dans deux variables
data = [13.2, 4.6, 49.5, 8.8, 1.6, 22.4]
labels = ['Aliments pour animaux', 'Pertes','Nourriture',
          'Autres Utilisations','Semences','Traitement']
colors = sns.color_palette('pastel')
```

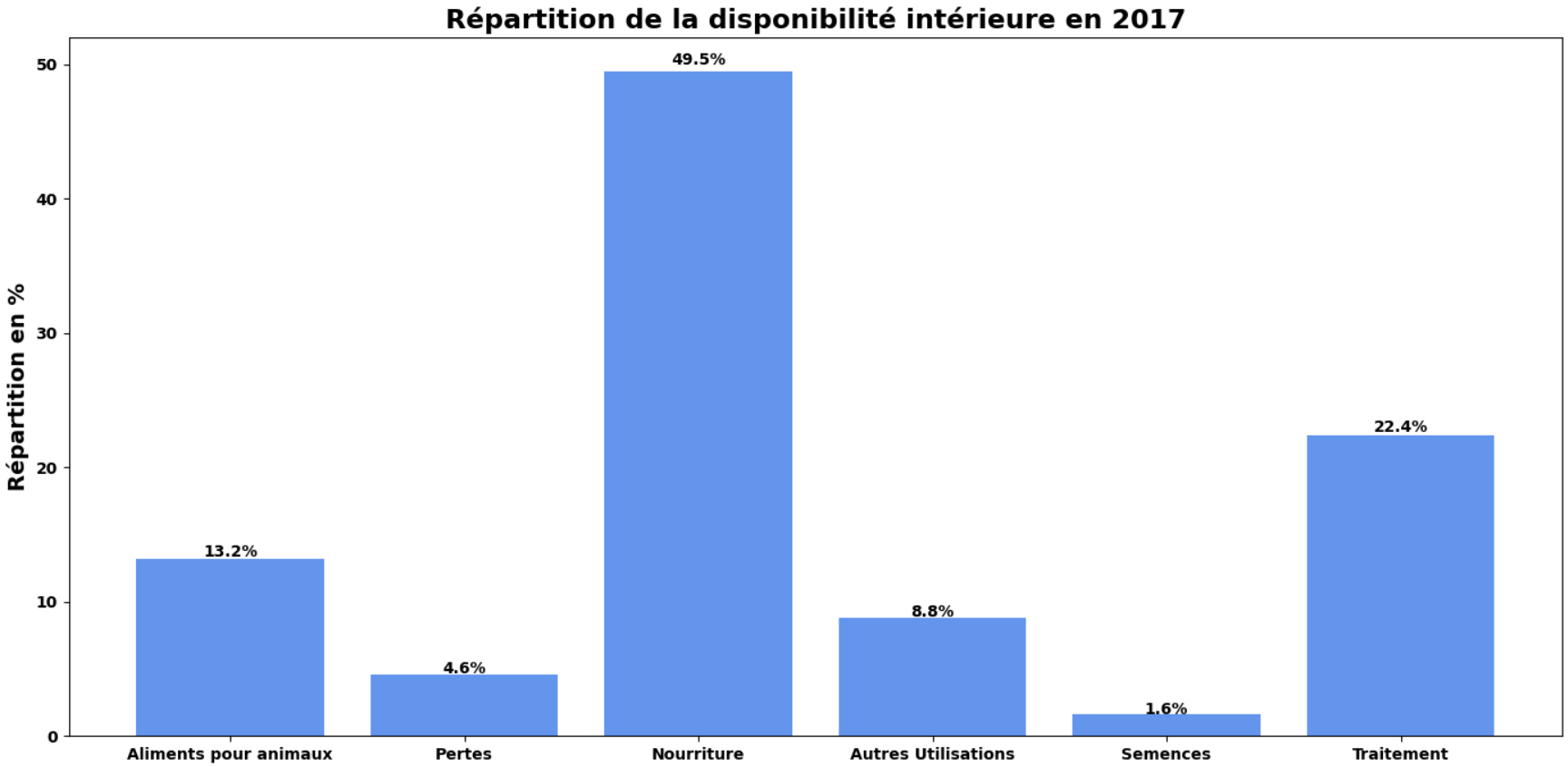
In [54]:
```
# Création et paramétrage de la figure
plt.rcParams['figure.figsize'] = [17,8]
plt.rcParams['text.color'] = 'black'
plt.rcParams['font.weight'] ='bold'
fig = plt.pie(data,
        colors = colors,
        labels = labels,
        autopct = '%.1F%%'
       )
title = "Répartition de la disponibilité intérieure en 2017"
plt.title(title,
          fontweight ='bold',
          loc = 'center',
          color = "black"
```

```
        )
plt.legend(bbox_to_anchor = [1, 1],
           labelcolor = "black",
           facecolor = "white"
          )
plt.show()
```



In [55]:
```
# FIG : Répartition de la dispo intérieure - Diagramme à barres
# Création et paramétrage de la figure"""
line1 = plt.bar(labels, data, color = 'cornflowerblue')

plt.title('Répartition de la disponibilité intérieure en 2017',
```

```
          fontsize=17,
          fontweight='bold'
         )
plt.xlabel('',
           fontsize=14,
           fontweight='bold'
          )
plt.ylabel('Répartition en %',
           fontsize=14,
           fontweight='bold'
          )
plt.yticks(np.arange(0, 60, step = 10))

i = 0
for p in line1:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()

    plt.text(x+width/2,
             y+height*1.01,
             str(data[i])+'%',
             ha='center',
             weight='bold')
    i += 1
plt.show()
```



3.5 - Utilisation des céréales

In [56]:
```
# Création d'une liste avec toutes les variables
liste_cereales = ['Blé','Riz (Eq Blanchi)','Orge','Maïs','Seigle',
                  'Avoine','Millet','Sorgho','Céréales,Autres']
```

In [57]:
```
# Création d'un dataframe avec les informations uniquement pour ces céréales
dispo_alimentaire1 = pd.DataFrame(dispo_alimentaire.groupby(['Produit'])
                                  .sum()
                                  .reset_index()
                                 )
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\1523312852.py:3: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_on
ly or select only columns which should be valid for the function.
  .sum()
```

In [58]:
```
# Création du dataframe céréales
cereales = pd.DataFrame(dispo_alimentaire1.loc[
    dispo_alimentaire1['Produit'].isin(liste_cereales)]
                        .reset_index(drop=True)
                       )
```

In [59]:
```
# Affichage de la proportion d'alimentation humaine
# Calcul de la proportion d'alimentation humaine globale en valeur et en pourcentage
proportion_humaine_valeur = ('Valeurs :',round((cereales['Nourriture']).sum(),0))
```

In [60]:
```
proportion_humaine_pourcent = (round((cereales['Nourriture'].sum()*100)
                                     /(cereales['Disponibilité intérieure'].sum()),0)
                               )

print('La proportion d\'alimentation humaine est de {} % par rapport '
      'à la disponibilité intérieure sur les produits céréaliers.'
      .format(proportion_humaine_pourcent))
```

```
La proportion d'alimentation humaine est de 43.0 % par rapport à la disponibilité intérieure sur les produits céréalie
rs.
```

In [61]:
```
# Calcul de la proportion d'alimentation humaine par type de céréales en pourcentage
cereales = cereales.assign(
    Proportion_humaine = round(100*cereales['Nourriture']
                               /cereales['Disponibilité intérieure'],2))
```

In [62]:
```
# Affichage de la proportion d'alimentation animale
# Calcul de la proportion d'alimentation animale globale en valeur et en pourcentage
proportion_animale_valeur = ('Valeurs :',round((cereales['Aliments pour animaux']).sum(),0))
```

In [63]:
```
proportion_animale_pourcent = (round((cereales['Aliments pour animaux'].sum()*100)
                                     /(cereales['Disponibilité intérieure'].sum()),1)
                               )

print('La proportion d\'alimentation animale est de {} % par rapport à la disponibilité '
      'intérieure sur les produits céréaliers.'
      .format(proportion_animale_pourcent))
```

La proportion d'alimentation animale est de 35.9 % par rapport à la disponibilité intérieure sur les produits céréaliers.

In [64]:
```
# Calcul de la proportion d'alimentation animale par type de céréales en pourcentage
cereales = cereales.assign(
    Proportion_animale = round(100*cereales['Aliments pour animaux']
                               /cereales['Disponibilité intérieure'],2))
```

In [65]:
```
# Calcul de la proportion d'alimentation autres par type de céréales en pourcentage
cereales = cereales.assign(
    Proportion_autres_utilisations = round(100*cereales['Autres Utilisations']
                               /cereales['Disponibilité intérieure'],2))

cereales = cereales.assign(
    Proportion_semences = round(100*cereales['Semences']
                               /cereales['Disponibilité intérieure'],2))

cereales = cereales.assign(
    Proportion_traitement = round(100*cereales['Traitement']
                               /cereales['Disponibilité intérieure'],2))

cereales = cereales.assign(
    Proportion_pertes = round(100*cereales['Pertes']
                               /cereales['Disponibilité intérieure'],2))

cereales = cereales.assign(
    Proportion_total_autres = cereales['Proportion_autres_utilisations']+cereales['Proportion_semences']
                              +cereales['Proportion_traitement']+cereales['Proportion_pertes'])
```

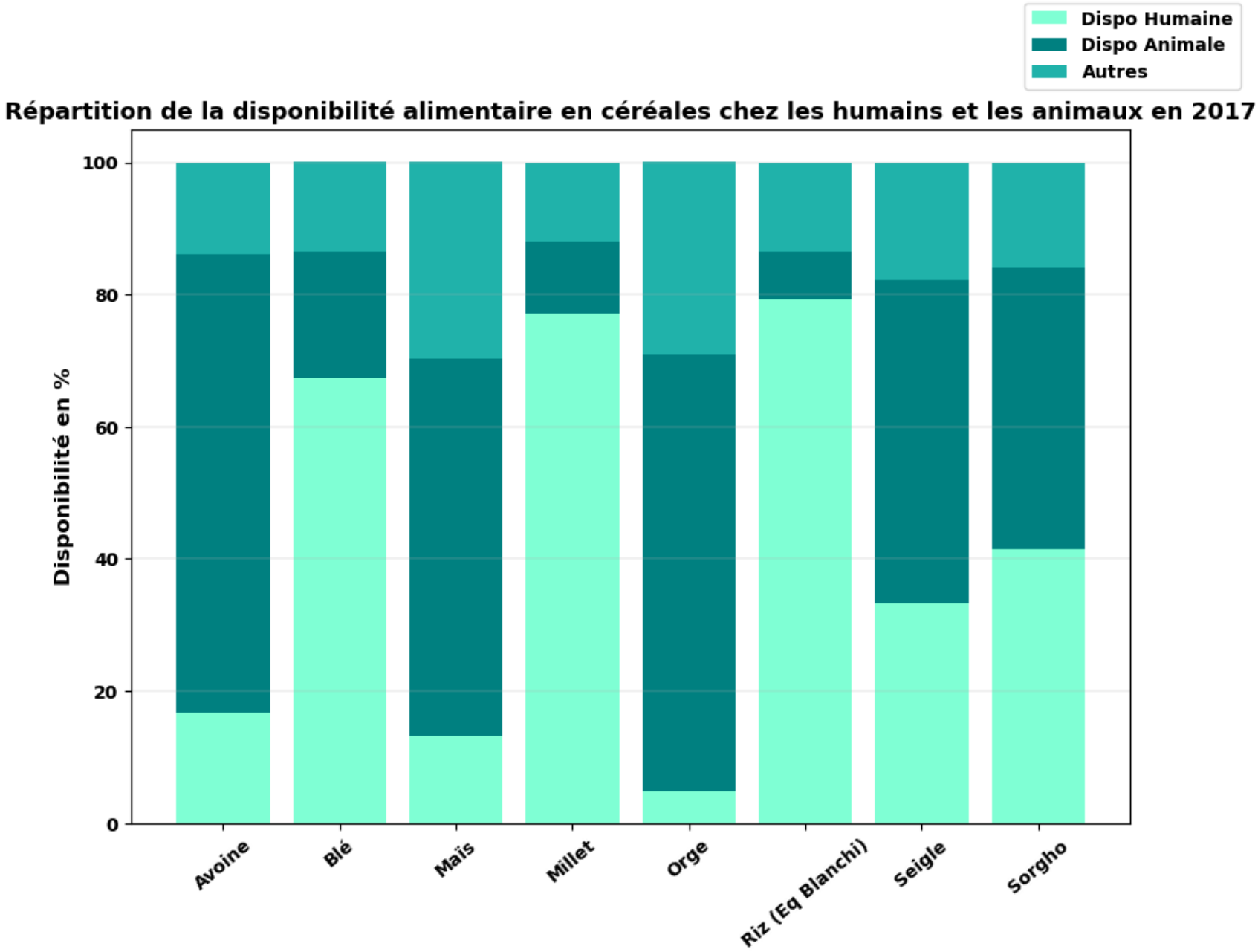
In [66]:
```
# Affichage des proportions animale et végétale des céréales
colonnes = ['Produit','Proportion_humaine','Proportion_animale','Proportion_autres_utilisations',
            'Proportion_semences','Proportion_traitement','Proportion_pertes']

display(cereales[colonnes])
```

	Produit	Proportion_humaine	Proportion_animale	Proportion_autres_utilisations	Proportion_semences	Proportion_traitement	Proportion_
0	Avoine	16.67	69.43	0.88	10.62	0.15	
1	Blé	67.38	19.08	3.30	5.05	1.16	
2	Maïs	13.10	57.14	19.81	0.72	4.99	
3	Millet	77.03	11.05	0.66	2.28	1.35	
4	Orge	4.84	65.98	0.47	6.27	19.16	
5	Riz (Eq Blanchi)	79.32	7.06	3.88	2.56	1.42	
6	Seigle	33.21	48.89	0.12	7.62	6.76	
7	Sorgho	41.47	42.60	3.52	1.35	6.34	

In [67]:
```
# FIG : Répartition de la disponibilité alimentaire en céréales pour les humains et les animaux
# Stocker les données à exploiter dans plusieurs variables
cereales_animaux = cereales['Proportion_animale']
cereales_humains = cereales['Proportion_humaine']
cereales_autres = (cereales['Proportion_autres_utilisations']+cereales['Proportion_semences']
                   +cereales['Proportion_traitement']+cereales['Proportion_pertes'])
```

In [68]:
```
# Création et paramétrage d'un diagramme à barres empilées
plt.figure(figsize=(10,7))

plt.bar(cereales.Produit,
        cereales_humains,
        color="aquamarine",
        label="Dispo Humaine"
       )
plt.bar(cereales.Produit,
        cereales_animaux,
        color="teal",
        bottom=cereales_humains,
        label="Dispo Animale"
       )
plt.bar(cereales.Produit,
        cereales_autres,
        color="lightseagreen",
        bottom=[sum(data) for data in zip(cereales_humains,cereales_animaux)],
        label="Autres"
```

```
       )
plt.title('Répartition de la disponibilité alimentaire en céréales chez les humains '
          'et les animaux en 2017',
          fontsize=13,
          fontweight='bold'
         )
plt.xticks(rotation = 40)

plt.ylabel('Disponibilité en %',
           fontsize=12,
           fontweight='bold'
          )
plt.legend(loc="center right",
           bbox_to_anchor=(1.12,1.12)
          )
plt.grid(axis='y',linewidth = 0.25)

plt.show()
```



In [69]:
```
colonnes_graph = ['Produit','Proportion_humaine','Proportion_animale','Proportion_total_autres']

prop_cereales = cereales[colonnes_graph]
```

In [70]:
```
prop_cereales
```

Out[70]:

	Produit	Proportion_humaine	Proportion_animale	Proportion_total_autres
0	Avoine	16.67	69.43	13.85
1	Blé	67.38	19.08	13.56
2	Maïs	13.10	57.14	29.78
3	Millet	77.03	11.05	11.90
4	Orge	4.84	65.98	29.20
5	Riz (Eq Blanchi)	79.32	7.06	13.62
6	Seigle	33.21	48.89	17.89
7	Sorgho	41.47	42.60	15.93

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

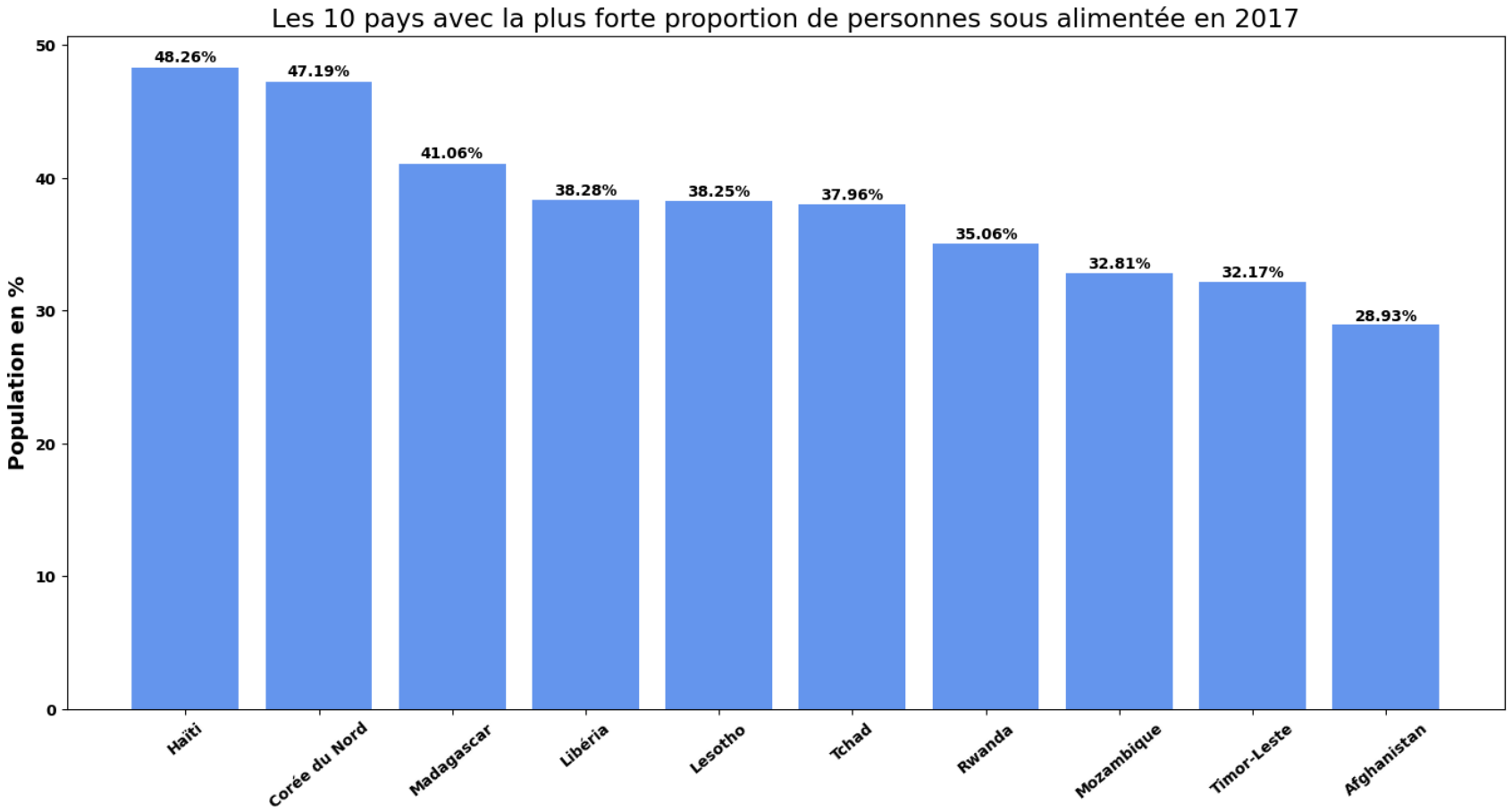
In [71]:
```
# Création de la colonne proportion par pays
data_pop_sous_nutrition_2017 = data_pop_sous_nutrition_2017.assign(
    Proportion_pop_sous_nutrie = round(
        100*data_pop_sous_nutrition_2017['sous_nutrition'] /data_pop_sous_nutrition_2017['Population'],2))

data_pop_sous_nutrition_2017 = data_pop_sous_nutrition_2017.assign(
    Proportion_pop_nutrie = 100 - data_pop_sous_nutrition_2017['Proportion_pop_sous_nutrie'])

data_pop_sous_nutrition_2017['Somme']  = (data_pop_sous_nutrition_2017['Proportion_pop_sous_nutrie']
                                          + data_pop_sous_nutrition_2017['Proportion_pop_nutrie'])
```

In [72]:
```
# Affichage après tri des 10 pires pays
pires_pays = data_pop_sous_nutrition_2017.sort_values(
    by = 'Proportion_pop_sous_nutrie', ascending = False).reset_index(drop = True).head(10)
```

In [73]:
```
# Création d'un tableau de chiffres
variables = ['Zone','Proportion_pop_sous_nutrie','Proportion_pop_nutrie','Somme']

tableau_pires_pays = (pires_pays[variables]).sort_values(
    by = 'Proportion_pop_sous_nutrie',ascending = False)

tableau_pires_pays.to_csv('tableau_pires_pays.csv')
```

In [74]:
```
# FIG : 10 pays avec la plus forte proportion de personnes sous alimentées
tableau_pires_pays['Zone'] = tableau_pires_pays['Zone'].replace(
    ['République populaire démocratique de Corée'],'Corée du Nord')
```

In [75]:
```
# Création et paramétrage d'un diagramme à barres empilées"""
line1 = plt.bar(tableau_pires_pays.Zone,
                tableau_pires_pays.Proportion_pop_sous_nutrie,
                color = 'cornflowerblue'
               )
plt.title('Les 10 pays avec la plus forte proportion de personnes sous alimentée en 2017',
          fontsize=17
         )
plt.xlabel('',
           fontsize=14,
           fontweight='bold'
          )
plt.ylabel('Population en %',
           fontsize=14,
           fontweight='bold'
          )
plt.yticks(np.arange(0, 60, step = 10))
plt.xticks(rotation = 40)

i = 0
for p in line1:
    width = p.get_width()
    height = p.get_height()
    x, y = p.get_xy()

    plt.text(x+width/2,
             y+height*1.01,
             str(tableau_pires_pays.Proportion_pop_sous_nutrie[i])+'%',
             ha='center',
             weight='bold')
    i += 1
plt.show()
```



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

In [76]:
```
# Calcul du total de l'aide alimentaire par pays
aide_alim_pays = aide_alimentaire.groupby(['Zone']).sum().reset_index()
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\2087332427.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 nume
ric_only or select only columns which should be valid for the function.
  aide_alim_pays = aide_alimentaire.groupby(['Zone']).sum().reset_index()
```

In [77]:
```
# Affichage après trie des 10 pays qui ont bénéficié le plus de l'aide alimentaire
variables = ['Zone','Valeur']

tableau = aide_alim_pays[variables]

aide_alim_2013 = tableau.sort_values(
    by = 'Valeur', ascending = False).reset_index(drop = True).head(10)

aide_alim_2013.head(10)
```

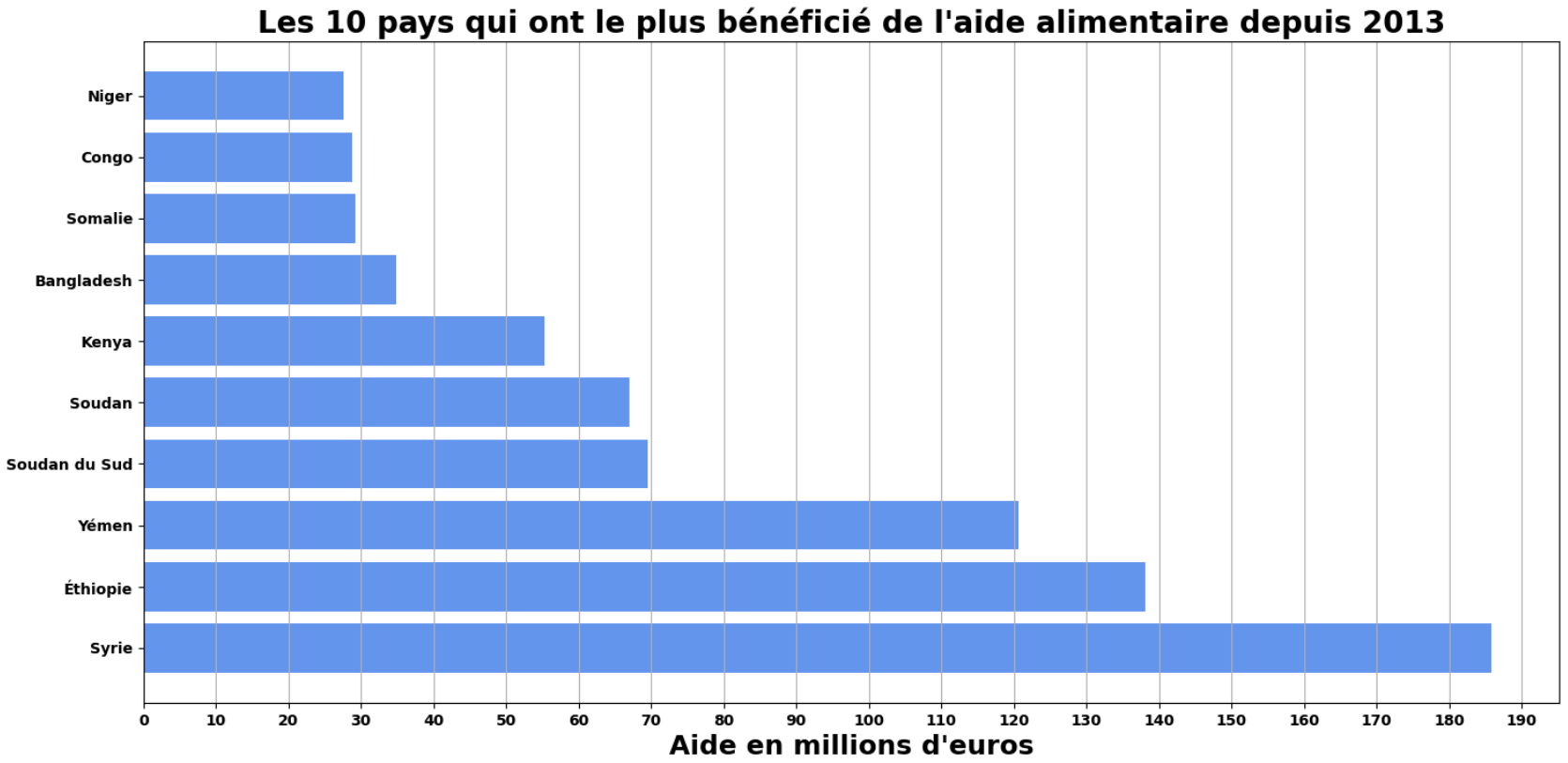
Out[77]:

	Zone	Valeur
0	République arabe syrienne	1858943000
1	Éthiopie	1381294000
2	Yémen	1206484000
3	Soudan du Sud	695248000
4	Soudan	669784000
5	Kenya	552836000
6	Bangladesh	348188000
7	Somalie	292678000
8	République démocratique du Congo	288502000
9	Niger	276344000

In [78]:
```
# Renommer les pays
aide_alim_2013['Zone'] = aide_alim_2013['Zone'].replace(
    ['République arabe syrienne'], 'Syrie'
)
aide_alim_2013['Zone'] = aide_alim_2013['Zone'].replace(
    ['République démocratique du Congo'], 'Congo'
)
```

In [79]:
```
# FIG : les 10 pays qui ont le plus bénéficié de l'aide alimentaire
"""Stockage des variables Valeur (à la bonne échelle) et Zone"""
aide = (aide_alim_2013['Valeur']/10_000_000)

pays_aide = aide_alim_2013['Zone']
```

In [80]:
```
# Création du diagramme à barres horizontal
plt.barh(pays_aide,
         aide,
         color = 'cornflowerblue'
        )
plt.title('Les 10 pays qui ont le plus bénéficié de l\'aide alimentaire depuis 2013',
          fontsize=20,
          fontweight='bold'
         )
plt.xlabel('Aide en millions d\'euros',
           fontsize = 18
          )
plt.xticks(np.arange(0, 200, step = 10))

plt.grid(axis='x')

plt.show()
```



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

In [81]:
```
# Création d'un dataframe avec la zone, l'année et l'aide alimentaire puis groupby sur zone et année
aide_alim_2013_2016 = pd.DataFrame(aide_alimentaire.loc[
    aide_alimentaire.Année.isin([2013,2014,2015,2016]),:]
                                   )
```

In [82]:
```
#Faire le groupby sur zone et année
aide_alim_2013_2016 = aide_alim_2013_2016.groupby(['Zone','Année']).sum().reset_index()
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\3876596678.py:2: FutureWarning: The default value of numeric_onl
y in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify n
umeric_only or select only columns which should be valid for the function.
  aide_alim_2013_2016 = aide_alim_2013_2016.groupby(['Zone','Année']).sum().reset_index()
```

In [83]:
```
# Création d'une liste contenant les 5 pays qui ont le plus bénéficiées de l'aide alimentaire
aide_alim_5pays = ['République arabe syrienne','Éthiopie',
                   'Yémen','Soudan du Sud','Soudan']
```

In [84]:
```
# On filtre sur le dataframe avec notre liste
aide_alim_liste_5_pays = aide_alim_2013_2016.loc[
    aide_alim_2013_2016.Zone.isin(aide_alim_5pays)
]
```

In [85]:
```
# Affichage des pays avec l'aide alimentaire par année
aide_alim_liste_5_pays.head(20).reset_index()
```

Out[85]:

	index	Zone	Année	Valeur
0	157	République arabe syrienne	2013	563566000
1	158	République arabe syrienne	2014	651870000
2	159	République arabe syrienne	2015	524949000
3	160	République arabe syrienne	2016	118558000
4	189	Soudan	2013	330230000
5	190	Soudan	2014	321904000
6	191	Soudan	2015	17650000
7	192	Soudan du Sud	2013	196330000
8	193	Soudan du Sud	2014	450610000
9	194	Soudan du Sud	2015	48308000
10	214	Yémen	2013	264764000
11	215	Yémen	2014	103840000
12	216	Yémen	2015	372306000
13	217	Yémen	2016	465574000
14	225	Éthiopie	2013	591404000
15	226	Éthiopie	2014	586624000
16	227	Éthiopie	2015	203266000

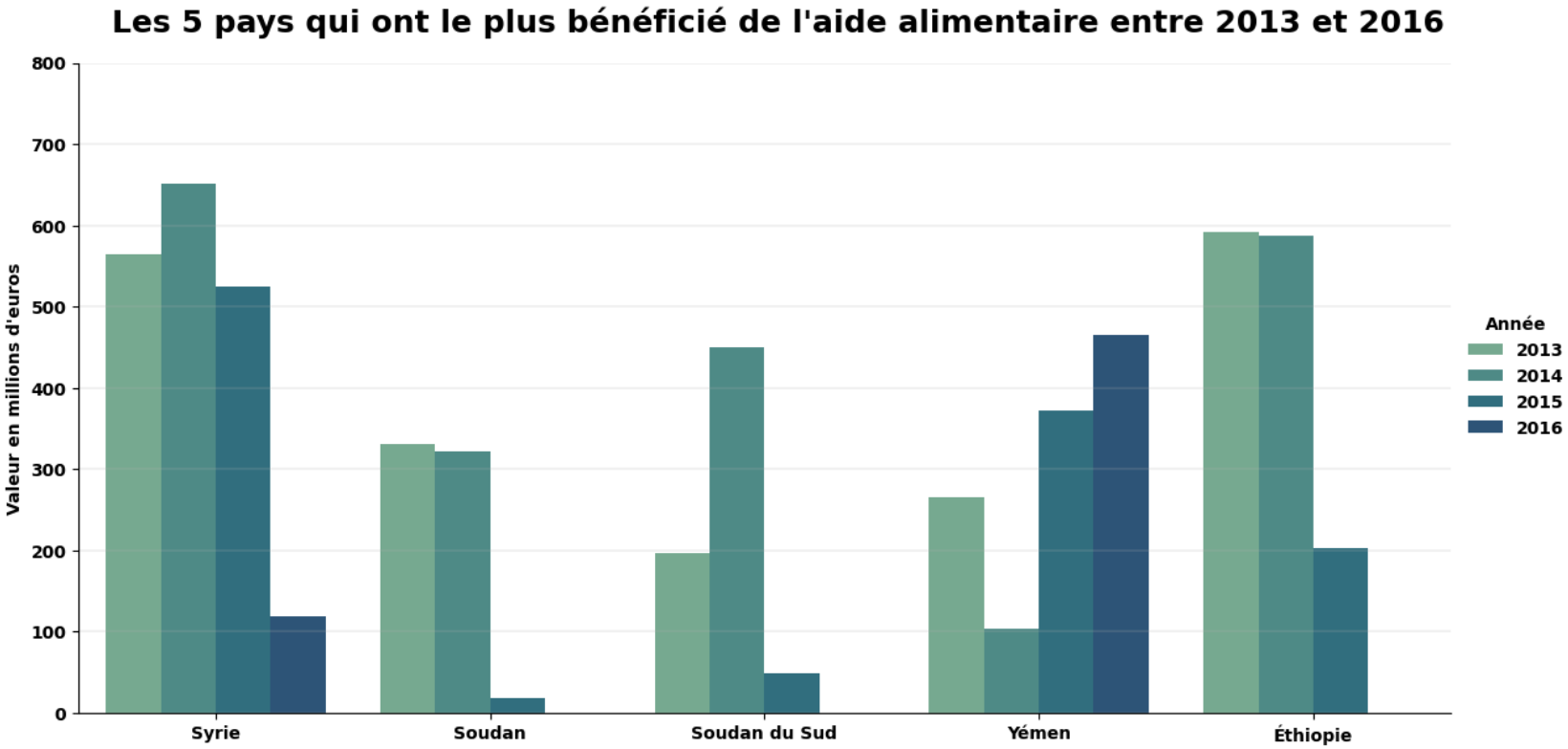
In [86]:

```
# FIG : les 5 pays qui ont le plus bénéficié de l'aide alimentaire
aide_alim_liste_5_pays['Zone'] = aide_alim_liste_5_pays['Zone'].replace(
    ['République arabe syrienne'], 'Syrie'
)

"""Création d'une nouvelle colonne Valeur Million pour mise à l'échelle"""
aide_alim_liste_5_pays['Valeur Million'] = aide_alim_liste_5_pays['Valeur']/1_000_000
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\4172208517.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  aide_alim_liste_5_pays['Zone'] = aide_alim_liste_5_pays['Zone'].replace(
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\4172208517.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy
  aide_alim_liste_5_pays['Valeur Million'] = aide_alim_liste_5_pays['Valeur']/1_000_000
```

In [87]:

```python
# Création du diagramme à barres
g = sns.catplot(
    data = aide_alim_liste_5_pays,
    kind = "bar",
    x = 'Zone',
    y = 'Valeur Million',
    hue = 'Année',
    palette = "crest",
    height = 6,
    aspect = 20/10,
    legend_out = True
)
g.fig.suptitle(
    'Les 5 pays qui ont le plus bénéficié de l\'aide alimentaire entre 2013 et 2016',
    fontsize = 18,
    fontweight = 'bold',
    y = 1.05
  )
g.set_axis_labels("", "Valeur en millions d\'euros")

g.set(ylim = (0, 800))

plt.grid(axis = 'y', linewidth = 0.25)
```



3.9 - Pays avec le moins de disponibilité par habitant

In [88]:
```
# Calcul de la disponibilité en kcal par personne par jour par pays
dispo_alimentaire_pays = dispo_alimentaire.groupby(["Zone"]).sum().reset_index()
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\923643618.py:2: FutureWarning: The default value of numeric_on
ly in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specif
y numeric_only or select only columns which should be valid for the function.
  dispo_alimentaire_pays = dispo_alimentaire.groupby(["Zone"]).sum().reset_index()
```

In [89]:
```
# Affichage des 10 pays qui ont le moins de dispo alimentaire par personne
dispo_10_pays = dispo_alimentaire_pays.sort_values(
```

```
    by = 'Disponibilité alimentaire (Kcal/personne/jour)', ascending = True
).reset_index(drop=True).head(10)
```

In [90]:
```
# Création d'un tableau de chiffres
variables = ['Zone','Disponibilité alimentaire (Kcal/personne/jour)']

tableau_pays_moins_dispo = dispo_10_pays[variables]

display(tableau_pays_moins_dispo)

tableau_pays_moins_dispo.to_csv('tableau_moins_dispo.csv')
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	République centrafricaine	1879.0
1	Zambie	1924.0
2	Madagascar	2056.0
3	Afghanistan	2087.0
4	Haïti	2089.0
5	République populaire démocratique de Corée	2093.0
6	Tchad	2109.0
7	Zimbabwe	2113.0
8	Ouganda	2126.0
9	Timor-Leste	2129.0

3.10 - Pays avec le plus de disponibilité par habitant

In [91]:
```
# Affichage des 10 pays qui ont le plus de dispo alimentaire par personne
dispo_forte_10_pays = dispo_alimentaire_pays.sort_values(
    by = 'Disponibilité alimentaire (Kcal/personne/jour)', ascending = False
).reset_index(drop=True).head(10)
```

In [92]:
```
# Création d'un tableau de chiffres
variables = ['Zone','Disponibilité alimentaire (Kcal/personne/jour)']

tableau_pays_forte_dispo = dispo_forte_10_pays[variables]

display(tableau_pays_forte_dispo)

tableau_pays_forte_dispo.to_csv('tableau_pays_forte_dispo.csv')
```

	Zone	Disponibilité alimentaire (Kcal/personne/jour)
0	Autriche	3770.0
1	Belgique	3737.0
2	Turquie	3708.0
3	États-Unis d'Amérique	3682.0
4	Israël	3610.0
5	Irlande	3602.0
6	Italie	3578.0
7	Luxembourg	3540.0
8	Égypte	3518.0
9	Allemagne	3503.0

3.11 - Exemple de la Thaïlande pour le Manioc

In [93]:
```
# Création d'un dataframe avec uniquement la Thaïlande
thailande = pd.merge(data_pop_sous_nutrition_2017.loc[(data_pop_sous_nutrition_2017['Zone'] == "Thaïlande")],
                     dispo_alimentaire_pays,
                     on = 'Zone',
                     how = 'left'
                    )
```

In [94]:
```
# Calcul de la sous-nutrition en Thaïlande
calcul3 = round((thailande['sous_nutrition'].sum()/
                 thailande['Population'].sum()*100),0)

print(f'En 2017, presque {calcul3} % de la population Thaïlandaise était sous alimentée.')
```

```
En 2017, presque 9.0 % de la population Thaïlandaise était sous alimentée.
```

In [95]:
```
# On calcule la proportion exportée en fonction de la production
info_manioc = dispo_alimentaire.loc[(dispo_alimentaire['Produit'] == 'Manioc')
                                    & (dispo_alimentaire['Zone'] == 'Thaïlande')].reset_index(drop=True)
```

In [96]:
```
calcul_manioc = round((info_manioc.loc[0, 'Exportations - Quantité']*100
                       /info_manioc.loc[0, 'Production']),0)
```

In [97]:
```
# On calcule la proportion importée en fonction de l'exportation
import_export = round((info_manioc.loc[0, 'Importations - Quantité']*100
                       /info_manioc.loc[0, 'Exportations - Quantité']),0)
```

In [98]:
```
print(f'En 2017, {calcul_manioc} % de la production thaïlandaise de manioc a été exportée, et {import_export}
```

```
En 2017, 83.0 % de la production thaïlandaise de manioc a été exportée, et 5.0 % a été importée dans le pays.
```

In [99]:
```
# FIG : Répartition de la production de manioc en Thaïlande
liste_disponible = ['Pertes','Nourriture','Exportations - Quantité']

for var in liste_disponible :
    print(var,': \n', '--- Proportion :',round((info_manioc.loc[0,var])
                                        /info_manioc.loc[0,'Production']*100,1),'%')
```

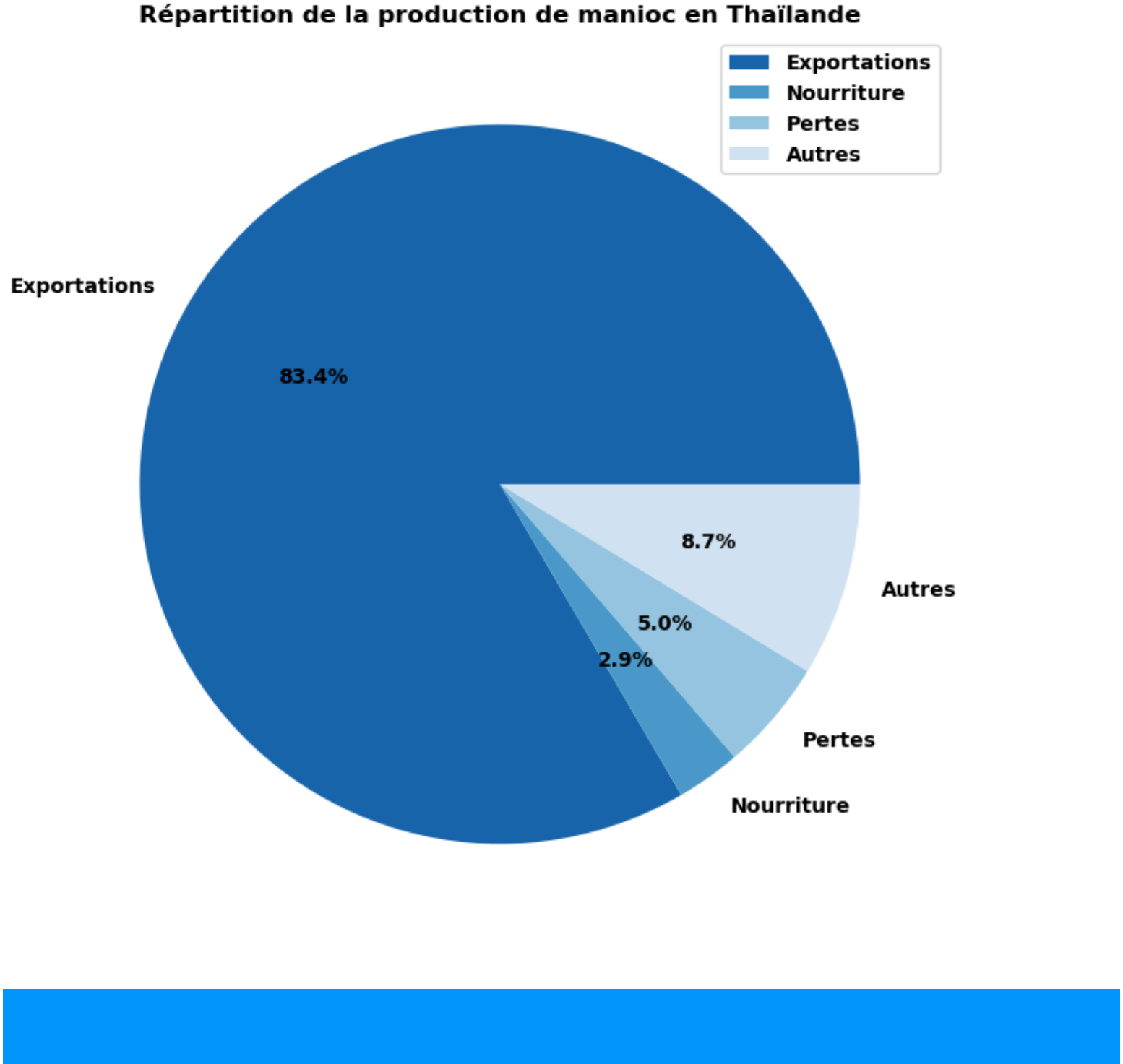
```
Pertes :
 --- Proportion : 5.0 %
Nourriture :
 --- Proportion : 2.9 %
Exportations - Quantité :
 --- Proportion : 83.4 %
```

In [100…

```
# Création d'un diagramme circulaire
data = [83.4,2.9,5,8.7]
labels = ['Exportations','Nourriture','Pertes', 'Autres']
colors = sns.color_palette('Blues_r',4)
```

In [101…

```
plt.rcParams['figure.figsize'] = [16,8]
plt.rcParams['text.color'] = 'k'
plt.rcParams['font.weight'] = 'bold'

fig1 = plt.pie(data,
               colors = colors,
               labels = labels,
               autopct = '%.1F%%'
              )
title = "Répartition de la production de manioc en Thaïlande"

plt.title(title,
          fontweight ='bold',
          loc = 'center',
          color = "k"
         )
plt.legend(bbox_to_anchor = [1,1],
           labelcolor = "black",
           facecolor = "white"
          )
plt.show()
```


Répartition de la production de manioc en Thaïlande
Exportations
Nourriture
Pertes
Autres
Exportations
83.4%
8.7%
Autres
5.0%
2.9%
Pertes
Nourriture

Etape 6 - Analyse complémentaires

In [102…
```
#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"
```

In [103…
```
# Disponibilité par habitant de la Thaïlande
dispo_hab_thai = thailande['Disponibilité alimentaire (Kcal/personne/jour)'].iloc[0]
print('En 2017, la disponibilité alimentaire en Thaïlande était de {} kcal par jour et par personne'
      .format(round(dispo_hab_thai),0))
```

```
En 2017, la disponibilité alimentaire en Thaïlande était de 2785 kcal par jour et par personne
```

In [104…
```
# Part d'alimentation végétale/animale par rapport à la production dans les pays les plus en difficulté 2017
pays_defavorises = ['République centrafricaine','Zambie','Madagascar','Afghanistan',
                    'Haïti','République populaire démocratique de Corée','Tchad','Zimbabwe',
                    'Ouganda','Timor-Leste']
```

In [105…
```
# Selection des pays défavorisés via une liste
focus_pays_defav = dispo_alimentaire.loc[
    dispo_alimentaire.Zone.isin(pays_defavorises)
]
```

In [106…
```
# Groupby sur la zone et l'origine
focus_pays_defav = focus_pays_defav.groupby(
    ['Zone','Origine']).sum().reset_index()
focus_pays_defav
```

```
C:\Users\pauli\AppData\Local\Temp\ipykernel_19128\2283305055.py:3: FutureWarning: The default value of numer
ic_only in DataFrameGroupBy.sum 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.
  ['Zone','Origine']).sum().reset_index()
```

Out[106]:

	Zone	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	animale	123000.0	0.0	216.0	79.92	15.23	12.21
1	Afghanistan	vegetale	645000.0	415000.0	1871.0	271.49	18.27	46.05
2	Haïti	animale	2000.0	6000.0	154.0	44.16	9.94	10.15
3	Haïti	vegetale	479000.0	109000.0	1935.0	353.62	38.98	37.55
4	Madagascar	animale	20000.0	4000.0	149.0	46.61	10.13	10.03
5	Madagascar	vegetale	822000.0	2518000.0	1907.0	378.46	14.36	36.66
6	Ouganda	animale	2000.0	0.0	183.0	64.84	12.25	12.38
7	Ouganda	vegetale	346000.0	68000.0	1943.0	470.74	34.32	40.26
8	République centrafricaine	animale	0.0	1000.0	206.0	55.41	13.94	17.12
9	République centrafricaine	vegetale	1000.0	26000.0	1673.0	398.32	45.47	28.92
10	République populaire démocratique de Corée	animale	1000.0	444000.0	130.0	32.80	9.34	10.07
11	République populaire démocratique de Corée	vegetale	815000.0	251000.0	1963.0	426.13	27.07	44.92
12	Tchad	animale	7000.0	0.0	118.0	40.07	7.36	9.71
13	Tchad	vegetale	97000.0	2000.0	1991.0	279.38	39.74	53.04
14	Timor-Leste	animale	0.0	2000.0	245.0	57.04	18.06	17.74
15	Timor-Leste	vegetale	9000.0	0.0	1884.0	312.06	33.02	39.77
16	Zambie	animale	6000.0	1000.0	106.0	33.65	6.99	9.23
17	Zambie	vegetale	402000.0	68000.0	1818.0	316.47	35.06	45.99
18	Zimbabwe	animale	46000.0	3000.0	178.0	57.67	12.54	11.76

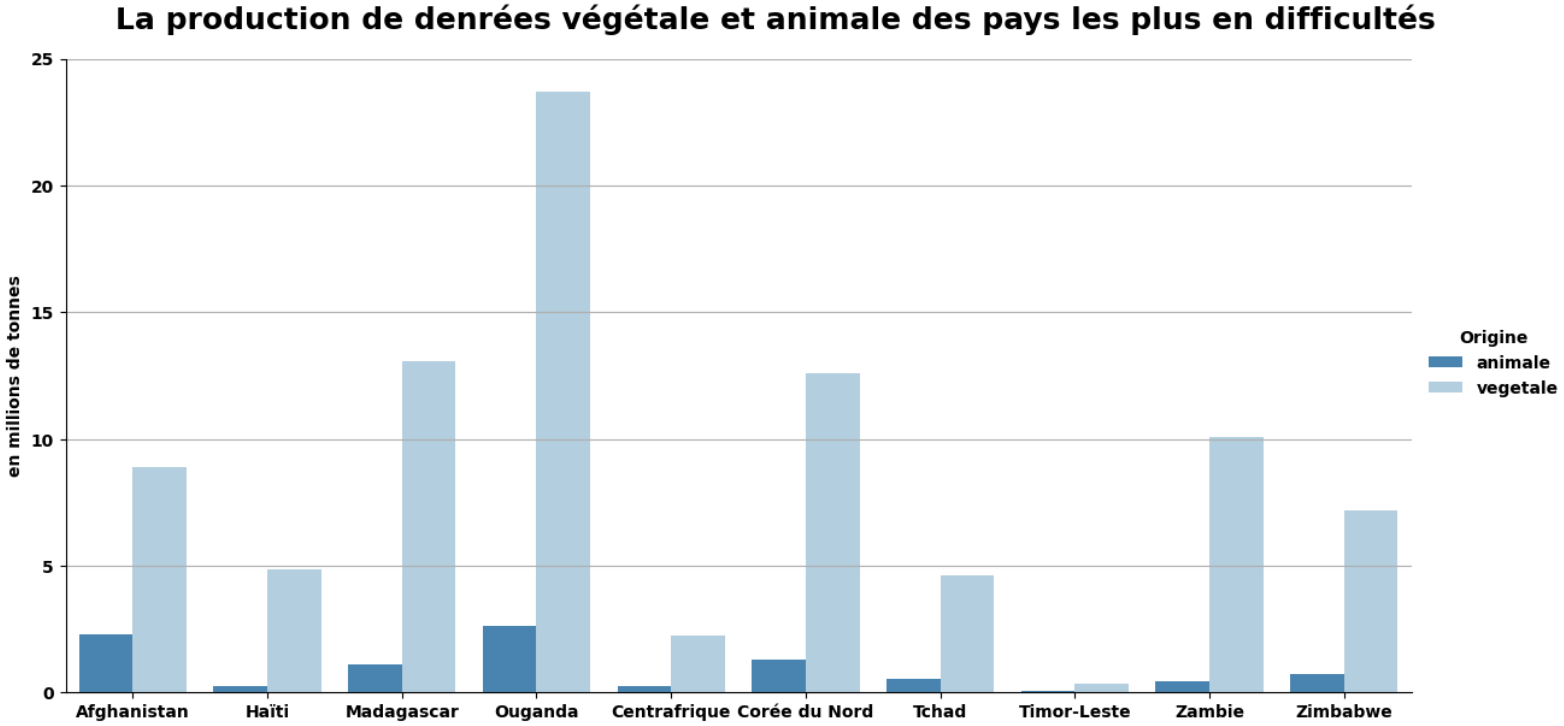
	Zone	Origine	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)
19	Zimbabwe	vegetale	85000.0	20000.0	1935.0	278.92	44.64	36.56

In [107…
```
# FIG : La production de denrées végétale et animale des pays les plus en difficultés
"""Renommer deux pays pour raccourcir le nom et gagner en lisibilité sur le graphique et mettre à l'échelle
focus_pays_defav['Zone'] = focus_pays_defav['Zone'].replace(
    ['République populaire démocratique de Corée'], 'Corée du Nord'
)
```

In [108…
```
focus_pays_defav['Zone'] = focus_pays_defav['Zone'].replace(
    ['République centrafricaine'], 'Centrafrique'
)
```

In [109…
```
focus_pays_defav['Prod Millions Tonnes'] = focus_pays_defav['Production']/1_000_000
```

In [110…
```
# Création d'un diagramme à barres
h = sns.catplot(
    data = focus_pays_defav,
    kind = "bar",
    x = 'Zone',
    y = 'Prod Millions Tonnes',
    hue = 'Origine',
    palette = 'Blues_r',
    height = 6,
    aspect = 20/10,
    legend_out = True
)
h.fig.suptitle(
    'La production de denrées végétale et animale des pays les plus en difficultés',
    fontsize = 18,
    fontweight = 'bold',
    y = 1.05
  )
h.set_axis_labels("", "en millions de tonnes")
h.set(ylim = (0, 25))
plt.grid(axis = 'y')
plt.show()
```



In [111…

```
focus_pays_defav['Proportion_prod_mondiale'] = round((focus_pays_defav['Production']/dispo_alimentaire['Prod
focus_pays_defav
```

Out[111]:

	Zone	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)	D
0	Afghanistan	animale	123000.0	0.0	216.0	79.92	15.23	12.21	
1	Afghanistan	vegetale	645000.0	415000.0	1871.0	271.49	18.27	46.05	
2	Haïti	animale	2000.0	6000.0	154.0	44.16	9.94	10.15	
3	Haïti	vegetale	479000.0	109000.0	1935.0	353.62	38.98	37.55	
4	Madagascar	animale	20000.0	4000.0	149.0	46.61	10.13	10.03	
5	Madagascar	vegetale	822000.0	2518000.0	1907.0	378.46	14.36	36.66	
6	Ouganda	animale	2000.0	0.0	183.0	64.84	12.25	12.38	
7	Ouganda	vegetale	346000.0	68000.0	1943.0	470.74	34.32	40.26	
8	Centrafrique	animale	0.0	1000.0	206.0	55.41	13.94	17.12	
9	Centrafrique	vegetale	1000.0	26000.0	1673.0	398.32	45.47	28.92	
10	Corée du Nord	animale	1000.0	444000.0	130.0	32.80	9.34	10.07	
11	Corée du Nord	vegetale	815000.0	251000.0	1963.0	426.13	27.07	44.92	
12	Tchad	animale	7000.0	0.0	118.0	40.07	7.36	9.71	
13	Tchad	vegetale	97000.0	2000.0	1991.0	279.38	39.74	53.04	
14	Timor-Leste	animale	0.0	2000.0	245.0	57.04	18.06	17.74	
15	Timor-Leste	vegetale	9000.0	0.0	1884.0	312.06	33.02	39.77	
16	Zambie	animale	6000.0	1000.0	106.0	33.65	6.99	9.23	
17	Zambie	vegetale	402000.0	68000.0	1818.0	316.47	35.06	45.99	
18	Zimbabwe	animale	46000.0	3000.0	178.0	57.67	12.54	11.76	
19	Zimbabwe	vegetale	85000.0	20000.0	1935.0	278.92	44.64	36.56	

In [112…

```
# Création d'un tableau de chiffres
variables = ['Zone','Origine','Prod Millions Tonnes','Proportion_prod_mondiale']

tableau_prod_vege_animal = focus_pays_defav[variables]

display(tableau_prod_vege_animal)

tableau_prod_vege_animal.to_csv('tableau_prod_vege_animal.csv')
```

	Zone	Origine	Prod Millions Tonnes	Proportion_prod_mondiale
0	Afghanistan	animale	2.280	0.023
1	Afghanistan	vegetale	8.891	0.089
2	Haïti	animale	0.239	0.002
3	Haïti	vegetale	4.833	0.048
4	Madagascar	animale	1.101	0.011
5	Madagascar	vegetale	13.059	0.130
6	Ouganda	animale	2.603	0.026
7	Ouganda	vegetale	23.718	0.237
8	Centrafrique	animale	0.249	0.002
9	Centrafrique	vegetale	2.254	0.023
10	Corée du Nord	animale	1.307	0.013
11	Corée du Nord	vegetale	12.576	0.126
12	Tchad	animale	0.525	0.005
13	Tchad	vegetale	4.628	0.046
14	Timor-Leste	animale	0.044	0.000
15	Timor-Leste	vegetale	0.327	0.003
16	Zambie	animale	0.430	0.004
17	Zambie	vegetale	10.096	0.101
18	Zimbabwe	animale	0.740	0.007
19	Zimbabwe	vegetale	7.174	0.072

In [113…

```
#Différents calculs par rapport à la production et à la dispo intérieure
dispo_10_pays = dispo_10_pays.assign(
    taux_Export = round((dispo_10_pays['Exportations - Quantité']/dispo_10_pays['Production']*100),2)
)
dispo_10_pays = dispo_10_pays.assign(
    taux_Import = round((dispo_10_pays['Importations - Quantité']/dispo_10_pays['Disponibilité intérieure']*
)
```

```
dispo_10_pays = dispo_10_pays.assign(
    taux_Perte = round((dispo_10_pays['Pertes']/dispo_10_pays['Production']*100),2)
)
dispo_10_pays = dispo_10_pays.assign(
    taux_Traitement = round((dispo_10_pays['Traitement']/dispo_10_pays['Production']*100),2)
)
dispo_10_pays = dispo_10_pays.assign(
    Part_Alimentation_Animale = round((dispo_10_pays['Aliments pour animaux']/dispo_10_pays['Production']*10
)
dispo_10_pays = dispo_10_pays.assign(
    Part_Alimentation_Humaine = round((dispo_10_pays['Nourriture']/dispo_10_pays['Production']*100),2)
)
dispo_10_pays = dispo_10_pays.assign(
    Part_prod_mondiale = round((dispo_10_pays['Production']/dispo_alimentaire['Production'].sum()*100),2)
)
```

In [115…

```
# Création d'un tableau de chiffres
dispo_10_pays['Zone'] = dispo_10_pays['Zone'].replace(
    ['République populaire démocratique de Corée'], 'Corée du Nord')
dispo_10_pays['Zone'] = dispo_10_pays['Zone'].replace(
    ['République centrafricaine'], 'Centrafrique')

variables = ['Zone','taux_Export','taux_Import','taux_Perte', 'taux_Traitement',
             'Part_Alimentation_Animale','Part_Alimentation_Humaine',
             'Part_prod_mondiale']

tableau_taux = dispo_10_pays[variables]

display(tableau_taux)

tableau_taux.to_csv('tableau_taux.csv')
```

	Zone	taux_Export	taux_Import	taux_Perte	taux_Traitement	Part_Alimentation_Animale	Part_Alimentation_Humaine	P
0	Centrafrique	0.00	2.94	4.87	11.11	0.04	83.58	
1	Zambie	6.41	3.78	2.30	39.69	3.88	48.34	
2	Madagascar	1.67	6.07	6.82	5.96	5.95	68.79	
3	Afghanistan	2.49	22.11	10.16	0.56	6.87	96.10	
4	Haïti	0.30	19.50	13.39	15.02	9.48	80.86	
5	Corée du Nord	0.60	6.68	7.68	4.88	5.88	82.29	
6	Tchad	0.00	5.47	8.91	10.52	2.02	79.51	
7	Zimbabwe	3.51	14.12	2.38	54.71	1.66	60.17	
8	Ouganda	4.63	4.72	3.77	18.93	1.32	76.44	
9	Timor-Leste	6.20	22.37	3.77	0.81	2.43	112.94	

In [116…
```
# Création du diagramme à barres empilées pour montrer le nombre de
# personnes sous-nutrie par rapport à la population totale des 10 pays les plus en difficulté
pires_pays['Zone'] = pires_pays['Zone'].replace(
    ['République populaire démocratique de Corée'],'Corée du Nord')
```

In [117…
```
# Stocker les pays dans une variable
pays = pires_pays['Zone']
```

In [118…
```
# Ajout colonnes pop_totale, pop_sous_nutrition et somme
pires_pays['pop_totale'] = (pires_pays['Population']/10_000_000)

pires_pays['pop_sous_nutrition'] = (pires_pays['sous_nutrition']/10_000_000)
```

In [119…
```
# Effectuer un tri sur les sommes calculées afin de pouvoir ordonner les données du tableau
pires_pays = pires_pays.sort_values(
    by = 'pop_sous_nutrition',ascending = False).reset_index(drop = True)
pires_pays
```

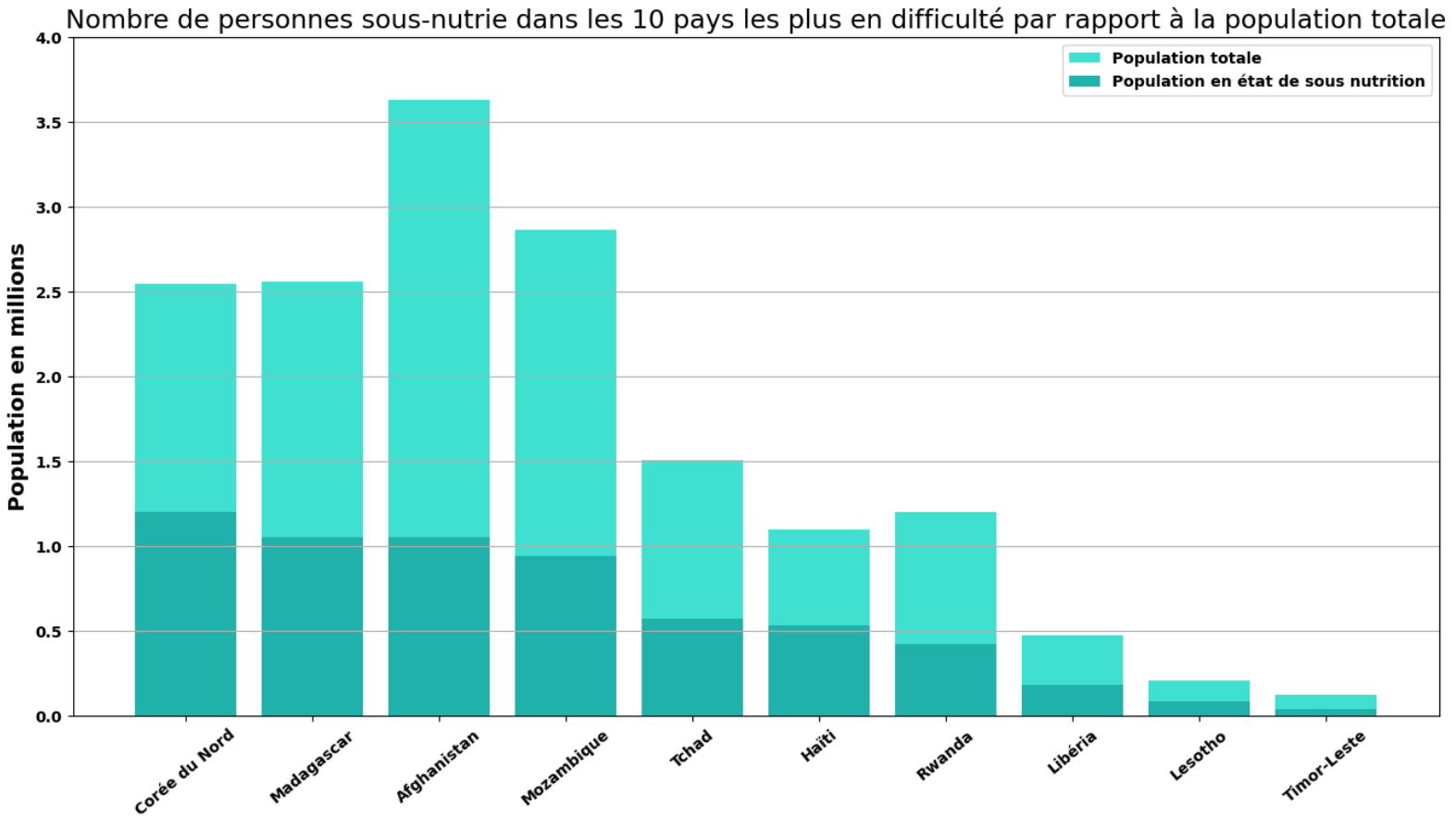
Out[119]:

	Zone	Population	sous_nutrition	Proportion_pop_sous_nutrie	Proportion_pop_nutrie	Somme	pop_totale	pop_sous_nu
0	Corée du Nord	25429825.0	12000000.0	47.19	52.81	100.0	2.542982	
1	Madagascar	25570512.0	10500000.0	41.06	58.94	100.0	2.557051	
2	Afghanistan	36296113.0	10500000.0	28.93	71.07	100.0	3.629611	
3	Mozambique	28649018.0	9400000.0	32.81	67.19	100.0	2.864902	
4	Tchad	15016753.0	5700000.0	37.96	62.04	100.0	1.501675	
5	Haïti	10982366.0	5300000.0	48.26	51.74	100.0	1.098237	
6	Rwanda	11980961.0	4200000.0	35.06	64.94	100.0	1.198096	
7	Libéria	4702226.0	1800000.0	38.28	61.72	100.0	0.470223	
8	Lesotho	2091534.0	800000.0	38.25	61.75	100.0	0.209153	
9	Timor-Leste	1243258.0	400000.0	32.17	67.83	100.0	0.124326	

In [120…

```
# Création et paramétrage d'un diagramme à barres empilées
line1 = plt.bar(pires_pays.Zone,
                pires_pays.pop_totale,
                color = 'turquoise',
                label = 'Population totale'
               )
line2 = plt.bar(pires_pays.Zone,
                pires_pays.pop_sous_nutrition,
                color = 'lightseagreen',
                label = 'Population en état de sous nutrition'
               )
plt.title('Nombre de personnes sous-nutrie dans les 10 pays '
          'les plus en difficulté par rapport à la population totale',
          fontsize=17
         )
plt.xlabel('',
           fontsize=14,
           fontweight='bold'
          )
plt.ylabel('Population en millions',
           fontsize=14,
           fontweight='bold'
```

```
        )
plt.yticks(np.arange(0, 4.2, step = 0.5))
plt.xticks(rotation = 40)
plt.grid(axis='y')
plt.legend(handles = [line1, line2])
plt.show()
```



In [121…

```
pires_pays
```

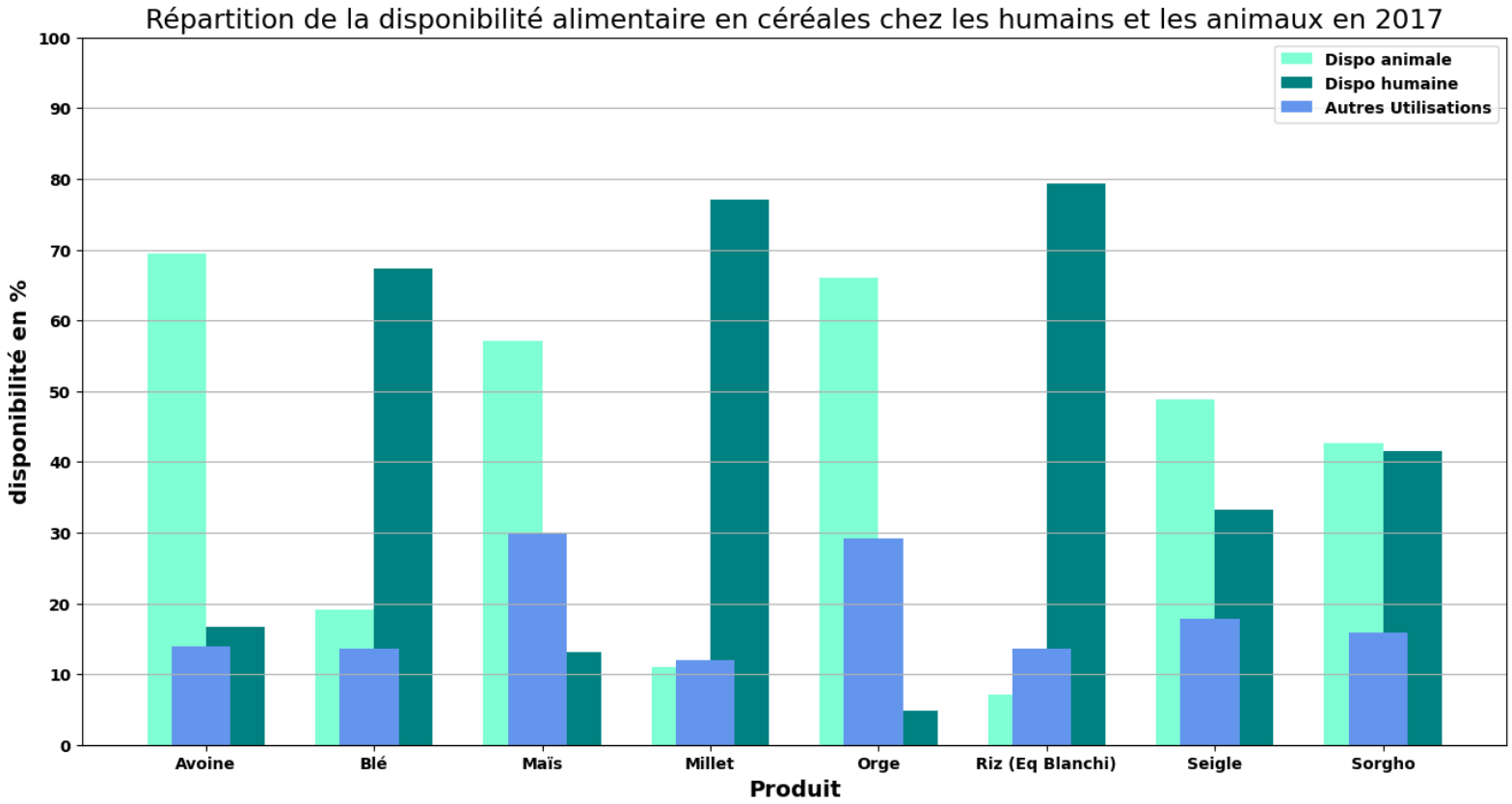
Out[121]:

	Zone	Population	sous_nutrition	Proportion_pop_sous_nutrie	Proportion_pop_nutrie	Somme	pop_totale	pop_sous_nu
0	Corée du Nord	25429825.0	12000000.0	47.19	52.81	100.0	2.542982	
1	Madagascar	25570512.0	10500000.0	41.06	58.94	100.0	2.557051	
2	Afghanistan	36296113.0	10500000.0	28.93	71.07	100.0	3.629611	
3	Mozambique	28649018.0	9400000.0	32.81	67.19	100.0	2.864902	
4	Tchad	15016753.0	5700000.0	37.96	62.04	100.0	1.501675	
5	Haïti	10982366.0	5300000.0	48.26	51.74	100.0	1.098237	
6	Rwanda	11980961.0	4200000.0	35.06	64.94	100.0	1.198096	
7	Libéria	4702226.0	1800000.0	38.28	61.72	100.0	0.470223	
8	Lesotho	2091534.0	800000.0	38.25	61.75	100.0	0.209153	
9	Timor-Leste	1243258.0	400000.0	32.17	67.83	100.0	0.124326	

In [122…

```
# Création du diagramme en bâtons (bâtons côte à côte) utilisation des céréales
"""Création et paramétrage d'un diagramme en bâtons"""
produit = cereales['Produit'].value_counts().index
pos = np.arange(len(produit))
width = 0.35

plt.rcParams['figure.figsize'] = [16,8]
plt.rcParams['text.color'] = 'k'
plt.rcParams['font.weight'] = 'bold'

line1 = plt.bar(pos - width/2,
                cereales_animaux,
                width,
                color = 'aquamarine',
                label = 'Dispo animale'
               )
line2 = plt.bar(pos + width/2,
                cereales_humains,
                width,
                color = 'teal',
                label = 'Dispo humaine'
               )
```

```
line3 = plt.bar(pos - width/12,
                cereales_autres,
                width,
                color = 'cornflowerblue',
                label = 'Autres Utilisations'
               )

plt.xlabel('Produit',
           fontsize=14
          )
plt.ylabel('disponibilité en %',
           fontsize=14
          )
plt.yticks(np.arange(0, 101, step = 10),
           fontweight = 'bold'
          )
plt.xticks(range(len(produit)),
           produit,
           fontweight = 'bold'
          )
plt.title('Répartition de la disponibilité alimentaire en céréales chez les humains '
          'et les animaux en 2017'
          ,fontsize=17
         )
plt.legend(handles=[line1, line2, line3],
           bbox_to_anchor=(1, 1)
          )
plt.grid(axis='y')
plt.show()
```



In [123…
```
#  Proportion de personnes sous-nutrie dans les pays en sous-nutrition
data_pays_sous_nutrition = data_pop_sous_nutrition_2017.loc[
    (data_pop_sous_nutrition_2017['sous_nutrition'] > 0)].reset_index(drop=True)
```

In [124…
```
somme_pop_pays_sous_nutrie = data_pays_sous_nutrition.sous_nutrition.sum()
```

In [125…
```
somme_pop_totale_pays_sous_nutrie = data_pays_sous_nutrition.Population.sum()
```

In [126…
```
prop_sous_nutrition = round((somme_pop_pays_sous_nutrie
                             / somme_pop_totale_pays_sous_nutrie*100),0)
```

In [127…
```
print('Il y a', round(somme_pop_totale_pays_sous_nutrie/1_000_000_000,2),
      'milliards de personnes dans les pays où on constate de la sous-nutrition,'
      'dont',round(somme_pop_pays_sous_nutrie/1_000_000,2),'millions de personnes en sous-nutrition,'
      ' ce qui représente', prop_sous_nutrition,'% de la population dans les pays en difficulté.')
```

Il y a 4.17 milliards de personnes dans les pays où on constate de la sous-nutrition,dont 535.7 millions de personnes en sous-nutrition, ce qui représente 13.0 % de la population dans les pays en difficulté.

In [128…
```
dispo_10_pays
```

Out[128]:

	Zone	Aliments pour animaux	Autres Utilisations	Disponibilité alimentaire (Kcal/personne/jour)	Disponibilité alimentaire en quantité (kg/personne/an)	Disponibilité de matière grasse en quantité (g/personne/jour)	Disponibilité de protéines en quantité (g/personne/jour)	Disponibilité intérieure
0	Centrafrique	1000.0	27000.0	1879.0	453.73	59.41	46.04	2582000.0
1	Zambie	408000.0	69000.0	1924.0	350.12	42.05	55.22	10041000.0
2	Madagascar	842000.0	2522000.0	2056.0	425.07	24.49	46.69	15216000.0
3	Afghanistan	768000.0	415000.0	2087.0	351.41	33.50	58.26	13515000.0
4	Haïti	481000.0	115000.0	2089.0	397.78	48.92	47.70	6175000.0
5	Corée du Nord	816000.0	695000.0	2093.0	458.93	36.41	54.99	14876000.0
6	Tchad	104000.0	2000.0	2109.0	319.45	47.10	62.75	5304000.0
7	Zimbabwe	131000.0	23000.0	2113.0	336.59	57.18	48.32	9507000.0
8	Ouganda	348000.0	68000.0	2126.0	535.58	46.57	52.64	26624000.0
9	Timor-Leste	9000.0	2000.0	2129.0	369.10	51.08	57.51	447000.0

10 rows × 23 columns